ENTERPRISE ARCHITECT

Série de Guides d'Utilisateur

# Ingénierie de Logiciel

Author: Sparx Systems
Date: 7/11/2024
Version: 17.0

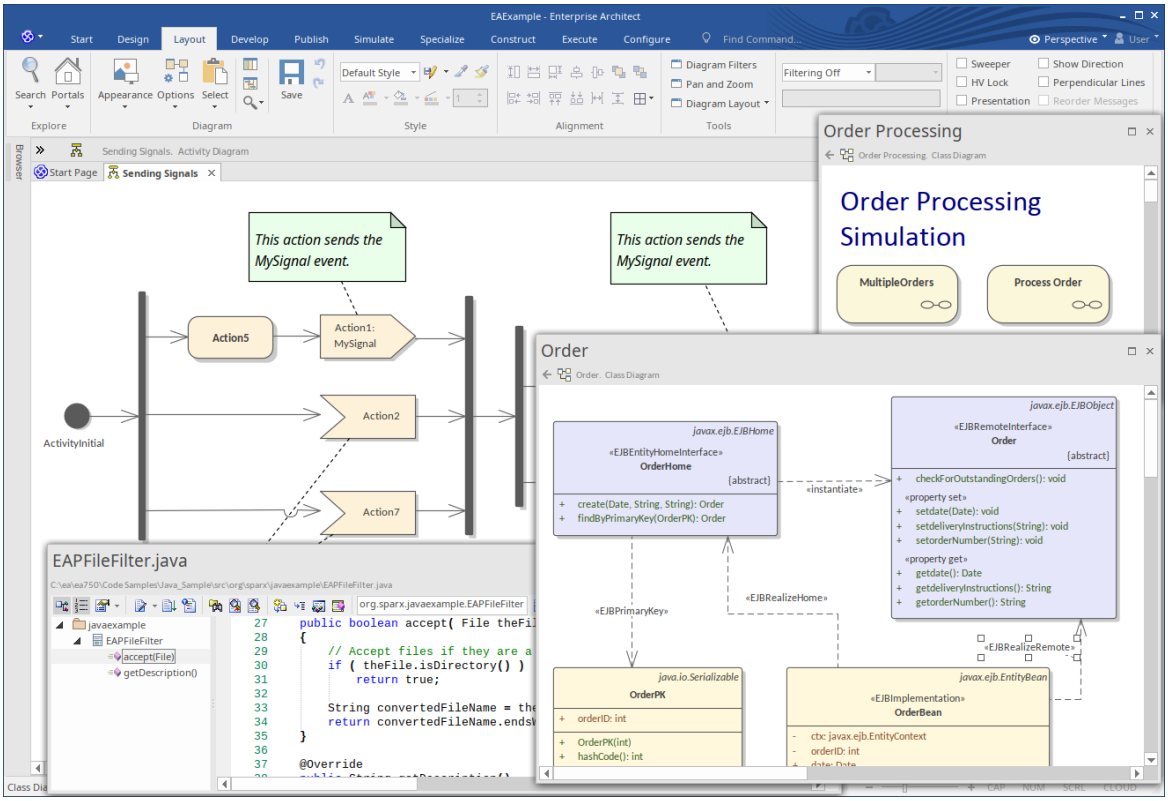
CRÉÉ AVEC ENTERPRISE ARCHITECT

# Table des Matières

# Ingénierie de Logiciel

**Créer et gérer des modèles structurels et Comportementale de logiciels efficaces et productifs**

L'ingénierie logicielle est la discipline de conception, d'implémentation et de maintenance des logiciels. Le processus d'ingénierie logicielle commence par des exigences et des contraintes en tant qu'entrées, et aboutit à un code de programmation et à des schémas qui sont déployés sur diverses plates-formes, créant ainsi des systèmes opérationnels.



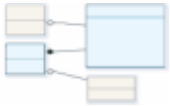
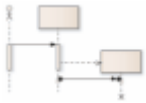

Enterprise Architect dispose d'un ensemble complet d'outils et fonctionnalités qui aident les ingénieurs logiciels à effectuer leur travail efficacement et à réduire le nombre d'erreurs dans les solutions mises en œuvre. Les fonctionnalités comprennent des outils de conception pour créer des modèles de logiciels, la génération de code automatisée, la rétro-ingénierie du code source, des binaires et des schémas, ainsi que des outils pour synchroniser le code source avec les modèles de conception. Le code de programmation peut être visualisé et modifié directement dans les Éditeurs de Code intégrés à Enterprise Architect , qui fournissent Intelli-sense et d'autres fonctionnalités pour aider au codage.

Un autre aspect convaincant de l'environnement est la capacité de retracer les classes d'implémentation jusqu'aux éléments de conception et architecture , puis jusqu'aux exigences, aux contraintes et aux autres spécifications, et enfin jusqu'aux parties prenantes et à leurs objectifs et visions.

Enterprise Architect supporte une large gamme de langages et de plates-formes de programmation et offre une intégration légère et transparente avec les deux environnements de développement intégrés les plus répandus : Visual Studio et Eclipse. De plus, il existe un Analyseur d'Exécution complet qui permet à l'ingénieur logiciel de concevoir, de créer, de déboguer et de tester des modules logiciels directement dans Enterprise Architect .

## Facilités

| Facilité | Description |
|---|---|
| Outils de développement | Découvrez l'environnement de développement étroitement intégré avec des outils et |

| | |
|---|---|
|  | des fonctionnalités exceptionnels. |
| Coder, construire et Déboguer  | Modèle , développer, déboguer, profiler et gérer une application depuis l'environnement modélisation . |
| Analyse visuelle du code en cours d'exécution  | Comprenez votre base de code en analysant visuellement le code en cours d'exécution. Utilisez les points Test , le profilage et la génération automatisée diagramme . |
| Générer Code Source  | Découvrez quelques façons de générer du code source pour une classe unique, une sélection de classes ou un Paquetage entier. Générer à partir de modèles structurels ou comportementaux. |
| Importation du code source  | Examinez les systèmes existants en important le code source dans Enterprise Architect . Vue et modifiez les définitions dialogue . Synchronisez le modèle avec les dernières mises à jour du code source. |

# Démarrage

Enterprise Architect propose une suite complète d'outils et fonctionnalités conçus pour améliorer la productivité et la précision des ingénieurs logiciels. Ces outils aident à créer des modèles logiciels, à générer automatiquement du code, à effectuer une rétro-ingénierie du code source et des schémas, et à synchroniser le code avec les modèles de conception.

## Paramètres de configuration

## Sélection de la perspective

Enterprise Architect partitionne les fonctionnalités étendues de l'outil en Perspectives , ce qui vous permet de vous concentrer sur une tâche spécifique et de travailler avec les outils dont vous avez besoin sans être distrait par d'autres fonctionnalités . Pour travailler avec fonctionnalités Modèle logiciel, vous devez d'abord sélectionner l'une de ces Perspectives :

L'ensemble Ingénierie de Logiciel :

 <nom de la perspective> > Ingénierie de Logiciel > Code Engineering

 <nom de la perspective> > Ingénierie de Logiciel > GoF Motifs

 <nom de la perspective> > Ingénierie de Logiciel > ICONIX

L'ensemble de conception UX :

 <nom de la perspective> > Conception UX > Modèles UI Win 32

La définition de la perspective garantit que le Case Management Model and Notation diagrammes de notation, leurs boîtes à outils et autres fonctionnalités de la perspective seront disponibles par défaut.

## Exemple Diagramme

Un exemple diagramme fournit une introduction visuelle au sujet et vous permet de voir certains des éléments et connecteurs importants que vous utilisez pour spécifier ou décrire des classes pour la visualisation de logiciels et l'ingénierie directe et inverse vers et depuis un large éventail de langages de programmation.

## Développement intégré

Dans cette rubrique, vous apprendrez à utiliser l'environnement de développement intégré aux fonctionnalités complètes. Vous apprendrez à créer des modèles structurels et comportementaux d'artefacts logiciels dans un éditeur de code riche, à générer et à rétroconcevoir du code, à personnaliser la manière dont le code est généré, exécuter des scripts d'analyse pour optimiser le code, à utiliser le débogueur et à définir des tests unitaires et bien plus encore.

## Modèles Comportementale

Dans cette rubrique, vous apprendrez à générer du code pour les langages de description de logiciels, de systèmes et de matériels directement à partir de diagrammes comportementaux, notamment : Diagrammes Statemachine , Séquence et d'activités. Cela ajoutera de nouvelles dimensions et précisions à votre façon de travailler avec les logiciels et les systèmes d'ingénierie.

## Motifs de Gang of Four (GoF)

Ce sujet présente les vingt-trois motifs de conception célèbres, regroupés sous le nom motifs Gang of Four (GoF), qui font référence à leurs quatre auteurs. Vous aurez à portée de main les solutions aux problèmes courants auxquels sont confrontés les ingénieurs logiciels et pourrez injecter ces motifs dans vos propres modèles, ajoutant ainsi à la qualité et à la rigueur de vos systèmes logiciels.

## Boîtes de dialogue Interface Utilisateur Win32

Dans cette rubrique, vous apprendrez à utiliser la fonctionnalité modélisation Interface Utilisateur d' Enterprise Architect qui vous permet de modéliser des écrans d'interface utilisateur à l'aide de contrôles Win32®. Les modèles peuvent être conçus en amont ou en aval et peuvent également fournir une interface pour la simulation diagramme Statemachine et Activity, leur permettant de recevoir et de traiter les entrées utilisateur.

## Cadre de code Gabarit

Dans cette rubrique, vous apprendrez à travailler avec le framework Code Gabarit qui régit la manière dont les modèles sont convertis en code. Il existe un ensemble standard de gabarits , mais vous pouvez les étendre pour créer vos propres gabarits et générer du code adapté à vos besoins. Il existe également gabarits qui contrôlent les transformations et la génération du langage de définition de base de données (DDL).

## Cadre grammatical

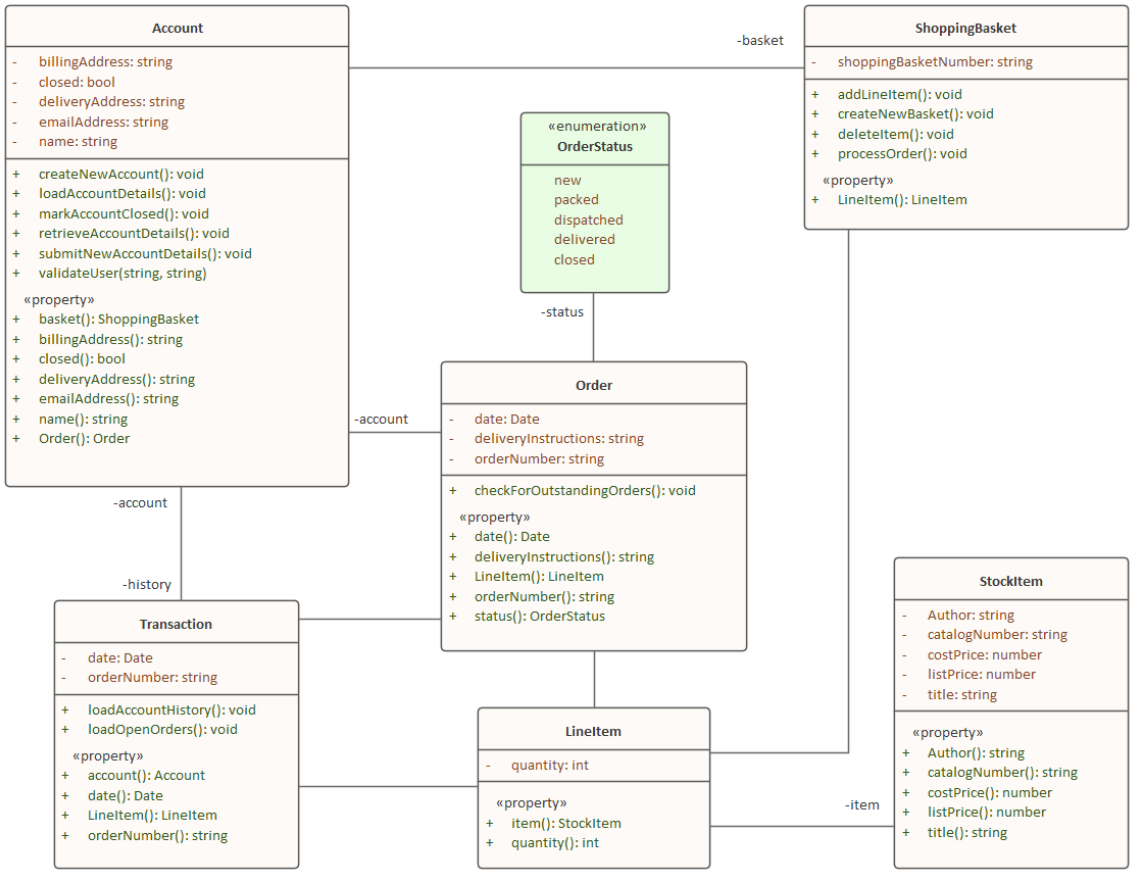
Dans cette rubrique, vous apprendrez à créer une grammaire pour convertir un langage de programmation non pris en charge en modèle UML . Enterprise Architect a intégré la support d'un large éventail de langages de programmation, mais si vous devez travailler avec un langage non pris en charge, vous pouvez utiliser le framework Grammar pour écrire votre propre analyseur. La grammaire est utilisée pour effectuer une rétro-ingénierie du code de programmation sous forme de texte et est le complément direct du framework Code Gabarit qui vous permet de spécifier comment un modèle UML pour un langage non pris en charge est converti en code.

## Cadre Code Miner

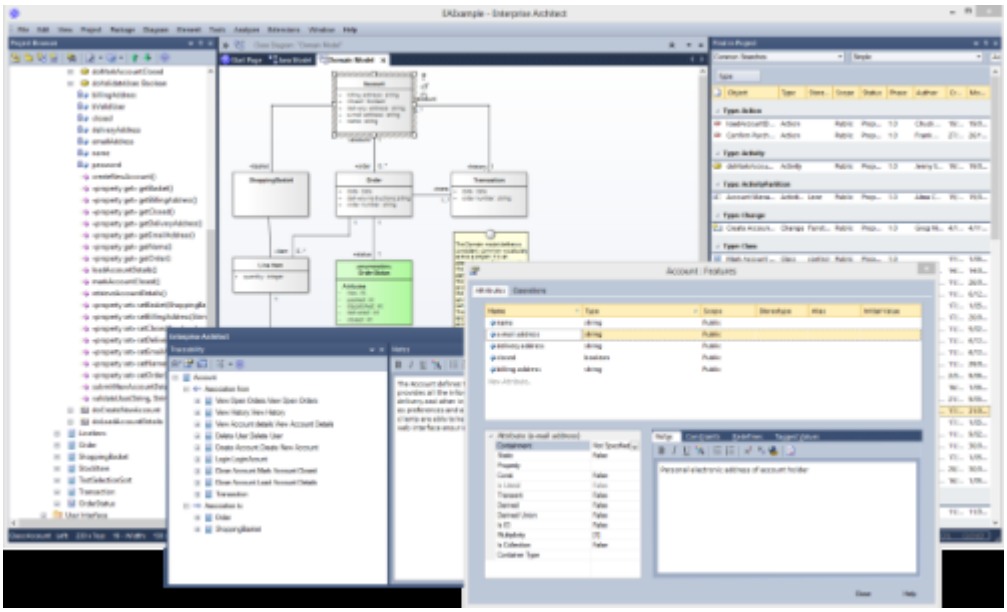
Dans cette rubrique, vous apprendrez à travailler avec une base de données de code source qui permet d'accéder aux données cachées dans le code source de manière rapide et efficace. Le code source est analysé pour créer une structure arborescente qui peut être utilisée pour analyser la structure du programme, calculer des métriques, tracer des relations et même effectuer une refactorisation.

# Exemple Diagramme

diagrammes logiciels vous permettent de modéliser la structure et le comportement des logiciels, y compris les interfaces utilisateur. Enterprise Architect a pour base support fondamental pour la modélisation des logiciels et l'outil supporte une large gamme de langages et de paradigmes de programmation. Dans ce diagramme nous voyons les classes utilisées pour modéliser une boutique en ligne, y compris les classes qui contiennent des compartiments pour Attributes , les opérations et Propriétés . Une énumération a également été utilisée pour modéliser le statut de la commande.



# Développement intégré



Enterprise Architect fournit un ensemble inégalé d'outils et fonctionnalités pour l'ingénieur logiciel, afin de l'aider à créer des systèmes logiciels robustes et sans erreur. L'ingénieur peut commencer par définir l' architecture et s'assurer qu'elle correspond aux exigences et aux spécifications. Les modèles neutres sur le plan technologique peuvent être transformés pour cibler une gamme complète de langages de programmation. L'environnement de développement piloté par Modèle s'adapte à diverses technologies.

## Fonctionnalités

**Outils de développement**

- Développement piloté par Modèle avec les meilleurs outils UML de leur catégorie
- Générer et rétro-ingénierie du code
- Personnaliser la génération de code avec gabarits
- Scripts d'Analyseur pour gérer vos applications
- Éditeurs de code pour créer la base de code
- Débogueurs pour enquêter sur le comportement
- Des profileurs pour visualiser le comportement
- Analyseurs pour enregistrer le comportement
- Testpoints pour la validation des contrats de programmation
- Intégration avec jUnit et nUnit
- Intégration Eclipse ou Visual Studio si nécessaire

**Traçabilité**

Traçabilité en un coup d'œil des généralisations, réalisations, associations, dépendances et plus encore. Personnalisez les vues de relation. Naviguez facilement entre les éléments associés dans le modèle.

**Usage**

Parcourez rapidement l'utilisation des éléments dans tous diagrammes . Effectuez des recherches d'éléments efficaces à l'aide de requêtes sophistiquées.

**Langues populaires**

- C/ C++

- Java
- Famille Microsoft .NET
- ADA
- Python
- Perl
- PHP

**Boîtes à outils** Des boîtes à outils sont fournies pour une vaste gamme de technologies modélisation et de langages de programmation.

**Motifs d'application** Enterprise Architect fournit des projets de démarrage complets, y compris des informations sur le modèle, du code et des scripts de construction, pour plusieurs types d'applications de base.

# Aperçu Fonctionnalité

L'ingénierie de code avec Enterprise Architect englobe largement divers processus pour la conception, la génération et la transformation du code à partir de votre modèle UML .

## Fonctionnalités

**Ingénierie de code pilotée par Modèle**

- Génération de code source et rétro-ingénierie pour de nombreux langages populaires, notamment C++, C# , Java, Delphi, VB.Net, Visual Basic, ActionScript, Python et PHP
- Un éditeur de code source « surlignage de syntaxe » intégré
- gabarits de génération de code, qui vous permettent de personnaliser le code source généré selon les spécifications de votre entreprise

**Des transformations pour un développement rapide**

- Transformations Model Driven Architecture (MDA) avancée à l'aide gabarits de transformation
- Transformations intégrées pour DDL, C# , Java, EJB et XSD
- Un Modèle indépendant de la plate-forme peut être utilisé pour générer et synchroniser plusieurs modèles spécifiques à la plate-forme, offrant ainsi une augmentation significative de la productivité
- diagramme de transformation XSL, boîte à outils, éditeur et débogueur.

**Analyse d'Exécution Visuelle / Débogage, Vérification et Visualisation**

- Exécuter des scripts de build, de test, de débogage, exécuter et de déploiement
- Intégrer le développement et modélisation UML au développement et à la compilation des sources
- Générer des classes de test NUnit et JUnit à partir de classes sources à l'aide de transformations MDA
- Intégrer le processus de test directement dans l'IDE Enterprise Architect
- Déboguer les applications .NET , Mono, Java et Microsoft Native (C, C++ et Visual Basic)
- Concevoir et exécuter des suites Test basées sur les principes de la programmation par contrat
- Débogage de la feuille de style XSL

**Modélisation de base de données**

Enterprise Architect vous permet de :

- Reverse engineering à partir de nombreux SGBD populaires, notamment SQL Server, My SQL, Access, PostgreSQL et Oracle
- Modèle tableaux de base de données, des colonnes, des clés, foreign keys et des relations complexes à l'aide UML et d'un profil modélisation de données intégré
- Générer des scripts DDL pour créer des structures de base de données cibles

**Ingénierie de la technologie XML**

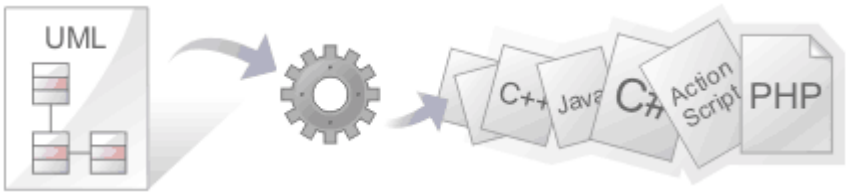
Enterprise Architect vous permet de modéliser, d'effectuer l'ingénierie directe et inverse rapidement sur deux technologies XML W3C clés :

- Schéma XML (XSD)
- Langage de définition de service Web (WSDL)

support de XSD et WSDL est essentielle au développement d'une Service Oriented Architecture (SOA) complète, et le couplage d' UML 2.5 et XML fournit le mécanisme naturel pour la mise en œuvre d'artefacts SOA basés sur XML au sein

d'une organisation.

# Générer Code Source




La génération de code source est le processus de création de code de programmation à partir d'un modèle UML . Cette approche présente de nombreux avantages, car les Paquetages , classes et interfaces de code source sont automatiquement créés et élaborés avec des variables et des méthodes.

Enterprise Architect peut également générer du code à partir d'un certain nombre de modèles comportementaux, notamment diagrammes Statemachine , Séquence et Activity. Il existe un mécanisme gabarit extrêmement flexible qui permet à l'ingénieur de personnaliser entièrement la manière dont le code source est généré, y compris les en-têtes de commentaires dans les méthodes et les classes de collection utilisées.

D'un point de vue technique et qualité, l'avantage le plus convaincant de cette approche est que les modèles UML et donc l' architecture et la conception sont synchronisés avec le code de programmation. Un chemin traçable ininterrompu peut être créé à partir des objectifs, des moteurs commerciaux et des exigences des parties prenantes jusqu'aux méthodes du code de programmation.

## Facilités

| Facilité | Description |
|---|---|
| Langues | Enterprise Architect supporte la génération de code dans chacun de ces langages logiciels :<br>• Scénario Action<br>• Ada<br>• ArcGIS<br>• C<br>• C# (pour .NET 1.1, .NET 2.0 et .NET 4.0)<br>• C++ (standard, plus extensions C++ gérées .NET )<br>• Delphes<br>• Java (y compris Java 1.5, Aspects et Génériques)<br>• JavaScript<br>• mFQL<br>• MySQL<br>• PHP<br>• Python<br>• Teradata SQL<br>• Visual Basic<br>• Visual Basic .NET<br>• Script de flux de travail<br>Vous pouvez également générer du code Hardware Definition Language dans ces langages :<br>• VHDL<br>• Verilog |

| | |
|---|---|
| | • SystèmeC |
| Éléments | Le code est généré à partir d'éléments de modèle de classe ou d'interface. Vous devez donc créer les éléments de classe et d'interface requis pour la génération. Tous les autres types d'éléments contribuant au code (tels que Statemachines ou les activités) doivent être des éléments enfants d'une classe.<br>Ajoutez des attributs (qui deviennent des variables) et des opérations (qui deviennent des méthodes). Les contraintes et les réceptions sont également prises en charge dans le code. |
| Paramètres | Avant de générer du code, vous devez vous assurer que les paramètres par défaut pour la génération de code correspondent à vos besoins ; configurez les valeurs par défaut pour qu'elles correspondent à la langue et aux préférences requises.<br>Les préférences que vous pouvez définir incluent les constructeurs et destructeurs par défaut, les méthodes pour les interfaces et les options Unicode pour les langues créées.<br>Des langages tels que Java support les « espaces de noms » et peuvent être configurés pour spécifier une racine d'espace de noms.<br>En plus des paramètres par défaut pour la génération de code, Enterprise Architect facilite la définition d'options de génération spécifiques pour chacun des langages pris en charge. |
| Cadre de code Gabarit | Le Code Gabarit Framework (CTF) vous permet de personnaliser la manière dont Enterprise Architect génère le code source et permet également la génération de langages qui ne sont pas spécifiquement pris en charge par Enterprise Architect . |
| Chemins locaux | Les noms de chemin locaux vous permettent de remplacer les noms de répertoire par étiquettes . |
| Code Comportementale | Vous pouvez également générer du code logiciel à partir de trois paradigmes modélisation comportementale UML :<br>• diagrammes d'interaction ( Séquence )<br>• diagrammes d'activité<br>• diagrammes Statemachine (en utilisant Legacy Statemachine Gabarits dans les opérations de génération de code sous « Tâches »)<br>• diagrammes Statemachine (utilisant un artefact Statemachine Exécutable ) |
| Génération de code en direct | Dans le menu déroulant « Développer > Code source > Options », vous avez la possibilité de mettre à jour votre code source instantanément lorsque vous apportez des modifications à votre modèle. |
| Tâches | Lorsque vous générez du code, vous effectuez une ou plusieurs de ces tâches :<br>• Générer une classe unique<br>• Générer un groupe de classes<br>• Générer un Paquetage<br>• Mettre à jour le contenu Paquetage |

## Notes

- La plupart des outils fournis par Enterprise Architect pour l'ingénierie et le débogage du code sont disponibles dans

les éditions Professional et supérieures d' Enterprise Architect ; la génération de code Comportementale est disponible dans les éditions Unified et Ultimate

- Lorsque la sécurité est activée, vous avez besoin des autorisations d'accès « Générer Code Source et le DDL » et « Rétro-ingénierie à partir du DDL et du code source »

# Générer une classe unique

Avant de générer du code pour une seule classe, vous :

- Compléter la conception de l'élément de modèle (Classe ou Interface)
- Créer des connecteurs d'héritage vers les parents et des associations vers d'autres classes utilisées
- Créez des connecteurs d'héritage vers les interfaces que votre classe implémente ; le système fournit une option pour générer des stubs de fonction pour toutes les méthodes d'interface qu'une classe implémente

## Générer du code pour une seule Classe

| Étape | Action |
|---|---|
| 1 | Ouvrez le diagramme contenant la classe ou l'interface pour laquelle générer du code. |
| 2 | Cliquez sur la classe ou l'interface requise et sélectionnez l'option de ruban « Développer > Code source > Générer > Générer un élément unique » ou appuyez sur F11.<br>La dialogue « Générer du code » s'affiche, grâce à laquelle vous pouvez contrôler comment et où votre code source est généré. |
| 3 | Dans le champ « Chemin », cliquez sur le bouton  et sélectionnez un nom de chemin vers lequel votre code source doit être généré. |
| 4 | Dans le champ « Langue cible », cliquez sur la flèche déroulante et sélectionnez la langue à générer ; cela devient l'option permanente pour cette classe, donc modifiez-la à nouveau si vous n'effectuez qu'un seul passage dans une autre langue. |
| 5 | Cliquez sur le bouton Avancé.<br>La dialogue « Options Object » s'affiche, fournissant des sous-ensembles des pages d'options « Ingénierie du code source » et de langage de code dans la dialogue « Préférences ». |
| 6 | Définissez les options personnalisées (pour cette classe uniquement), puis cliquez sur le bouton Fermer pour revenir à la dialogue « Générer le code ». |
| 7 | Dans les champs « Import(s) / En-tête(s) », saisissez des instructions d'importation, des #includes ou d'autres informations d'en-tête.<br>Note que dans le cas de Visual Basic, ces informations sont ignorées ; dans le cas de Java, les deux zones de texte d'importation sont fusionnées ; et dans le cas de C++, la première zone de texte d'importation est placée dans le fichier d'en-tête et la seconde dans le fichier corps (.cpp). |
| 8 | Cliquez sur le bouton Générer pour créer le code source. |
| 9 | Une fois terminé, cliquez sur le bouton Vue pour voir ce qui a été généré.<br>Note que vous devez d'abord configurer votre visualiseur/éditeur par défaut pour chaque type de langue ; vous pouvez également configurer l'éditeur par défaut sur la page « Éditeurs de Code » de la fenêtre Préférences (' Démarrer > Application > Préférences > Préférences > Source Code Engineering > Éditeurs de Code '). |

# Générer un groupe de classes

En plus de pouvoir générer du code pour une classe individuelle, vous pouvez également sélectionner un groupe de classes pour la génération de code par lots. Lorsque vous faites cela, vous acceptez toutes les options de génération de code par défaut pour chaque classe de l'ensemble.

## Générer un groupe de classe

| Étape | Détail |
|---|---|
| 1 | Sélectionnez un groupe de classes et/ou d'interfaces dans un diagramme . |
| 2 | Cliquez sur un élément du groupe et sélectionnez l'option de ruban « Développer > Code source > Générer > Générer les éléments sélectionnés » (ou appuyez sur Maj+F11).<br>Si aucun code n'existe pour les éléments sélectionnés, la dialogue « Enregistrer sous » s'affiche, dans laquelle vous spécifiez le chemin d'accès et le nom de chaque fichier de code ; saisissez ces informations et cliquez sur le bouton Enregistrer. |
| 3 | La dialogue « Génération par lots » s'affiche, indiquant l'état du processus au fur et à mesure de son exécution (le processus peut être trop rapide pour voir cette dialogue ).<br>Si le code existe déjà pour les éléments de classe sélectionnés et que des modifications ont été apportées au nom ou à la structure de la classe, la boîte de dialogue « Synchroniser l'élément < nom paquetage >.< nom de l'élément> » peut également dialogue ; cette dialogue permet de synchroniser le modèle et le code. |

## Notes

- Si l'un des éléments sélectionnés n'est pas une classe ou une interface, l'option de génération de code n'est pas disponible

# Générer un Paquetage

En plus de générer du code source à partir de classes uniques et de groupes de classes, vous pouvez générer du code à partir d'un Paquetage . Cette fonctionnalité fournit des options pour générer de manière récursive du code à partir de Paquetages enfants et générer automatiquement des structures de répertoires basées sur la hiérarchie Paquetage . Cela vous aide à générer du code pour toute une branche de votre modèle de projet en une seule étape.

## Accéder

| | |
|---|---|
| Ruban | Développer > Code Source > Générer > Générer Tout |
| Raccourcis Clavier | Ctrl+Alt+K |

## Générer du code à partir d'un Paquetage , sur la dialogue Générer le Code Source Paquetage

| Étape | Action |
|---|---|
| 1 | Dans le champ « Synchroniser », cliquez sur la flèche déroulante et sélectionnez l'option de synchronisation appropriée :<br>• « Synchroniser le modèle et le code » : le code des classes avec des fichiers existants est synchronisé avec ce fichier ; le code des classes sans fichier existant est généré vers le fichier cible affiché<br>• « Écraser le code » : tous les fichiers cibles sélectionnés sont écrasés (générés vers l'avant)<br>• « Ne pas générer » : Générer du code uniquement pour les classes sélectionnées qui n'ont pas de fichier existant ; toutes les autres classes sont ignorées |
| 2 | Mettez en surbrillance les classes pour lesquelles générer du code ; laissez non sélectionnées celles pour lesquelles vous ne souhaitez pas générer de code.<br>Si vous souhaitez afficher plus d'informations dans la disposition , vous pouvez redimensionner le dialogue et ses colonnes. |
| 3 | Pour qu'Enterprise Enterprise Architect génère automatiquement des répertoires et des noms de fichiers en fonction de la hiérarchie Paquetage , cochez la case « Générer automatiquement des fichiers » ; cela active le champ « Répertoire racine », dans lequel vous sélectionnez un répertoire racine sous lequel les répertoires sources doivent être générés.<br>Par défaut, la fonctionnalité « Générer automatiquement des fichiers » ignore tous les chemins de fichiers déjà associés à une classe ; vous pouvez modifier ce comportement en cochant également la case « Conserver les chemins de fichiers existants ». |
| 4 | Pour inclure le code de tous les sous-packages dans la sortie, cochez la case 'Inclure Pacquetages Enfants' . |
| 5 | Cliquez sur le bouton Générer pour commencer à générer du code.<br>Au fur et à mesure de la génération du code, Enterprise Architect affiche des messages de progression. Si une classe nécessite un nom de fichier de sortie, le système vous prompts à en saisir un au moment opportun (en supposant que l'option Générer automatiquement les fichiers n'est pas sélectionnée). Par |

| | |
|---|---|
| | exemple, si les classes sélectionnées incluent des classes partielles, une prompt s'affiche pour vous demander de saisir le nom de fichier dans lequel générer le code pour la deuxième classe partielle. |

## Plus d'informations sur le dialogue

| Option | Action |
|---|---|
| Paquetage de racines | Vérifiez le nom du Paquetage pour lequel le code doit être généré. |
| Synchroniser | Sélectionnez les options qui spécifient comment les fichiers existants doivent être régénérés. |
| Générer automatiquement des fichiers | Spécifiez si Enterprise Architect doit générer automatiquement des noms de fichiers et des répertoires, en fonction de la hiérarchie Paquetage . |
| Répertoire racine | Si Générer automatiquement les fichiers est sélectionnée, affichez le chemin sous lequel les structures de répertoire générées sont créées. |
| Conserver les chemins de fichiers existants | Si Générer automatiquement les fichiers est sélectionnée, indiquez si vous souhaitez utiliser les chemins de fichiers existants associés aux classes.<br>Si l' Générer automatiquement les fichiers n'est pas sélectionnée, Enterprise Architect génère automatiquement le code de classe vers les chemins déterminés, que les fichiers sources soient déjà associés aux classes ou non. |
| Inclure tous Paquetages enfants | Générez également du code pour toutes les classes dans tous les sous-packages du Paquetage cible dans la liste.<br>Cette option facilite la génération récursive de code pour un Paquetage donné et ses sous-Packages. |
| Sélectionner les objets à Générer | Répertoriez toutes les classes disponibles pour la génération de code sous les Paquetages cibles ; seul le code des classes sélectionnées (mises en surbrillance) est généré.<br>Les classes sont répertoriées avec leur fichier source cible. |
| Sélectionner tout | Marquez toutes les classes de la liste comme sélectionnées. |
| Ne rien sélectionner | Marquer toutes les classes de la liste comme non sélectionnées. |
| Générer | Démarrer la génération du code pour toutes les classes sélectionnées. |
| Annuler | Quittez la dialogue ' Générer Paquetage Source Code' ; aucun code de classe n'est généré. |

# Mettre à jour le contenu Paquetage

Outre la génération et l'importation de code, Enterprise Architect offre la possibilité de synchroniser le modèle et le code source, en créant un modèle qui représente les dernières modifications du code source et vice versa. Vous pouvez utiliser soit le modèle comme source, soit le code comme source.

Le comportement et les actions de la synchronisation dépendent des paramètres que vous avez sélectionnés dans la page « Attributes et opérations » de la dialogue « Préférences ». En utilisant ces paramètres, vous pouvez soit protéger, soit supprimer automatiquement les informations du modèle qui ne sont pas présentes dans le code, et prompt une décision sur fonctionnalités du code qui ne sont pas présentes dans le modèle. Dans ces deux exemples, les cases à cocher appropriées ont été sélectionnées pour une protection maximale des données :

- Vous avez généré du code source, mais vous avez apporté des modifications ultérieures au modèle ; lorsque vous générez à nouveau du code, Enterprise Architect ajoute tous les nouveaux attributs ou méthodes au code source existant, laissant intact ce qui existe déjà, ce qui signifie que les développeurs peuvent travailler sur le code source, puis générer des méthodes supplémentaires selon les besoins à partir du modèle, sans que leur code ne soit écrasé ou détruit
- Vous avez peut-être apporté des modifications à un fichier de code source, mais le modèle contient notes et des caractéristiques détaillées que vous ne souhaitez pas perdre ; en synchronisant le code source dans le modèle, vous importez des attributs et des méthodes supplémentaires, mais ne modifiez pas les autres éléments du modèle.

Grâce aux méthodes de synchronisation, il est simple de maintenir le code source et les éléments du modèle à jour et synchronisés.

## Accéder

| | |
|---|---|
| Ruban | Développer > Code source > Synchroniser > Synchroniser Paquetage |

## Synchronisez le contenu Paquetage avec le code source

| Champ/Bouton | Action |
|---|---|
| Type de mise à jour | Sélectionnez le bouton radio pour effectuer une ingénierie directe ou une ingénierie inverse des classes Paquetage , selon le cas. |
| Inclure paquetages enfants dans la génération | Cochez la case pour inclure Paquetages enfants dans la synchronisation. |
| OK | Cliquez sur le bouton pour démarrer la synchronisation.<br>Enterprise Architect utilise les noms de répertoire spécifiés lors de la première importation/génération de la source du projet et met à jour le modèle ou le code source en fonction de l'option choisie. Si :<br>• Exécution de la synchronisation vers l'avant AND<br>• Il existe des différences entre le modèle et le code AND<br>• La case à cocher « Lors de la synchronisation en avant, prompt la suppression fonctionnalités de code qui ne sont pas dans le modèle » est sélectionnée dans la boîte de dialogue « Options - Attributes et opérations »<br>Ensuite, la dialogue « Synchroniser l'élément < nom paquetage >.< nom de l'élément> » s'affiche. |

| | |
|---|---|
| | Dans le cas contraire, aucune autre action n’est requise. |

## Notes

- La synchronisation du code ne modifie pas les corps des méthodes ; le code comportemental ne peut pas être synchronisé et la génération de code ne fonctionne que lors de la génération de l'intégralité du fichier
- Dans les éditions Corporate , Unified et Ultimate d' Enterprise Architect , si la sécurité est activée, vous devez disposer de l'autorisation « Générer Code Source et DDL » pour synchroniser le code source avec les éléments du modèle.

# Synchroniser Modèle et le code

Vous pouvez soit :

- Synchroniser le code d'un Paquetage de Classes avec le modèle dans la fenêtre Navigateur , ou
- Régénérer le code à partir d'un lot de classes dans le modèle

Dans de tels processus, il peut y avoir des éléments dans le code qui ne sont pas présents dans le modèle.

Si vous souhaitez piéger ces éléments et les résoudre manuellement, cochez la case « Lors de la synchronisation directe, prompt la suppression fonctionnalités de code non présentes dans le modèle » dans la boîte de dialogue « Options - Attributes et opérations », de sorte que la dialogue « Synchroniser l'élément < nom paquetage >.< nom de l'élément> » s'affiche, fournissant des options pour répondre à chaque élément.

## Synchroniser Items

| Bouton | Détail |
|---|---|
| Sélectionner tout | Mettez en surbrillance et sélectionnez tous les éléments dans la colonne Fonctionnalité . |
| Tout Effacer | Désélectionnez et supprimez la surbrillance de tous les éléments de la colonne Fonctionnalité . |
| Supprimer | Marquez les fonctionnalités de code sélectionnées à supprimer du code (la valeur dans la colonne Action change en Supprimer). |
| Réaffecter | Marquez les fonctionnalités de code sélectionnées à réaffecter aux éléments du modèle.<br>Cela n'est possible que lorsqu'un élément de modèle approprié est présent et n'est pas déjà défini dans le code.<br>La dialogue Sélectionner la Fonctionnalité de classe correspondante s'affiche. Vous pouvez alors sélectionner la classe à laquelle réaffecter la fonctionnalité . Cliquez sur le bouton OK pour marquer la fonctionnalité à réaffecter. |
| Ignorer | Marquez les éléments de code sélectionnés non présents dans le modèle à ignorer complètement (valeur par défaut ; la valeur dans la colonne Action reste ou passe à <none> ). |
| Réinitialiser par défaut | Réinitialisez les éléments sélectionnés sur Ignorer (la valeur dans la colonne Action passe à <none> ). |
| OK | Apportez les modifications attribuées aux éléments et fermez le dialogue . |

# Namespaces

Les langages tels que Java support les structures Paquetage ou les espaces de noms. Dans Enterprise Architect vous pouvez spécifier un Paquetage comme racine d'espace de noms, ce qui indique où commence la structure d'espace de noms de votre modèle de classe ; tous Paquetages subordonnés situés sous une racine d'espace de noms formeront la hiérarchie d'espace de noms pour les classes et les interfaces contenues.

Pour définir un Paquetage comme racine d'espace de noms, cliquez sur le Paquetage dans la fenêtre Navigateur et sélectionnez l'option de ruban « Développer > Code source > Options > Définir comme racine Namespace ». L'icône Paquetage dans la fenêtre Navigateur change pour afficher un coin coloré indiquant que ce Paquetage est une racine d'espace de noms.

Le code source Java généré, par exemple, ajoutera automatiquement une déclaration Paquetage au début du fichier généré, indiquant l'emplacement de la classe dans la hiérarchie Paquetage sous la racine de l'espace de noms.

Pour effacer une racine d'espace de noms existante, cliquez sur la racine d'espace de noms Paquetage dans la fenêtre Navigateur et décochez l'option de ruban « Développer > Code source > Options > Définir comme racine Namespace »

Pour afficher une liste d'espaces de noms, sélectionnez l'option de ruban « Paramètres > Données de référence > Paramètres > Racines Namespace ; la dialogue « Namespaces » s'affiche. Si vous double-cliquez sur un espace de noms dans la liste, le Paquetage est mis en surbrillance dans la fenêtre Navigateur ; vous pouvez également cliquez-droit sur l'espace de noms et sélectionner l'option « Localiser Paquetage dans Navigateur ».

Vous pouvez également effacer la racine de l'espace de noms sélectionné en sélectionnant l'option « Effacer l'attribut Namespace ».

Pour omettre un Paquetage subordonné d'une définition d'espace de noms, sélectionnez l'option de ruban « Développer > Code source > Options > Supprimer Namespace » ; pour inclure à nouveau le Paquetage dans l'espace de noms, désélectionnez l'option de ruban.

## Notes

- Lors de la génération de code, tout nom Paquetage contenant des caractères d'espacement est automatiquement traité comme une racine d'espace de noms.

# Importation du code source


La possibilité de visualiser simultanément le code de programmation et les modèles dont il est dérivé apporte de la clarté à la conception d'un système. L'une des fonctionnalités pratiques d'ingénierie de code d' Enterprise Architect est la possibilité de procéder à une rétro-ingénierie du code source dans un modèle UML . Une large gamme de langages de programmation sont pris en charge et il existe des options qui régissent la manière dont les modèles sont générés. Une fois le code dans le modèle, il est possible de le maintenir synchronisé avec le modèle, que les modifications aient été apportées directement dans le code ou dans le modèle lui-même. Les structures de code sont mappées dans leurs représentations UML ; par exemple, une classe Java est mappée dans un élément de classe UML , les variables sont définies comme des attributs, les méthodes modélisées comme des opérations et les interactions entre les classes Java représentées par les connecteurs appropriés.

La représentation du code de programmation sous forme de constructions de modèles vous aide à mieux comprendre la structure du code et la manière dont il implémente la conception, architecture et les exigences, et finalement comment il fournit la valeur commerciale.

Il est important de note que si un système n'est pas bien conçu, la simple importation de la source dans Enterprise Architect ne le transforme pas en un modèle UML facilement compréhensible. Lorsque vous travaillez avec un système mal conçu, il est utile d'évaluer le code en unités gérables en examinant les Paquetages ou éléments de modèle individuels générés à partir du code ; par exemple, en faisant glisser une classe spécifique d'intérêt sur un diagramme , puis en utilisant l'option « Insérer des éléments associés » à un niveau pour déterminer les relations immédiates entre cette classe et d'autres classes. À partir de ce point, il est possible de créer des cas d'utilisation qui identifient l'interaction entre les classes de code source, offrant ainsi une vue d'ensemble du fonctionnement de l'application.

Plusieurs options guident la manière dont le code est rétroconçu, notamment si les commentaires sont importés dans notes et comment ils sont formatés, comment les méthodes de propriété sont reconnues et si des relations de dépendance sont créées pour les types de retour d'opération et de paramètre.

## Propriété des droits d'auteur

Les situations qui se prêtent généralement à la rétro-ingénierie ont tendance à fonctionner sur du code source qui :

- Vous avez déjà développé
- Fait partie d'une bibliothèque tierce pour laquelle vous avez obtenu l'autorisation d'utiliser
- Fait partie d'un cadre que votre organisation utilise
- Est développé quotidiennement par vos développeurs

Si vous examinez un code dont vous ou votre organisation n’êtes pas propriétaire ou pour lequel vous n’avez pas l’autorisation spécifique de copier et de modifier, vous devez vous assurer que vous comprenez et respectez les restrictions de droits d’auteur sur ce code avant de commencer le processus de rétro-ingénierie.

## Langues supportées pour Rétro-ingénierie

| Langue |
| --- |
| Scénario Action |
| |

| |
|---|
| Ada 2012 (éditions Unified et Ultimate ) |
| C |
| C# |
| C++ |
| CORBA IDL ( MDG Technologie ) |
| Delphes |
| Java |
| PHP |
| Python |
| SystemC (éditions Unified et Ultimate ) |
| Verilog (éditions Unified et Ultimate ) |
| VHDL (éditions Unified et Ultimate ) |
| Visual Basic |
| Visual Basic .NET |

## Notes

- Rétro-ingénierie est supporté dans les éditions Professional , Corporate , Unified et Ultimate d' Enterprise Architect
- Si la sécurité est activée, vous devez disposer de l'autorisation « Reverse engineering à partir de DDL et du code source » pour effectuer la rétro-ingénierie du code source et synchroniser les éléments du modèle avec le code
- À l'aide d' Enterprise Architect , vous pouvez également importer certains types de fichiers binaires, tels que les fichiers Java .jar et les fichiers .NET PE.
- Rétro-ingénierie d'autres langages est actuellement disponible grâce à l'utilisation de MDG Technologies répertoriées sur les pages MDG Technologie du site Web Sparx Systems

# Projets d'importation

Enterprise Architect support d'importer des projets logiciels créés dans Visual Studio, Mono, Eclipse et NetBeans. L'importation et le travail sur des projets dans Enterprise Architect présentent de nombreux avantages, notamment l'accès immédiat aux outils modélisation et fonctionnalités de gestion réputés d' Enterprise Architect , mais également l'accès aux outils de développement tels que la simulation, le débogage et le profilage.

## Accéder

| | |
|---|---|
| Ruban | Développer > Code source > Solutions > Importer un <type de projet> |

## Importer une solution Visual Studio

Cette option vous permet d'importer un ou plusieurs projets à partir d'un fichier de solution Visual Studio existant ou d'une instance en cours d'exécution de Visual Studio. L' assistant génère un modèle de classe pour chacun des projets et les Scripts d'Analyseur appropriés pour chaque configuration de Visual Studio.

## Importer une solution mono

Cette option vous permet d'importer des projets Mono à partir d'un fichier solution. La dialogue qui s'affiche est la même que la dialogue « Importation de Visual Studio », mais vous pouvez choisir de cibler Linux ou Windows . L' assistant génère un modèle de classe pour chacun des projets et les configure pour le débogage. Les Scripts d'Analyseur générés font référence à msbuild pour construire les projets.

## Importer un projet Eclipse

L' Assistant Eclipse peut effectuer une rétro-ingénierie d'un projet Java décrit par son fichier Eclipse .project et sa compilation ANT. La fonctionnalité produira un modèle de classe UML et Scripts d'Analyseur pour chacune des cibles ANT que vous sélectionnez. Le processus générera également un script pour chaque protocole de débogage que vous sélectionnez via l' Assistant . Vous aurez le choix entre JDWP (Java Déboguer Wire Protocol), adapté aux serveurs, et JVMTI (Java Virtual Machine Tools Interface), adapté aux applications Java autonomes. Ces scripts doivent être utilisés pour déboguer le projet dans Enterprise Architect .

## Importer un projet NetBeans

L' Assistant NetBeans peut effectuer une rétro-ingénierie d'un projet Java décrit par un fichier de projet XML NetBeans et une compilation ANT. L' Assistant crée un modèle de classe UML du projet et Scripts d'Analyseur pour chacune des cibles ANT que vous sélectionnez. Le processus génère également un script pour chaque protocole de débogage que vous sélectionnez via l' Assistant . Ces scripts doivent être utilisés pour déboguer le projet dans Enterprise Architect . Vous aurez le choix entre JDWP (Java Déboguer Wire Protocol), adapté aux serveurs, et JVMTI (Java Virtual Machine Tools Interface), qui convient aux applications Java autonomes.

## Options d'importation

Lorsque vous choisissez d'importer une solution Visual Studio ou Mono, la dialogue « Importer une solution Visual Studio » s'affiche. Remplissez les champs comme indiqué dans ce tableau .

Lorsque vous choisissez d'importer une solution Eclipse ou Netbeans, l'écran de démarrage Assistant approprié s'affiche. Parcourez les écrans en suivant les prompts s'affichent sur chaque écran.

| Option | Description |
|---|---|
| <liste des projets> | Après avoir sélectionné le fichier de solution, les projets de la solution sont répertoriés dans le panneau. Sélectionnez les projets à importer par l' Assistant .<br>Vous pouvez utiliser le bouton Tous pour sélectionner tous les projets et le bouton Aucun pour effacer la sélection de projets. |
| Sélectionner le fichier de solution | Recherchez et sélectionnez le fichier de solution à partir duquel effectuer l'importation. Les fichiers de solution Mono et les fichiers de solution Visual Studio ont une extension de fichier .sln. |
| Effectuer un Exécuter à Sec | Sélectionnez cette option pour effectuer l'importation en tant exécuter à sec, afin de vérifier les éventuelles erreurs dans le processus ou la sortie avant de répéter l'importation pour modifier le contenu du modèle. Cliquez sur le bouton Journal Vue pour vérifier le log de l'importation. |
| Créer Paquetage par fichier | Sélectionnez cette option pour effectuer l'importation avec une granularité plus fine, en créant un Paquetage distinct pour chaque fichier. |
| Importer | Cliquez sur ce bouton pour démarrer le processus d'importation. |
| Demande de définitions de macros manquantes | Non applicable aux importations de Mono Solution.<br>Pour les projets C++ dans Visual Studio, l'analyseur peut rencontrer des macros non reconnues. Si vous sélectionnez cette option, vous serez averti lorsqu'un tel événement se produit et aurez la possibilité de définir la macro. Si vous ne sélectionnez pas cette option, le modèle de classe résultant peut manquer de certains éléments. |
| Créer Diagramme pour chaque Paquetage | Lorsque cette option est sélectionnée, un diagramme de classe est créé, représentant le modèle de classe pour chaque Paquetage . Le résultat est un modèle plus grand mais plus coloré. La désélection de cette option entraînera l'omission de la création diagramme et l' exécuter de l'importation sera plus rapide. |
| Générer Scripts d'Analyseur | Pour les solutions Visual Studio, la sélection de cette option génèrera Scripts d'Analyseur pour chaque configuration de projet en plus des scripts pour chaque configuration de solution. Les scripts permettront de construire et de déboguer le(s) programme(s) décrit(s) par la solution immédiatement après la fin de l'importation. Cochez la case « Windows » ; si vous ne sélectionnez pas cette option, aucune fonctionnalités Analyseur d'Exécution ne sera configurée.<br>Pour les solutions Mono, cette option vous permet de cibler Linux ou Windows . Si vous sélectionnez Linux, il est supposé que la machine sur laquelle s'exécute Enterprise Architect est Linux, que la plate-forme (Java ou Mono) y est installée et que les programmes compilés exécuter sous Linux. |
| Projet de démarrage | Lorsque cette option est sélectionnée, le script de ce projet deviendra le modèle par défaut. Les outils de débogage, le ruban Exécuter et les boutons de la barre d'outils cibleront automatiquement ce programme. |

# Importer le code source

Vous pouvez importer du code source dans votre modèle Enterprise Architect pour procéder à la rétro-ingénierie d'un module. Au fur et à mesure de l'importation, Enterprise Architect fournit des informations sur la progression. Une fois tous les fichiers importés, Enterprise Architect effectue une deuxième passe pour résoudre les associations et les relations d'héritage entre les classes importées.

## Procédure - Importer le code source

| Étape | Action |
|---|---|
| 1 | Dans la fenêtre Navigateur , sélectionnez (ou ajoutez) un diagramme dans lequel importer les classes. |
| 2 | Cliquez sur l'arrière-plan diagramme et soit :<br>• Sélectionnez l'option de ruban « Développer > Code source > Fichiers » et cliquez sur la langue appropriée, ou<br>• Si la barre d'outils de génération de code s'affiche, cliquez sur la flèche déroulante « Importer » et sélectionnez la langue à importer<br>La liste des langues inclura toutes les langues personnalisées pour lesquelles vous avez créé des structures de modèle. |
| 3 | Dans le navigateur de fichiers qui apparaît, recherchez et sélectionnez un ou plusieurs fichiers de code source à importer. |
| 4 | Cliquez sur le bouton Ouvrir pour démarrer le processus d'importation. |

# Notes sur l'importation du code source

Vous pouvez importer du code dans votre projet Enterprise Architect , dans une gamme de langages de programmation. Enterprise Architect supporte la plupart des constructions et des mots-clés pour chaque langage de programmation. Vous sélectionnez le type de fichier source approprié pour le langage, comme code source à importer.

Si vous avez besoin support une fonctionnalité particulière pour laquelle vous estimez qu'elle manque, veuillez contacter Sparx Systems .

## Notes

- Lors de la rétro-ingénierie d'attributs avec des substitutions de paramètres (attributs basés sur des modèles) :
  - Si une classe avec des définitions de paramètres gabarit appropriées est trouvée, un connecteur d'association est créé et ses substitutions de paramètres sont configurées
  - Un connecteur d'association est également créé si une entrée correspondante est définie comme une classe de collection ou
  dans l'option « Classes de collection supplémentaires » (pour C# , C++ et Java) ; pour un exemple, voir *Exemple Utilisation des classes de collection*

## notes sur le langage de programmation

| Langue | Notes |
|---|---|
| ActionScript | Type de fichier source approprié : fichier .as code. |
| C | Type de fichier source approprié : fichiers d'en-tête .h et/ou fichiers .c.<br>Lorsque vous sélectionnez un fichier d'en-tête, Enterprise Architect recherche automatiquement le fichier d'implémentation .c correspondant à importer, en fonction des options d'extension et du chemin de recherche spécifiés dans les options C.<br>Enterprise Architect ne développe pas les macros qui ont été utilisées, celles-ci doivent être ajoutées à la liste interne des macros de langage. |
| C++ | Type de fichier source approprié : fichier d'en-tête .h.<br>Enterprise Architect recherche automatiquement le fichier d'implémentation .cpp en fonction de l'extension et du chemin de recherche définis dans les options C++ ; lorsqu'il trouve le fichier d'implémentation, il peut l'utiliser pour résoudre les noms de paramètres et notes de méthode si nécessaire.<br>Lors de l'importation de code source C++, Enterprise Architect ignore les déclarations de pointeur de fonction.<br>Pour les importer dans votre modèle, vous pouvez créer un typedef pour définir un type de pointeur de fonction, puis déclarer des pointeurs de fonction en utilisant ce type ; les pointeurs de fonction déclarés de cette manière sont importés en tant qu'attributs du type de pointeur de fonction.<br>Enterprise Architect ne développe pas les macros qui ont été utilisées ; celles-ci doivent être ajoutées à la liste interne des macros de langage. |
| C# | Type de fichier source approprié : .cs. |
| Delphes | Type de fichier source approprié : .pas. |

| | |
|---|---|
| Java | Type de fichier source approprié : .java.<br>Enterprise Architect supporte les extensions du langage AspectJ.<br>  <br><br>Les aspects sont modélisés à l'aide de classes avec le stéréotype aspect ; ces aspects peuvent alors contenir des attributs et des méthodes comme pour une classe normale.<br>Si un attribut ou une opération intertype est requis, vous pouvez ajouter une étiquette 'className' dont la valeur est le nom de la classe à laquelle elle appartient.<br>Les points de coupure sont définis comme des opérations avec le stéréotype <<pointcut>> et peuvent se produire dans n'importe quelle classe, interface ou aspect Java ; les détails du point de coupure sont inclus dans le champ « comportement » de la méthode.<br>Un conseil est défini comme une opération avec le stéréotype <<conseil>> ; le point de coupure sur lequel ce conseil opère se trouve dans le champ « comportement » et agit comme une partie de la signature unique de la méthode.<br>afterAdvice peut également avoir l'une des Valeur Étiquetés revenant ou jetant. |
| PHP | Type de fichier source approprié : .php, .php4 ou .inc.<br>La syntaxe de condition imbriquée si est activée. |
| Python | Type de fichier source approprié : .py. |
| Visual Basic | Type de fichier source approprié : fichier de classe .cls. |
| Visual Basic .NET | Type de fichier source approprié : fichier de classe .vb. |

# Script d'importation de ressources

Enterprise Architect supporte l'importation et l'exportation de Scripts de ressources Microsoft Windows (sous forme de fichiers .rc), qui contiennent les définitions dialogue Win32® (celles avec le stéréotype « win32Dialog ») pour l'interface utilisateur graphique d'une application. Les ressources Dialogue sont importées et exportées pour une langue spécifique, en utilisant par défaut les paramètres régionaux du système informatique actuel.

## Accéder

| | |
|---|---|
| Ruban | Développer > Code source > Fichiers > Importer un script de ressources |
| Raccourcis Clavier | F7 (synchroniser l'élément avec le code) |

## Importer des ressources dialogue à partir d'un fichier .rc

| Option | Action |
|---|---|
| Fichier de ressources | Cliquez sur le bouton  et localisez le fichier .rc pour importer les éléments de l'écran. |
| ID de la ressource | Soit:<br>• Laissez la valeur par défaut « Tous » pour importer tous les éléments de l'écran à partir du fichier, ou<br>• Cliquez sur la flèche déroulante et sélectionnez l' ID d'écran d'une dialogue spécifique à importer |
| Langue | Cliquez sur la flèche déroulante et sélectionnez la version linguistique (par exemple, Anglais - States Unis) du le dialogue à importer. |
| Importer | Cliquez sur ce bouton pour importer les écrans du fichier de ressources.<br>La progression de l'importation est indiquée dans le champ situé sous le champ « Langue ». |

## Exporter un dialogue vers un fichier .rc

| Option | Action |
|---|---|
| ID d'écran | Valeur par défaut de l' ID Win32UI Valeur Étiquetée de l'élément d'écran sélectionné.<br>(Si le dialogue n'a pas cet ID , ouvrez la page 'Win32UI' de la dialogue ' Propriétés ' de l'élément et fournissez une valeur pour l' étiquette ID .) |
| | |

| | |
|---|---|
| Fichier de ressources | Cliquez sur le bouton ... et recherchez le fichier .rc dans lequel exporter les éléments de l'écran.<br>Si l'élément a été précédemment importé, ce champ correspond par défaut au fichier source. |
| Langue | Cliquez sur la flèche déroulante et sélectionnez la version linguistique (par exemple, Anglais - States Unis) du dialogue exporté. |
| Exporter | Cliquez sur ce bouton pour exporter les écrans du fichier de ressources.<br>La progression de l'exportation est indiquée dans le champ situé sous le champ « Langue ». |

## Notes

- Les nouvelles boîtes de dialogue sont exportées vers un fichier .rc existant
- Lors d'une exportation vers un fichier .rc existant, aucune boîte de dialogue n'est jamais supprimée du fichier, même lorsqu'elles sont supprimées du modèle
- Lors d'une importation, aucune boîte de dialogue n'est supprimée du modèle, même si elle est omise du fichier .rc d'origine

# Importer une structure de répertoire

Vous pouvez importer tous les fichiers sources dans une structure de répertoire complète, ce qui vous permet d'importer ou de synchroniser plusieurs fichiers dans une arborescence de répertoires en un seul passage.

Enterprise Architect crée les Paquetages et diagrammes nécessaires pendant le processus d'importation.

## Accéder

| | |
|---|---|
| Ruban | Développer > Code source > Fichiers > Importer le répertoire source |
| Raccourcis Clavier | Ctrl+Maj+U |

## Importer une structure de répertoire à l'aide de la dialogue « Importer le répertoire source »

| Champ | Action |
|---|---|
| Répertoire racine | Type ou recherchez le nom du répertoire à importer. |
| Type source | Type ou sélectionnez dans la liste déroulante la langue de codage des fichiers à importer dans le répertoire source. |
| Déposer | Type ou sélectionnez dans la liste déroulante les extensions de fichier à inclure dans l'importation. Utilisez un « ; » pour séparer les valeurs. |
| Effectuer un Exécuter à Sec | Si vous souhaitez effectuer l'importation en tant exécuter à sec lorsque vous cliquez sur le bouton OK , cochez cette case. Une fois le traitement terminé, cliquez sur le bouton Vue Log pour vérifier le résultat prévu du processus. |
| Traiter les sous-répertoires de manière récursive | Si vous souhaitez inclure le contenu des sous-répertoires dans le processus d'importation, cochez cette case. |
| Importer des composants depuis | Si vous souhaitez importer des fichiers supplémentaires (comme décrit dans la dialogue « Importer les types de composants »), cochez cette case. Vous complétez ensuite l' prompt pour spécifier la provenance des composants. |
| Ne pas importer de membres privés | Si vous souhaitez exclure les membres privés du modèle lors de l'importation de bibliothèques, cochez cette case. |
| Demande de définitions de macros manquantes | Lors de l'importation, l'analyseur peut rencontrer des macros non reconnues. Si vous cochez cette case, vous serez averti lorsqu'un tel événement se produit et vous aurez la possibilité de définir la macro. Si vous ne cochez pas cette option, la structure Paquetage résultante peut manquer de certains éléments. |
| Structure Paquetage | Sélectionnez le bouton radio approprié pour créer un Paquetage pour chaque répertoire, chaque espace de noms ou chaque fichier ; cela peut être restreint en |

| | |
|---|---|
| | fonction du type de source sélectionné. |
| Créer Diagramme pour chaque Paquetage | Cochez cette case pour créer un diagramme dans chaque Paquetage créé lors de l'import. Cliquez sur le bouton Options pour identifier les fonctionnalités des éléments à inclure dans les diagrammes . |
| Synchronisation | Sélectionnez le bouton radio approprié pour synchroniser les classes existantes ou écraser les classes existantes.<br>Si une classe de modèle est trouvée qui correspond à celle du code :<br>• « Synchroniser » met à jour la classe du modèle pour inclure les détails de celle du code, ce qui préserve les informations non représentées dans le code, telles que l'emplacement des classes dans diagrammes<br>• « Overwrite » supprime la classe du modèle et en génère une nouvelle à partir du code ; les informations supplémentaires ne sont pas conservées.<br>Si l'option « Utiliser les horodatages » est sélectionnée, la représentation avec l'horodatage le plus récent (modèle ou code) aura la priorité. |
| Supprimer les classes non trouvées dans le code | Sélectionnez le bouton radio approprié pour spécifier comment gérer les classes de modèle existantes qui ne sont pas présentes dans le code importé.<br>• « Ne jamais supprimer » conserve toutes les classes existantes dans le modèle.<br>• « Invite à l'action » vous permet de révision les classes individuellement<br>• « Toujours » supprimer » supprime du modèle toute classe qui n'est pas présente dans le code importé. |
| OK | Cliquez sur ce bouton pour démarrer l'importation. |

# Importer un module binaire

Enterprise Architect vous permet de procéder à la rétro-ingénierie de certains types de modules binaires.

## Accéder

| | |
|---|---|
| Ruban | Développer > Code source > Fichiers > Importer un module binaire |

## Utiliser

Actuellement, les types autorisés sont :

- Archive Java (.jar)
- Fichier PE .NET (.exe, .dll) - Les fichiers DLL et EXE natifs Windows ne sont pas pris en charge, seuls les fichiers PE contenant des données d'assemblage .NET
- Fichier de langue intermédiaire (.il)

Enterprise Architect crée les Paquetages et diagrammes nécessaires pendant le processus d'importation ; la sélection de la case à cocher « Ne pas importer les membres privés » exclut les membres privés des bibliothèques de l'importation dans le modèle.

Lors de l'importation de fichiers .NET , vous pouvez importer via la réflexion ou via le désassemblage, ou laisser le système sélectionner la meilleure méthode - cela peut entraîner l'utilisation des deux types.

L'importateur basé sur la réflexion s'appuie sur un programme .NET et nécessite l'installation de l'environnement d'exécution .NET .

L'importateur basé sur le désassembleur s'appuie sur un programme Windows natif appelé Ildasm.exe, qui est un outil fourni avec le SDK MS .NET ; le SDK peut être téléchargé à partir du site Web de Microsoft.

Un choix de méthodes d'importation est disponible car certains fichiers ne sont pas compatibles avec la réflexion (comme mscorlib.dll) et ne peuvent être ouverts qu'à l'aide du désassembleur ; cependant, l'importateur basé sur la réflexion est généralement beaucoup plus rapide.

Vous pouvez également configurer :

- S'il faut synchroniser ou écraser les classes existantes lorsqu'elles sont trouvées ; si une classe de modèle est trouvée correspondant à celle du fichier :
  - Synchroniser les mises à jour de la classe modèle pour inclure les détails de celle du fichier, qui conserve les informations non représentées dans le fichier, telles que l'emplacement des classes dans diagrammes
  - Overwrite supprime la classe du modèle et en génère une nouvelle à partir du fichier, ce qui supprime et ne remplace pas les informations complémentaires
- Faut-il créer un diagramme pour chaque Paquetage
- Ce qui est affiché sur diagrammes créés par l'importation

# Classes non trouvées lors de l'importation

Lors de la rétro-ingénierie de votre code, il peut arriver que des classes soient délibérément supprimées de votre code source.

La fonctionnalité « Importer le répertoire source » garde une trace des classes avec lesquelles elle s'attend à se synchroniser et, dans la dialogue « Importer la structure du répertoire », fournit des options sur la façon de gérer les classes qui n'ont pas été trouvées.

Vous pouvez sélectionner l'option appropriée pour Enterprise Architect , à la fin de l'importation, ignore les classes manquantes, les supprime automatiquement ou vous prompt à les gérer.

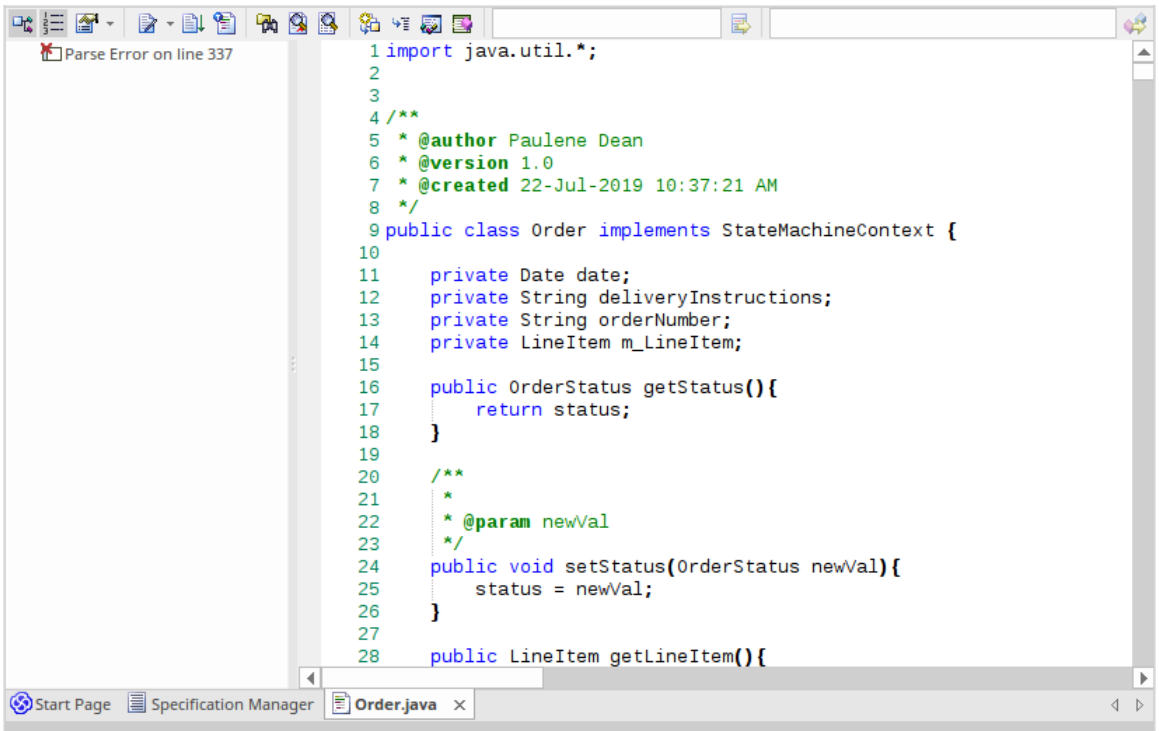
Dans la dialogue « Importer la structure du répertoire », si vous sélectionnez le bouton radio « Demander Action » pour révision manuellement les classes manquantes, une dialogue s'affiche dans laquelle vous spécifiez la gestion de chaque classe manquante dans le code importé.

Par défaut, toutes les classes sont marquées pour suppression ; pour conserver une ou plusieurs classes, sélectionnez-les et cliquez sur le bouton Ignorer.

# Modification du code source

Enterprise Architect contient un éditeur de code source riche en fonctionnalités qui vous permet de visualiser, de modifier et de gérer votre code source directement dans l'outil. Une fois le code source généré pour une ou plusieurs classes, il peut être visualisé dans cet environnement d'édition flexible. Le fait de voir le code dans le contexte des modèles UML dont il est dérivé apporte de la clarté au code et aux modèles, et comble le fossé entre la conception et la mise en œuvre qui a historiquement introduit des erreurs dans les systèmes logiciels.

L' Éditeur de Code source est complet, avec une arborescence de structure pour une navigation facile des attributs, propriétés et méthodes. Les numéros de ligne peuvent être affichés et les options de surbrillance de la syntaxe peuvent être configurées. De nombreuses fonctionnalités que les ingénieurs logiciels connaissent dans leur IDE préféré, telles qu'Intelli-sense et la saisie semi-automatique du code, sont incluses dans l'éditeur. Il existe de nombreuses fonctionnalités supplémentaires, telles que l'enregistrement de macros qui facilite la gestion du code source dans Enterprise Architect . Il existe également de nombreuses options de gestion du code, disponibles via le menu contextuel de l'éditeur de code, la barre d'outils et les touches de fonction.




Pour la plupart des langages de programmation, un seul fichier est créé à partir d'une classe UML , mais dans le cas de C++, les classes d'en-tête et d'implémentation sont créées et l'éditeur de code source affiche ces fichiers dans des onglets séparés.

Un certain nombre d'options modifient le fonctionnement de l'éditeur de code source ; elles peuvent être modifiées à l'aide de la dialogue « Préférences » disponible dans le ruban Démarrer :

' Démarrer > Apparence > Préférences > Préférences > Ingénierie du code source > Éditeurs de Code '

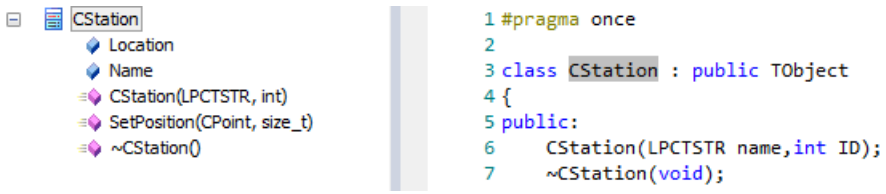
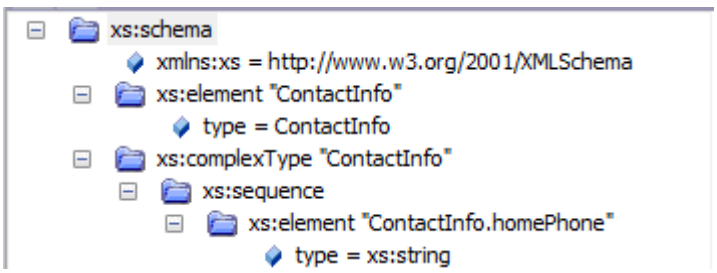
Il existe des variantes de l' Éditeur de Code source, avec différentes méthodes d'accès. Les variantes sont présentées dans la rubrique *Comparer les éditeurs* .

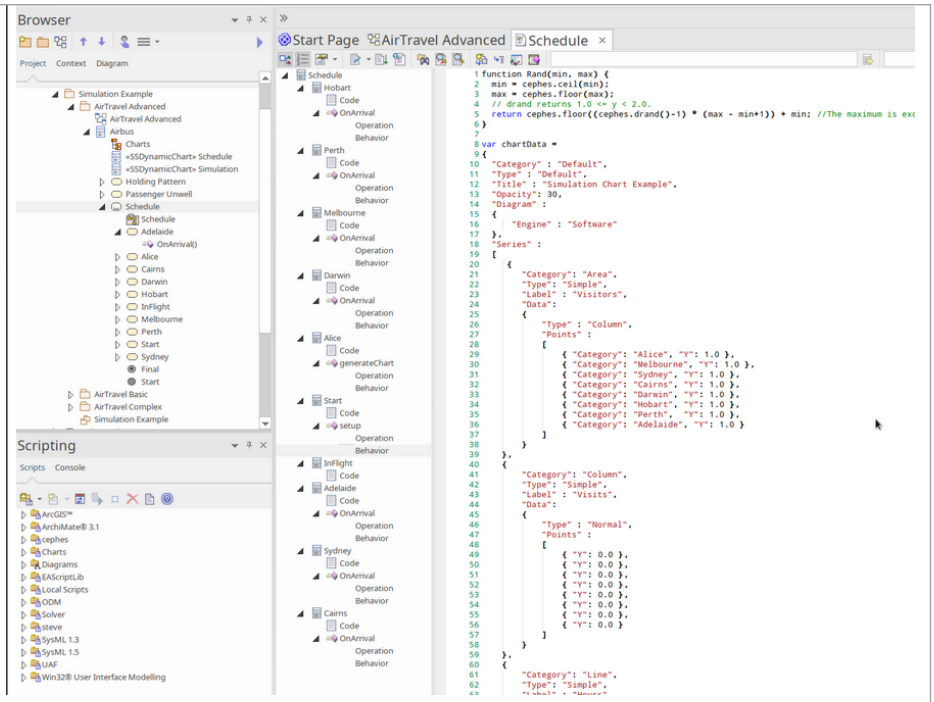
## Accéder

| | |
|---|---|
| Ruban | Exécuter > Source > Modifier > Ouvrir le fichier source (fichier externe) ou<br>Exécuter > Source > Modifier > Modifier la source de l'élément (pour un fichier source existant) ou |

| | |
|---|---|
| | Exécuter > Source > Modifier > Modifier le nouveau fichier source ou<br>Conception > Élément > Comportement ou<br>Développer > Code source > Comportement |
| Raccourcis Clavier | F12 ou Ctrl+E (pour le code existant pour les éléments du modèle)<br>Ctrl+Alt+O (pour localiser les fichiers externes) |

## Facilités

| Facilité | Description |
|---|---|
| Éditeur de code source | Par défaut, l'éditeur de code source est défini sur :<br>• Analyser tous les fichiers ouverts et afficher une arborescence des résultats<br>• Afficher les numéros de ligne<br>  <br><br>Si vous modifiez un fichier XML, l'arborescence de la structure reflète l'ordre et la structure exacts du document.<br>  <br> |
| Structure arborescente | L'arborescence de la structure des fichiers est disponible pour les fichiers de langues prises en charge, tels que C++, C# , Java et XML. L'arborescence peut être utile pour parcourir rapidement le contenu, de la même manière qu'un tableau de contenu le ferait pour d'autres documents. |
| Comportements Simulation | Si vous éditez les comportements des éléments d'un diagramme Statemachine ou d'activités, l' Éditeur de Code vous permet de lister et d'éditer les comportements de tous les éléments du diagramme ensemble, en utilisant une arborescence de structure. |




Dans cette illustration, vous pouvez voir un certain nombre d' States au sein d'une Statemachine , chacun d'eux ayant des opérations et des comportements, et tous répertoriés ensemble et pouvant être sélectionnés sans quitter ou modifier la fenêtre de l'éditeur.

## Notes

- Pour la plupart des éléments sélectionnés, vous pouvez utiliser les touches F12 ou Ctrl+E pour afficher le code source.
- Lorsque vous sélectionnez un élément pour afficher le code source, si l'élément n'a pas de fichier de génération (c'est-à-dire que le code n'a pas été ou ne peut pas être généré, comme pour un cas d'utilisation), Enterprise Architect vérifie si l'élément a un lien vers une opération ou un attribut d'un autre élément - si un tel lien existe et que cet autre élément a du code source, le code de cet élément s'affiche
- Vous pouvez également localiser le répertoire contenant un fichier source qui a été créé ou importé dans Enterprise Architect et le modifier ou ses fichiers associés à l'aide d'un éditeur externe tel que le Bloc-notes ou Visual Studio ; cliquez sur l'élément dans la fenêtre Navigateur et appuyez sur Ctrl+Alt+Y

# Langues prises en charge

Les Éditeurs de Code Source peuvent afficher du code dans un large éventail de langages, comme indiqué ici. Pour chaque langage, l'éditeur met en évidence - en texte coloré - la syntaxe standard du code.

- Ada (.ada, .ads, .adb)
- ActionScript (.as)
- Document BPEL (.bpel)
- C++ (.h, .hh, .hpp, .c, .cpp, .cxx)
- C# (.cs)
- Langage Query structuré DDL (.sql)
- Delphes/Pascal (.pas)
- Fichiers de différences/ Patch (.diff, .patch)
- Définition Type de document (.dtd)
- Fichiers batch DOS (.bat)
- Scripts de commande DOS (.cmd)
- HTML (.html)
- Langage de définition d'interface (.idl, .odl)
- Java (.java)
- JavaScript (.javascript)
- JScript (.js)
- Grammaire de la forme Backus-Naur modifiée (.mbnf)
- PHP (.php, .php4, .inc)
- Python (.py)
- Langage de balisage généralisé standard (.sgml)
- SystèmeC (.sc)
- Visual Basic 6 (.bas)
- VB.NET (.vb)
- VBScript (.vbs)
- Verilog (.v)
- Langage de description du matériel VHSIC (.vhdl)
- Configuration des ressources de Visual Studio (.rc)
- XML (langage de balisage extensible) (.xml)
- XSD (définition de schéma XML)
- XSL (langage de feuille de style XML)

# Configurer les associations de fichiers

Si vous êtes un utilisateur Windows®, vous pouvez configurer Enterprise Architect comme gestionnaire de documents par défaut pour vos fichiers sources de langue.

## Accéder

| | |
|---|---|
| Ruban | Démarrer > Apparence > Préférences > Préférences > Ingénierie du code source > Éditeurs de Code : Configurer les associations de fichiers Enterprise Architect... |

## Actes

Pour chaque type de fichier que vous souhaitez ouvrir dans Enterprise Architect , cliquez sur la case à cocher à gauche du nom du type de fichier. Après avoir sélectionné tous les types de documents souhaités, cliquez sur le bouton Enregistrer.

Après cela, cliquez sur n'importe quel fichier correspondant dans l'Explorateur Windows® pour l'ouvrir dans Enterprise Architect .

## Notes

- Vous pouvez modifier les programmes par défaut, ou les documents gérés par eux, directement via l'option « Programmes par défaut » dans le Panneau de configuration Windows ®.

# Comparer les éditeurs

Enterprise Architect propose quatre principales variantes d'éditeur de code, disponibles via plusieurs chemins d'accès. Les options d'accès les plus directes sont identifiées dans ces descriptions.

Les trois premières variantes d'éditeur de code répertoriées ont le même format d'affichage, la même barre d'outils d'options, les mêmes options de menu contextuel et les mêmes touches de fonction internes. Elles diffèrent par leur méthode d'accès et leur mécanisme d'affichage.

## Variantes de l'éditeur

| Variante | Détails |
|---|---|
| Code source Vue | F12<br>Ctrl+E<br>Menu contextuel de la classe \| « Code source Vue »<br>Description : Affiche le code sur un onglet du Diagramme Vue ; l'étiquette de l'onglet affiche le nom du fichier et l'extension (comme .java) ; encore une fois, pour C++, il existe deux onglets pour les fichiers d'en-tête et d'implémentation.<br>Vous pouvez afficher le code source d'autres classes sur des onglets supplémentaires, en resélectionnant l'option de menu/les touches de la classe suivante. |
| Fenêtre de code source (ancrable) | Alt+7<br>« Exécuter > Source > Modifier > Ouvrir le fichier source »<br>Description : Affiche le contenu du fichier source d'une classe sélectionnée (sauf si le langage est C++, lorsque la fenêtre affiche un onglet pour le fichier d'en-tête et un onglet pour le fichier d'implémentation).<br>Si vous sélectionnez une classe différente, la fenêtre change pour afficher le code de la nouvelle classe (à moins que la première classe n'appelle la seconde, auquel cas la fenêtre défile jusqu'au code de la seconde classe). |
| Éditeur interne, code source externe | Ctrl+Alt+O<br>Option du ruban « Exécuter > Source > Modifier > Ouvrir le fichier source »<br>Description : utilisez cette option si vous avez l'intention de modifier du code externe, des fichiers XML ou DDL (c'est-à-dire du code non importé ou généré dans Enterprise Architect ).<br>Affiche un navigateur externe, puis ouvre le fichier de code sélectionné spécifique sous forme d'onglet du Diagramme Vue (pour C++, pas deux fichiers de code) ; sinon, cela est identique à l'option F12. |
| Éditeur externe, code source interne ou externe | Ctrl+Alt+Y<br>Menu contextuel de la classe \| Répertoire Open Source<br>Description : affiche un navigateur de fichiers externe, ouvert sur le répertoire contenant les fichiers sources de la classe sélectionnée ; vous pouvez ouvrir les fichiers dans le Bloc-notes, Visual Studio ou d'autres outils que vous pourriez avoir sur votre système. |

# Barre d'outils Éditeur de Code

Lorsque vous réviser le code d'une partie de votre modèle dans l'éditeur de code source, vous pouvez accéder à un large éventail de fonctions d'affichage et d'édition à partir de la barre d'outils de l'éditeur.

## Barre d'outils Éditeur de Code


## Options de la barre d'outils

**Structure arborescente** — Cliquez sur cette icône pour afficher ou masquer le panneau de hiérarchie des éléments (le panneau de gauche de l'éditeur de code source).

**Numéros de ligne** — Cliquez sur cette icône pour afficher ou masquer les numéros de ligne par rapport aux lignes de code.

**Propriétés d'ingénierie du code source** — Cliquez sur la flèche déroulante pour afficher un menu d'options permettant de sélectionner les pages individuelles « Ingénierie du code source » de la boîte de dialogue « Préférences », à partir desquelles vous pouvez configurer les options d'affichage et de comportement pour l'ingénierie du code source :

- Langue
- Options de mise en évidence de la syntaxe
- Éditeur de Code Options
- Options d'ingénierie de code
- Raccourcis clavier Éditeur de Code

**Fonctions de l'éditeur** — Cliquez sur la flèche déroulante pour afficher un menu donnant accès à une gamme de fonctions d'édition de code :

- Ouvrir le fichier correspondant (Ctrl+Maj+O) - ouvre l'en-tête ou le fichier d'implémentation associé au fichier actuellement ouvert
- Accéder à l'accolade correspondante (Ctrl+E) - pour une accolade ouvrante ou fermante sélectionnée, met en surbrillance l'accolade fermante ou ouvrante correspondante dans la paire
- Aller à la ligne (Ctrl+G) - affiche une dialogue dans laquelle vous sélectionnez le numéro de la ligne à mettre en surbrillance ; cliquez sur le bouton OK pour déplacer le curseur sur cette ligne
- Historique du curseur précédent (Ctrl+-) - le visualiseur de code source conserve un historique des 50 positions de curseur précédentes, créant un enregistrement lorsque le curseur est déplacé soit de plus de 10 lignes de sa position précédente, soit lors d'une opération de recherche et de remplacement ; l'option de menu déplace le curseur vers la position dans l'enregistrement d'historique du curseur immédiatement précédent
- Historique du curseur suivant (Ctrl+Maj+-) - si vous êtes passé à une position de curseur antérieure, cette option déplace le curseur vers la position dans l'enregistrement d'historique du curseur immédiatement suivant
- Rechercher (Ctrl+F) - affiche une dialogue dans laquelle vous définissez une string de texte et des options de recherche pour localiser cette string de texte

dans le code

- Remplacer (Ctrl+R) - affiche une dialogue dans laquelle vous définissez une string de texte et des options de recherche pour localiser cette string de texte dans le code et la remplacer par une autre string de texte ; le dialogue propose des options pour localiser et remplacer chaque occurrence selon votre choix, ou pour remplacer toutes les occurrences immédiatement
- Mettre en surbrillance les mots correspondants - (Ctrl+3) Active ou désactive la mise en surbrillance des mots correspondants lors d'une opération de recherche ; par défaut, cette option est activée
- Enregistrer une macro - enregistre vos prochaines frappes de touches pour les sauvegarder sous forme de macro
- Arrête d'Enregistrer et Enregistrer la macro - arrête l'enregistrement des frappes et affiche la dialogue « Enregistrer la macro » dans laquelle vous spécifiez un nom pour la macro
- Lire la macro - affiche la dialogue « Ouvrir une macro » à partir de laquelle vous sélectionnez et exécutez une macro enregistrée, pour répéter les frappes de touches enregistrées
- Basculer le commentaire de ligne (Ctrl+Maj+C) - commente (//) ou rétablit le code pour chaque ligne complète dans laquelle le texte est mis en surbrillance
- Basculer le commentaire de flux (Ctrl+Maj+X) - insère un commentaire de flux (/* */) à la position du curseur (commente uniquement les caractères et les lignes mis en surbrillance) ou rétablit le texte commenté sous forme de code
- Activer/désactiver les caractères d'espacement (Ctrl+Maj+W) - affiche ou masque les caractères d'espacement : --> (espace de tabulation) et . (espace de caractère)
- Basculer les caractères de fin de ligne (Ctrl+Maj+L) - affiche ou masque les caractères de fin de ligne : CR (retour chariot) et LF (saut de ligne)
- Basculer la synchronisation de l'arborescence - sélectionne automatiquement l'élément de l'arborescence lorsque le contexte change dans l'éditeur de code
- Ouvrir le dossier contenant - ouvre le navigateur de fichiers dans le dossier contenant le fichier de code ; vous pouvez ouvrir d'autres fichiers dans votre éditeur externe par défaut à des fins de comparaison et de travail parallèle

**Enregistrer la source et resynchroniser la classe** — Cliquez sur cette icône pour enregistrer le code source et resynchroniser le code et la classe dans le modèle.

**Code Gabarits** — Cliquez sur cette icône pour accéder à l'éditeur Gabarits de code, pour éditer ou créer gabarits de code pour la génération de code.

**Rechercher dans Projet Navigateur** — Pour une ligne de code sélectionnée, cliquez sur cette icône pour mettre en surbrillance la structure correspondante dans la fenêtre Navigateur . S'il existe plusieurs possibilités, la dialogue « Correspondances possibles » s'affiche, répertoriant les occurrences de la structure parmi lesquelles vous pouvez sélectionner celle qui vous intéresse.

**Rechercher dans les fichiers** — Cliquez sur cette icône pour rechercher le nom object sélectionné dans les fichiers associés et afficher les résultats de la recherche dans la fenêtre Recherche de fichiers. Vous pouvez affiner et actualiser la recherche en spécifiant des critères dans la barre d'outils de la fenêtre Rechercher dans les fichiers.

**Recherche dans Modèle** — Cliquez sur cette icône pour rechercher le texte sélectionné dans tout le modèle et afficher les résultats de la recherche dans la vue Rechercher dans Projet .

**Accéder à la déclaration** — Cliquez sur cette icône pour localiser la déclaration d'un symbole dans le code source.

| | |
|---|---|
| **Aller à la définition** | Cliquez sur cette icône pour localiser la définition d'un symbole dans le code source (applicable aux langages tels que C++ et Delphi, où les symboles sont déclarés et définis dans des fichiers séparés). |
| **Liste de saisie semi-automatique** | Cliquez sur cette icône pour afficher la liste d'autocomplétion des valeurs possibles ; double-cliquez sur une valeur pour la sélectionner. |
| **Informations sur les paramètres** | Lorsque le curseur se trouve entre les parenthèses de la liste des paramètres d'une opération, cliquez sur cette icône pour afficher la signature de l'opération, mettant en évidence le paramètre actuel. |
| **Rechercher la classe actuelle dans la fenêtre Navigateur** | Cliquez sur cette icône pour afficher le nom de la classe actuellement sélectionnée dans le code et mettez en surbrillance ce nom dans la fenêtre Navigateur ; s'il existe plusieurs possibilités, la dialogue « Correspondances possibles » s'affiche, répertoriant les occurrences de la classe parmi lesquelles vous pouvez sélectionner celle requise. |
| **Trouver un membre** | Cliquez sur cette icône pour afficher le nom de l'attribut ou de la méthode actuellement sélectionné dans le code, et mettez en surbrillance ce nom dans la fenêtre Navigateur ; s'il existe plusieurs possibilités, la dialogue « Correspondances possibles » s'affiche, répertoriant les occurrences de la fonctionnalité parmi lesquelles vous pouvez sélectionner celle requise. |

## Notes

- L'option « Enregistrer la macro » désactive Intelli-sense pendant l'enregistrement de la macro
- Vous pouvez attribuer des touches pour exécuter la macro, au lieu d'utiliser la liste déroulante de la barre d'outils et dialogue « Ouvrir la macro »

# Éditeur de Code Menu Contexte

Lorsque vous travaillez sur un fichier avec un éditeur de code, vous pouvez effectuer un certain nombre d'opérations de recherche et d'édition de code pour révision le contenu du fichier. Ces options sont disponibles via le menu contextuel de l'éditeur et peuvent varier en fonction de l'éditeur de code que vous utilisez.

## Accéder

| | |
|---|---|
| Menu Contexte | Cliquez-droit sur la string de texte du code sur laquelle vous travaillez |

## Options

**Accéder à la déclaration** — Localisez et mettez en surbrillance la déclaration d'un symbole dans le code source.

**Aller à la définition** — Localisez et mettez en évidence la définition d'un symbole dans le code source (applicable aux langages tels que C++ et Delphi, où les symboles sont déclarés et définis à des endroits séparés).

**Ouvrir dans l'éditeur de grammaire** — Ouvre une vue qui vous permet d'examiner ou de valider le code à l'aide de la grammaire appropriée.

**Synchroniser l'arbre avec l'éditeur** — Recherche et affiche l'élément courant (méthode par exemple) dans l'arborescence de la structure.

**Synchronisation automatique de l'arbre et de l'éditeur** — Une fois sélectionné, l'arborescence de la structure affichera automatiquement l'élément en cours de traitement dans l'éditeur.

**Validation du schéma XML** — Permet de valider un schéma XML.

**Rechercher « < string > »** — Afficher un sous-menu fournissant des options permettant de localiser la string de texte sélectionnée dans une plage d'emplacements.

- ' Rechercher dans Projet Navigateur ' - Mettre en surbrillance l' object contenant le texte sélectionné dans la fenêtre Navigateur
- « Rechercher dans les fichiers ouverts » : recherchez la string de texte sélectionnée dans les fichiers ouverts associés et affichez les résultats de la recherche dans la fenêtre Rechercher dans les fichiers ; vous pouvez affiner et actualiser la recherche en spécifiant des critères dans la barre d'outils de la fenêtre Rechercher dans les fichiers
- « Rechercher dans les fichiers » : recherchez la string de texte sélectionnée dans tous les fichiers associés (fermés ou ouverts) et affichez les résultats de la recherche dans la fenêtre Rechercher dans les fichiers ; vous pouvez affiner et actualiser la recherche en spécifiant des critères dans la fenêtre Rechercher dans Barre d'outils de la fenêtre Fichiers (raccourci clavier : F12)
- 'Rechercher dans Modèle ' - Effectuez une recherche par 'Nom d'élément' dans la recherche Modèle facilité et affichez les résultats dans l'onglet Recherche Modèle

- 'Rechercher dans Scripts ' - (Disponible lorsque vous travaillez dans l' Éditeur de Script ) Ouvrez la fenêtre Rechercher dans les fichiers, définissez le champ 'Chemin de recherche' sur 'Rechercher dans Scripts ' et le champ 'Texte de recherche' sur le texte sélectionné, puis recherchez tous les scripts pour la string de texte et afficher les résultats de la recherche ; vous pouvez affiner et actualiser la recherche en spécifiant des critères dans la barre d'outils de la fenêtre Rechercher dans les fichiers
- 'EA User Guide' - Afficher la description de l'élément de code dans le *Guide d'Utilisateur d'Enterprise Architect*
- 'Google' - Afficher les résultats d'une recherche Google sur le texte
- « MSDN » – Afficher les résultats d'une recherche sur le texte dans le réseau des développeurs Microsoft (MSDN)
- 'Sun Java SE' - Affichage des résultats d'une recherche sur le texte dans la facilité 'Sun Search' de Sun Microsystems
- « Wikipédia » : affiche n'importe quelle entrée sur l' object sur le site Web de Wikipédia
- 'Koders' - Affiche les résultats d'une recherche de la string de texte sur Koders.com

**Recherche Intelli-sense**

Effectuez une recherche sur la string spécifiée à l'aide du service Code Miner ou de la bibliothèque spécifiée dans le script Analyzer actuel. Les résultats s'affichent dans l'onglet « Code Miner » de la fenêtre Rechercher dans les fichiers.

Raccourci clavier : Maj+F12

**Régler Débogueur sur Ligne**

(Si le débogueur est en cours d'exécution et a atteint un point d'arrêt.) Déplacez le point d'exécution vers la ligne en cours. Vérifiez que vous n'ignorez aucun code ou déclaration qui affecte la section suivante du code en cours de débogage.

**Variable d'affichage**

(Si le débogueur est en cours d’exécution.) Ouvrez la fenêtre Variables locales et mettez en surbrillance la variable locale pour le point actuel dans le code.

**Afficher dans la visionneuse String**

Affichez le contenu complet d'une string variable dans le visualiseur String .

**Créer un cas d'utilisation pour '< string >'**

Affichez la dialogue « Créer un cas d'utilisation pour la méthode », à travers laquelle vous créez un cas d'utilisation pour la méthode contenant la string de texte.

**Point d'Arrêt**

Affiche un sous-menu d'options permettant de créer un marqueur d'enregistrement sur la ligne de code sélectionnée. Les marqueurs d'enregistrement que vous pouvez ajouter sont les suivants :

- Point d'Arrêt
- Démarrer Enregistrement Marqueur
- Fin de Enregistrement Marqueur
- Marqueur de capture automatique de pile
- Méthode d'enregistrement automatique Marqueur
- Point de Trace

**Testpoints**

Options d'affichage pour ajouter un nouveau Testpoint , afficher le gestionnaire Testpoints (fenêtre Testpoints ) ou modifier un Testpoint existant si un ou plusieurs sont déjà définis à l'emplacement sélectionné.

(Les sous-options dépendent du type de fichier de code que vous réviser .)

**Validation XML**

Permet de vérifier la conformité d'un document XML avec ses propres références de schéma ou à l'aide d'un schéma spécifié par l'utilisateur ; soit un fichier de

| | |
|---|---|
| | schéma local, soit une URL. |
| **Ouvrir (Fermer) l'IME** | Ouvrez (ou fermez) l'éditeur de méthode de saisie pour pouvoir saisir du texte dans une langue étrangère sélectionnée, comme le japonais. Vous facilité définir la langue du clavier à l'aide du Panneau de configuration Windows - Options régionales et linguistiques. |
| **Copier le lien hypertexte de position** | Copie la position du curseur sous forme de lien hypertexte pouvant être collé dans les éditeurs Rich Notes , comme un message dans l'onglet « Chat » de la fenêtre Chat & Mail. Utilisez simplement l'option de menu contextuel « Coller » dans le message et spécifiez le texte du lien.<br>Le lecteur peut cliquer sur le lien pour ouvrir le fichier source et déplacer le curseur à la position du curseur sélectionnée dans le fichier. |
| **Copier le texte du lien hypertexte** | Copie la string de texte sélectionnée sous forme de lien hypertexte pouvant être collé dans les éditeurs Rich Notes , comme un message dans l'onglet « Chat » de la fenêtre Chat & Mail. Utilisez simplement l'option de menu contextuel « Coller » dans le message.<br>Le lecteur peut cliquer sur le lien pour ouvrir le fichier source et déplacer le curseur sur la première occurrence de cette string de texte dans le fichier. |
| **Numéros de ligne** | ( Éditeur de Script uniquement.) Affichez ou masquez les numéros de ligne de code sur le côté gauche de l'écran de l'éditeur. |
| **Annuler**<br>**Couper**<br>**Copie**<br>**Coller**<br>**Supprimer**<br>**Sélectionner tout** | Ces six options fournissent des fonctions simples pour éditer le code. |

## Notes

- Les options dans la moitié inférieure du sous-menu « Rechercher <string> » (après « Rechercher dans Scripts ») sont configurables ; vous pouvez ajouter de nouveaux outils de recherche ou supprimer des outils existants en modifiant le fichier searchProviders.xml dans le dossier Sparx Systems > EA > Config - ce fichier est au format de document de description OpenSearch

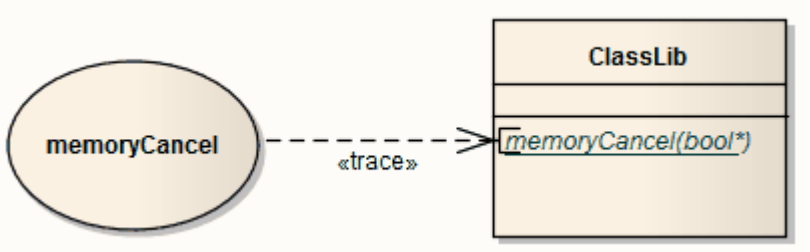
# Créer un cas d'utilisation pour la méthode

À l'aide du menu contextuel de l'éditeur de code, vous pouvez créer un élément de cas d'utilisation pour une méthode que vous sélectionnez dans le code. Vous pouvez également :

- Lier le cas d'utilisation directement à la méthode
- Ajoutez la classe parent à un diagramme (si elle n'est pas déjà dans le diagramme sélectionné) et/ou ajoutez l'élément Cas d'utilisation au diagramme
- Bloc d'affichage de tous les attributs ou méthodes qui ne sont pas également les cibles des liens fonctionnalité

## Créer un cas d'utilisation pour une méthode, via l'éditeur de code

| Étape | Action |
|---|---|
| 1 | (Si vous souhaitez représenter le cas d'utilisation et son lien avec la méthode dans un diagramme ), cliquez sur le nom diagramme dans la fenêtre Navigateur . |
| 2 | Dans l'éditeur de code, cliquez-droit sur le nom de la méthode ou sur n'importe quelle partie du corps de la méthode et sélectionnez l'option « Créer une méthode pour <methodname> ».<br>La dialogue « Créer un cas d'utilisation pour la méthode » s'affiche. |
| 3 | La fonction de base de cette dialogue est de créer un cas d'utilisation pour la méthode sélectionnée :<br>• Si c'est tout ce qui est requis, cliquez sur le bouton OK ; l'élément Use Case est créé dans la fenêtre Navigateur , dans le même Paquetage que la classe parent de la méthode et avec le même nom que la méthode<br>• Si vous souhaitez rendre la relation tangible, continuez la procédure |
| 4 | Pour créer un connecteur Trace reliant le cas d'utilisation à la méthode, cochez la case « Lier le cas d'utilisation à la méthode ». |
| 5 | Pour ajouter la classe parent de la méthode au diagramme , si elle n'est pas déjà là, cochez la case « Ajouter une classe au Diagramme ». |
| 6 | Pour ajouter le cas d'utilisation nouvellement créé au diagramme , cochez la case « Ajouter un cas d'utilisation au Diagramme » ; cela affichera maintenant le cas d'utilisation, la classe et le connecteur Trace sur le diagramme . |
| 7 | Pour afficher uniquement les fonctionnalités (attributs et méthodes) de la classe parent qui sont les cibles des relations « lien vers fonctionnalité », cochez la case « Afficher uniquement fonctionnalités liées dans la classe ».<br>La classe peut contenir n'importe quel nombre d'attributs et de méthodes, mais ceux sans relation « lien vers fonctionnalité » sont masqués. |
| 8 | Cliquez sur le bouton OK pour créer et représenter le cas d'utilisation et la relation ; si vous avez sélectionné toutes les options, le diagramme contient maintenant des éléments liés ressemblant à cette illustration : |


ClassLib
memoryCancel
«trace»
memoryCancel(bool*)

# Éditeur de Code Functions

L' Éditeur de Code commun fournit une variété de fonctions pour aider au processus d'édition de code, notamment :

- Mise en évidence de la syntaxe
- Signets
- Historique du curseur
- Correspondance des accolades
- Indentation automatique
- Commenter les sélections
- Guides de portée
- Zoom
- Sélection de ligne
- Intelli-sens
- Rechercher et remplacer
- Rechercher dans les fichiers

Une gamme de ces fonctions est disponible via des combinaisons de touches du clavier et/ou des options de menu contextuel.

Vous pouvez personnaliser plusieurs fonctionnalités de l' Éditeur de Code en définissant des propriétés dans les fichiers de configuration Éditeur de Code ; par exemple, par défaut, la ligne contenant le curseur est toujours surlignée, mais vous pouvez désactiver la surbrillance.

# Détails de la fonction

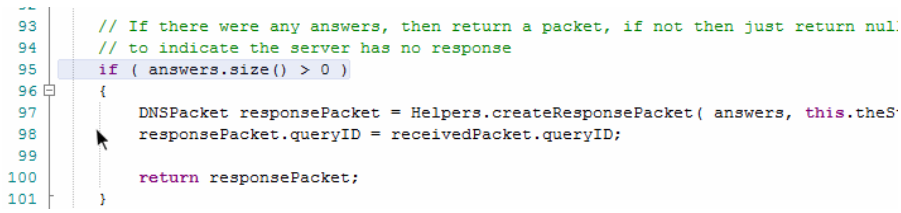
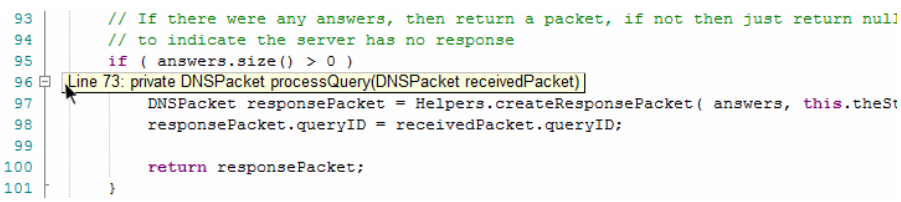
## Éditeur de Code Functions

| Fonction | Description |
|---|---|
| Mise en évidence de la syntaxe | L' Éditeur de Code met en évidence - en texte coloré - la syntaxe de code standard de tous les formats de fichiers de langage pris en charge par Enterprise Architect<br>`#pragma once`<br>`#include "afxwin.h"`<br>`#include "afxcmn.h"`<br><br>`// CToolBox dialog`<br><br>`class CToolBox : public CDialog`<br>`{`<br>`DECLARE_DYNAMIC(CToolBox)`<br>`CRect m_rect;`<br>`int m_offset;`<br>Vous pouvez définir comment l' Éditeur de Code implémente la coloration syntaxique pour chaque langage, via la page ' Éditeurs de Code ' de la dialogue 'Préférences'. |
| Signets | Les signets indiquent une ligne intéressante dans le document ; vous pouvez les activer et les désactiver pour une ligne particulière en appuyant sur Ctrl+F2.<br>De plus, vous pouvez appuyer sur F2 et Maj+F2 pour accéder au signet suivant ou précédent dans le document.<br>Pour effacer tous les signets dans le fichier de code, appuyez sur Ctrl+Maj+F2. |
| Historique du curseur | L' Éditeur de Code Control conserve un historique des 50 positions précédentes du curseur ; une entrée dans la liste d'historique est créée lorsque :<br>• Le curseur est déplacé de plus de 10 lignes par rapport à sa position précédente<br>• Le curseur est déplacé lors d'une opération de recherche/remplacement<br>Vous pouvez accéder à un point antérieur dans l'historique du curseur en appuyant sur Ctrl+-, et à un point ultérieur en appuyant sur Ctrl+Maj+-. |
| Correspondance des accolades | Lorsque vous placez le curseur sur une accolade ou un crochet, l' Éditeur de Code met en surbrillance son partenaire correspondant ; vous pouvez ensuite naviguer jusqu'à l'accolade correspondante en appuyant sur Ctrl+E.<br>`function ProtectedFunctionTest: boolean;`<br>`procedure ProtectedProcedureTest(a: WideString);` |
| Indentation automatique | Pour chaque langue prise en charge, l' Éditeur de Code ajuste l'indentation d'une nouvelle ligne en fonction de la présence d'instructions de contrôle ou de jetons de bloc de portée dans les lignes menant à la position du curseur. |

```
358 {
359     for(size_t t = 0; t < Stations.size(); t++)
360     {
361         if(Stations[t]->Location == loc)
362             return Stations[t];
363     }
364     return NULL;
365 }
```

Les niveaux de retrait sont indiqués par des lignes horizontales pâles.

Vous pouvez également mettre en retrait manuellement les lignes et les blocs de code sélectionnés en appuyant sur la touche Tab ; pour annuler l'indentation du code sélectionné, appuyez sur Maj+Tab.

| | |
|---|---|
| Commenter les sélections | Pour les langages qui support les commentaires, l' Éditeur de Code peut commenter des sélections entières de code.<br>L' Éditeur de Code reconnaît deux types de commentaires :<br>• Commentaire de ligne - des lignes entières sont commentées dès le début (par exemple :<br>// Ceci est un commentaire)<br>• Commentaires de flux - les sections d'une ligne sont commentées à partir d'un point de départ spécifié jusqu'à un point de fin spécifié (par exemple :<br>/* Ceci est un commentaire */)<br>Vous pouvez basculer les commentaires sur la ligne ou la sélection actuelle en appuyant sur :<br>• Ctrl+Maj+C pour les commentaires de ligne, ou<br>• Ctrl+Maj+X pour les commentaires de flux |
| Guides de portée | Si le curseur est placé sur un marqueur d'indentation, l' Éditeur de Code effectue un « retour en arrière » pour trouver la ligne qui a démarré la portée à ce niveau d'indentation ; si la ligne est trouvée et est actuellement à l'écran, elle est surlignée en bleu clair.<br>  <br>Alternativement, si la ligne est hors écran, une info-bulle s'affiche indiquant le numéro de la ligne et son contenu :<br>  |
| Zoom | Vous pouvez zoomer et dézoomer sur le contenu de l' Éditeur de Code en utilisant :<br>• Ctrl+clavier + et<br>• Ctrl+clavier -<br>Le zoom peut être rétabli à 100 % en utilisant Ctrl+clavier /. |

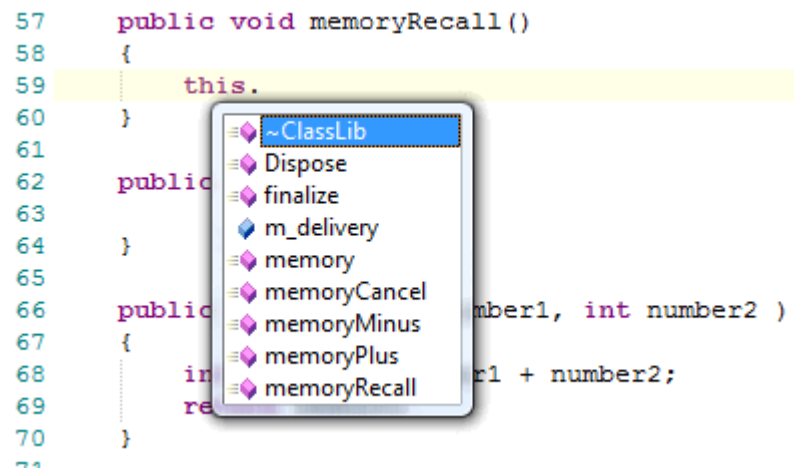
| | |
|---|---|
| Sélection de ligne | Si vous souhaitez déplacer le curseur sur une ligne de code spécifique, appuyez sur Ctrl+G et, en réponse à l' prompt , saisissez le numéro de ligne.<br>Appuyez sur le bouton OK ; l'éditeur affiche la ligne de code spécifiée avec le curseur à gauche. |

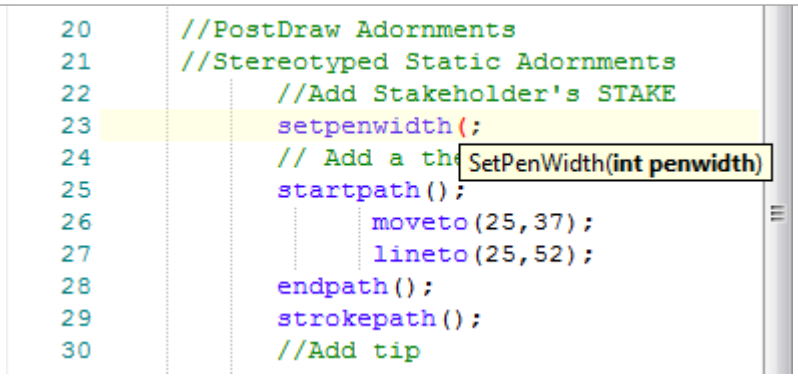
# Intelli-sens

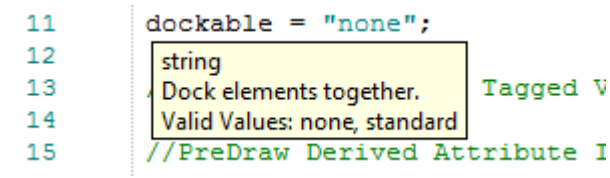
Intelli-sense est une fonctionnalité qui permet de choisir les éléments et les valeurs de code à mesure que vous tapez. Tous les éditeurs de code n'utilisent pas Intelli-sense ; par exemple, Intelli-sense est désactivé lorsque vous enregistrez une macro dans la Visionneuse de code source .

Intelli-sense vous fournit une assistance contextuelle via des listes de saisie semi-automatique, des info-bulles et des informations de survol de la souris.

## Facilités

| Facilité | Description |
|---|---|
| Liste de saisie semi-automatique | Une liste de saisie semi-automatique fournit une liste de saisies semi-automatiques possibles pour le texte actuel ; la liste est automatiquement appelée lorsque vous entrez un jeton d'accès (tel qu'un point ou un pointeur d'accès) après un object ou un type contenant des membres.<br>  <br>Vous pouvez également invoquer la liste de saisie semi-automatique manuellement en appuyant sur Ctrl+Espace ; l' Éditeur de Code recherche alors les correspondances pour le mot menant au point d'invocation.<br>Sélectionnez un élément dans la liste et appuyez sur la touche Entrée ou sur la touche Tab pour insérer l'élément dans le code ; pour fermer la liste de saisie semi-automatique, appuyez sur Échap. |
| Conseils d'appel | Les info-bulles affichent la signature de la méthode actuelle lorsque vous saisissez le jeton de la liste de paramètres (par exemple, en ouvrant une parenthèse) ; si la méthode est surchargée, l'info-bulle affiche des flèches que vous pouvez utiliser pour naviguer dans les différentes signatures de méthode |

<table>
<tr>
<td></td>
<td>


</td>
</tr>
<tr>
<td>Informations sur le survol de la souris</td>
<td>Vous pouvez afficher la documentation de support pour les éléments de code (par exemple, les attributs et les méthodes) en passant le curseur sur l'élément en question.


</td>
</tr>
</table>

# Rechercher et remplacer

Chacun des éditeurs de code d' Enterprise Architect facilite la recherche et le remplacement de termes dans l'éditeur, via la dialogue « Rechercher et remplacer ».

## Accéder

| | |
|---|---|
| Raccourcis Clavier | Mettez en surbrillance la string de texte requise et appuyez sur :<br>• Ctrl+F pour les contrôles de recherche uniquement, ou<br>• Ctrl+R pour les commandes de recherche et de remplacement<br>Dans chaque cas, le champ « Rechercher » est renseigné avec le texte actuellement sélectionné dans l'éditeur. Si aucun texte n'est sélectionné dans l'éditeur, le champ « Rechercher » est renseigné avec le mot se trouvant à la position actuelle du curseur. Si aucun mot n'existe à la position actuelle du curseur, le dernier terme recherché est utilisé. |

## Opérations de base – Commandes

| Commande | Action |
|---|---|
| Trouver la suite | Recherchez et mettez en surbrillance l'instance suivante (par rapport à la position actuelle du curseur) du texte spécifié dans le champ « Rechercher ». |
| Remplacer | Remplacez l'instance actuelle du texte spécifié dans le champ « Rechercher » par le texte spécifié dans le champ « Remplacer par », puis recherchez et mettez en surbrillance l'instance suivante (par rapport à la position actuelle du curseur) du texte spécifié dans le champ « Rechercher ». |
| Remplacer tout | Remplacez automatiquement toutes les instances du texte spécifié dans le champ « Rechercher » par le texte spécifié dans le champ « Remplacer par ». |

## Opérations de base – Options

| Option | Action |
|---|---|
| Étui à allumettes | Spécifiez que la casse de chaque caractère de la string de texte dans le champ « Rechercher » est significative lors de la recherche de correspondances dans le code. |
| Faire correspondre le mot entier | Spécifiez que la string de texte dans le champ « Rechercher » est un mot complet et ne doit pas correspondre à des instances du texte qui font partie d'une string plus longue.<br>Par exemple, les recherches pour ARE ne doivent pas correspondre à ces lettres |

| | |
|---|---|
| | dans les instances des mots AREA ou ARENA. |
| Rechercher vers le haut | Effectuez la recherche à partir de la position actuelle du curseur jusqu'au début du fichier, plutôt que dans la direction par défaut de la position actuelle du curseur jusqu'à la fin du fichier. |
| Utiliser des expressions régulières | Évaluez des séquences de caractères spécifiques dans les champs « Rechercher » et « Remplacer par » en tant qu'expressions régulières. |

## Concepts

| Concept | Description |
|---|---|
| Expressions régulières | Une expression régulière est une définition formelle d'un Motif de recherche, qui peut être utilisée pour faire correspondre des caractères, des mots ou motifs de caractères spécifiques.<br>Par souci de simplicité, le mécanisme « rechercher et remplacer » de Éditeur de Code ne supporte qu'un sous-ensemble de la grammaire standard des expressions régulières.<br>Le texte dans les champs « Rechercher » et « Remplacer par » n'est interprété comme une expression régulière que si la case à cocher « Utiliser les expressions régulières » est sélectionnée dans la boîte dialogue « Rechercher et remplacer ». |
| Métaséquences | Si la case à cocher « Utiliser les expressions régulières » est sélectionnée, la plupart des caractères du champ « Rechercher » sont traités comme des littéraux (c'est-à-dire qu'ils ne correspondent qu'à eux-mêmes).<br>Les exceptions sont appelées métaséquences ; chaque métaséquence reconnue dans la dialogue « Rechercher et remplacer » Éditeur de Code est décrite dans ce tableau :<br>• \< - Indique que le texte est le début d'un mot ; par exemple : \<cat correspond à *catastrophe* et *cataclysm* , mais n'est pas *concaténé*<br>• \> - Indique que le texte est la fin d'un mot ; par exemple : hat\> correspond à *that* et *chat* , mais pas *à hate*<br>• (...) - Indique des caractères uniques alternatifs qui peuvent être mis en correspondance - les caractères peuvent être spécifiques (chr) ou dans une plage alphabétique ou numérique (am) ; par exemple : (hc) at correspond à *hat* et *cat* mais pas *à bat* , et (am) Class correspond à n'importe quel nom dans la plage *aClass-mClass*<br>• (^...) - Indique des caractères uniques alternatifs qui doivent être exclus d'une correspondance - les caractères peuvent être spécifiques (^chr) ou dans une plage alphabétique ou numérique (^am) ; par exemple : (^hc) at correspond à *rat* et *bat* , mais *hat* et *cat* sont exclus, et (^am) Class correspond à n'importe quel nom dans la plage *nClass* à *zClass* , mais *aClass* à *mClass* sont exclus<br>• ^ - Correspond au début d'une ligne<br>• $ - Correspond à la fin d'une ligne<br>• * - Correspond au caractère précédent (ou au jeu de caractères) 0 fois ou plus ; par exemple : ba*t correspond à *bt* , *bat* , *baat* , *baaat* et ainsi de suite, et b(ea)*t correspond à *bt* , *bet* , *bat* , *beat* , *beet* , *baat* et ainsi de suite<br>• + - Correspond au caractère précédent (ou au jeu de caractères) 1 ou plusieurs fois ; par exemple : ba+t correspond à *bat* , *baat* et *baaat* mais pas à *bt* , et |

| | |
|---|---|
| | b(ea) +t correspond à *bet* , *bat* , *beat* , *beet* et *baat* mais pas à *bt*<br>Si une métaséquence à caractère unique est précédée d'une barre oblique inverse (\), elle est traitée comme un caractère littéral : c\(at\) correspond à c(at) car les crochets sont traités littéralement.<br>Lorsque la case à cocher « Utiliser les expressions régulières » est sélectionnée, un menu d'aide à la métaséquence est disponible à droite des champs « Rechercher » et « Remplacer par » ; la sélection d'une métaséquence dans ce menu insère la métaséquence dans le champ, remplaçant ou enveloppant le texte actuellement sélectionné selon le cas. |
| Régions marquées | Lors de la « recherche et du remplacement » avec des expressions régulières, jusqu'à neuf sections du terme d'origine peuvent être substituées dans le terme de remplacement.<br>Les métaséquences '\(' et '\)' indiquent le début et la fin d'une région balisée ; la section du texte correspondant qui se trouve dans la région balisée peut être incluse dans le texte de remplacement avec la métaséquence '\n' (où *n* est le numéro de la région balisée entre 1 et 9).<br>Par exemple:<br>Trouver : *les choses de* \((A-Za-z) +\)<br>Remplacer par *des éléments appartenant à \1*<br>Texte original : *Ce sont toutes les affaires de Michael* .<br>Texte remplacé : *Ce sont tous les éléments qui appartiennent à Michael.* |

# Rechercher dans les fichiers

Les recherches de texte de fichier sont proposées par la fenêtre Rechercher dans les fichiers et depuis les Éditeurs de Code , pour rechercher des fichiers à partir de noms de données et de structures. Ces fichiers peuvent être des fichiers de code externes, des fichiers de code que vous avez déjà ouverts dans Enterprise Architect , des scripts de modèles internes ou le sous-système d'aide.

L'onglet « Recherche de fichiers » conserve un historique des chemins de fichiers que vous avez explorés, vous aidant à revenir rapidement aux dossiers fréquemment utilisés dans votre système de fichiers. Vous pouvez également sélectionner une string de recherche précédemment utilisée, si vous devez répéter une recherche plusieurs fois. Lorsque vous recherchez des fichiers de code, vous pouvez également limiter la recherche à des fichiers de types spécifiques, en sélectionnant les extensions de fichier, et en incluant uniquement le dossier sélectionné ou tous ses sous-dossiers. Une autre facilité utile consiste à pouvoir choisir d'afficher les résultats de la recherche sous la forme d'une liste de toutes les instances de la string , ou d'une liste de fichiers contenant la string avec les instances regroupées sous le fichier dans lequel elles se trouvent.

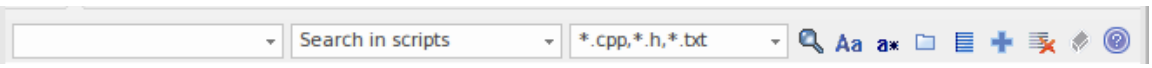
Pour toutes les recherches, vous pouvez préciser que la recherche doit être sensible à la casse et/ou faire correspondre la string de recherche à des mots complets.

## Accéder

| | |
|---|---|
| Ruban | Explorer > Rechercher > Fichiers<br>Exécuter > Source > Rechercher<br>Exécuter > Source > Modifier > Rechercher dans les fichiers |
| Menu Contexte | Cliquez-droit sur le texte sélectionné \| Rechercher <texte sélectionné> \| Rechercher dans les fichiers |
| Raccourcis Clavier | F12, Ctrl+Maj+Alt+F |

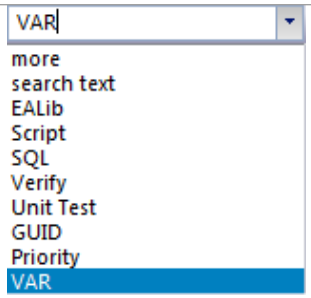
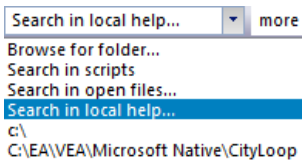
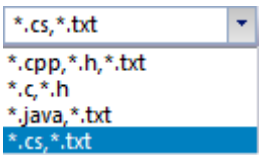
## Barre d'outils de recherche

Vous pouvez utiliser les options de la barre d'outils dans la fenêtre Rechercher dans les fichiers pour contrôler l'opération de recherche. L'état de chaque bouton persiste dans le temps pour toujours refléter vos critères de recherche précédents.




## Options

| Option | Action |
|---|---|
| | Le champ « Texte à rechercher ». Type la string de texte à rechercher.<br>Tout texte que vous saisissez est automatiquement enregistré dans la liste déroulante, jusqu'à un maximum de dix chaînes ; le texte ajouté après cela remplace la string de texte la plus ancienne de la liste. Vous pouvez cliquer sur la flèche déroulante et sélectionner l'une de ces chaînes de texte enregistrées, si vous préférez. |

| | |
|---|---|
|  | |
|  | Le champ « Chemin de recherche ». Spécifiez le dossier dans lequel effectuer la recherche ou le type de recherche.<br>Vous pouvez saisir le chemin du dossier à rechercher directement dans la zone de texte, ou cliquer sur la flèche déroulante et sélectionner « Parcourir le dossier » pour effectuer une recherche à l'aide de la dialogue « Parcourir le dossier ».<br>Tous les chemins que vous saisissez sont automatiquement enregistrés dans la liste déroulante, jusqu'à un maximum de dix chemins ; les chemins ajoutés après cela remplacent le chemin le plus ancien de la liste. Vous pouvez sélectionner l'un de ces chemins enregistrés si vous préférez.<br>Outre « Rechercher un dossier », il existe trois autres options fixes dans la liste déroulante :<br>• « Rechercher dans les scripts », qui recherche les scripts locaux et définis par l'utilisateur dans la fenêtre Scriptant<br>• « Rechercher dans les fichiers ouverts », qui limite la recherche aux fichiers que vous avez ouverts dans Enterprise Architect<br>• « Rechercher dans l'aide locale », qui recherche les fichiers d'aide locaux qui ont été installés à partir du site Web Sparx Systems ; les résultats répertorient les rubriques d'aide contenant le terme de recherche, ainsi que le numéro de ligne et la ligne dans laquelle le texte apparaît<br>Ces options désactivent la liste déroulante « Rechercher les types de fichiers ». |
|  | Le champ « Rechercher les types de fichiers ». Cliquez sur la flèche déroulante et sélectionnez les types de fichiers (extensions de fichiers) à rechercher. |
|  | Cliquez sur cette icône pour lancer la recherche.<br>Pendant la recherche, tous les autres boutons de la barre d'outils sont désactivés. Vous pouvez annuler la recherche à tout moment en cliquant à nouveau sur le bouton Rechercher.<br>Si vous changez l'un de ces boutons à bascule, vous devez exécuter à nouveau la recherche pour modifier la sortie. |
|  | Cliquez sur cette icône pour activer ou désactiver la sensibilité à la casse de la recherche. Le message d'info-bulle identifie le paramètre actuel. |
|  | Cliquez sur cette icône pour basculer entre la recherche de n'importe quelle correspondance et la recherche des seules correspondances qui forment un mot entier. Le message d'info-bulle identifie le paramètre actuel. |
|  | Cliquez sur cette icône pour basculer entre la limitation de la recherche à un seul chemin et l'inclusion de tous les sous-dossiers sous ce chemin. Le message d'info-bulle identifie le paramètre actuel. |
| | |

| | |
|---|---|
|  | Cliquez sur cette icône pour sélectionner le format de présentation des résultats de la recherche ; vous avez deux options :<br>• Liste Vue - (comme indiqué) chaque ligne de résultat se compose du chemin d'accès au fichier et du numéro de ligne, suivis du texte de la ligne ; plusieurs lignes d'un fichier sont répertoriées sous forme d'entrées distinctes<br>• Vue arborescente - (  ) chaque ligne de résultat se compose du chemin d'accès au fichier qui correspond aux critères de recherche et du nombre de lignes correspondant au texte de recherche dans ce fichier ; vous pouvez développer l'entrée pour afficher le numéro de ligne et le texte de chaque ligne |
|  | Cliquez sur cette icône pour ajouter un nouvel onglet de recherche. Vous pouvez créer jusqu'à quatre nouveaux onglets de recherche. Les recherches peuvent également exécuter simultanément. |
|  | Cliquez sur cette icône pour effacer les résultats. |
|  | Si nécessaire, cliquez sur cette icône pour supprimer toutes les entrées dans les listes déroulantes Chemin de recherche, Texte de recherche et Types de fichiers de recherche. |

# Rechercher un fichier

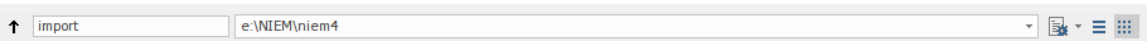
L'onglet « Rechercher un fichier » de la fenêtre Rechercher dans les fichiers fournit un outil qui peut vous aider à trouver des fichiers plus rapidement. L'onglet agit comme un explorateur de système de fichiers et offre une alternative rapide à la dialogue d'ouverture de fichier habituelle. Les recherches de fichiers sont rapides et simples, vous permettant de rechercher les fichiers qui vous intéressent sans perdre votre flux de travail actuel. L'affichage peut être basculé entre la vue rapport et la vue liste.

## Accéder

| | |
|---|---|
| Ruban | Explorer > Rechercher > Fichiers > Rechercher un fichier |
| Raccourcis Clavier | Ctrl+Maj+Alt+F |

## Barre d'outils

La barre d'outils propose un filtre de recherche et une zone de liste déroulante de navigation dans les dossiers. La barre d'outils propose des options permettant de mémoriser les emplacements de recherche et d'alterner entre les vues de liste et de rapport.




## Options

| | |
|---|---|
| | Cliquez pour accéder au dossier parent. |
| | Le contrôle de filtrage vous permet d'exclure les fichiers qui ne correspondent pas aux critères que vous saisissez. Le symbole générique * est automatiquement ajouté au texte, il n'est donc pas nécessaire de l'ajouter vous-même. Pour rechercher tous les fichiers qui contiennent le terme « jvm », saisissez simplement « jvm ». Pour trouver des images .png contenant le terme « red », vous pouvez saisir *red*.png. Appuyez sur la touche Entrée pour mettre à jour les résultats. |
| | Entrez le chemin d'un répertoire et appuyez sur la touche Entrée pour afficher les fichiers à cet emplacement<br>Utilisez la liste déroulante pour sélectionner les emplacements marqués d'un signet pour le modèle actuel. Les emplacements peuvent être gérés à l'aide du menu de la barre d'outils. |
| | Permet de gérer les emplacements affichés dans le combo de répertoires.<br>• Mémoriser le chemin - stocke la valeur actuelle du champ « Répertoire » de sorte que, lorsque vous revenez ultérieurement à la fenêtre Rechercher dans les fichiers, le champ « Répertoire » utilise par défaut cette valeur (s'il s'agit de la seule valeur « mémorisée ») ou propose la valeur dans la liste déroulante |

| | |
|---|---|
| | • Oublier le chemin - efface la valeur actuelle de la mémoire afin qu'elle ne soit pas proposée comme valeur possible pour le champ « Répertoire »<br>• Mémoriser le filtre - stocke la valeur actuelle dans le champ « Filtre » de sorte que lorsque vous revenez ultérieurement à la fenêtre Rechercher dans les fichiers, le champ « Filtre » utilise par défaut cette valeur<br>• Oublier le filtre - supprime la valeur du champ « Filtre » de la mémoire afin qu'elle ne soit pas placée dans le champ la prochaine fois que vous accédez à la fenêtre |
| | Dans cette vue, la liste affiche les colonnes « Nom », « Date de modification », « Type » et « Taille ».<br>Les colonnes peuvent être triées par ordre croissant ou décroissant. Cliquez une troisième fois sur la colonne pour supprimer l'ordre de tri. |
| | La vue de liste supprime les colonnes et est pratique lorsqu'un dossier contient de nombreux fichiers. |

## Raccourcis Clavier

| | |
|---|---|
| | Ensembles se concentrent sur le contrôle du filtre. |
| | Accède au dossier parent. |
| | Accède au dossier parent. |
| | Si un dossier est sélectionné, ouvre le dossier, sinon ouvre les fichiers sélectionnés. |

# Recherche Intelli-sense

Les fonctionnalités Intelli-sense d' Enterprise Architect sont développées à l'aide de l'outil Code Miner de Sparx Systems . Code Miner offre un accès rapide et complet aux informations d'une base de code existante. Le système offre un accès complet à tous les aspects du code source d'origine, soit « à la volée », comme on pourrait le faire avec un éditeur de code, soit sous forme de résultats de recherche produits par des requêtes écrites dans le langage mFQL Code Miner .

## Accéder

Dans la fenêtre Rechercher dans les fichiers, cliquez sur l'onglet « Code Miner ».

| | |
|---|---|
| Ruban | Explorer > Rechercher > Fichiers |
| Raccourcis Clavier | Ctrl+Maj+Alt+F |

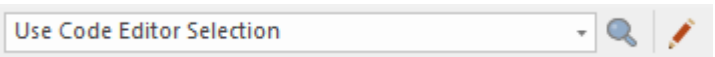
## Le contrôle Code Miner

Ce contrôle présente une interface permettant d'effectuer des requêtes sur plusieurs bases de code à la fois. Les bases de code qu'il utilise sont des bases de données construites à l'aide de l'outil Code Miner d' Enterprise Architect . Ces bases de données forment une bibliothèque, qui peut également être partagée lorsqu'elle est déployée en tant que service. Les requêtes pouvant être exécuter sont répertoriées et sélectionnées à l'aide de la barre d'outils, ce qui permet un accès facile au code source des requêtes, pour l'édition et la composition. Les requêtes n'ont pas besoin d'être compilées ; elles sont visualisées, modifiées et enregistrées comme n'importe quel fichier de code source. Les requêtes qui prennent un seul paramètre peuvent utiliser n'importe quelle sélection dans un éditeur de code ouvert. L'interface supporte la saisie manuelle des paramètres pour les requêtes qui prennent plusieurs arguments.

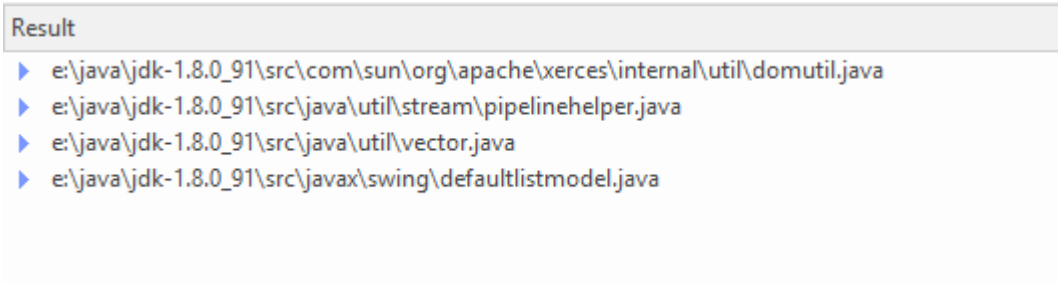
Le premier contrôle de la barre d'outils répertorie les espaces de noms disponibles. La sélection d'un espace de noms limite les requêtes affichées à celles qui se trouvent dans cet espace de noms.


Le contrôle suivant fournit une liste déroulante de toutes les requêtes dans le fichier de requête pour l'espace de noms sélectionné.


Le troisième contrôle est une zone de liste déroulante d'édition. Par défaut, un seul paramètre de requête est extrait du texte sélectionné dans un éditeur de code ouvert, mais vous pouvez également saisir le(s) paramètre(s) directement dans ce champ. Les paramètres multiples doivent être séparés par des virgules. Ceci est suivi du bouton Rechercher pour exécuter la requête. Les requêtes peuvent être modifiées à tout moment à l'aide du bouton Modifier à côté du bouton Rechercher.




Le panneau « Résultat » est un contrôle arborescent qui répertorie les résultats de la requête regroupés par fichier.




## Bibliothèques Code Miner

Les bibliothèques Code Miner sont un ensemble de bases de données qui peuvent être utilisées par les fournisseurs Enterprise Architect Intelli-sense pour obtenir et interroger des informations sur plusieurs bases de code. Chaque base de données est créée à partir du répertoire de code source racine d'une base de code, à l'aide d'une grammaire spécialisée adaptée à son langage (C++, Java ou C# ).

Les bibliothèques sont créées, mises à jour, supprimées ou ajoutées dans l' Éditeur de Script Analyseur . Un scénario typique d'utilisation de cette fonctionnalité serait de créer une base de données pour un projet de développement et des bases de données supplémentaires pour les frameworks référencés par le projet. Votre base de données de développement peut être mise à jour fréquemment au fur et à mesure que les modifications de code s'accumulent, tandis que les frameworks statiques seront mis à jour moins souvent. Les bibliothèques peuvent être recherchées de manière similaire à l'outil « Recherche de fichiers », mais Code Miner offre des capacités de recherche avancées grâce à son langage mFQL.

- Plusieurs domaines/cadres peuvent être recherchés à la fois
- Une requête peut être exécuter en une fraction du temps requis pour une recherche de fichier
- Les requêtes peuvent être codées pour faciliter la recherche de critères complexes
- Les requêtes peuvent prendre plusieurs paramètres
- Tous les fichiers sont indexés sur la base de constructions UML équivalentes, permettant des recherches intelligentes produisant des résultats significatifs dans un cadre modélisation

## Fichiers Query Code Miner

Les requêtes Code Miner sont conservées dans un fichier de code source unique qui doit avoir l'extension .mFQL. Un ensemble de requêtes de base est fourni avec chaque installation Enterprise Architect ; celles-ci peuvent être situées dans le sous-répertoire config\codeminer. Ce fichier de requête doit être nommé par défaut dans tout script Analyzer que vous modifiez.

Avant de modifier une requête, il est conseillé de copier ce fichier dans un emplacement de travail et de nommer la copie dans tout script Analyzer que vous utilisez. De cette façon, vous disposerez toujours d'un fichier de référence auquel vous référer.

Les requêtes sont mieux considérées comme des fonctions écrites dans le langage mFQL. En tant que telles, elles ont des noms uniques, peuvent être qualifiées par un espace de noms unique et peuvent spécifier des paramètres. Le fichier fournit les requêtes répertoriées dans la barre d'outils du contrôle Intelli-sense. Chaque fois que des modifications apportées à un fichier de requête sont enregistrées, les requêtes répertoriées dans la zone de liste déroulante de la barre d'outils de recherche sont mises à jour en conséquence. Cette image est un exemple de requête simple écrite en mFQL.

```
188
189 namespace java
190 {
191 //
192 // Find all references
193 //
194 query::findByName($param1)
195 {
196     distinct(GetByValue( $param1 +))
197 }
198
199 query::findMethodByName($name)
200 {
201     move( 1, "METHOD", intersect( GetByNode("NAME"), GetByValue( $name ) )  )
202 }
203
204 query::findMethodCall($name)
205 {
206     filter( "METHOD_ACCESS", intersect(GetByNode("NAME"), GetByValue( $name )) )
207 }
208
```

# Raccourcis clavier Éditeur de Code

## Clés

| Clé | Description |
|---|---|
| Ctrl+G | Déplacer le curseur vers une ligne spécifiée |
| ↓ | Déplacer le curseur d'une ligne vers le bas |
| Maj+↓ | Étendre la sélection vers le bas d'une ligne |
| Ctrl+↓ | Faites défiler une ligne vers le bas |
| Alt+Maj+↓ | Étendre la sélection rectangulaire d'une ligne vers le bas |
| ↑ | Déplacer le curseur d'une ligne vers le haut |
| Maj+↑ | Étendre la sélection d'une ligne vers le haut |
| Ctrl+↑ | Faire défiler une ligne vers le haut |
| Alt+Maj+↑ | Étendre la sélection rectangulaire d'une ligne vers le haut |
| Ctrl+( | Déplacer le curseur d'un paragraphe vers le haut |
| Ctrl+Maj+( | Étendre la sélection jusqu'à un paragraphe |
| Ctrl+) | Déplacer le curseur d'un paragraphe vers le bas |
| Ctrl+Maj+) | Étendre la sélection d'un paragraphe vers le bas |
| ← | Déplacer le curseur d'un caractère vers la gauche |
| Maj+← | Étendre la sélection d'un caractère vers la gauche |
| Ctrl+← | Déplacer le curseur d'un mot vers la gauche |
| Ctrl+Maj+← | Étendre la sélection d'un mot vers la gauche |
| Alt+Maj+← | Étendre la sélection rectangulaire d'un caractère vers la gauche |
| → | Déplacer le curseur d'un caractère vers la droite. |
| Maj+→ | Étendre la sélection d'un caractère vers la droite |
| Ctrl+→ | Déplacer le curseur d'un mot vers la droite |
| Ctrl+Maj+→ | Étendre la sélection d'un mot à droite |

| | |
|---|---|
| Alt+Maj+→ | Étendre la sélection rectangulaire d'un caractère vers la droite |
| Ctrl+/ | Déplacer le curseur d'une partie du mot vers la gauche |
| Ctrl+Maj+/ | Étendre la sélection à gauche d'une partie du mot |
| Ctrl+\ | Déplacer le curseur vers la droite d'une partie du mot |
| Ctrl+Maj+\ | Étendre la sélection à droite d'une partie du mot |
| Maison | Déplacer le curseur au début de la ligne actuelle |
| Maj+Accueil | Étendre la sélection jusqu'au début de la ligne actuelle |
| Ctrl+Accueil | Déplacer le curseur au début du document |
| Ctrl+Maj+Accueil | Étendre la sélection jusqu'au début du document |
| Alt+Accueil | Déplacer le curseur au début absolu de la ligne |
| Alt+Maj+Accueil | Étendre la sélection rectangulaire jusqu'au début de la ligne |
| Fin | Déplacer le curseur à la fin de la ligne actuelle |
| Maj+Fin | Étendre la sélection jusqu'à la fin de la ligne actuelle |
| Ctrl+Fin | Déplacer le curseur à la fin du document |
| Ctrl+Maj+Fin | Étendre la sélection jusqu'à la fin du document |
| Alt+Fin | Déplacer le curseur jusqu'à la fin absolue de la ligne |
| Alt+Maj+Fin | Étendre la sélection rectangulaire jusqu'à la fin de la ligne |
| Page précédente | Déplacer le curseur vers le haut d'une page |
| Maj+Page précédente | Étendre la sélection jusqu'à une page |
| Alt+Maj+Page précédente | Étendre la sélection rectangulaire sur une page |
| Page suivante | Déplacer le curseur vers le bas d'une page |
| Maj+Page suivante | Étendre la sélection vers le bas d'une page |
| Alt+Maj+Page suivante | Étendre la sélection rectangulaire sur une page |
| Supprimer | Supprimer le caractère à droite du curseur |
| Maj+Suppr | Sélection de coupe |
| | |

| | |
|---|---|
| Ctrl+Suppr | Supprimer le mot à droite du curseur |
| Ctrl+Maj+Suppr | Supprimer jusqu'à la fin de la ligne |
| Insérer | Activer/désactiver la frappe |
| Maj+Insertion | Coller |
| Ctrl+Insertion | Sélection de copie |
| Retour arrière | Supprimer le caractère à gauche du curseur |
| Maj+Retour arrière | Supprimer le caractère à gauche du curseur |
| Ctrl+Retour arrière | Supprimer le mot à gauche du curseur |
| Ctrl+Maj+Retour arrière | Supprimer du début de la ligne jusqu'au curseur |
| Alt+Retour arrière | Annuler supprimer |
| Languette | Indenter le curseur d'une tabulation |
| Ctrl+Maj+I | Indenter le curseur d'une tabulation |
| Maj+Tab | Annuler l'indentation du curseur d'une tabulation |
| Ctrl+clavier (+) | Agrandir |
| Ctrl+clavier(-) | Zoom arrière |
| Ctrl+clavier (/) | Restaurer le zoom |
| Ctrl+Z | Annuler |
| Ctrl+Y | Refaire |
| Ctrl+X | Sélection de coupe |
| Ctrl+C | Sélection de copie |
| Ctrl+V | Coller |
| Ctrl+L | Ligne de coupe |
| Ctrl+T | Transposer la ligne |
| Ctrl+Maj+T | Copier la ligne |
| Ctrl+A | Sélectionner le document entier |
| Ctrl+D | Sélection en double |

| | |
|---|---|
| Ctrl+U | Convertir la sélection en minuscules |
| Ctrl+Maj+U | Convertir la sélection en majuscules |
| Ctrl+E | Déplacer le curseur vers l'accolade correspondante |
| Ctrl+Maj+E | Étendre la sélection à l'accolade correspondante |
| Ctrl+Maj+C | Basculer la ligne de commentaire sur la sélection |
| Ctrl+Maj+X | Basculer le commentaire du flux sur la sélection. |
| Ctrl+F2 | Activer/désactiver le signet |
| F2 | Aller au signet suivant |
| Maj+F2 | Aller au signet précédent |
| Ctrl+Maj+F2 | Effacer tous les signets du fichier actuel |
| Ctrl+Maj+W | Activer/désactiver les caractères d'espacement |
| Ctrl+Maj+L | Basculer les caractères EOL |
| Ctrl+Espace | Invoquer la saisie semi-automatique. |
| Ctrl+- | Revenir en arrière dans l'historique du curseur |
| Ctrl+Maj+- | Avancer dans l'historique du curseur |
| F12 | Démarrer /Annuler la recherche de mot-clé dans le(s) fichier(s). |
| Ctrl+F | Rechercher du texte |
| Ctrl+R | Remplacer le texte |

## Notes

- En plus de ces touches, vous pouvez attribuer des combinaisons de touches (Ctrl+Alt+<n>) à des macros que vous définissez dans la Source Éditeur de Code

# Motifs d'application ( Modèle + Code)

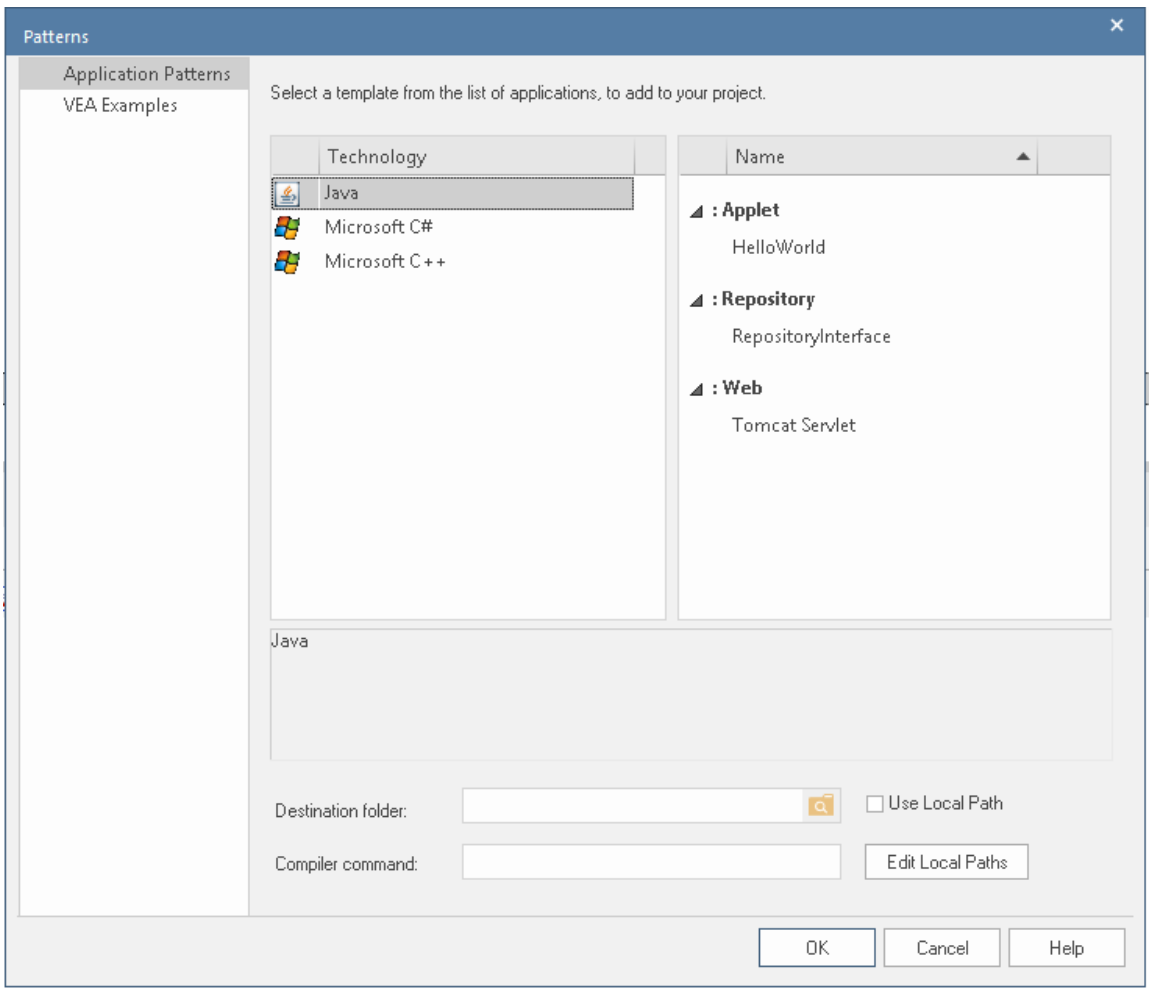
Pour vous permettre de démarrer un projet basé sur du code le plus rapidement possible, Enterprise Architect vous aide à générer des projets de démarrage comprenant des informations sur le modèle, du code et des scripts de construction pour l'un des nombreux types d'applications de base. Motifs incluent :

- Applications Windows MFC
- Programmes Java
- Services Web ASP.NET

## Accéder

| | |
|---|---|
| Ruban | Développer > Code source > Créer à partir de Motif > Motifs d'application |

## Générer des modèles




| Option | Action |
|---|---|
| | |

| | |
|---|---|
| Technologie | Sélectionnez la technologie appropriée. |
| Nom | Affiche les Motifs d'application disponibles pour la technologie sélectionnée ; sélectionnez le Motif requis à importer. |
| <description> | Affiche une description du Motif sélectionné. |
| Dossier de destination | Recherchez et sélectionnez le répertoire dans lequel charger le code source de l'application. |
| Utiliser le chemin local | Activez la sélection d'un chemin local existant pour placer le code source sous ; modifiez le champ « Dossier de destination » en une sélection déroulante. |
| Commande Compilateur | Affiche le chemin de commande du compilateur par défaut pour la technologie sélectionnée ; vous devez :<br>• Confirmez que le compilateur peut être trouvé à ce chemin, ou<br>• Modifier le chemin d'accès à l'emplacement du compilateur |
| Modifier les chemins locaux | De nombreux Motifs d'application spécifient leur compilateur à l'aide d'un chemin local.<br>La première fois que vous utilisez un Motif vous devez cliquer sur ce bouton pour vous assurer que le chemin local pointe vers le bon emplacement.<br>La dialogue « Chemins locaux » s'affiche. |

## Notes

- Si nécessaire, vous pouvez publier Motifs d'application personnalisés en ajoutant des fichiers au répertoire *AppPatterns* dans lequel Enterprise Architect est installé ; les répertoires de niveau supérieur sont répertoriés comme Technologies et peuvent contenir un fichier d'icône pour personnaliser l'icône affichée pour la technologie
  Les répertoires situés en dessous sont définis comme des groupes dans la liste Motifs ; les Motifs sont identifiés par la présence de quatre fichiers portant un nom correspondant : un fichier zip (.zip), un fichier XMI (.xml), un fichier de configuration (.cfg) et une icône facultative (.ico)
- Le fichier de configuration supporte ces champs :
  - [fournisseur], [langue], [plateforme], [url], [description], [version] - tous affichés dans la <description> champ
  - [xmirootpaths] - le chemin racine du code source dans le XMI exporté ; il est remplacé par le dossier de destination sélectionné lorsque l'utilisateur applique le Motif d'application

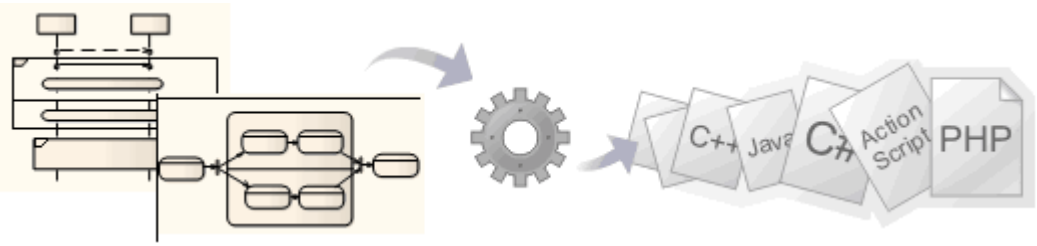
# Intégration MDG et Ingénierie de Code

MDG Integration for Eclipse et MDG Integration for Visual Studio sont des produits qui vous aident à créer et à maintenir vos modèles UML directement dans ces deux environnements de développement intégrés populaires, à l'aide de la fenêtre Enterprise Architect Navigateur . Les modèles peuvent être générés en code source à l'aide du moteur gabarit riche et flexible qui donne à l'ingénieur un contrôle complet sur la manière dont le code est généré. Le code source existant peut également être rétroconçu et synchronisé avec les modèles UML . Une fois l'intégration installée, l'IDE deviendra une plate-forme modélisation riche en fonctionnalités, économisant du temps et des efforts et réduisant le risque d'erreur en reliant la gestion des exigences, Architecture et la conception à l'ingénierie du code source.

Une documentation riche et expressive peut être générée automatiquement dans une large gamme de formats, notamment DOCX, PDF et HTML. La documentation peut inclure diagrammes d'exigences, de conception et architecture ainsi que des descriptions de code source, mettant le code source en contexte.

Vous pouvez acheter MDG Integration for Eclipse ™ et MDG Integration for Visual Studio ™ ou télécharger les éditions d'essai à partir du site Web Sparx Systems .

# Génération de code Modèle Comportementale




Les fonctionnalités d'ingénierie système polyvalentes d'Enterprise Architect permettent de générer du code pour les langages de description de logiciels, de systèmes et de matériels directement à partir de modèles comportementaux, tels que diagrammes Statemachine , Séquence (Interaction) et Activity. Les langages pris en charge incluent C(OO), C++, C# , Java, VB.Net, VHDL, Verilog et SystemC.

Le code logiciel peut être généré à partir de diagrammes Statemachine , Séquence et Activity, et des langages de description matérielle à partir de diagrammes Statemachine (en utilisant les gabarits Legacy Statemachine ).

## Accéder

| | |
|---|---|
| Ruban | Développer > Code Source > Générer |

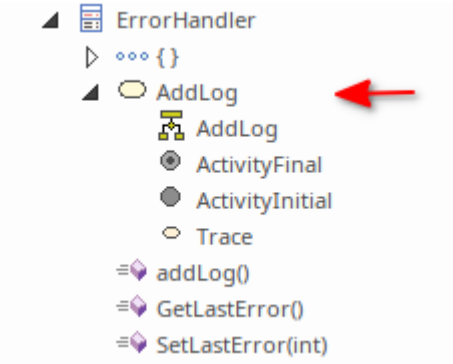
## Spécificités Modèle Comportementale

La génération de code de modèle Comportementale est prise en charge pour les trois principaux types de modèles comportementaux. Cependant, chaque type de modèle comportemental possède ses propres caractéristiques en fonction du type d'élément concerné. Ces rubriques fournissent guidage et des références pour les principaux types d'éléments utilisés.

| Type | Description |
|---|---|
| Activité | Un aperçu des principaux types d’actions et des détails sur leur utilisation dans la génération de code. |
| Interaction | Détails couvrant l'utilisation des messages et des fragments pour la génération de code des diagrammes d'interaction ( Séquence ). |
| Statemachines | Détails couvrant les options de définition du code à générer à l'aide States , y compris les comportements - Entrée/Sortie/Do et Transitions dans une Statemachine . |

## Structure

La génération de code du modèle Comportementale nécessite principalement que toutes les constructions comportementales soient contenues dans une classe (en tant qu'enfant de cette classe).




Si des constructions comportementales font référence à des éléments externes en dehors du Paquetage actuel, vous devez ajouter un connecteur Import du Paquetage actuel au Paquetage contenant les éléments externes. Pour plus de détails, consultez le connecteur *Import* -type dans la rubrique d'aide *Diagramme Paquetage* .

## Générer du code à partir de diagrammes comportementaux en utilisant le projet EAExample

| Étape | Action |
|---|---|
| 1 | Ouvrez le fichier EAExample.eap en sélectionnant l'option de ruban ' Démarrer > Aide > Aide > Ouvrir le Modèle d'exemple '. |
| 2 | Depuis la fenêtre Navigateur , sélectionnez l'un de ces Paquetages :<br>**Exemples de langage logiciel :**<br>• Exemple Modèle > Ingénierie de Logiciel > Modèle Java avec comportements<br>Générer les classes Account et Order<br>• Exemple Modèle > Ingénierie des Systèmes > Modèle d'implémentation > Logiciel > C#<br>Générer la classe DataProcessor<br>• Exemple Modèle > Ingénierie des Systèmes > Exemple SysML > Modèle d'implémentation > Logiciel > C++<br>Générer la classe IO<br>• Exemple Modèle > Ingénierie des Systèmes > Exemple SysML > Modèle d'implémentation > Logiciel > Java<br>Générer la classe IO<br>• Exemple Modèle > Ingénierie des Systèmes > Exemple SysML > Modèle d'implémentation > Logiciel > VBNet<br>Générer la classe IO<br>**Exemples de langage matériel :**<br>• Exemple Modèle > Ingénierie des Systèmes > Exemple SysML : Lecteur Audio Portable > Modèle d'implémentation > Matériel > SystemC<br>Générer la classe PlayBack<br>• Exemple Modèle > Ingénierie des Systèmes > Exemple SysML : Lecteur Audio Portable > Modèle d'implémentation > Matériel > VHDL<br>Générer la classe PlayBack<br>• Exemple Modèle > Ingénierie des Systèmes > Exemple SysML : Lecteur Audio Portable > Modèle d'implémentation > Matériel > Verilog<br>Générer la classe PlayBack |

| | |
|---|---|
| 3 | Une fois terminé :<br>• Sélectionnez la classe qui a été utilisée pour la génération<br>• Appuyez sur Ctrl+E pour ouvrir le code source généré.<br>Vous devriez voir des méthodes générées dans le code. |

## Notes

- La génération de code logiciel à partir de modèles comportementaux est disponible dans les éditions Unified et Ultimate d' Enterprise Architect
- La génération de code matériel à partir de modèles Statemachine est disponible dans les éditions Unified et Ultimate d' Enterprise Architect
- Pour C(OO), sur la page « Spécifications C » de la dialogue « Gérer les options Modèle », définissez l'option « Support orientée Object » sur Vrai.
  Voir la rubrique d'aide *Options C - Modèle* .
- La synchronisation du code n'est pas prise en charge pour le code comportemental.

# Génération de code - Diagrammes d'activités

La génération de code à partir de diagrammes d'activité dans une classe nécessite une phase de validation, au cours de laquelle Enterprise Architect utilise l'optimiseur de graphe d'ingénierie système pour analyser le diagramme et le restituer sous forme de différentes constructions à partir desquelles du code peut être généré. Enterprise Architect transforme également les constructions en l'un des différents types d'action (le cas échéant), de manière similaire aux constructions diagramme d'interaction.

## Actes

| Action | Description |
|---|---|
| Actions d'appel (actions d'invocation) | Utilisé pour invoquer des opérations ou des comportements dans un diagramme d'activité ; les deux principales variantes d'appels d'actions prises en charge dans la génération de code comportemental sont :<br>• Action CallOperation - utilisée pour invoquer des opérations, qui peuvent être dans la même classe ou dans d'autres classes au sein du même Paquetage ; si vous référencez des opérations d'autres classes au sein du même Paquetage , vous devez avoir une cible à laquelle la demande est transmise<br>• Action CallBehavior - utilisée pour invoquer une autre activité dans un flux d'activités ; l'activité référencée est censée être dans la même classe<br>**Arguments**<br>Les actions d'appel peuvent spécifier des valeurs d'argument correspondant aux paramètres du comportement ou de la fonctionnalité comportementale associée.<br>Vous pouvez ajouter les arguments manuellement ou les créer automatiquement à l'aide du bouton Synchroniser de la dialogue « Arguments ». |
| Créer un objet action | Utilisé pour désigner une création object dans le flux d'activité ; vous pouvez définir le Pin de résultat de CreateObjectAction comme l' object à créer, en utilisant la fenêtre Propriétés de l'élément Action .<br>Le classificateur de CreateObjectAction signifie le classificateur pour lequel une instance doit être créée. |
| Action de destruction d'objet | Utilisé pour indiquer une suppression object dans le flux d'activité ; vous pouvez définir le Pin cible de DestroyObjectAction comme l' object à détruire, en utilisant la fenêtre Propriétés de l'élément Action . |
| Boucles | L'optimiseur de graphiques d'ingénierie système d' Enterprise Architect est également capable d'analyser et d'identifier les boucles ; une boucle identifiée est rendue en interne sous la forme d'une boucle Action , qui est traduite par les macros de génération de code EASL pour générer le code requis.<br>Vous pouvez avoir une boucle unique, des boucles imbriquées et plusieurs niveaux de boucles imbriquées. |
| Déclarations conditionnelles | Pour modéliser une instruction conditionnelle, vous utilisez les nœuds Décision / Merge.<br>Alternativement, vous pouvez impliquer des décisions/fusions en interne ; l'optimiseur de graphique attend un nœud de fusion associé pour chaque nœud Décision , afin de faciliter le suivi efficace des différentes branches et l'analyse des constructions de code qu'elles contiennent. |

## Notes

- Pour pouvoir générer du code à partir de modèles comportementaux, toutes les constructions comportementales doivent être contenues dans une classe

# Génération de code - Diagrammes d'interaction

Lors de la génération de code à partir de diagrammes d'interaction ( Séquence ) dans une classe, Enterprise Architect applique son optimiseur de graphes d'ingénierie système pour transformer les constructions de classe en paradigmes programmatiques. Les messages et les fragments sont identifiés comme deux des nombreux types d'actions en fonction de leur fonctionnalité, et Enterprise Architect utilise les gabarits de génération de code pour restituer leur comportement en conséquence.

## Actes

| Action | Description |
|---|---|
| Appel Action | Un message qui appelle une opération. |
| Action Créer | Un message avec cycle de vie = Nouveau. |
| Action Détruire | Un message avec un cycle de vie = Supprimer. |
| Boucle Action | Un fragment combiné avec Type = Alt. |
| Action si | Un fragment combiné avec Type = boucle. |
| Affecter à | Un message d'appel avec un attribut cible valide défini à l'aide du champ « Attribuer à » est rendu dans le code comme l'attribut cible d'une Action d'appel. |

## Notes

- Pour pouvoir générer du code à partir de modèles comportementaux, toutes les constructions comportementales doivent être contenues dans une classe
- Pour un diagramme d'interaction ( Séquence ), le moteur de génération de code comportemental s'attend à ce que le diagramme Séquence et tous ses messages et fragments d'interaction associés soient encapsulés dans un élément d'interaction.

# Génération de code – Statemachines

Une Statemachine illustre comment un object (représenté par une classe) peut changer d'état, chaque changement d'état étant une transition initiée par un déclencheur découlant d'un événement, souvent sous des conditions ou des contraintes définies comme des gardes. Au fur et à mesure que vous modélisez la façon dont l' object change d'état, vous pouvez générer et construire (compiler) du code à partir de celui-ci dans le langage logiciel approprié et exécuter le code, en visualisant l'exécution via le simulateur Modèle .

Il est également possible, dans Enterprise Architect , de combiner les Statemachines d'objets distincts mais liés pour voir comment ils interagissent (via Broadcast Événements ), et de créer et générer rapidement du code à partir de variantes du modèle. Par exemple, vous pouvez modéliser le comportement de :

- La roue arrière côté passager d'un véhicule en modes traction arrière et traction avant (un seul Statemachine )
- Le volant et les quatre roues motrices d'un véhicule en mode 4 roues motrices (cinq Statemachines )
- Les roues d'un véhicule tout-terrain et d'une voiture de sport (deux artefacts, instances d'une combinaison de Statemachines )

L'élément Statemachine Exécutable Artifact est d'une importance cruciale pour générer et tester le code de toutes ces options. Il agit comme conteneur et unité de génération de code pour vos modèles Statemachine .

Vous n'utilisez pas cette méthode pour générer du code pour les langages de définition de matériel, mais vous pouvez également générer du code HDL et du code logiciel à partir de Statemachines à l'aide des facilités génériques de génération de code dans Enterprise Architect (voir les procédures *Générer Code Source* ).

## Prérequis

- Sélectionnez « Paramètres > Modèle > Options > Ingénierie du code source » et, pour le langage de codage logiciel approprié (Java, C, C# ou ANSI C++), définissez l'option « Utiliser la nouvelle Statemachine Gabarit » sur « Vrai »
- Si vous travaillez en C++, sélectionnez « Paramètres > Modèle > Options > Ingénierie du code source > C++ » et définissez l'option « Version C++ » sur « ANSI »

Cette méthode de génération de code ne s'applique pas aux gabarits de génération de code Legacy Statemachine développés avant Enterprise Architect Release 11.0, ni à la génération de code Hardware Definition Language.

## Accéder

Faites glisser un artefact Statemachine Exécutable depuis la page « Simulation » de la boîte à outils Diagramme vers votre diagramme . La page « Simulation » de la boîte à outils Diagramme est accessible à l'aide de l'une des méthodes décrites dans ce tableau .

| | |
|---|---|
| Ruban | Conception > Diagramme > Boîte à outils > Simulation |
| Raccourcis Clavier | Ctrl+Maj+3 > Simulation |
| Autre | Vous pouvez afficher ou masquer la boîte à outils Diagramme en cliquant sur les icônes » ou « à l'extrémité gauche de la barre de légende en haut de la Vue Diagramme . |

## Préparez votre (vos) diagramme (s) Statemachine

| Étape | Action |
|---|---|
| 1 | Pour chaque Statemachine vous souhaitez modéliser, créez un diagramme de classe. |
| 2 | Depuis la page « Classe » de la boîte à outils Diagramme , faites glisser l'icône « Classe » sur votre diagramme et donnez à l'élément un nom approprié. |
| 3 | Cliquez-droit sur l'élément Class et sélectionnez l'option de menu contextuel 'Nouveau Diagramme enfant \| Statemachine '.<br>Donnez au diagramme de la Statemachine un nom approprié. |
| 4 | Créez le modèle Statemachine pour refléter les transitions appropriées entre States . |

## Configurer l'artefact Statemachine Exécutable

| Étape | Action |
|---|---|
| 1 | Créez un nouveau diagramme de classe pour contenir la ou les Statemachine modélisées à partir desquelles vous avez l'intention de générer du code. |
| 2 | Depuis la page « Simulation » de la boîte à outils Diagramme , faites glisser l'icône « Statemachine Exécutable » sur le diagramme pour créer l'élément Artefact. Nommez l'élément et faites glisser ses bordures pour l'agrandir. |
| 3 | Depuis la fenêtre Navigateur , faites glisser l'élément Class (premier) contenant un diagramme Statemachine sur l'élément Artifact du diagramme .<br>La dialogue « Coller <nom de l'élément> » s'affiche. Dans le champ « Déposer sous », cliquez sur la flèche déroulante et sélectionnez la valeur « Propriété ».<br>(Si le dialogue ne s'affiche pas, appuyez sur Ctrl pendant que vous faites glisser l'élément Classe depuis la fenêtre Navigateur .) |
| 4 | Cliquez sur le bouton OK . L'élément Class est collé à l'intérieur de l'artefact en tant que partie. |
| 5 | Répétez les étapes 3 et 4 pour toutes les autres classes avec Statemachines que vous souhaitez combiner et pour lesquelles vous souhaitez générer du code. Il peut s'agir de :<br>• Répétez les « gouttes » de la même classe et de Statemachine , modélisation d'objets parallèles<br>• Différentes classes et Statemachines , modélisation d'objets interactifs séparés |
| 6 | Cliquez-droit sur l'élément Artifact et sélectionnez l'option ' Propriétés > Propriétés ', développez la catégorie 'Avancé' et, dans le champ 'Langue', cliquez sur la flèche déroulante et définissez la langue du code sur la même langue que celle définie pour les éléments Class.<br>Vous pouvez maintenant faire glisser cet élément Statemachine Exécutable Artifact de la fenêtre Navigateur sur le diagramme autant de fois que vous le souhaitez et modifier les parties pour modéliser des variantes du système ou du processus, ou le même système ou processus avec différents langages de programmation. |

## Générer du code à partir d'un artefact

| Étape | Action |
|---|---|
| 1 | Cliquez sur l'élément Statemachine Exécutable Artifact et sélectionnez l'option de ruban 'Simulate > States Exécutables > Statemachine > Générer '.<br>La dialogue « Génération de code Statemachine exécutable » s'affiche. |
| 2 | Dans le champ « Répertoire de sortie du projet », saisissez ou recherchez le chemin du répertoire sous lequel créer les fichiers de sortie.<br>Lors de la génération de code, tous les fichiers existants dans ce répertoire sont supprimés. |
| 3 | Sélectionnez le système cible. Si vous travaillez sous Windows sélectionnez l'option « Local ». Si vous travaillez sous Linux, choisissez l'option « À distance ». Le choix affecte les scripts générés pour support la Simulation . |
| 4 | Dans le champ « Emplacement du répertoire d'installation de <compilateur> », saisissez ou recherchez le chemin du répertoire d'installation du compilateur, qui sera automatiquement mappé au chemin local (affiché à gauche du champ). Pour chaque langage de programmation, les chemins peuvent ressembler à ces exemples :<br>• Java<br>JAVA_HOME C:\Program Files (x86)\Java\jdk1.7.0_17<br>• C/C++<br>VC_HOME C:\Program Files (x86)\Microsoft Visual Studio 9.0<br>• C#<br>CS_HOME C:\ Windows \Microsoft.NET\Framework\V3.5 |
| 5 | Cliquez sur le bouton Générer . Les fichiers de code sont créés en fonction du langage de programmation.<br>La fenêtre Sortie système s'affiche avec un onglet « Sortie Statemachine Exécutable », indiquant la progression et l'état de la génération.<br>Lors de la génération du code, une fonction de validation automatique est exécutée pour vérifier les erreurs diagramme ou de modèle par rapport aux contraintes UML . Les éventuelles erreurs sont identifiées par des messages d'erreur dans l'onglet ' Statemachine Exécutable Output'.<br>Double-cliquez sur un message d'erreur pour afficher la structure modélisation dans laquelle l'erreur se produit et corrigez l'erreur avant de régénérer le code. |
| 6 | Lorsque le code est généré sans erreur, cliquez sur l'élément Artifact et sélectionnez l'option de ruban « Simuler > States Exécutables > Statemachine > Construire » pour compiler le code.<br>La fenêtre Sortie système s'affiche avec un onglet « Créer », indiquant la progression et l'état de la compilation. Notez que la compilation inclut la configuration de l'opération de simulation. |

## Macros de génération de code

Vous pouvez également utiliser deux macros dans la génération de code pour Statemachines .

| Nom de la macro | Description |
|---|---|
| ENVOYER_EVENEMENT | Envoyer un événement à un récepteur (la partie). Par exemple :<br>%SEND_EVENT("événement1", "Partie1")% |
| | |

| | |
|---|---|
| ÉVÉNEMENT DIFFUSÉ | Diffuser un événement à tous les récepteurs. Par exemple :<br>%BROADCAST_EVENT("événement2")% |

## Exécuter/simuler du code à partir d'un artefact

| Étape | Action |
|---|---|
| 1 | Sélectionnez l'option du ruban « Simuler > Simulation Dynamique > Simulateur > Appliquer l'espace de travail » pour afficher ensemble la fenêtre Simulation et la fenêtre Simulation Événements<br>Placez les deux fenêtres dans une zone pratique de l'écran. |
| 2 | Sur le diagramme ou la fenêtre Navigateur , cliquez sur l'élément Artifact et sélectionnez l'option de ruban 'Simulate > States Exécutables > Statemachine > Exécuter '.<br>Le premier diagramme Statemachine de la série s'affiche avec la simulation du processus déjà démarrée. Dans la fenêtre Simulation , les étapes de traitement sont indiquées dans ce format :<br>[03516677] Partie 1 [Classe 1]. Initial_367_TO_State4_142 Effet<br>[03516683] Partie 1 [Classe 1]. StateMachine_State4 ENTRÉE<br>[03516684] Partie 1 [Classe 1].StateMachine_State4 DO<br>[03518375] Bloqué |
| 3 | Cliquez sur les boutons appropriés de la barre d'outils de la fenêtre de Simulation pour parcourir la simulation comme vous le souhaitez.<br>Lorsque la simulation se termine au niveau de l'élément Quitter ou Terminer, cliquez sur le bouton Arrêter dans la barre d'outils de la fenêtre Simulation . |
| 4 | Lorsque la trace indique Bloqué, la simulation a atteint un point où un événement Déclencheur doit se produire avant que le traitement puisse continuer. Dans la fenêtre Simulation Événements , dans la colonne « Déclencheurs en Attente », double-cliquez sur le Déclencheur approprié.<br>Lorsque le Déclencheur est déclenché, la simulation continue jusqu'au prochain point de pause, Déclencheur ou sortie. |

## Notes

- Si vous apportez de petites modifications à un modèle Statemachine existant, vous pouvez combiner les opérations de génération, de construction et exécuter de code en sélectionnant l'option de ruban « Simuler > States Exécutables > Statemachine > Générer , construire et exécuter ».
- Vous pouvez également générer du code en JavaScript

# Statemachine héritée Gabarits

La génération de code fonctionne à l'aide d'un ensemble de gabarits de génération. À partir de la version 11.0 d' Enterprise Architect , un ensemble différent de gabarits est disponible par défaut pour la génération de code logiciel à partir d'un diagramme Statemachine en code Java, C, ANSI C++ ou C# . Vous pouvez toujours utiliser les gabarits d'origine, comme décrit ici, pour les modèles développés dans les versions antérieures d' Enterprise Architect , si vous ne souhaitez pas facilités mettre à niveau pour les nouveaux gabarit .

## Basculer entre gabarits Legacy et Release 11

## Accéder

Affichez la dialogue « Gérer les options Modèle », puis affichez la page « Spécifications de langue » pour la langue choisie, en utilisant l'une des méthodes décrites dans ce tableau . Si nécessaire, développez le groupement « Ingénierie Statemachine (pour le modèle actuel) » et définissez l'option « Utiliser la nouvelle Statemachine Gabarit » sur True (pour utiliser les gabarits ultérieurs) ou False (pour utiliser les gabarits hérités).

| | |
|---|---|
| Ruban | Paramètres > Modèle > Options > Ingénierie du code source > [nom de la langue] |

## Transformations héritées Gabarit

Une Statemachine dans une classe génère en interne un certain nombre de constructions dans des langages logiciels pour assurer l'exécution efficace des comportements des States (do, entry et exit) et également pour coder l'effet de transition approprié lorsque cela est nécessaire.

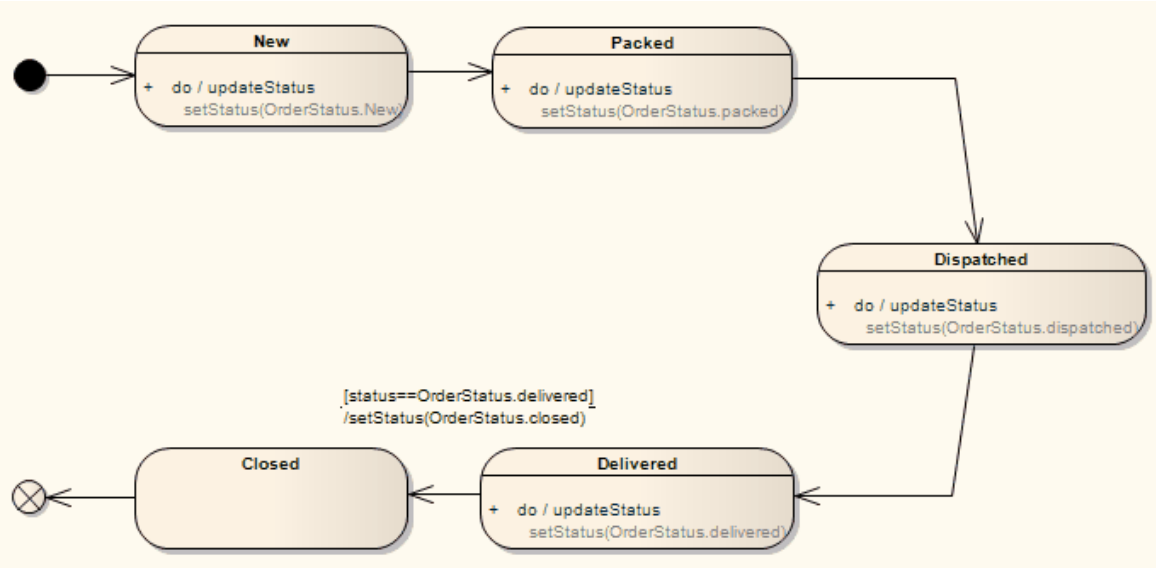
| Objets Modèle | Objets de code |
|---|---|
| Énumérations | • StateType - se compose d'une énumération pour chacun des States contenus dans la Statemachine<br>• TransitionType – consiste en une énumération pour chaque transition à laquelle est associé un effet valide ; par exemple, ProcessOrder_Delivered_to_ProcessOrder_Closed<br>• CommandType – consiste en une énumération pour chacun des types de comportement qu'un State peut contenir (Do, Entry, Exit) |
| Attributes | • currState:StateType - une variable pour contenir les informations sur State actuel<br>• nextState:StateType - une variable pour contenir les informations de State suivant, définies par les transitions de chaque State en conséquence<br>• currTransition:TransitionType - une variable pour contenir les informations de transition actuelles ; elle est définie si la transition est associée à un effet valide<br>• transcend:Boolean - un indicateur utilisé pour indiquer si une transition est impliquée dans la transcendance entre différentes Statemachines (ou états de sous-machines)<br>• xx_history:StateType - une variable d'historique pour chaque State Statemachine /sous-machine, pour contenir des informations sur le dernier |

| | |
|---|---|
| | State à partir duquel la transition a eu lieu |
| Opérations | • StatesProc - une procédure States , contenant une carte entre l'énumération d'un State et son fonctionnement ; elle déréférence les informations de State actuel pour invoquer la fonction de State respectif<br>• TransitionsProc - une procédure de transition, contenant une carte entre l'énumération de la transition et son effet ; elle invoque l'effet de la transition<br>• <<État>> - une opération pour chacun des States contenus dans la Statemachine ; cela restitue les comportements d'un State en fonction du CommandType d'entrée et exécute également ses transitions<br>• initializeStateMachine - une fonction qui initialise tous les attributs liés au framework<br>• runStateMachine - une fonction qui parcourt chaque State et exécute leurs comportements et transitions en conséquence |

## Notes

- Pour pouvoir générer du code à partir de modèles comportementaux, toutes les constructions comportementales doivent être contenues dans une classe

# Code Java généré à partir d' Statemachine héritée Gabarit




énumération privée StateType : int
{
ProcessusCommande_Livrée,
ProcessusCommande_Emballée,
ProcessusCommande_Fermée,
ProcessusOrder_Dispatched,
ProcessOrder_New,
ST_NOSTATE
}
énumération privée TransitionType : int
{
ProcessOrder_Delivered_to_ProcessOrder_Closed,
TT_NOTRANSITION
}
énumération privée CommandType
{
Faire,
Entrée,
Sortie
}
État privéType currState;
État privéType nextState;
TransitionType privé currTransition;
transcendance booléenne privée ;
État privé Type ProcessOrder_history ;
processus privé voidOrder_Delivered(CommandType commande)

```
{
commutateur(commande)
{
affaire à faire :
{
// Faire des comportements..
setStatus(Livré);
// Les transitions State
si((statut==Livré))
{
nextState = StateType.ProcessOrder_Closed;
currTransition = TransitionType.ProcessOrder_Delivered_to_ProcessOrder_Closed;
}
casser;
}
défaut:
{
casser;
}
}
}
processus privé voidOrder_Packed(CommandType commande)
{
commutateur(commande)
{
affaire à faire :
{
// Faire des comportements..
setStatus(Emballé);
// Les transitions State
nextState = StateType.ProcessOrder_Dispatched;
casser;
}
défaut:
{
casser;
}
}
}
processus privé voidOrder_Closed(CommandType commande)
{
commutateur(commande)
```

```
{
affaire à faire :
{
// Faire des comportements..
// Les transitions State
casser;
}
défaut:
{
casser;
}
}
}
processus privé voidOrder_Dispatched(CommandType commande)
{
commutateur(commande)
{
affaire à faire :
{
// Faire des comportements..
setStatus(Expédié);
// Les transitions State
nextState = StateType.ProcessOrder_Delivered;
casser;
}
défaut:
{
casser;
}
}
}
processus privé voidOrder_New(CommandType commande)
{
commutateur(commande)
{
affaire à faire :
{
// Faire des comportements..
setStatus(nouveau);
// Les transitions State
nextState = StateType.ProcessOrder_Packed;
casser;
```

```
}
défaut:
{
casser;
}
}
}
privé void StatesProc(StateType currState, CommandType commande)
{
commutateur(état actuel)
{
cas ProcessOrder_Delivered :
{
processOrder_Delivered(commande);
casser;
}
cas ProcessOrder_Packed :
{
processOrder_Packed(commande);
casser;
}
dossier ProcessOrder_Closed :
{
processOrder_Closed(commande);
casser;
}
cas ProcessOrder_Dispatched :
{
processOrder_Dispatched(commande);
casser;
}
affaire ProcessOrder_New :
{
processOrder_New(commande);
casser;
}
défaut:
casser;
}
}
privé void TransitionsProc(TransitionType transition)
{
```

```
commutateur (transition)
{
dossier ProcessOrder_Delivered_to_ProcessOrder_Closed :
{
setStatus(fermé);
casser;
}
défaut:
casser;
}
}
privé void initalizeStateMachine()
{
currState = StateType.ProcessOrder_New;
État suivant = Type d'État.ST_NOSTATE;
currTransition = TransitionType.TT_NOTRANSITION;
}
privé void runStateMachine()
{
tandis que (vrai)
{
si (currState == StateType.ST_NOSTATE)
{
casser;
}
currTransition = TransitionType.TT_NOTRANSITION;
ÉtatsProc(currState, CommandType.Do);
// puis vérifiez s'il existe une transition valide attribuée après le comportement do
si (nextState == StateType.ST_NOSTATE)
{
casser;
}
si (currTransition != TransitionType.TT_NOTRANSITION)
{
TransitionsProc(currTransition);
}
si (état actuel != état suivant)
{
ÉtatsProc(currState, CommandType.Exit);
StatesProc(nextState, CommandType.Entry);
currState = État suivant;
}
```

```
}
}
```

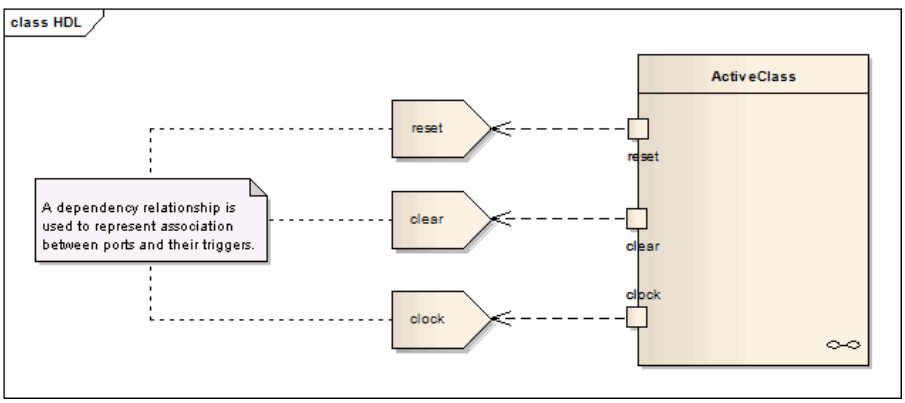
# Modélisation Statemachine pour les HDL

Pour générer efficacement du code HDL (Hardware Description Language) à partir de modèles Statemachine , appliquez les pratiques de conception décrites dans cette rubrique. Les langages de description du matériel incluent VHDL, Verilog et SystemC.

Dans un modèle de Statemachine HDL, vous pouvez vous attendre à :

- Désigner Déclencheurs de Conduite
- Établir une cartographie port-déclencheur
- Ajouter à la logique State Actif

## Opérations

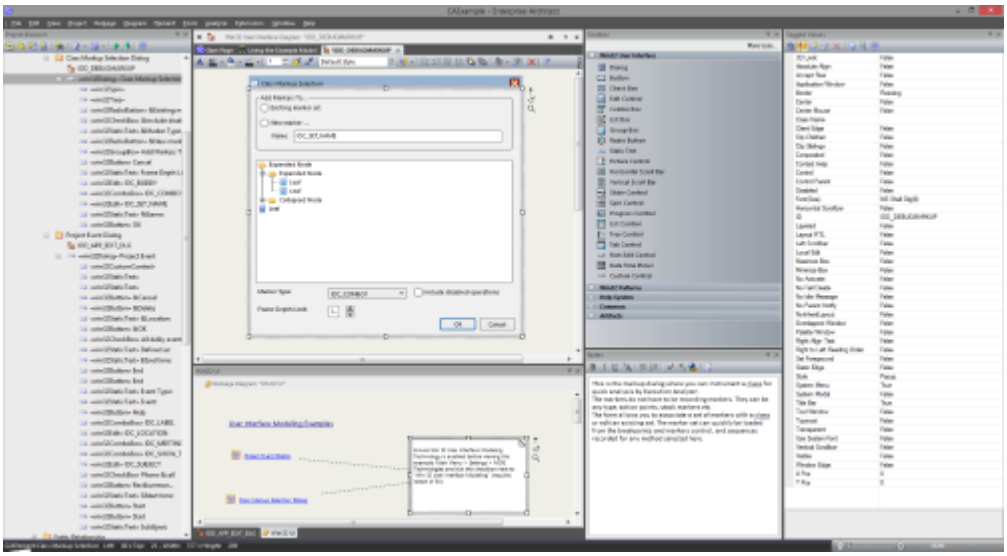
| Opération | Description |
|---|---|
| Désigner Déclencheurs de Conduite | • Un Déclencheur « Change » est considéré comme un déclencheur asynchrone si :<br>- Il y a une transition de l' State actuel de la sous-machine (qui encapsule la logique réelle) qu'elle déclencheurs , et<br>- L' State cible de cette transition a une auto-transition déclenchée par le même Déclencheur<br>• Déclencheurs Asynchrones doivent être modélisés selon ce motif :<br>- Le Déclencheur doit être de type Change (spécification : Vrai / Faux)<br>- L' State actif ( State de sous-machine) doit avoir une transition déclenché par cela<br>- L' State cible de la transition déclenchée doit avoir un auto transition avec le même Déclencheur<br>• Un Déclencheur de type 'Time', qui déclencheurs les transitions vers l'état actif (SubMachine State ), est considéré comme l'Horloge ; la spécification de ce déclencheur doit être conforme au langage cible :<br>- VHDL - front montant / front descendant<br>- Verilog - posedge / negedge<br>- SystemC - positif / négatif |
| Établir une cartographie des déclencheurs de port | Après modélisation avec succès les différents modes de fonctionnement du composant, et les Déclencheurs qui leur sont associés, vous devez associer les Déclencheurs aux Ports du composant.<br>Une relation de dépendance du port au Déclencheur associé est utilisée pour signifier cette association.<br>  |
| | |

| | |
|---|---|
| Logique State Actif | La désignation du Déclencheur pilote et l'établissement du mapping Port-Trigger mettent en place les préliminaires nécessaires à l'interprétation efficace des composants matériels.<br>Nous modélisons maintenant la logique réelle Statemachine dans l' State Actif (SubMachine). |

## Notes

- Pour pouvoir générer du code à partir de modèles comportementaux, toutes les constructions comportementales doivent être contenues dans une classe
- Le moteur de génération de code actuel supporte qu'un seul Déclencheur d'horloge par composant

# Boîtes de dialogue Interface Utilisateur Win32



Grâce à la technologie MDG Win32 UI , vous pouvez concevoir des écrans d'interface utilisateur qui s'affichent sous forme de contrôles Win32®. L'interface utilisateur produite peut être utilisée dans n'importe quel script de définition de ressources. Les scripts de définition de ressources, ou fichiers RC, sont une technologie Microsoft qui, comme pour d'autres codes, peut être compilée et les ressources utilisées par les applications de bureau natives. Les écrans ou boîtes de dialogue d'interface utilisateur peuvent être créés à partir de zéro ou rétroconçus. Les modèles d'interface utilisateur peuvent également être rétroconçus à l'aide de la fonction de synchronisation de code (F7). modélisation d'interface se déroule sur diagrammes exactement de la même manière que vous travailleriez avec n'importe quelle technologie dans Enterprise Architect . Un aspect intéressant de la conception Interface Utilisateur dans Enterprise Architect est que les composants peuvent jouer un rôle actif dans la simulation de Statemachines et d'activités, permettant à une simulation d'interagir avec les utilisateurs, à la manière d'un vrai programme !

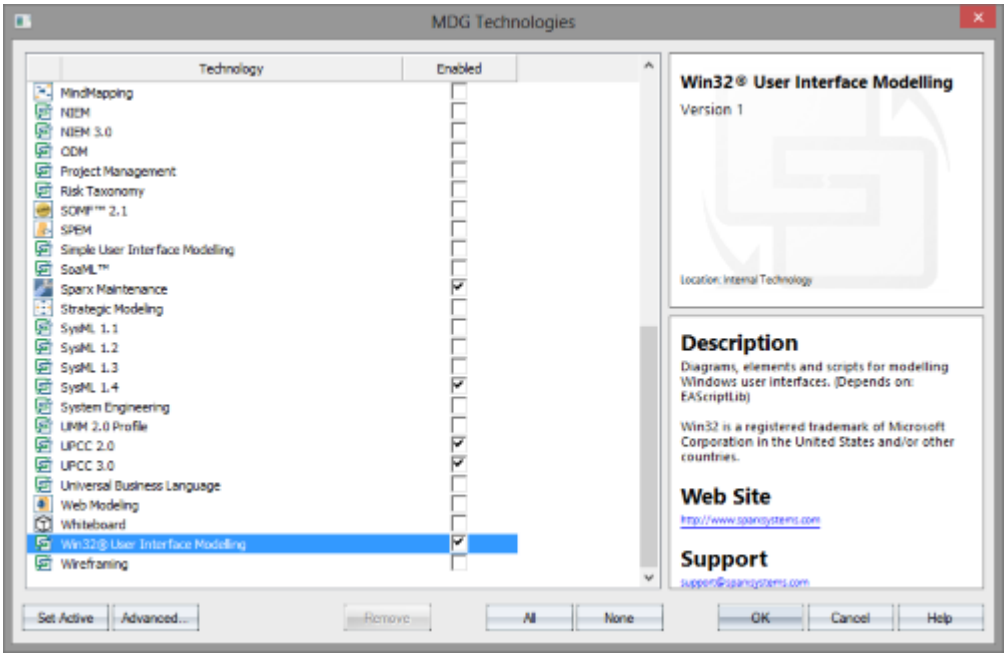
## Accéder

| | |
|---|---|
| Ruban | Conception > Diagramme > Ajouter Diagramme > Type > Interface Utilisateur Win32 |
| Menu Contexte | Cliquez-droit sur Paquetage \| Ajouter Diagramme > Type \| Interface Utilisateur Win32 |
| Autre | Menu de la barre de légende de la fenêtre du Navigateur \| Nouveau Diagramme \| Interface Utilisateur Win32 |

## Support

La technologie MDG Win32® Interface Utilisateur est disponible dans les éditions Enterprise Architect Professional , Corporate , Unified et Ultimate

## Activation de la technologie Interface Utilisateur Win32



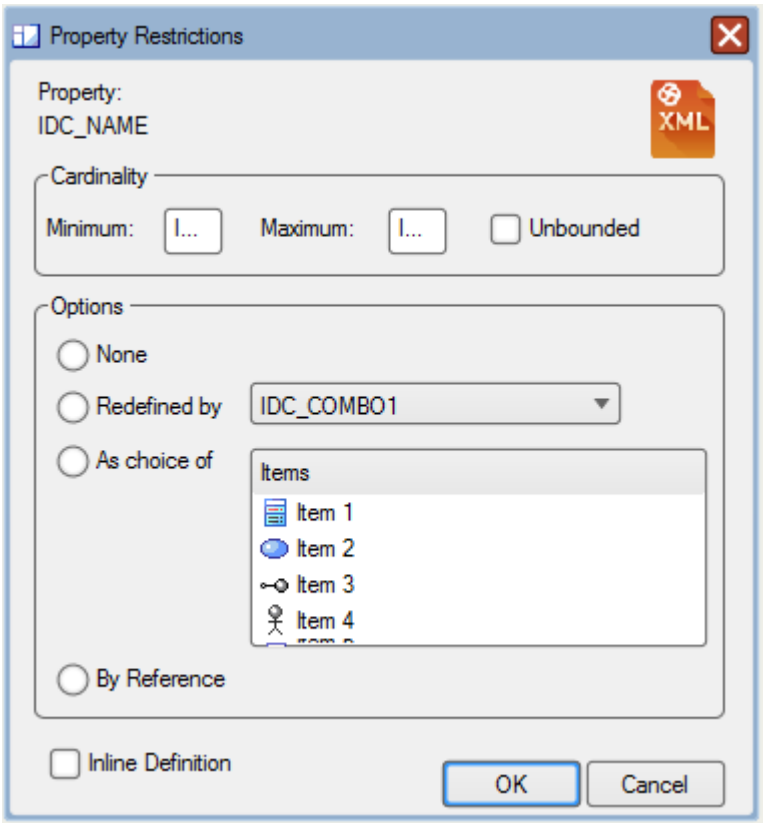

La technologie UI Win32® dans Enterprise Architect est activée ou désactivée à l'aide de la dialogue « MDG Technologies » (sélectionnez l'option de ruban « Spécialiser > Technologies > Gérer la technologie »).

## Technologie par défaut

Vous pouvez définir la technologie UI MDG Win32® comme technologie par défaut active pour accéder directement aux pages de la boîte à outils.

# Modélisation des dialogues UI

L' Interface Utilisateur Win32 MDG Technologie fournit les outils pour vous aider à concevoir une interface utilisateur qui émule étroitement le style visuel et les options disponibles pour les boîtes de dialogue Windows .




## Dialogue Win32

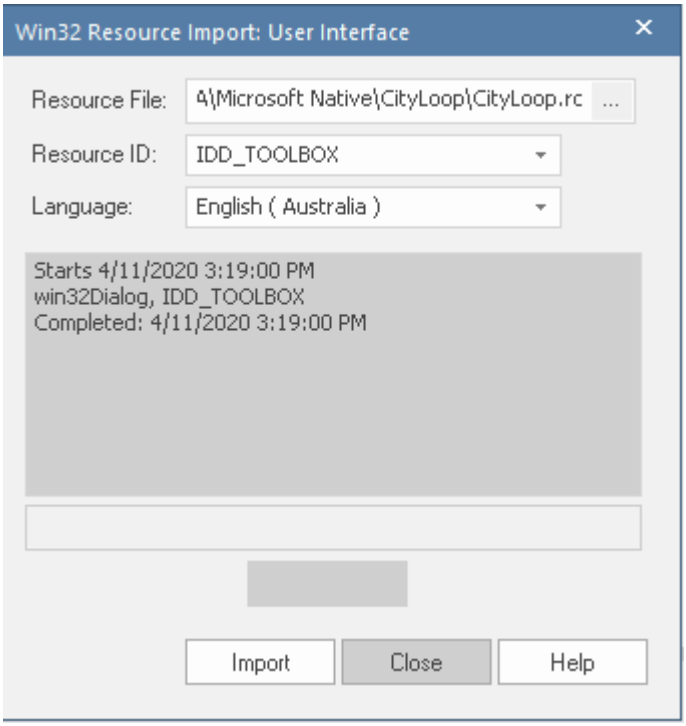
Ces composants d'interface utilisateur sont pris en charge, chacun correspondant à la ressource RC portant le nom équivalent.

| Composant | Détails |
|---|---|
| dialogue win32 | L'équivalent des ressources au format RC DIALOG et DIALOGEX. |
| Texte statique win32 | L'équivalent des ressources au format RC LTEXT, RTEXT, CTEXT. |
| win32Modifier | L'équivalent de la ressource EDITTEXT au format RC. |
| Bouton win32 | L'équivalent du format RC BUTTON, DEFPUSHBUTTON et d'autres ressources. |
| case à cocher win32 | L'équivalent de la ressource CHECKBOX au format RC. |
| barre de défilement win32H | L'équivalent de la ressource SCROLLBAR au format RC avec le style SBS_HORZ |
| barre de défilement win32V | L'équivalent de la ressource SCROLLBAR au format RC avec le style SBS_VERT. |

| | |
|---|---|
| Boîte de groupe win32 | L'équivalent de la ressource GROUPBOX au format RC. |
| boîte de dialogue Win32ComboBox | L'équivalent de la ressource COMBOBOX au format RC.<br>Note : Lorsque vous faites glisser l'icône « Combo Box » (de type « Liste déroulante » ou « Liste déroulante ») sur un diagramme , la « poignée de suivi » du milieu de chaque côté de l'élément est blanche, ce qui indique que vous ne pouvez ajuster que la largeur de l'élément. Pour ajuster la hauteur de l'élément ainsi que sa largeur, cliquez sur la partie flèche déroulante de l'image ; la « poignée de suivi » du milieu sur le bord inférieur est maintenant blanche, ce qui indique que vous pouvez faire glisser la base vers le bas pour définir la hauteur virtuelle (la hauteur de l'élément lorsqu'il est développé pour afficher toutes les valeurs possibles dans la liste déroulante). |
| Liste de listes win32 | L'équivalent de la ressource LISTBOX au format RC. |
| Bouton radio win32 | L'équivalent de la ressource RADIOBUTTON au format RC. |
| Panneau d'onglets win32 | L'équivalent de la ressource TABPANE au format RC. |
| win32Image | L'équivalent de la ressource STATIC au format RC avec le style SS_BITMAP.<br>Le contrôle peut restituer une image lorsqu'il est appliqué à partir de votre modèle. Une image peut être appliquée en la sélectionnant d'abord et en appuyant sur Ctrl+Maj+W pour afficher le Gestionnaire d'images. Ensuite, vous devrez peut-être modifier la valeur de l' ID de ressource dans la Valeur Étiquetée appropriée. |
| Contrôle personnalisé win32 | L'équivalent de la ressource CONTROL au format RC. |

# Importer Dialogue unique à partir d'un fichier RC

Vous pouvez rapidement importer un seul dialogue par nom.



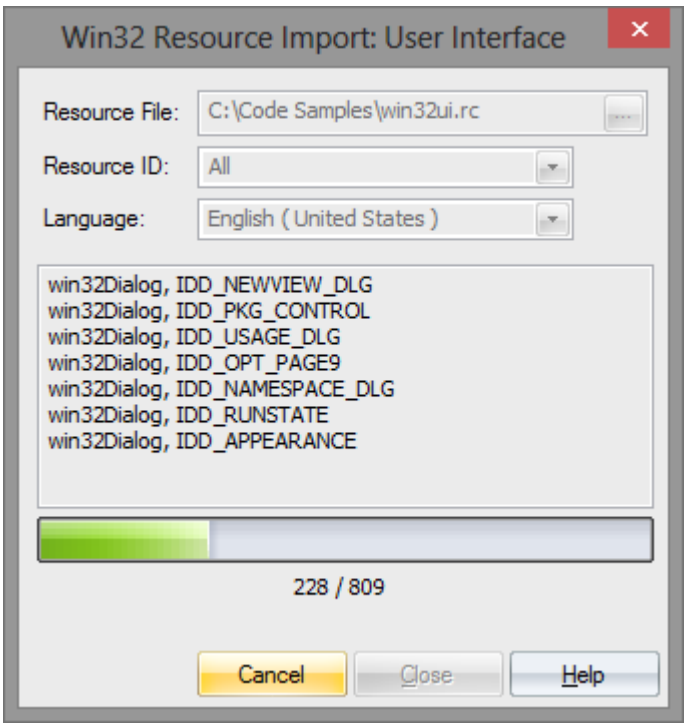

## Accéder

Dans la fenêtre Navigateur , cliquez sur la cible Paquetage .

| | |
|---|---|
| Ruban | Développer > Code source > Fichiers > Importer un script de ressources |

# Importer toutes les boîtes de dialogue à partir du fichier RC

Toutes les boîtes de dialogue d'un seul fichier RC peuvent être importées dans votre modèle. Cette image a été capturée une minute après le début de l'importation, moment auquel plus de 200 grandes définitions dialogue avaient été importées.

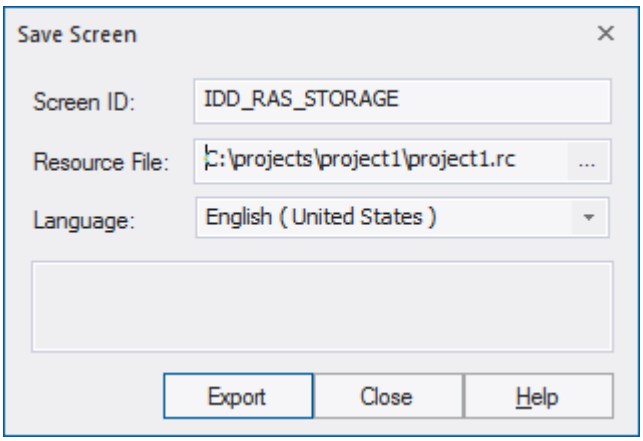



## Accéder

| | |
|---|---|
| Ruban | Développer > Code source > Fichiers > Importer un script de ressources |

# Exporter Dialogue vers un fichier RC

Une fois la conception d'un écran modifiée ou une nouvelle créée, vous souhaiterez peut-être la récupérer dans le fichier RC que vous avez utilisé pour créer votre application, afin de pouvoir voir à quoi elle ressemble avec des données réelles. Commencez par sélectionner l'élément Win32Dialog dans la fenêtre Navigateur , puis utilisez le ruban pour effectuer la synchronisation.




## Accéder

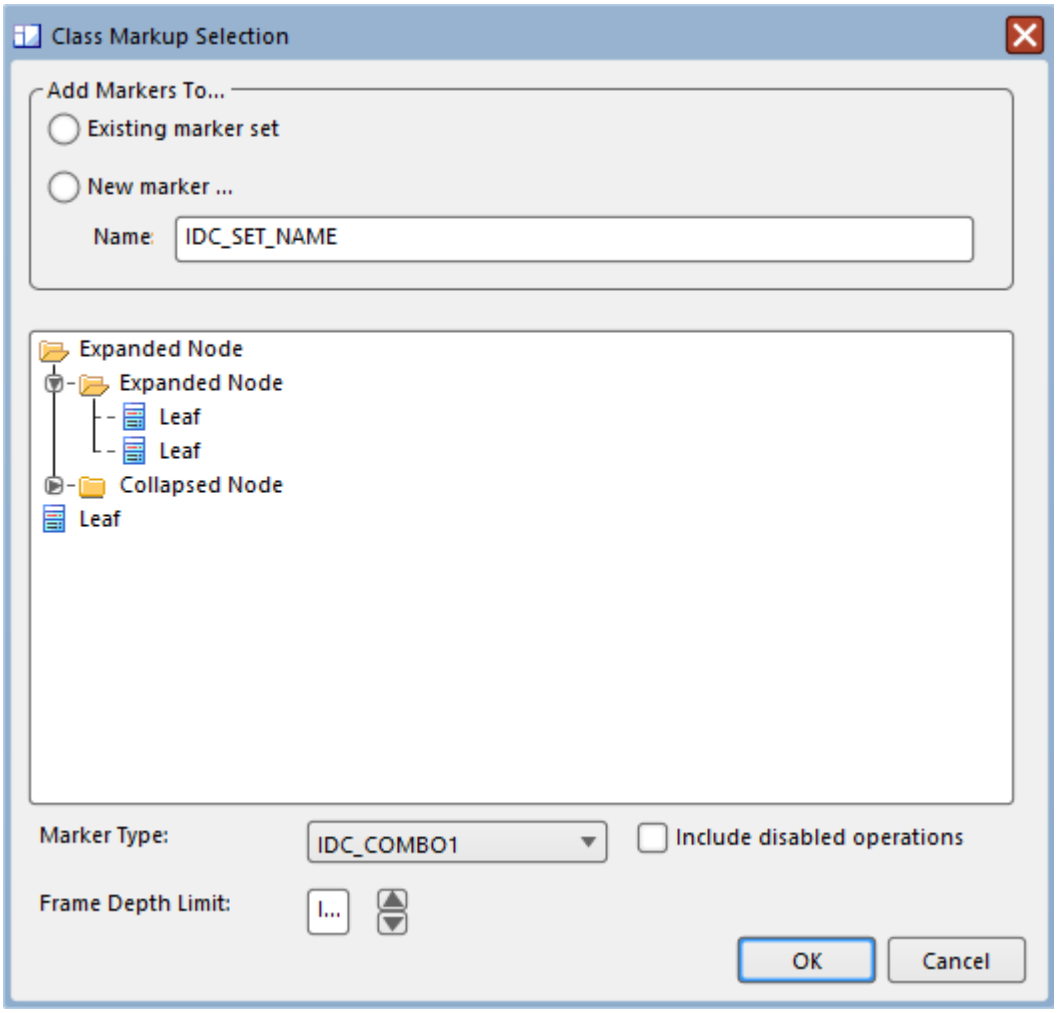
Cliquez sur l'élément win32Dialog.

| | |
|---|---|
| Ruban | Développer > Code Source > Générer > Générer un seul élément |
| Raccourcis Clavier | F11 |

# Concevoir un nouveau Dialogue

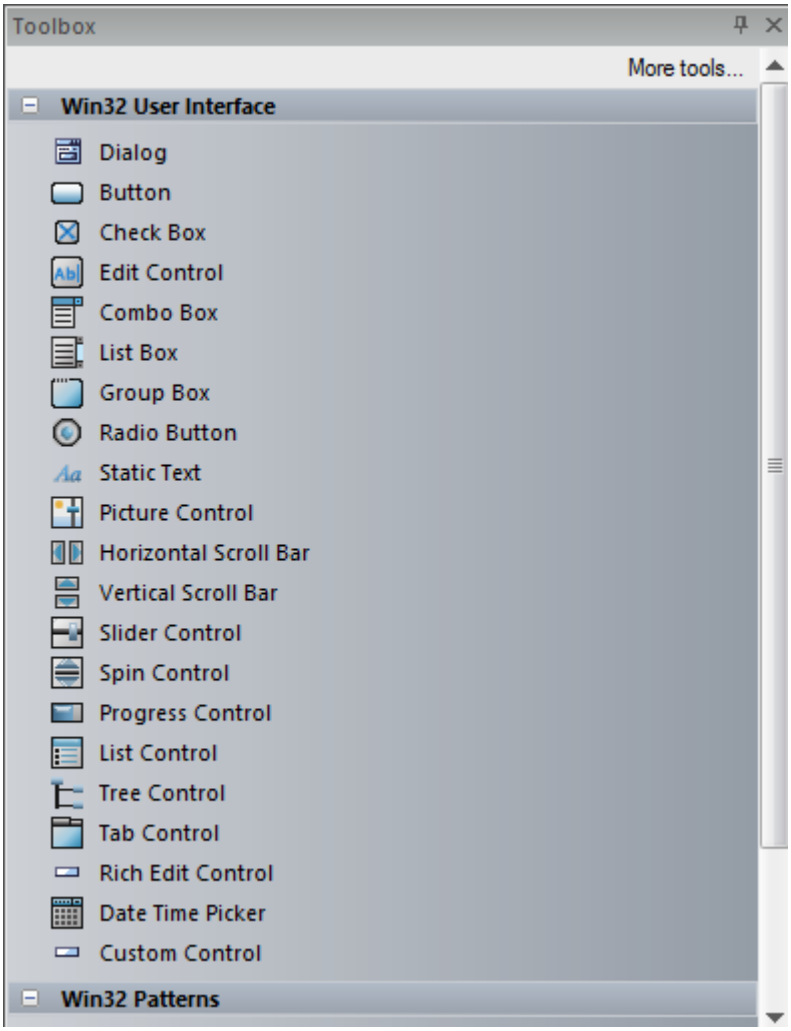
Créer une nouvelle dialogue Win32 est simple et principalement visuelle. Vous aurez probablement besoin d'un espace de travail qui affiche :

- Le nouveau diagramme (sélectionnez le chemin du ruban 'Conception > Diagramme > Ajouter Diagramme > Interface Utilisateur - Win32 > Interface Utilisateur - Win32')
- La boîte à outils Interface Utilisateur Win32 (sélectionnez l'option de ruban « Conception > Diagramme > Boîte à outils ») et
- L'onglet Valeur Étiquetés de la fenêtre Propriétés



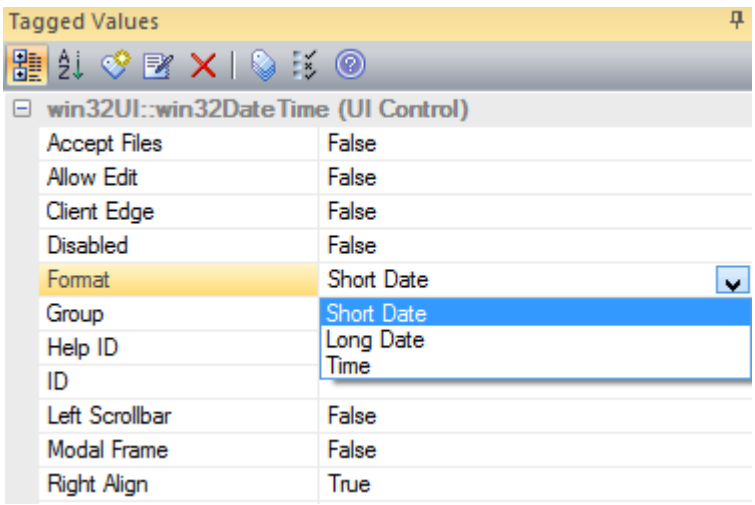

## La boîte à outils UI

Tous les éléments RC courants peuvent être trouvés dans la boîte à outils UI




## L'onglet Étiquettes

Cet onglet est fourni dans la fenêtre Propriétés et dialogue « Propriétés » d'un object , et c'est là que toutes les propriétés d'un contrôle peuvent être visualisées et modifiées.



## Utilisation du contrôle d'image

Les images de votre modèle (voir *Gestionnaire d'images* ) peuvent être appliquées en sélectionnant le contrôle dans le dialogue et en appuyant sur Ctrl+Maj+W. Vous devrez peut-être saisir la valeur de l' ID de ressource dans la Valeur Étiquetée appropriée.

## Note

- Vous pouvez copier et coller Paquetages dialogue

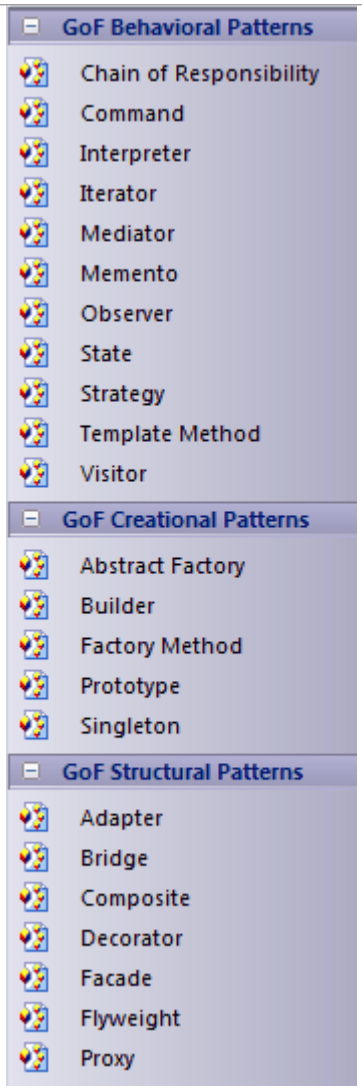
# Motifs de Gang of Four (GoF)

Un Motif de conception est un gabarit permettant de résoudre des problèmes de conception récurrents. Il se compose d'une série d'éléments et de connecteurs qui peuvent être réutilisés dans un nouveau contexte. L'avantage d'utiliser Motifs est qu'ils ont été testés et affinés dans un certain nombre de contextes et constituent donc généralement des solutions robustes aux problèmes courants. Enterprise Architect fournit le Gang of Four Motifs en tant que MDG Technologie qui peut être chargée dans le référentiel actuel.

Les Motifs Gang of Four (Gof) sont un groupe de vingt-trois Motifs de conception publiés à l'origine dans un livre fondateur intitulé *Design Motifs : Elements of Reusable Object-Oriented Software* ; le terme « Gang of Four » fait référence aux quatre auteurs. Enterprise Architect affiche ces Motifs dans son moteur Motif , vous aidant à visualiser les éléments du Motif et à ajuster le Motif au contexte de votre problème de conception de logiciel.

## Motifs GoF dans Enterprise Architect

| Fonctionnalités | Description |
|---|---|
| Facilités Motif GoF | Les Motifs GoF sont fournis sous la forme de :<br>• Pages Motifs Comportementale GoF, Motifs Créationnels GoF et Motifs Structurels GoF dans la Boîte à Outils<br>• Entrées Motif Gang of Four dans le menu contextuel de la boîte à outils<br>**Pages de la boîte à outils GoF Motif**<br>Vous pouvez accéder aux pages « GoF Motif » de la Boîte à Outils en cliquant sur  pour afficher la dialogue « Trouvez Item de Boîte à Outils » et en spécifiant « GoF Motifs » ; ces icônes sont disponibles : |




Lorsque vous faites glisser l'un des éléments Motif sur un nouveau diagramme , la dialogue « Ajouter Motif GoF < groupe motif >< type motif > » s'affiche ; si nécessaire, modifiez l'action et/ou la valeur par défaut des éléments composants, puis cliquez sur le bouton OK pour créer un diagramme basé sur le Motif .

# Icône

Le processus ICONIX est une méthodologie de développement logiciel propriétaire basée sur UML . Le processus est axé sur les cas d'utilisation et utilise diagrammes basés sur UML pour définir quatre étapes. La principale fonctionnalité du processus est un concept appelé modélisation de la robustesse, basé sur les premiers travaux d'Ivar Jacobson, qui permet de combler le fossé entre l'analyse et la conception.

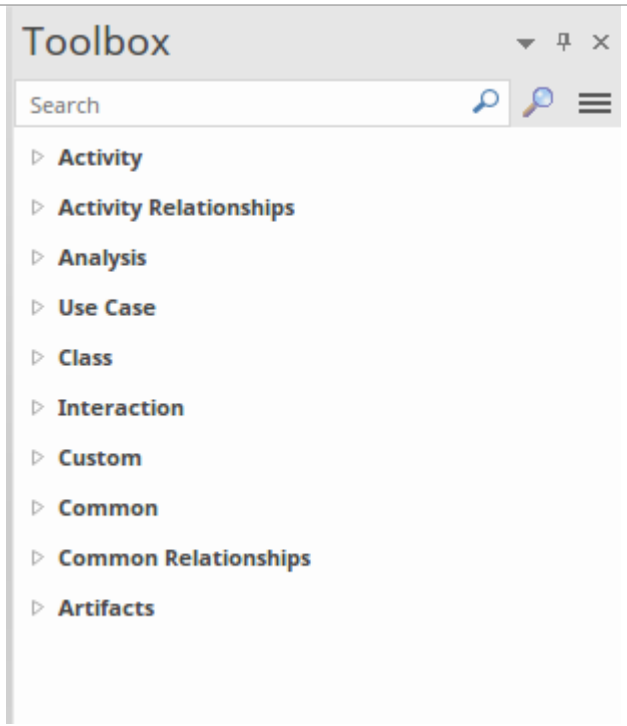
Ce texte est dérivé de l'entrée ICONIX dans Wikipédia en ligne :

« Le processus ICONIX est une approche minimaliste et rationalisée de modélisation UML axée sur les cas d'utilisation qui utilise un sous-ensemble de base de diagrammes et de techniques UML pour fournir une couverture complète de l'analyse et de la conception orientées objet. Son activité principale est l'analyse de robustesse, une méthode permettant de combler le fossé entre l'analyse et la conception. L'analyse de robustesse réduit l'ambiguïté dans les descriptions de cas d'utilisation, en garantissant qu'elles sont écrites dans le contexte d'un modèle de domaine d'accompagnement. Ce processus rend les cas d'utilisation beaucoup plus faciles à concevoir, à tester et à estimer. »

Le processus ICONIX a été développé par Doug Rosenberg ; pour plus d'informations sur ICONIX, consultez ICONIX Ingénierie de Logiciel Inc.

## Aspects

| Aspect | Détail |
|---|---|
| ICONIX dans Enterprise Architect | Enterprise Architect vous permet de développer des modèles sous ICONIX rapidement et simplement, grâce à l'utilisation d'une MDG Technologie intégrée à l'installateur Enterprise Architect .<br>Les facilités ICONIX sont fournies sous la forme de :<br>• Un ensemble de pages ICONIX dans la boîte à outils<br>• Entrées d'éléments et de relations ICONIX dans le menu « Raccourci de la boîte à outils » et Quick Linker<br>Pour vous aider davantage à développer et à gérer un projet sous ICONIX, Enterprise Architect fournit également un livre blanc sur la Feuille de Route ICONIX. |
| Pages de la boîte à outils ICONIX | Dans la boîte à outils, Enterprise Architect fournit des versions ICONIX des pages pour les diagrammes d'analyse UML , de cas d'utilisation, de classe, d'interaction ( Séquence ), d'activité et personnalisés (qui constituent souvent la base diagrammes diagrammes robustesse).<br>Par rapport aux pages de la boîte à outils standard, celles-ci ont des ensembles d'éléments et de relations légèrement différents ; vous pouvez y accéder de l'une des manières suivantes :<br>• Spécifier 'ICONIX' dans la dialogue ' Trouvez Item de Boîte à Outils ' et sélectionner une page de boîte à outils spécifique<br>• Sélection de l'option « ICONIX » dans le champ déroulant de la barre d'outils Outils par défaut, qui ajoute les six pages à la boîte à outils ; toutes les pages sont fermées |


Toolbox
Search
Activity
Activity Relationships
Analysis
Use Case
Class
Interaction
Custom
Common
Common Relationships
Artifacts

# Paramètres de configuration

Vous pouvez définir les options de code par défaut telles que les éditeurs pour chacun des langages de programmation disponibles pour Enterprise Architect et les options spéciales pour la génération ou la rétro-ingénierie du code source. Ces options sont définies selon qu'elles s'appliquent à :

- Tous les utilisateurs du modèle actuel, définis dans la dialogue « Gérer les options Modèle », ou
- Tous les modèles auxquels vous accédez (les autres utilisateurs peuvent définir leurs propres paramètres qui s'appliquent aux mêmes modèles), définis dans la dialogue « Préférences »

Vous pouvez également :

- Pour chaque langage de programmation utilisé dans le modèle, pour tous les utilisateurs travaillant sur le modèle, définissez des classes de collection pour générer du code à partir de connecteurs d'association où le rôle cible a un paramètre de multiplicité supérieur à 1
- Définissez un chemin local pour vous-même, à l'aide de la dialogue « Chemin local » ; ces paramètres s'appliquent à tous les modèles Enterprise Architect auxquels vous accédez
- Définir des macros de langage dans le modèle, qui sont utiles dans la rétro-ingénierie et peuvent être exportées et importées vers le modèle

# Options d'ingénierie du code source

Les options « Ingénierie du code source » s'appliquent aux langages dans lesquels vous générez du code à partir d' Enterprise Architect . Elles sont divisées en options spécifiques au modèle et options spécifiques à l'utilisateur, comme expliqué ici.

## Options spécifiques au modèle

Ces options sont définies dans la dialogue « Gérer les options Modèle ».

## Accéder

| | |
|---|---|
| Ruban | Paramètres > Modèle > Options > Ingénierie du code source |

## Types d'options

| Type d'option | Détail |
|---|---|
| Options de génération de code source | Vous pouvez définir un certain nombre de paramètres pour générer du code dans le modèle, tels que la langue par défaut dans laquelle générer du code et le jeu de caractères Unicode pour la génération de code. |
| Options - Durée de vie Object | Vous pouvez configurer diverses options concernant la durée de vie Object . |
| Options de langage de code | Pour chacun des langages de code supporte Enterprise Architect , vous pouvez définir les options spécifiques au modèle et définir les classes de collection requises. |

## Options spécifiques à l'utilisateur

Ces options sont définies dans la dialogue « Préférences ».

## Accéder

Dans la dialogue « Préférences », cliquez sur « Ingénierie du code source » dans la liste de gauche.

| | |
|---|---|
| Ruban | Démarrer > Apparence > Préférences > Préférences |
| Raccourcis Clavier | Ctrl+F9 |

## Types d'options

| Type d'option | Détail |
|---|---|
| Options de génération de code source | Vous pouvez définir un certain nombre de paramètres pour générer du code dans n'importe quel modèle auquel vous accédez sous le même ID utilisateur. |
| Éditeurs de Code | Voici les options permettant d'accéder et de configurer l'éditeur de code source. |
| Attributes / Opérations | Utilisez ces options pour configurer les attributs et les opérations. |
| Options de langage de code | Pour chacun des langages de code supporte Enterprise Architect , vous pouvez définir les options spécifiques à l'utilisateur qui s'appliquent à tout modèle auquel vous accédez sous votre ID utilisateur. |

# Options de génération de code

Lorsque vous générez du code pour votre modèle, vous pouvez définir certaines options. Celles-ci incluent :

- La langue par défaut
- S'il faut générer des méthodes pour les interfaces implémentées
- Les options Unicode pour la génération de code

## Accéder

| | |
|---|---|
| Ruban | Paramètres > Modèle > Options > Ingénierie du code source |

## Configurer les options de génération de code

| Option | Action |
|---|---|
| Toujours synchroniser avec le fichier existant (recommandé) | Sélectionnez le bouton radio pour synchroniser le code importé avec un fichier existant. |
| Remplacer (écraser) le fichier source existant | Sélectionnez le bouton radio pour écraser le fichier source existant avec le code importé. |
| Types de composants | Cliquez sur ce bouton pour ouvrir la dialogue « Importer les types de composants » afin de configurer l'importation des types de composants. |
| Langue par défaut pour la génération de code | Cliquez sur la flèche déroulante et sélectionnez la langue par défaut pour la génération de code. |
| Nom DDL Gabarits | Cliquez sur le bouton ... pour définir les noms gabarit pour les gabarits Primary Key , de contrainte unique, Foreign Key et de nom d'index Foreign Key . |
| Nom par défaut pour l'attribut associé | Type un nom par défaut à générer à partir des attributs importés. |
| Générer des méthodes pour les interfaces implémentées | Cochez la case pour indiquer que les méthodes sont générées pour les interfaces implémentées. |
| Page de code pour l'édition de la source | Cliquez sur la flèche déroulante et sélectionnez le format d’intégration de caractères Unicode approprié à appliquer. |

## Notes

- Il est utile de configurer ces paramètres, car ils servent de valeurs par défaut pour toutes les classes du modèle ; vous pouvez remplacer la plupart d'entre eux pour chaque classe à l'aide des paramètres personnalisés (à partir de la dialogue « Génération de code »).

# Types de composants d'importation

À l'aide de la dialogue « Importer les types de composants », vous pouvez configurer les éléments que vous souhaitez créer pour les fichiers de n'importe quelle extension trouvés lors de l'importation d'un répertoire de code source.

## Accéder

| | |
|---|---|
| Ruban | Paramètres > Modèle > Options > Ingénierie du code source : Types de composants |

## Définir les types de composants d'importation

| Option | Action |
|---|---|
| Extension | Type le nom de l'extension pour un type de composant. |
| Type | Cliquez sur la flèche déroulante et sélectionnez le type de composant. |
| Stéréotype | Type n'importe quel nom de stéréotype qui identifie davantage un composant de ce type. |
| Liste des composants | Répertorie les types de composants actuellement définis. |
| Sauvegarder | Cliquez sur ce bouton pour enregistrer la définition du composant et l'ajouter à la liste des composants. |
| Nouveau | Cliquez sur ce bouton pour effacer le dialogue afin de pouvoir définir un nouveau type de composant. |
| Supprimer | Cliquez sur ce bouton pour supprimer le type de composant sélectionné de la liste des composants. |

## Notes

- Vous pouvez transporter ces types de composants d'importation entre les modèles, en utilisant les options de ruban « Paramètres > Modèle > Transférer > Exporter les données de référence » et « Importer les données de référence ».

# Options du code source

Vous pouvez définir un large éventail d'options pour générer du code dans les modèles avec lesquels vous travaillez. Celles-ci incluent :

- Comment formater le code généré
- Comment réagir à certains événements lors de la génération de code
- Faut-il générer un diagramme à partir du code ?

## Accéder

Dans la dialogue « Préférences », sélectionnez l'option « Ingénierie du code source »

| | |
|---|---|
| Ruban | Démarrer > Apparence > Préférences > Préférences |
| Raccourcis Clavier | Ctrl+F9 |

## Configurer les options de génération de code

| Champ | Action |
|---|---|
| Envelopper les longues lignes de commentaires à | Type le nombre de caractères à autoriser dans une ligne de commentaire avant de renvoyer le texte à la ligne suivante. |
| Diagramme Disposition automatique à l'importation | Cliquez sur la flèche déroulante et sélectionnez si et quand un diagramme est automatiquement généré lors de l'importation de code. |
| Type de Diagramme Disposition par défaut | Cliquez sur la flèche déroulante et sélectionnez le type disposition à appliquer aux diagrammes générés à partir du code. |
| Les fichiers de sortie utilisent à la fois CR et LF | Cochez la case pour inclure les retours chariot et les sauts de ligne ; définissez cette option en fonction du système d'exploitation actuellement utilisé, car le code peut ne pas s'afficher correctement. |
| Invite lors de la synchronisation (inversion) | Cochez la case pour afficher une prompt lorsque la synchronisation se produit. |
| Supprimer les sauts de ligne des commentaires lors de l'importation | Cochez la case pour supprimer les sauts de ligne des sections commentées lors de l'importation. |
| Générer automatiquement les noms de rôle lors de la création du code | Cochez la case pour générer des noms de rôle lors de la création de code. |
| Ne pas générer de membres lorsque la direction de l'association est « Non | Cochez la case pour empêcher la génération de membres si la direction de l'association n'est pas spécifiée. |

| | |
|---|---|
| spécifiée » | |
| Créer des dépendances pour les retours d'opération et les types de paramètres | Cochez la case pour générer des dépendances pour les retours d'opération et les types de paramètres. |
| Commentaires : Générer | Cochez la case pour générer des commentaires. |
| Commentaires : Inversé | Cochez la case pour générer des commentaires inversés. |
| Supprimer les préfixes lors de la génération des propriétés Get/Set | Type les préfixes, séparés par des points-virgules, utilisés dans vos conventions de nommage de variables, à supprimer dans les fonctions get/set correspondantes des variables. |
| Traiter comme des suffixes | Cochez la case pour utiliser les préfixes définis dans le champ « Supprimer les préfixes lors de la génération des propriétés Get/Set » comme suffixes. |
| Nom d'attribut en majuscule pour Propriétés | Cochez la case pour mettre en majuscule les noms d'attributs des propriétés. |
| Utiliser « Is » pour la propriété booléenne Get() | Cochez la case pour utiliser le mot-clé Is pour la propriété booléenne Get(). |

## Notes

- Il est utile de configurer ces paramètres, car ils servent de valeurs par défaut pour toutes les classes du modèle ; vous pouvez remplacer la plupart d'entre eux pour chaque classe à l'aide des paramètres personnalisés (à partir de la dialogue « Génération de code »).

# Options - Éditeurs de Code

Vous pouvez accéder aux options de l'éditeur de code source via la page « DDL » de la dialogue « Préférences ». Sur cette page, vous pouvez configurer les options de l'éditeur interne d' Enterprise Architect , ainsi que l'éditeur par défaut pour les scripts DDL. Vous pouvez configurer des éditeurs externes pour les langages de code sur chaque page d'options de langage.

## Accéder

Dans la dialogue « Préférences », sélectionnez l'option « Ingénierie du code source > Éditeurs de Code ».

| | |
|---|---|
| Ruban | Démarrer > Apparence > Préférences > Préférences |
| Raccourcis Clavier | Ctrl+F9 |

## Options

| Option | Action |
|---|---|
| Éditeur DDL | La valeur par défaut est vide, indiquant que l'éditeur de code Enterprise Architect est l'éditeur DDL utilisé.<br>Vous pouvez sélectionner un autre éditeur par défaut si nécessaire ; cliquez sur le bouton  pour rechercher et sélectionner l'éditeur DDL requis. Le nom de l'éditeur s'affiche alors dans le champ « Éditeur DDL ». |
| Base de données par défaut | Cliquez sur la flèche déroulante et sélectionnez la base de données par défaut à utiliser. |
| Moteur de stockage MySQL | Cliquez sur la flèche déroulante et sélectionnez le moteur de stockage MySQL à utiliser. |
| Utiliser l'éditeur intégré si aucun éditeur externe n'est défini | Cochez la case pour utiliser l'éditeur intégré pour le code dans n'importe quelle langue si aucun éditeur externe n'est défini pour cette langue dans les options spécifiques à l'utilisateur. |
| Afficher les numéros de ligne | Cochez la case pour afficher les numéros de ligne dans l'éditeur. |
| Afficher l'arborescence de la structure | Cochez la case pour afficher un arbre avec les résultats de l'analyse du fichier ouvert (si le fichier est analysé avec succès). |
| Rétro-ingénierie automatique lors de l'enregistrement du fichier | Si vous cochez cette case, appuyez sur Ctrl+S pour enregistrer dans l'éditeur de code source pour effectuer automatiquement une rétro-ingénierie du code de la même manière que le bouton Enregistrer la source et resynchroniser la classe. |
| N'analysez pas les fichiers plus grands que | Cliquez sur la flèche déroulante et sélectionnez la limite supérieure de la taille du fichier pour l'analyse.<br>La définition de cette option empêche la diminution des performances due à |

| | |
|---|---|
| | l'analyse de fichiers très volumineux. |
| Police , style et mise en évidence de la syntaxe | Cliquez sur le bouton  pour afficher la dialogue « Propriétés de la langue de l'éditeur », dans laquelle vous pouvez définir les propriétés de la langue de l'éditeur globales et spécifiques à la langue. |
| Configurer les associations de fichiers Enterprise Architect | Cliquez sur le bouton  pour afficher la dialogue « Définir des associations pour un programme » et sélectionnez les extensions de fichier pour les fichiers que vous souhaitez ouvrir via le gestionnaire de documents Enterprise Architect . |

# Propriétés de la langue de l'éditeur

À l'aide de la dialogue « Propriétés du langage de l'éditeur », vous pouvez spécifier les propriétés de mise en surbrillance de la syntaxe pour n'importe lequel des langages de programmation supporte par Enterprise Architect lors de l'installation.

## Accéder

Dans la dialogue « Préférences », sélectionnez l'option « Ingénierie du code source | Éditeurs de Code » et cliquez sur le bouton  à côté de « Options de mise en évidence de la syntaxe ».

| | |
|---|---|
| Ruban | Démarrer > Apparence > Préférences > Préférences, sélectionnez l'option « Source Code Engineering \| Éditeurs de Code » > cliquez sur le bouton  à côté de « Options de mise en surbrillance de la syntaxe » |
| Autre | Dans la fenêtre Éditeur de Code , cliquez sur l'icône de la barre d'outils  \| Options de mise en surbrillance de la syntaxe |

## Options

| Panneau | Description |
|---|---|
| Panneau de langue | Le panneau sur la gauche de le dialogue répertorie les langues pour lesquelles vous pouvez définir des propriétés.<br>En haut de la liste se trouvent trois options non linguistiques :<br>• (Thème sombre) - attribue un arrière-plan sombre aux champs de propriété et au panneau de code dans l'écran de l'éditeur de code (vous pouvez appliquer une couleur différente à des propriétés spécifiques)<br>• (Thème clair) - attribue un arrière-plan pâle aux champs de propriété et au panneau de code dans l'écran de l'éditeur de code (vous pouvez appliquer une couleur différente à des propriétés spécifiques)<br>Vous pouvez également définir les thèmes d'arrière-plan dans la dialogue « Apparence de l'application »<br>• (Global) fournit des propriétés que vous pouvez définir pour tous les langages de programmation ; cependant, vous pouvez réinitialiser une propriété globale à une valeur différente pour un langage particulier, dans les propriétés spécifiques à ce langage<br>La réinitialisation d'une propriété globale pour une langue n'affecte pas valeur de cette propriété pour les autres langues<br>Cliquez sur la langue souhaitée dans la liste pour afficher les propriétés de cette langue :<br>• Propriétés affichées en gras indiquent qu'il s'agit du niveau le plus élevé auquel cette propriété peut être définie (pour la plupart des options de langage autres que « Global », il s'agit en fait du seul point auquel la propriété est définie)<br>• Propriétés affichées en police normale sont généralement les propriétés globales que vous pouvez réinitialiser uniquement pour la langue actuelle |
| | |

| | |
|---|---|
| Panneau Propriétés | Faites défiler les catégories de propriétés et les propriétés individuelles de la langue. Vous pouvez réduire et développer les catégories selon vos besoins, à l'aide de la zone d'extension située à côté du nom de la catégorie ( ).<br>Lorsque vous cliquez sur le nom d'une propriété, une explication de cette propriété s'affiche dans le panneau en bas à droite de le dialogue .<br>Pour définir une propriété, cliquez sur le champ valeur suivant le nom de la propriété ; selon le type de propriété, soit le champ est activé pour l'édition directe, soit une flèche déroulante ou un bouton  s'affiche (comme décrit pour l'onglet 'Tags' de la fenêtre Propriétés ) afin que vous puissiez sélectionner les valeurs pour définir la propriété.<br>Sélectionnez ou saisissez les valeurs requises.<br>Utilisez les icônes de la barre d'outils pour :<br>• Enregistrez vos modifications dans les propriétés<br>• Réinitialiser tous les champs de propriétés aux paramètres par défaut fournis avec Enterprise Architect<br>• Réinitialiser le champ de style actuel au paramètre par défaut (non activé pour les champs sans style) |
| Affecter des touches aux macros | Dans la catégorie 'Macros' des propriétés, vous pouvez attribuer des combinaisons de touches (Ctrl+Alt+<n>) à des macros de codage que vous avez créées vous-même dans la ' Visionneuse de code source '.<br>Lorsque vous cliquez sur le bouton Parcourir dans un champ « Macro » sélectionné, la dialogue « Ouvrir une macro » s'affiche ; cette dialogue répertorie les macros existantes et, si une combinaison de touches a été attribuée à une macro, quelle est cette combinaison de touches.<br>Cliquez sur le nom de la macro et sur le bouton Ouvrir pour attribuer les touches sélectionnées à la macro. |

## Notes

- Vous ne pouvez actuellement pas définir de propriétés pour les langues supplémentaires que vous incluez via une MDG Technologie
- Vous pouvez redimensionner cette dialogue , si nécessaire

# Options - Durée de vie Object

Vous pouvez utiliser ces options pour configurer divers paramètres de durée de vie Object tels que :

- Définition des détails du constructeur lors de la génération de code
- Spécifier s'il faut créer un constructeur de copie
- Définition des détails du destructeur

## Accéder

| | |
|---|---|
| Ruban | Paramètres > Modèle > Options > Ingénierie du code source > Durée de vie Object |

## Options

| Option | Action |
|---|---|
| Constructeur | Si nécessaire, cochez les cases pour spécifier qu'un constructeur est généré et (pour C++) que le constructeur est en ligne.<br>Cliquez sur la flèche déroulante et sélectionnez la visibilité appropriée du constructeur par défaut : Privé, Protégé ou Public. |
| Constructeur de copie | Si nécessaire, cochez les cases pour spécifier qu'un constructeur de copie est généré et (pour C++) que le constructeur de copie est en ligne.<br>Cliquez sur la flèche déroulante et sélectionnez la visibilité appropriée du constructeur de copie par défaut : Privé, Protégé ou Public. |
| Destructeur | Si nécessaire, cochez les cases pour spécifier qu'un destructeur est généré et (pour C++) que le destructeur est en ligne et/ou virtuel.<br>Cliquez sur la flèche déroulante et sélectionnez la visibilité appropriée du destructeur par défaut - Privé, Protégé ou Public. |

# Options - Attribut/Opérations

L'utilisation des attributs et des opérations peut être configurée de plusieurs manières. Vous pouvez définir des options pour :

- Supprimer les attributs du modèle non inclus dans le code lors de la synchronisation inverse
- Supprimer les méthodes de modèle non incluses dans le code lors de la synchronisation inverse
- Supprimer le code des fonctionnalités contenues dans le modèle lors de la synchronisation directe
- Supprimer les associations et agrégations de modèles qui correspondent à des attributs non inclus dans le code lors de la synchronisation inverse
- Définir si les corps des méthodes sont inclus et enregistrés dans le modèle lors de la rétro-ingénierie
- Créez fonctionnalités en succession rapide, en effaçant la fenêtre Propriétés lorsque vous cliquez sur « Enregistrer » afin de pouvoir saisir un autre nom fonctionnalité

Vous configurez ces options sur la page « Attribut/Opérations » de la dialogue « Préférences ».

## Accéder

Dans la dialogue « Préférences », sélectionnez l'option « Ingénierie du code source > Attribut/Opérations ».

| | |
|---|---|
| Ruban | Démarrer > Apparence > Préférences > Préférences |
| Raccourcis Clavier | Ctrl+F9 |

## Options

| Champ | Action |
|---|---|
| Lors de la synchronisation inverse, supprimez les attributs du modèle qui ne figurent pas dans le code | Cochez la case pour indiquer que lors de la synchronisation inverse, les attributs du modèle qui ne sont pas inclus dans le code sont automatiquement supprimés du modèle. |
| Lors de la synchronisation inverse, supprimez les associations de modèles qui ne figurent pas dans le code | Cochez la case pour indiquer que lors de la synchronisation inverse, les associations dans le modèle qui ne sont pas incluses dans le code sont automatiquement supprimées du modèle. |
| Lors de la synchronisation inverse, supprimez les méthodes de modèle qui ne figurent pas dans le code | Cochez la case pour indiquer que lors de la synchronisation inverse, les méthodes du modèle qui ne sont pas incluses dans le code sont automatiquement supprimées du modèle. |
| Inclure les corps de méthode dans le modèle lors de la rétro-ingénierie | Cochez la case pour indiquer que dans le code de rétro-ingénierie, les corps de méthode du code sont inclus dans votre modèle. |
| Après avoir enregistré, resélectionnez l'élément | Cochez la case pour indiquer qu'après l'enregistrement d'un attribut ou d'une opération, la définition des propriétés continue d'afficher les détails de la |

| | |
|---|---|
| modifié | fonctionnalité sélectionnée.<br>Si cette option est désélectionnée, cela indique que les champs de la définition des propriétés seront effacés afin que vous puissiez saisir immédiatement un autre nom et des détails d'attribut ou d'opération. |
| Lors de la synchronisation directe, prompt à supprimer fonctionnalités de code non présentes dans le modèle | Cochez la case pour indiquer que, lors de la synchronisation directe, la dialogue « Synchroniser l'élément < nom paquetage >.< nom de l'élément> » s'affiche, afin que vous puissiez ignorer, réaffecter ou supprimer fonctionnalités du code qui ne sont pas dans le modèle. |

# Conventions Modélisation

La synchronisation entre les modèles UML et le code de programmation est réalisée à l'aide d'un ensemble de conventions modélisation (mappings) entre les constructions UML et la syntaxe du code de programmation. Il est conseillé à l'ingénieur logiciel de se familiariser avec ces conventions afin de travailler avec le processus de génération de code pour les langages de programmation qu'il souhaite cibler. Il existe toute une gamme de constructions utilisées, notamment les éléments, fonctionnalités , les connecteurs, les extrémités de connecteur, les stéréotypes et Valeur Étiquetés . Le nouveau venu aura besoin d'un peu de temps pour se familiariser avec ces conventions, mais après peu de temps, il saura traduire sans effort entre le code de programmation et les constructions UML .

## Langues prises en charge

| Langue |
| --- |
| Scénario Action |
| Ada 2012 (éditions Unified et Ultimate ) |
| C |
| C# |
| C++ |
| Delphes |
| Java |
| PHP |
| Python |
| SystemC (éditions Unified et Ultimate ) |
| Verilog (éditions Unified et Ultimate ) |
| VHDL (éditions Unified et Ultimate ) |
| Visual Basic |
| Visual Basic .NET |

## Notes

Enterprise Architect intègre un certain nombre d'indicateurs de visibilité ou de valeurs de portée pour les langues prises en charge ; il s'agit notamment de :

- Toutes les langues - Public (+), Protégé (#) et Privé (-)
- Java - Paquetage (~)
- Delphi - Publié (^)
- C# - Interne (~), Interne protégé (^)
- ActionScript - Interne (~)
- VB.NET - Ami (~), ami protégé (^)
- PHP - Paquetage (~)
- Python - Paquetage (~)
- C - Paquetage (~)
- C++ - Paquetage (~)

# Conventions ActionScript

Enterprise Architect supporte l'ingénierie round d'ActionScript 2 et 3, lorsque ces conventions sont utilisées.

## Stéréotypes

| Stéréotype | S'applique à |
|---|---|
| littéral | Opération<br>Correspond à : une méthode littérale référencée par une variable. |
| obtenir une propriété | Opération<br>Correspond à : une propriété « lecture ». |
| ensemble de propriétés | Opération<br>Correspond à : une propriété « écriture ». |

## Valeur Étiquetés

| Étiquette | S'applique à |
|---|---|
| nom_attribut | Opération avec la propriété stéréotype get ou property set<br>Correspond à : Le nom de la variable derrière cette propriété. |
| dynamique | Classe ou interface<br>Correspond à : Le mot-clé « dynamique ». |
| final | ActionScript 3 : opération<br>Correspond à : Le mot-clé « final ». |
| intrinsèque | ActionScript 2 : classe<br>Correspond à : Le mot-clé « intrinsèque ». |
| espace de noms | ActionScript 3 : classe, interface, attribut, opération<br>Correspond à : l'espace de noms de l'élément actuel. |
| outrepasser | ActionScript 3 : opération<br>Correspond à : Le mot-clé « override ». |
| prototype | ActionScript 3 : Attribut<br>Correspond à : Le mot-clé « prototype ». |
| repos | ActionScript 3 : Paramètre<br>Correspond à : Le paramètre de repos (...) |

## Conventions communes

- Les qualificateurs Paquetage (ActionScript 2) et Paquetages (ActionScript 3) sont générés lorsque le Paquetage actuel n'est pas une racine d'espace de noms
- Un type non spécifié est modélisé comme « var » ou un champ « Type » vide

## Conventions d'ActionScript 3

- La propriété Is Leaf d'une classe correspond au mot-clé sealed
- Si une étiquette d'espace de noms est spécifiée, elle remplace la portée spécifiée

# Conventions Ada 2012

Enterprise Architect supporte l'ingénierie round retour d'Ada 2012, où ces conventions sont utilisées.

## Stéréotypes

| Stéréotype | S'applique à |
|---|---|
| Paquet ada | Classe<br>Correspond à : une spécification Paquetage dans Ada 2012 sans enregistrement balisé. |
| Procédure ada | Classe<br>Correspond à : une spécification de procédure dans Ada 2012. |
| déléguer | Opération<br>Correspond à : Accès à un sous-programme. |
| énumération | Classe intérieure<br>Correspond à : un type énuméré. |
| structure | Classe intérieure<br>Correspond à : une définition d'enregistrement. |
| définition de type | Classe intérieure<br>Correspond à : une définition de type, une définition de sous-type, une définition de type d'accès, un renommage. |

## Valeur Étiquetés

| Étiquette | S'applique à |
|---|---|
| Aspect | Classe interne avec typedef stéréotypé<br>Opération<br>Correspond à : Spécification d'aspect (Précondition et Postcondition de type de sous-programme 'invariant', sous-type 'prédicat'). |
| Type d'unité instanciée | Classe interne avec typedef stéréotypé<br>Correspond à : Le type de l'unité instanciée ( Paquetage / Procedure / Function). |
| Est-ce que Access | Paramètre<br>Correspond à : Déterminer si le paramètre est une variable d'accès. |
| Est aliasé | Paramètre de fonction |

| | |
|---|---|
| | Correspond à : Paramètre de fonction aliasé. |
| Discriminant | Classe interne avec typedef stéréotypé<br>Correspond à : Le discriminant du type. |
| Type de pièce | Classe interne avec typedef stéréotypé<br>Correspond à : Le type de pièce (« renomme » ou « nouveau »). |
| Type | Classe interne avec typedef stéréotypé<br>Correspond à : Si « Valeur » = « Sous-type », définir « sous-type »<br>Si « Valeur » = « Accès », définissez « type d'accès ». |

## Autres conventions

- Type approprié de fichiers sources : fichier de spécification Ada, .ads
- Ada 2012 importe Paquetages définis comme classe <<adaPackage>> ou classe, en fonction des paramètres des options Ada 2012
- Un Paquetage dans le fichier de spécifications Ada est importé en tant que Classe s'il contient un enregistrement balisé, dont le nom est régi par les options « Utiliser le nom de classe pour l'enregistrement balisé » et « Nom d'enregistrement balisé alternatif » ; tous les attributs définis dans cet enregistrement balisé sont absorbés en tant qu'attributs de la classe
- Une procédure/fonction dans un fichier de spécification Ada est considérée comme la fonction membre de la classe si son premier paramètre satisfait les conditions spécifiées dans les options « Style de paramètre de référence », « Ignorer le nom du paramètre de référence » et « Nom du paramètre de référence »
- L'option « Définir une référence pour un enregistrement balisé », si elle est activée, crée un type de référence pour la classe, dont le nom est déterminé par l'option « Nom Type de référence » ; par exemple :

Bonjour le monde.annonces

paquetage HelloWorld est

le type HelloWorld est un enregistrement tagué

Att1 : Naturel ;

Att3 : Integer ;

fin d'enregistrement;

-- Fonctions publiques

fonction MyPublicFunction (P: HelloWorld) renvoie String ;

procédure MyPublicFunction (P1 : entrée sortie HelloWorld ; AFlag : Boolean) ;

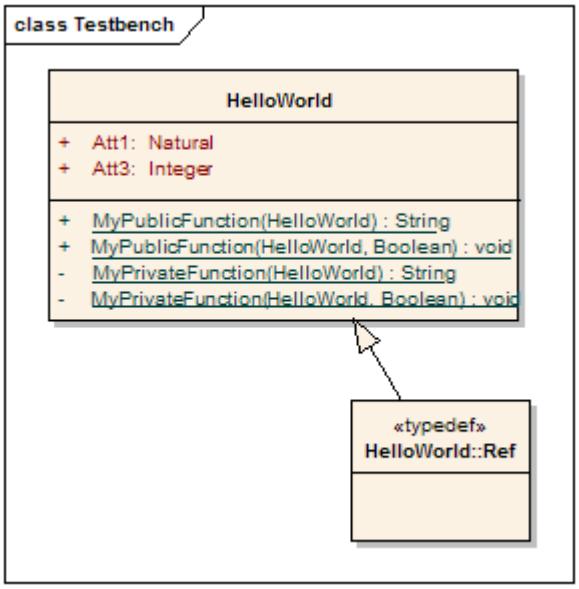
privé

-- Fonctions privées

fonction MyPrivateFunction (P: HelloWorld) renvoie String ;

procédure MyPrivateFunction (P1 : entrée sortie HelloWorld ; AFlag : Boolean) ;

fin HelloWorld;




## Notes

- support Ada 2012 est disponible dans les éditions Unified et Ultimate d' Enterprise Architect

# Conventions C

Enterprise Architect supporte l'ingénierie round retour de C, où ces conventions sont utilisées :

## Stéréotype

| Stéréotype | S'applique à |
|---|---|
| énumération | Classe intérieure<br>Correspond à : un type énuméré. |
| structure | Classe intérieure<br>Correspond à : un type « struct ». |
| Attribut | Une structure de mot-clé dans la définition de variable. |
| définition de type | Classe intérieure<br>Correspond à : une instruction « typedef », où le parent est le nom du type d'origine. |
| union | Classe intérieure<br>Correspond à : Un type d'union. |
| Attribut | Une union de mots clés dans la définition de variable. |

## Valeur Étiquetés

| Étiquette | S'applique à |
|---|---|
| anonyme | Classe contenant également le typedef Valeur Étiquetée<br>Correspond à : Le nom de cette classe étant défini uniquement par l'instruction typedef. |
| champ de bits | Attribut<br>Correspond à : la taille, en bits, autorisée pour le stockage de cet attribut. |
| Emplacement du corps | Opération<br>Correspond à : l'emplacement vers lequel le corps de la méthode est généré ; les valeurs attendues sont header, classDec ou classBody. |
| définition de type | Classe avec un stéréotype autre que « typedef »<br>Correspond à : Cette classe étant définie dans une instruction « typedef ». |
| typeSynonymes | Classe<br>Correspond à : le nom « typedef » et/ou les champs de ce type. |

## Génération de code C pour Modèle UML

| UML | Code C |
|---|---|
| Une classe | Une paire de fichiers C (.h + .c)<br>Notes : le nom du fichier est le même que le nom de la classe |
| Opération (publique et protégée) | Déclaration de fonction dans le fichier .h et définition dans le fichier .c<br>Notes : |
| Opération (privée) | Définition de fonction dans le fichier .c uniquement<br>Notes : |
| Fonctionnement (statique) | Définition de fonction dans le fichier .c uniquement<br>Notes : les fonctions statiques n'apparaîtront que dans le fichier .c, quelle que soit leur portée. |
| Attribut (public et protégé) | Définition de variable dans le fichier .h<br>Notes : |
| Attribut (privé) | Définition de variable dans le fichier .c<br>Notes : |
| Classe intérieure (sans stéréotype) | (N / A)<br>Notes : Cette classe interne serait ignorée |

## Capturez valeur #define à générer dans le code C

Par exemple, #define PI 3.14.

| Étape | Processus |
|---|---|
| 1 | Ajoutez un attribut à la classe, avec le nom = PI et la valeur initiale = 3,14. |
| 2 | Dans le panneau des propriétés de la page « Attributes », mettez à jour les champs « Statique » et « Const ». |
| 3 | Sur l'onglet ' Valeur Étiquetés ' de la page ' Attributes ', ajoutez une étiquette appelée 'define' avec la valeur True. |

## Notes

- Des conventions distinctes s'appliquent à la programmation orientée Object en C

# Programmation orientée Object en C

Dans Enterprise Architect , vous appliquez un certain nombre de conventions pour la programmation orientée objet en C.

Pour configurer le système afin de support de C, vous devez définir l'option « Support orientée Object » sur True dans la page « Spécifications C » de la dialogue « Préférences ».

## Stéréotypes

| Stéréotype | S'applique à |
|---|---|
| énumération | Classe<br>Correspond à : un type énuméré. |
| structure | Classe<br>Correspond à : un type « struct ». |
| Attribut | Une structure de mot-clé dans la définition de variable. |
| définition de type | Classe<br>Correspond à : une instruction « typedef », où le parent est le nom du type d'origine. |
| union | Classe<br>Correspond à : Un type d'union. |
| Attribut | Une union de mots clés dans la définition de variable. |

## Valeur Étiquetés

| Étiquette | S'applique à |
|---|---|
| anonyme | Classe avec le stéréotype « enumeration », « struct » ou « union »<br>Correspond à : Le nom de cette classe étant défini uniquement par l'instruction typedef. |
| Emplacement du corps | Opération<br>Correspond à : l'emplacement où le corps de la méthode est généré ; les valeurs attendues sont « header », « classDec » ou « classBody ». |
| définir | Attribut<br>Correspond à : instruction « #define ». |
| définition de type | Classe avec le stéréotype « enumeration », « struct » ou « union »<br>Correspond à : Cette classe étant définie dans une instruction « typedef ». |

## Génération de code C orienté objet pour Modèle UML

L'idée de base de l'implémentation d'une classe UML dans le code C est de regrouper les variables de données (attributs UML ) dans un type de structure ; cette structure est définie dans un fichier .h afin qu'elle puisse être partagée par d'autres classes et par le client qui y fait référence.

Une opération dans une classe UML est implémentée dans le code C en tant que fonction ; le nom de la fonction doit être un nom entièrement qualifié composé du nom de l'opération, ainsi que du nom de la classe pour indiquer que l'opération est destinée à cette classe.

Un délimiteur (spécifié dans l'option « Délimiteur Namespace » sur la page « Spécifications C ») est utilisé pour joindre le nom de la classe et le nom de la fonction (opération).

La fonction dans le code C doit également avoir un paramètre de référence à l' object Class - vous pouvez modifier les options « Référence comme paramètre d'opération », « Style de paramètre de référence » et « Nom du paramètre de référence » sur la page « Spécifications C » pour support ce paramètre de référence.

## Limitations de la programmation orientée objet en C

- Pas de mappage de portée pour un attribut : un attribut dans une classe UML est mappé à une variable de structure dans le code C et sa portée (privée, protégée ou publique) est ignorée
- Actuellement, une classe interne est ignorée : si une classe UML est la classe interne d'une autre classe UML , elle est ignorée lors de la génération du code C
- valeur initiale est ignorée : la valeur initiale d'un attribut dans une classe UML est ignorée dans le code C généré

# Conventions C#

Enterprise Architect supporte l'ingénierie round -retour de C# , où ces conventions sont utilisées.

## Stéréotypes

| Stéréotype | S'applique à |
|---|---|
| énumération | Classe<br>Correspond à : un type énuméré. |
| événement | Opération<br>Correspond à : Un événement. |
| extension | Opération<br>Correspond à : une méthode d'extension de classe, représentée dans le code par un paramètre « this » dans la signature. |
| indexeur | Opération<br>Correspond à : une propriété agissant comme un index pour cette classe. |
| partiel | Opération<br>Correspond à : Le mot clé « partial » sur une opération. |
| propriété | Opération<br>Correspond à : une propriété contenant éventuellement du code de lecture et d'écriture. |
| enregistrer | Classe<br>Correspond à : un type « enregistrement ». |
| structure_enregistrement | Classe<br>Correspond à : Un type « record struct ». |
| structure | Classe<br>Correspond à : un type « struct ». |

## Valeur Étiquetés

| Étiquette | S'applique à |
|---|---|
| argumentNom | Opération avec extension du stéréotype<br>Correspond à : Le nom donné à ce paramètre. |
| | |

| | |
|---|---|
| nom_attribut | Opération avec une propriété ou un événement stéréotypé<br>Correspond à : Le nom de la variable derrière cette propriété ou cet événement. |
| nom de la classe | Opération avec extension du stéréotype<br>Correspond à : la classe à laquelle cette méthode est ajoutée. |
| constante | Attribut<br>Correspond à : le mot-clé const. |
| définition | Opération avec stéréotype partiel<br>Correspond à : s'il s'agit de la déclaration de la méthode ou de la définition. |
| déléguer | Opération<br>Correspond à : Le mot-clé « délégué ». |
| enumType | Opération avec propriété de stéréotype<br>Correspond à : le type de données sous lequel la propriété est représentée. |
| expressionCorps | Opération, Opération avec propriété stéréotype ou indexeur<br>Correspond à : « Vrai » si le « Code de comportement » provient d'un membre de fonction du corps de l'expression. |
| extensionAttribut | Opération avec extension du stéréotype.<br>Correspond à : L'attribut donné à ce paramètre. |
| externe | Opération<br>Correspond à : Le mot-clé « extern ». |
| fixé | Attribut<br>Correspond à : Le mot-clé « fixe ». |
| générique | Opération<br>Correspond à : Les paramètres génériques pour cette opération. |
| contraintes génériques | Classe ou interface basée sur un modèle, opération avec étiquette « générique »<br>Correspond à : Les contraintes sur les paramètres génériques de ce type ou de cette opération. |
| Outils | Opération<br>Correspond à : Le nom de la méthode implémentée, y compris le nom de l'interface. |
| ImplémenteExplicit | Opération<br>Correspond à : La présence du nom de l'interface source dans cette déclaration de méthode. |
| initialiseur | Opération<br>Correspond à : une liste d’initialisation de constructeur. |
| nouveau | Classe, interface, fonctionnement |

| | |
|---|---|
| | Correspond à : Le mot-clé « nouveau ». |
| outrepasser | Opération<br>Correspond à : Le mot-clé « override ». |
| paramètres | Paramètre<br>Correspond à : une liste de paramètres utilisant le mot-clé « params ». |
| partiel | Classe, Interface<br>Correspond à : Le mot-clé « partial ». |
| Initialiseur de propriété | Opération avec propriété de stéréotype<br>Correspond à : un initialiseur de propriété. |
| lecture seule | Opération, classe <<struct>><br>Correspond à : Le mot-clé « readonly ». |
| Paramètres de position | Classe <<record>><br>Correspond à : Le paramètre de position dans la définition d'enregistrement. |
| réf | Opération, classe <<struct>><br>Correspond à : Le mot-clé « ref ». |
| scellé | Opération<br>Correspond à : Le mot-clé « scellé ». |
| statique | Classe<br>Correspond à : Le mot-clé « static ». |
| dangereux | Classe, Interface, Fonctionnement<br>Correspond à : Le mot clé « unsafe ». |
| virtuel | Opération<br>Correspond à : Le mot-clé « virtuel ». |
| écriture seule | Opération avec propriété de stéréotype<br>Correspond à : Cette propriété définit uniquement le code « écriture ». |

## Autres conventions

- Namespaces sont générés pour chaque Paquetage sous une racine d'espace de noms
- La propriété Const d'un attribut correspond au mot clé readonly, tandis que l' étiquette const correspond au mot clé const
- La valeur de inout pour la propriété Kind d'un paramètre correspond au mot-clé ref
- La valeur de out pour la propriété Kind d'un paramètre correspond au mot-clé out
- Les classes partielles peuvent être modélisées comme deux classes distinctes avec l' étiquette partielle

- La propriété Is Leaf d'une classe correspond au mot-clé sealed

# Conventions C++

Enterprise Architect supporte l'ingénierie round retour de C++, y compris les extensions Managed C++ et C++/CLI, où ces conventions sont utilisées.

## Stéréotypes

| Stéréotype | S'applique à |
|---|---|
| énumération | Classe<br>Correspond à : un type énuméré. |
| ami | Opération<br>Correspond à : Le mot-clé « ami ». |
| obtenir une propriété | Opération<br>Correspond à : une propriété « lecture ». |
| ensemble de propriétés | Opération<br>Correspond à : une propriété « écriture ». |
| structure | Classe<br>Correspond à : un type « struct ». |
| définition de type | Classe<br>Correspond à : une instruction « typedef », où le parent est le nom du type d'origine. |
| alias | Classe<br>Correspond à une déclaration « Alias », où le parent est le nom du type d'origine. |
| union | Classe<br>Correspond à : Un type d'union. |

## Valeur Étiquetés

| Étiquette | S'applique à |
|---|---|
| afx_msg | Opération<br>Correspond à : le mot-clé afx_msg. |
| anonyme | Classe contenant également le typedef Valeur Étiquetée<br>Correspond à : Le nom de cette classe étant défini uniquement par l'instruction typedef. |
| | |

| | |
|---|---|
| nom_attribut | Opération avec la propriété stéréotype get ou property set<br>Correspond à : Le nom de la variable derrière cette propriété. |
| champ de bits | Attribut<br>Correspond à : la taille, en bits, autorisée pour le stockage de cet attribut. |
| Emplacement du corps | Opération<br>Correspond à : l'emplacement vers lequel le corps de la méthode est généré ; les valeurs attendues sont header, classDec ou classBody. |
| rappel | Opération<br>Correspond à : une référence à la macro CALLBACK. |
| constpr | Attribut et fonctionnement<br>Correspond à : le mot-clé constexpr. |
| explicite | Opération<br>Correspond à : Le mot-clé « explicite ». |
| initialiseur | Opération<br>Correspond à : une liste d'initialisation de constructeur. |
| en ligne | Attribut et fonctionnement<br>Correspond à : le mot-clé « inline » et la génération en ligne de la définition de la variable membre et du corps de la méthode. |
| mutable | Attribut<br>Correspond à : Le mot-clé « mutable ». |
| portée | Classe avec énumération de stéréotypes<br>Correspond à : soit le mot-clé « class » soit le mot-clé « struct ». |
| lance | Opération<br>Correspond à : les exceptions levées par cette méthode. |
| définition de type | Classe avec un stéréotype autre que « typedef »<br>Correspond à : Cette classe étant définie dans une instruction « typedef ». |
| typeSynonymes | Classe<br>Correspond à : le nom « typedef » et/ou les champs de ce type. |
| volatil | Opération<br>Correspond à : Le mot-clé « volatile ». |

## Autres conventions

- Namespaces sont générés pour chaque Paquetage sous une racine d'espace de noms

- Les attributs par référence correspondent à un pointeur vers le type spécifié
- La propriété Transient d'un attribut correspond au mot clé volatile
- La propriété Abstract d'un attribut correspond au mot-clé virtual
- La propriété Const d'une opération correspond au mot-clé const, spécifiant un type de retour constant
- La propriété Is Query d'une opération correspond au mot-clé const, spécifiant que la méthode ne modifie aucun champ
- La propriété Pure d'une opération correspond à une méthode virtuelle pure utilisant la syntaxe "= 0"
- La propriété Fixed d'un paramètre correspond au mot-clé const

# Conventions C++ gérées

Ces conventions sont utilisées pour les extensions managées de C++ antérieures à C++/CLI. Pour configurer le système afin de générer du C++ managé, vous devez modifier la version C++ dans les options C++.

## Stéréotypes

| Stéréotype | S'applique à |
|---|---|
| propriété | Opération<br>Correspond à : Le mot-clé '__property'. |
| obtenir une propriété | Opération<br>Correspond à : le mot-clé '__property' et une propriété lue. |
| ensemble de propriétés | Opération<br>Correspond à : le mot-clé '_ _property' et une propriété 'write'. |
| référence | Classe<br>Correspond à : Le mot-clé « __gc ». |
| valeur | Classe<br>Correspond à : Le mot-clé « __value ». |

## Valeur Étiquetés

| Étiquette | S'applique à |
|---|---|
| Type géré | Classe avec référence stéréotypée, valeur ou énumération ; Interface<br>Correspond à : Le mot clé utilisé dans la déclaration de ce type ; les valeurs attendues sont « class » ou « struct ». |

## Autres conventions

- Les étiquettes typedef et anonymes du C++ natif ne sont pas prises en charge
- La propriété Pure d'une opération correspond au mot clé __abstract

# Conventions C++/CLI

Ces conventions sont utilisées pour modélisation des extensions C++/CLI vers C++. Pour configurer le système afin de générer du C++/CLI managé, vous devez modifier la version C++ dans les options C++.

## Stéréotypes

| Stéréotype | S'applique à |
|---|---|
| événement | Opération<br>Description : définit un événement pour fournir l'accès au gestionnaire d'événements pour cette classe. |
| propriété | Opération, Attribut<br>Description : Il s'agit d'une propriété contenant éventuellement du code de lecture et d'écriture. |
| référence | Classe<br>Description : Correspond au mot-clé « ref class » ou « ref struct ». |
| valeur | Classe<br>Description : Correspond au mot-clé « valeur class » ou « valeur struct ». |

## Valeur Étiquetés

| Étiquette | S'applique à |
|---|---|
| nom_attribut | Opération avec une propriété ou un événement stéréotypé<br>Description : Le nom de la variable derrière cette propriété ou cet événement. |
| générique | Opération<br>Description : Définit les paramètres génériques pour cette opération. |
| contraintes génériques | Classe ou interface modélisée, opération avec étiquette générique<br>Description : Définit les contraintes sur les paramètres génériques pour cette opération. |
| initonly | Attribut<br>Description : Correspond au mot-clé « initonly ». |
| littéral | Attribut<br>Description : Correspond au mot-clé littéral. |
| Type géré | Classe avec référence stéréotypée, valeur ou énumération ; Interface<br>Description : Correspond soit au mot-clé « class » soit au mot-clé « struct ». |

## Autres conventions

- Les étiquettes typedef et anonymes ne sont pas utilisées
- Les stéréotypes de propriété get/property set ne sont pas utilisés
- La propriété Pure d'une opération correspond au mot-clé abstract

# Conventions Delphi

Enterprise Architect supporte l'ingénierie round de Delphi, où ces conventions sont utilisées :

## Stéréotypes

| Stéréotype | S'applique à |
|---|---|
| constructeur | Opération<br>Correspond à : Un constructeur. |
| destructeur | Opération<br>Correspond à : Un destructeur. |
| interface d'affichage | Classe, Interface<br>Correspond à : une interface de répartition. |
| énumération | Classe<br>Correspond à : un type énuméré. |
| métaclasse | Classe<br>Correspond à : un type de métaclasse. |
| object | Classe<br>Correspond à : Un type object . |
| opérateur | Opération<br>Correspond à : Un opérateur. |
| obtenir une propriété | Opération<br>Correspond à : une propriété « lecture ». |
| ensemble de propriétés | Opération<br>Correspond à : une propriété « écriture ». |
| structure | Classe<br>Correspond à : un type d’enregistrement. |

## Valeur Étiquetés

| Étiquette | S'applique à |
|---|---|
| nom_attribut | Opération avec la propriété stéréotype get ou property set<br>Correspond à : Le nom de la variable derrière cette propriété. |

| | |
|---|---|
| surcharge | Opération<br>Correspond à : Le mot-clé « surcharge ». |
| outrepasser | Opération<br>Correspond à : Le mot-clé « override ». |
| emballé | Classe<br>Correspond à : Le mot-clé « packed ». |
| propriété | Classe<br>Correspond à : une propriété ; voir *Delphi Propriétés* pour plus d'informations. |
| réintroduire | Opération<br>Correspond à : Le mot-clé « réintroduire ». |

## Autres conventions

- La propriété Static d'un attribut ou d'une opération correspond au mot-clé 'class'
- La propriété Fixed d'un paramètre correspond au mot-clé 'const'
- La valeur de inout pour la propriété Kind d'un paramètre correspond au mot-clé 'Var'
- La valeur de out pour la propriété Kind d'un paramètre correspond au mot-clé 'Out'

# Conventions Java

Enterprise Architect supporte l'ingénierie round -retour de Java - y compris les extensions AspectJ - lorsque ces conventions sont utilisées.

## Stéréotypes

| Stéréotype | S'applique à |
|---|---|
| annotation | Interface<br>Correspond à : un type d'annotation. |
| Constructeur Compact | Opération<br>Correspond à : Un constructeur canonique compact pour l'enregistrement. |
| enregistrer | Classe<br>Correspond à : un type d'enregistrement. |
| défaut | Opération<br>Correspond à : Le mot clé « default ». |
| énumération | Attributes dans une énumération stéréotypée de classe<br>Correspond à : une option énumérée, distinguée des autres attributs qui n'ont pas de stéréotype. |
| énumération | Classe<br>Correspond à : un type énuméré. |
| opérateur | Opération<br>Correspond à : Un opérateur. |
| obtenir une propriété | Opération<br>Correspond à : une propriété « lecture ». |
| ensemble de propriétés | Opération<br>Correspond à : une propriété « écriture ». |
| statique | Classe ou interface<br>Correspond à : Le mot-clé « static ». |

## Valeur Étiquetés

| Étiquette | S'applique à |
|---|---|
| | |

| | |
|---|---|
| annotations | Rien<br>Correspond à : Les annotations sur la fonctionnalité de code actuelle. |
| arguments | Attribut avec énumération de stéréotypes<br>Correspond à : Les arguments qui s'appliquent à cette valeur énumérée. |
| nom_attribut | Opération avec la propriété stéréotype get ou property set<br>Correspond à : Le nom de la variable derrière cette propriété. |
| dynamique | Classe ou interface<br>Correspond à : Le mot-clé « dynamique ». |
| générique | Opération<br>Correspond à : Les paramètres génériques de cette opération. |
| non scellé | Classe, Interface<br>Correspond à : Le mot-clé « non scellé ». |
| permis | Classe, Interface<br>Correspond à : L'expression « permet ». |
| Liste de paramètres | Paramètre<br>Correspond à : une liste de paramètres avec la syntaxe .... |
| scellé | Classe, Interface<br>Correspond à : Le mot-clé « scellé ». |
| En-tête d'enregistrement | <<record>>Class<br>Correspond à : l'en-tête d'enregistrement de la définition d'enregistrement. |
| lance | Opération<br>Correspond à : les exceptions levées par cette méthode. |
| transitoire | Attribut<br>Correspond à : Le mot-clé « transient ». |

## Autres conventions

- Les instructions Paquetage sont générées lorsque le Paquetage actuel n'est pas une racine d'espace de noms
- La propriété Const d'un attribut ou d'une opération correspond au mot-clé final
- La propriété Transient d'un attribut correspond au mot clé volatile
- La propriété Fixed d'un paramètre correspond au mot clé final

# Conventions d'AspectJ

Ce sont les conventions utilisées pour prendre en charge les extensions AspectJ pour Java.

## Stéréotypes

| Stéréotype | S'applique à |
|---|---|
| conseil | Opération<br>Correspond à : Un conseil dans un aspect AspectJ. |
| aspect | Classe<br>Correspond à : Un aspectJ aspect. |
| coupe en pointe | Opération<br>Correspond à : un « pointcut » dans un aspect AspectJ. |

## Valeur Étiquetés

| Étiquette | S'applique à |
|---|---|
| nom de la classe | Attribut ou opération dans un aspect stéréotypé de classe<br>Correspond à : les classes auxquelles appartient ce membre intertype AspectJ. |

## Autres conventions

- Les spécifications d'un pointcut sont incluses dans le champ « Comportement » de la méthode

# Conventions PHP

Enterprise Architect supporte l'ingénierie round de PHP 4 et 5, où ces conventions sont utilisées.

## Stéréotypes

| Stéréotype | S'applique à |
|---|---|
| trait | Classe<br>Correspond à : Un « trait ». |
| obtenir une propriété | Opération<br>Correspond à : une propriété « lecture ». |
| ensemble de propriétés | Opération<br>Correspond à : une propriété « écriture ». |

## Valeur Étiquetés

| Étiquette | S'applique à |
|---|---|
| nom_attribut | Opération avec la propriété stéréotype get ou property set<br>Correspond à : Le nom de la variable derrière cette propriété. |
| final | Opérations en PHP 5<br>Correspond à : Le mot-clé « final ». |

## Conventions communes

- Un type non spécifié est modélisé comme var
- Méthodes renvoyant une référence sont générées en définissant le Type de retour sur var*
- Les paramètres de référence sont générés à partir de paramètres avec le paramètre Kind défini sur inout ou out

## Conventions PHP 5

- Le modificateur de classe final correspond à la propriété Is Leaf
- Le modificateur de classe abstrait correspond à la propriété Abstract
- L'indication du type de paramètre est prise en charge en définissant le Type d'un paramètre
- La valeur de inout ou out pour la propriété Kind d'un paramètre correspond à un paramètre de référence

# Conventions Python

Enterprise Architect supporte l'ingénierie round -retour de Python, où ces conventions sont utilisées.

## Stéréotypes

| Stéréotype | Correspond à |
|---|---|
| TypeAlias | Classe<br>Correspond à : Alias Type explicite |

## Valeur Étiquetés

| Étiquette | S'applique à |
|---|---|
| asynchrone | Opération<br>Correspond à : Le mot clé « async » dans la définition de fonction. |
| Décorateurs | Classe, Opération<br>Correspond à : Les décorateurs appliqués à cet élément dans la source. |

## Autres conventions

- Les membres Modèle avec une portée privée correspondent aux membres du code avec deux traits de soulignement au début
- Attributes ne sont générés que lorsque la valeur initiale n'est pas vide
- Tous les types sont rétro-conçus comme var

# Conventions SystemC

Enterprise Architect supporte l'ingénierie aller-retour de SystemC, où ces conventions sont utilisées.

## Stéréotypes

| Stéréotype | S'applique à |
|---|---|
| déléguer | Méthode<br>Correspond à : Un délégué. |
| énumération | Classe intérieure<br>Correspond à : un type d'énumération. |
| ami | Méthode<br>Correspond à : Une méthode amie. |
| propriété | Méthode<br>Correspond à : une définition de propriété. |
| sc_ctor | Méthode<br>Correspond à : un constructeur SystemC. |
| module_sc | Classe<br>Correspond à : un module SystemC. |
| sc_port | Attribut<br>Correspond à : Un port. |
| signal_sc | Attribut<br>Correspond à : Un signal. |
| structure | Classe intérieure<br>Correspond à : une structure ou une union. |

## Valeur Étiquetés

| Étiquette | S'applique à |
|---|---|
| gentil | Attribut (Port)<br>Correspond à : Type de port (cadencé, fifo, maître, esclave, résolu, vecteur). |
| mode | Attribut (Port)<br>Correspond à : Mode port (entrée, sortie, entrée-sortie). |

| | |
|---|---|
| remplace | Méthode<br>Correspond à : La liste d'héritage d'une déclaration de méthode. |
| lancer | Méthode<br>Correspond à : La spécification d'exception d'une méthode. |

## Autres conventions

- SystemC hérite également de la plupart des stéréotypes et Valeur Étiquetés du C++

## Pages de la boîte à outils SystemC

Pour modéliser une conception SystemC, faites glisser ces icônes sur un diagramme à partir de la page « Constructions SystemC » de la boîte à outils Diagramme .

| Page | Icône |
|---|---|
| SystèmeC | Module<br>Action : définit un module SystemC.<br>Un élément de classe stéréotypé sc_module. |
| Fonctionnalités de SystemC | Port<br>Action : définit un port SystemC.<br>Un attribut stéréotypé sc_port. |

## Accéder

| | |
|---|---|
| Ruban | Conception > Diagramme > Toolbox :  > Spécifiez 'SystemC Constructs' dans les boîtes de dialogue ' Trouvez Item de Boîte à Outils ' |
| Raccourcis Clavier | Ctrl+Shift+3 :  > Spécifiez 'SystemC Constructs' dans la dialogue ' Trouvez Item de Boîte à Outils ' |
| Autre | Vous pouvez afficher ou masquer la boîte à outils Diagramme en cliquant sur les icônes  ou  à l'extrémité gauche de la barre de légende en haut de la Vue Diagramme . |

# Conventions VB.NET

Enterprise Architect supporte l'ingénierie aller-retour de Visual Basic.NET, où ces conventions sont utilisées. Les versions antérieures de Visual Basic sont prises en charge en tant que langage différent.

## Stéréotypes

| Stéréotype | S'applique à |
|---|---|
| événement | Opération<br>Correspond à : une déclaration d'événement. |
| importer | Opération<br>Correspond à : une opération à importer depuis une autre bibliothèque. |
| module | Classe<br>Correspond à : Un module. |
| opérateur | Opération<br>Correspond à : une définition de surcharge d'opérateur. |
| partiel | Opération<br>Correspond à : Le mot clé « partial » sur une opération. |
| propriété | Opération<br>Correspond à : une propriété contenant éventuellement du code de lecture et d'écriture. |

## Valeur Étiquetés

| Étiquette | S'applique à |
|---|---|
| Alias | Opération avec importation de stéréotypes<br>Correspond à : l'alias de cette opération importée. |
| nom_attribut | Opération avec propriété de stéréotype<br>Correspond à : Le nom de la variable derrière cette propriété. |
| Jeu de caractères | Opération avec importation de stéréotypes<br>Correspond à : la clause de jeu de caractères pour cette importation - l'une des valeurs « Ansi », « Unicode » ou « Auto ». |
| déléguer | Opération<br>Correspond à : Le mot-clé « délégué ». |
| | |

| | |
|---|---|
| enumTag | Opération avec propriété de stéréotype<br>Correspond à : le type de données sous lequel cette propriété est représentée. |
| Poignées | Opération<br>Correspond à : la clause « handles » sur cette opération. |
| Outils | Opération<br>Correspond à : la clause « implements » sur cette opération. |
| Lib | Opération avec importation de stéréotypes<br>Correspond à : la bibliothèque d'où provient cette importation. |
| Doit remplacer | Opération<br>Correspond à : le mot-clé « MustOverride ». |
| Rétrécissement | Opération avec l'opérateur stéréotype<br>Correspond à : Le mot-clé « Rétrécissement ». |
| Non remplaçable | Opération<br>Correspond à : le mot-clé « NotOverrideable ». |
| Surcharges | Opération<br>Correspond à : Le mot-clé « surcharges ». |
| Remplacements | Opération<br>Correspond à : Le mot-clé « overrides ». |
| tableau de paramètres | Paramètre<br>Correspond à : une liste de paramètres utilisant le mot-clé « ParamArray ». |
| partiel | Classe, Interface<br>Correspond à : Le mot-clé « partial ». |
| lecture seule | Opération avec propriété de stéréotype<br>Correspond à : Cette propriété définit uniquement le code « lecture ». |
| ombres | Classe, interface, fonctionnement<br>Correspond à : Le mot-clé « Ombres ». |
| Commun | Attribut<br>Correspond à : Le mot-clé « Partagé ». |
| Élargissement | Opération avec l'opérateur stéréotype<br>Correspond à : Le mot-clé « Élargissement ». |
| écriture seule | Opération avec propriété de stéréotype<br>Correspond à : Cette propriété définit uniquement le code « écriture ». |

## Autres conventions

- Namespaces sont générés pour chaque Paquetage sous une racine d'espace de noms
- La propriété Is Leaf d'une classe correspond au mot-clé NotInheritable
- La propriété Abstract d'une classe correspond au mot-clé MustInherit
- La propriété Static d'un attribut ou d'une opération correspond au mot-clé Shared
- La propriété Abstract d'une opération correspond au mot clé MustOverride
- La valeur de in pour la propriété Kind d'un paramètre correspond au mot-clé ByVal
- La valeur de inout ou out pour la propriété Kind d'un paramètre correspond au mot-clé ByRef

# Conventions Verilog

Enterprise Architect supporte l'ingénierie aller-retour de Verilog, où ces conventions sont utilisées.

## Stéréotypes

| Stéréotype | S'applique à |
|---|---|
| asynchrone | Méthode<br>Correspond à : Un processus simultané. |
| énumération | Classe intérieure<br>Correspond à : un type d'énumération. |
| initialiseur | Méthode<br>Correspond à : un processus d'initialisation. |
| module | Classe<br>Correspond à : Un module. |
| partie | Attribut<br>Correspond à : une instanciation de composant. |
| port | Attribut<br>Correspond à : Un port. |
| synchrone | Méthode<br>Correspond à : Un processus séquentiel. |

## Valeur Étiquetés

| Étiquette | S'applique à |
|---|---|
| gentil | Attribut (signal)<br>Correspond à : Le type de signal (tel que registre, bus). |
| mode | Attribut (Port)<br>Correspond à : Le mode Port (in, out, inout). |
| Plan du port | Attribut (partie)<br>Correspond à : la carte générique/port du composant instancié. |
| sensibilité | Méthode<br>Correspond à : La liste de sensibilité d'un processus séquentiel. |

| | |
|---|---|
| taper | Attribut<br>Correspond à : la plage ou le type de valeur d'un attribut. |

## Pages de la boîte à outils Verilog

Accès : « Conception > Diagramme > Boîte à outils : icône « Hamburger » > HDL | Constructions Verilog »

Faites glisser ces icônes sur un diagramme pour modéliser une conception Verilog.

| Item | Action |
|---|---|
| Module | Définit un module Verilog. Un élément de classe stéréotypé de module. |
| Énumération | Définit un Type énuméré. Un élément d'énumération. |
| Port | Définit un port Verilog. Un attribut stéréotypé de port. |
| Partie | Définit une instanciation de composant Verilog. Un attribut partiellement stéréotypé. |
| Attribut | Définit un attribut. |
| Procédure | Définit un processus Verilog :<br>• Concurrent - Une méthode stéréotypée asynchrone<br>• Séquentiel - Une méthode synchrone-stéréotypée<br>• Initializer - Une méthode stéréotypée d'initialisation |

# Conventions VHDL

Enterprise Architect supporte l'ingénierie aller-retour de VHDL, lorsque ces conventions sont utilisées.

## Stéréotypes

| Stéréotype | S'applique à |
|---|---|
| architecture | Classe<br>Correspond à : Une architecture . |
| asynchrone | Méthode<br>Correspond à : Un processus asynchrone. |
| configuration | Méthode<br>Correspond à : Une configuration. |
| énumération | Classe intérieure<br>Correspond à : un type énuméré. |
| entité | Interface<br>Correspond à : Une entité. |
| partie | Attribut<br>Correspond à : une instanciation de composant. |
| port | Attribut<br>Correspond à : Un port. |
| signal | Attribut<br>Correspond à : Une déclaration de signal. |
| structure | Classe intérieure<br>Correspond à : une définition d'enregistrement. |
| synchrone | Méthode<br>Correspond à : Un processus synchrone. |
| définition de type | Classe intérieure<br>Correspond à : une définition de type ou de sous-type. |

## Valeur Étiquetés

| Étiquette | S'applique à |
|---|---|

| | |
|---|---|
| estgénérique | Attribut (port)<br>Correspond à : La déclaration « port » dans une interface générique. |
| estSubType | Classe interne (typedef)<br>Correspond à : une définition de sous-type. |
| gentil | Attribut (signal)<br>Correspond à : Le type de signal (tel que « registre », « bus »). |
| mode | Attribut (Port)<br>Correspond à : Le mode Port ('in', 'out', 'inout', 'buffer', 'linkage'). |
| plan du port | Attribut (partie)<br>Correspond à : la carte générique/port du composant instancié. |
| sensibilité | Méthode (synchrone)<br>Correspond à : La liste de « sensibilité » d'un processus synchrone. |
| taper | Classe interne (typedef)<br>Correspond à : L'indication « type » d'une déclaration « type ». |
| typeNameSpace | Attribut (partie)<br>Correspond à : l'espace de noms « type » du composant instancié. |

## Pages de la boîte à outils VHDL

## Accéder

Pour modéliser une conception VHDL, faites glisser les icônes depuis les pages de la boîte à outils VHDL et déposez-les sur votre diagramme .

| | |
|---|---|
| Ruban | Conception > Diagramme > Toolbox :  > Spécifiez 'VHDL Constructs' dans la dialogue ' Trouvez Item de Boîte à Outils ' |
| Raccourcis Clavier | Ctrl+Shift+3 :  > Spécifiez 'VHDL Constructs' dans la dialogue ' Trouvez Item de Boîte à Outils ' |
| Autre | Vous pouvez afficher ou masquer la boîte à outils Diagramme en cliquant sur les icônes  ou  à l'extrémité gauche de la barre de légende en haut de la Vue Diagramme . |

## Page de la boîte à outils VHDL

| Item | Action |
|---|---|
| Architecture | Définit une architecture à associer à une entité VHDL.<br>Un élément de classe stéréotypé en termes d'architecture. |
| Entité | Définit une entité VHDL pour contenir les définitions de port.<br>Un élément d'interface stéréotypé d'entité. |
| Énumération | Définit un Type énuméré.<br>Un élément d'énumération. |
| Structure | Définit un enregistrement VHDL.<br>Un élément de classe stéréotypé de type struct. |
| Définition de type | Définit un type ou sous-type VHDL.<br>Un élément de classe stéréotypé typedef. |

## Page Boîte à Outils Fonctionnalités VHDL

| Item | Action |
|---|---|
| Partie | Définit une instanciation de composant VHDL.<br>Un attribut partiellement stéréotypé. |
| Port | Définit un port VHDL.<br>Un attribut stéréotypé du port. |
| Signal | Définit un signal VHDL.<br>Un attribut stéréotypé du signal. |
| Procédure | Définit un processus VHDL :<br>• Concurrent - Une méthode stéréotypée asynchrone<br>• Séquentiel - Une méthode synchrone-stéréotypée<br>• Configuration - Une méthode stéréotypée de configuration |

# Conventions de Visual Basic

Enterprise Architect supporte l'ingénierie round de Visual Basic 5 et 6, où ces conventions sont utilisées.
Visual Basic .NET est pris en charge comme langage différent.

## Stéréotypes

| Stéréotype | S'applique à |
|---|---|
| mondial | Attribut<br>Correspond à : Le mot-clé « Global ». |
| importer | Opération<br>Correspond à : une opération à importer depuis une autre bibliothèque. |
| obtenir une propriété | Opération<br>Correspond à : Une propriété 'get'. |
| ensemble de propriétés | Opération<br>Correspond à : un « ensemble » de propriétés. |
| propriété à louer | Opération<br>Correspond à : Une propriété « let ». |
| avec des événements | Attribut<br>Correspond à : le mot-clé « WithEvents ». |

## Valeur Étiquetés

| Étiquette | S'applique à |
|---|---|
| Alias | Opération avec importation de stéréotypes<br>Correspond à : l'alias de cette opération importée. |
| nom_attribut | Opération avec propriété de stéréotype get, propriété set ou propriété let<br>Correspond à : Le nom de la variable derrière cette propriété. |
| Lib | Opération avec importation de stéréotypes<br>Correspond à : la bibliothèque d'où provient cette importation. |
| Nouveau | Attribut<br>Correspond à : Le mot-clé « nouveau ». |

## Autres conventions

- La valeur de in pour la propriété Kind d'un paramètre correspond au mot-clé ByVal
- La valeur de inout ou out pour la propriété Kind d'un paramètre correspond au mot-clé ByRef

# Options de langue

Vous pouvez configurer différentes options pour déterminer la manière dont Enterprise Architect gère un langage particulier lors de la génération et de la rétro-ingénierie du code. Ces options sont spécifiques à :

- Votre ID utilisateur, pour tous les modèles ou
- Le modèle dans lequel ils sont définis, pour tous les utilisateurs

## Accéder

| | |
|---|---|
| Ruban | Démarrer > Apparence > Préférences > Préférences > Ingénierie du code source > <nom de la langue><br>Paramètres > Modèle > Options > Ingénierie du code source > <nom de la langue> |
| Raccourcis Clavier | Ctrl+F9 ( dialogue « Préférences ») |

## Langues prises en charge

| Langue |
|---|
| Scénario Action |
| Ada 2012 (dans les éditions Unified et Ultimate d' Enterprise Architect ) |
| ArcGIS |
| Norme ANSI C |
| C# |
| C++ |
| Delphes |
| Java |
| PHP |
| Python |
| SystèmeC |
| Verilog (éditions Unified et Ultimate ) |
| VHDL (éditions Unified et Ultimate ) |
| |

| Visual Basic |
| --- |
| Visual Basic .NET |

# Options ActionScript - Utilisateur

Si vous avez l'intention de générer du code ActionScript à partir de votre modèle, vous pouvez configurer les options de génération de code à l'aide de la page « Spécifications ActionScript » de la dialogue « Préférences » pour :

- Spécifier le répertoire source par défaut
- Spécifier l'éditeur pour le code ActionScript

## Accéder

| | |
|---|---|
| Ruban | Démarrer > Apparence > Préférences > Préférences > Ingénierie du code source > ActionScript |
| Raccourcis Clavier | Ctrl+F9 |

## Options

| Option | Action |
|---|---|
| Désactiver la langue | Laissez cette case à cocher décochée pour support la génération de code ActionScript.<br>Cochez cette case pour désactiver support du code ActionScript. |
| Options pour l'utilisateur actuel | Dans les champs « Répertoire source par défaut » et « Éditeur », cliquez sur le bouton  et recherchez le répertoire source et l'éditeur de fichiers externes que vous utiliserez. |

## Notes

- Ces options s'appliquent à tous les modèles auxquels vous accédez

# Options ActionScript - Modèle

Si vous avez l'intention de générer du code ActionScript à partir de votre modèle, vous pouvez configurer les options de génération de code spécifiques au modèle à l'aide de la page « Spécifications ActionScript » de la dialogue « Gérer les options Modèle » pour :

- Spécifiez la version ActionScript par défaut à générer (AS2.0 ou AS3.0)
- Spécifier les extensions de fichier par défaut
- Spécifier les définitions de classe de collection pour les connecteurs d'association

## Accéder

| | |
|---|---|
| Ruban | Paramètres > Modèle > Options > Ingénierie du code source > ActionScript |

## Options

| Option | Action |
|---|---|
| Options pour le modèle actuel | Type la version ActionScript par défaut et l'extension de fichier par défaut à appliquer lors de la génération du code source ActionScript. |
| Cours de collection | Cliquez sur ce bouton pour ouvrir la dialogue « Classes de collection pour les rôles d'association », à travers laquelle vous spécifiez les définitions de classes de collection pour les connecteurs d'association. |

## Notes

- Ces options affectent tous les utilisateurs du modèle actuel ; cependant, elles ne s'appliquent pas aux autres modèles

# Options Ada 2012 - Utilisateur

Si vous souhaitez générer du code Ada 2012 à partir de votre modèle, vous pouvez configurer les options de génération de code à l'aide de la page « Ada » de la dialogue « Préférences » pour :

- Informez le processus d'ingénierie inverse si le nom de l'enregistrement balisé est le même que le nom Paquetage
- Indiquez au moteur le nom de l'enregistrement balisé alternatif à localiser
- Spécifiez si le moteur doit créer un type de référence pour l'enregistrement balisé (si aucun n'est défini)
- Indiquez le nom du type de référence à créer (la valeur par défaut est Ref)
- Spécifier le paramètre de référence d'un type Référence / Accès
- Dites au moteur d'ignorer le nom du paramètre de référence
- Indiquer le nom du paramètre de référence à localiser

## Accéder

| | |
|---|---|
| Ruban | Démarrer > Apparence > Préférences > Préférences > Ingénierie du code source > Ada |
| Raccourcis Clavier | Ctrl+F9 |

## Options

| Option | Action |
|---|---|
| Désactiver la langue | Laissez cette case à cocher décochée pour support la génération de code Ada 2012.<br>Cochez cette case pour désactiver support du code Ada 2012. |
| Options pour l'utilisateur actuel | Spécifie les options utilisées pour l'utilisateur actuel ; ces options s'appliquent à tous les modèles auxquels l'utilisateur accède. |

## Notes

- support Ada 2012 est disponible dans les éditions Unified et Ultimate d' Enterprise Architect

# Options Ada 2012 - Modèle

Si vous avez l'intention de générer du code Ada 2012 à partir de votre modèle, vous pouvez configurer les options de génération de code spécifiques au modèle à l'aide de la page « Ada » de la dialogue « Gérer les options Modèle » pour :

- Spécifiez l'extension de fichier par défaut et
- Spécifier les définitions de classe de collection pour les connecteurs d'association

## Accéder

| | |
|---|---|
| Ruban | Paramètres > Modèle > Options > Ingénierie du code source > Ada |

## Options

| Option | Action |
|---|---|
| Options pour le modèle actuel | Type l’extension de fichier par défaut à appliquer lors de la génération du code source Ada. |
| Cours de collection | Cliquez sur ce bouton pour ouvrir la dialogue « Classes de collection pour les rôles d'association », à travers laquelle vous spécifiez les définitions de classes de collection pour les connecteurs d'association. |

## Notes

- Ces options affectent tous les utilisateurs du modèle actuel ; cependant, elles ne s'appliquent pas aux autres modèles
- support Ada 2012 est disponible dans les éditions Unified et Ultimate d' Enterprise Architect

# Options ArcGIS - Utilisateur

Si vous avez l'intention de générer du code ArcGIS à partir de votre modèle, vous pouvez configurer les options de génération de code à l'aide de la page « ArcGIS » de la dialogue « Préférences » pour :

- Spécifier le répertoire source par défaut
- Spécifier l'éditeur pour le code ArcGIS

ArcGIS doit être activé dans la dialogue « MDG Technologies » (Spécialiser > Technologies > Gérer la technologie ») pour que la page « ArcGIS » soit disponible.

## Accéder

| | |
|---|---|
| Ruban | Démarrer >Apparence > Préférences > Préférences > Ingénierie du code source > ArcGIS |
| Raccourcis Clavier | Ctrl+F9 |

## Options

| Option | Action |
|---|---|
| Désactiver la langue | Laissez cette case à cocher décochée pour support la génération de code ArcGIS.<br>Cochez cette case pour désactiver support du code ArcGIS. |
| Options pour l'utilisateur actuel | Spécifie les options utilisées pour l'utilisateur actuel ; ces options s'appliquent à tous les modèles auxquels l'utilisateur accède. |

# Options ArcGIS - Modèle

Si vous avez l'intention de générer du code ArcGIS à partir de votre modèle, vous pouvez configurer les options de génération de code spécifiques au modèle à l'aide de la page « ArcGIS » de la dialogue « Gérer les options Modèle » pour :

- Spécifier les extensions de fichier par défaut
- Spécifier les définitions de classe de collection pour les connecteurs d'association

## Accéder

| | |
|---|---|
| Ruban | Paramètres > Modèle > Options > Ingénierie du code source > ArcGIS |

## Options

| Option | Action |
|---|---|
| Options pour le modèle actuel | Type l’extension de fichier par défaut à appliquer lors de la génération du code source ArcGIS. |
| Cours de collection | Cliquez sur ce bouton pour ouvrir la dialogue « Classes de collection pour les rôles d'association », à travers laquelle vous spécifiez les définitions de classes de collection pour les connecteurs d'association. |

## Notes

- Ces options affectent tous les utilisateurs du modèle actuel ; cependant, elles ne s'appliquent pas aux autres modèles

# Options C - Utilisateur

Si vous avez l'intention de générer du code C à partir de votre modèle, vous pouvez configurer les options de génération de code à l'aide de la page « Spécifications C » de la dialogue « Préférences ».

## Accéder

| | |
|---|---|
| Ruban | Démarrer > Apparence > Préférences > Préférences > Ingénierie du code source > C |
| Raccourcis Clavier | Ctrl+F9 |

## Options

| Option | Action |
|---|---|
| Désactiver la langue | Laissez cette case à cocher décochée pour support la génération de code C.<br>Sélectionnez cette option pour désactiver support du code C. |
| Options pour l'utilisateur actuel | Dans les champs valeur , spécifiez les options qui s'appliquent sous votre propre ID d'utilisateur dans tous les modèles auxquels vous accédez :<br>• Le type d'attribut par défaut à créer (fixé comme int )<br>• Si une constante #define est importée en tant qu'attribut dans le code C importé (si « Programmation orientée Object » est défini sur True sur la page « Spécifications C » de la dialogue « Gérer les options Modèle »)<br>• S'il faut générer des commentaires pour les méthodes C de la déclaration et effectuer une rétro-ingénierie des commentaires à partir de la déclaration<br>• S'il faut générer des commentaires pour les méthodes C de l'implémentation et effectuer une rétro-ingénierie des commentaires à partir de l'implémentation<br>• S'il faut mettre à jour les commentaires lors de la régénération du code à partir du modèle<br>• S'il faut mettre à jour le fichier d'implémentation lors de la régénération du code à partir du modèle<br>• L'emplacement du répertoire de code source par défaut (cliquez sur le bouton  )<br>• Les extensions de fichiers par défaut à lire lors de l'importation d'un répertoire de code C<br>• L' Éditeur de Code à utiliser (cliquez sur le bouton  )<br>• Le chemin de recherche du fichier d'implémentation par rapport au chemin du fichier d'en-tête |

# Options C - Modèle

Si vous avez l'intention de générer du code C à partir de votre modèle, vous pouvez configurer les options de génération de code spécifiques au modèle à l'aide de la page « Spécifications C » de la dialogue « Gérer les options Modèle » pour :

- Spécifier les extensions de fichier par défaut (en-tête et source)
- Définir support de la programmation orientée Object
- Définir les options d'ingénierie Statemachine
- Spécifier les définitions de classe de collection pour les connecteurs d'association

## Accéder

| | |
|---|---|
| Ruban | Paramètres > Modèle > Options > Ingénierie du code source > C |

## Options

| Option | Action |
|---|---|
| Options pour le modèle actuel | Dans les champs valeur , spécifiez ces options :<br>• Les extensions de fichier source et d'en-tête par défaut pour les fichiers de code<br>• Support de la programmation orientée Object ; si cette valeur est vraie, définissez :<br>- Le caractère délimiteur Namespace<br>- Si le premier paramètre d'une opération est une référence de classe<br>- Le style de référence des paramètres dans le code C généré<br>- Le nom du paramètre de référence dans le code généré<br>- Le nom du constructeur par défaut dans le code généré<br>- Le nom du destructeur par défaut dans le code généré |
| Ingénierie Statemachine | Dans les champs valeur , utilisez les flèches déroulantes pour définir les options sur True ou False ; ces options s'appliquent uniquement à la génération de code à partir de modèles Statemachine dans le modèle actuel :<br>• « Utiliser la nouvelle Statemachine Gabarit » : définissez sur True pour utiliser les gabarits de génération de code d' Enterprise Architect Release 11 et versions ultérieures, définissez sur False pour appliquer les gabarits hérités d'EASL<br>• Générer un code de trace - définissez sur True pour générer un code de trace, False pour l'omettre |
| Cours de collection | Cliquez sur ce bouton pour ouvrir la dialogue « Classes de collection pour les rôles d'association », à travers laquelle vous spécifiez les définitions de classes de collection pour les connecteurs d'association. |

## Notes

- Ces options affectent tous les utilisateurs du modèle actuel ; cependant, elles ne s'appliquent pas aux autres modèles

# Options C# - Utilisateur

Si vous avez l'intention de générer du code C# à partir de votre modèle, vous pouvez configurer les options de génération de code à l'aide de la page « Spécifications C# de la dialogue « Préférences »

## Accéder

| | |
|---|---|
| Ruban | Démarrer > Apparence > Préférences > Préférences > Ingénierie du code source > C# |
| Raccourcis Clavier | Ctrl+F9 |

## Options

| Option | Action |
|---|---|
| Désactiver la langue | Laissez cette case à cocher décochée pour support la génération de code C# .<br>Cochez cette case pour désactiver support du code C# . |
| Options pour l'utilisateur actuel | Dans les champs valeur , spécifiez les options qui s'appliquent sous votre propre ID d'utilisateur dans tous les modèles auxquels vous accédez :<br>• Le type d'attribut par défaut à créer<br>• Si Namespaces doivent être générés lors de la génération de classes C#<br>• S'il faut supprimer les nouvelles lignes (retours chariot durs) de l' étiquette de résumé lors de l'importation de commentaires de style XML.NET<br>• S'il faut générer une méthode Finalizer lors de la génération de code pour une classe C#<br>• S'il faut générer une méthode Dispose lors de la génération de code pour une classe C#<br>• L'emplacement du répertoire de code source par défaut (cliquez sur le bouton  )<br>• L' Éditeur de Code à utiliser (cliquez sur le bouton  ) |

# Options C# - Modèle

Si vous avez l'intention de générer du code C# à partir de votre modèle, vous pouvez configurer les options de génération de code spécifiques au modèle à l'aide de la page « Spécifications C# » de la dialogue « Gérer les options Modèle » pour :

- Spécifiez l'extension de fichier par défaut
- Indiquer des classes de collection supplémentaires - pour définir des classes de collection personnalisées, qui peuvent être de simples substitutions (telles que CArray<#TYPE#>) ou un mélange d'autres chaînes et substitutions (telles que Cmap<CString,LPCTSTR,#TYPE#*,#TYPE#*>) ; ces classes de collection sont définies par défaut :
  - Liste<#TYPE#>;Pile<#TYPE#>;File d'attente<#TYPE#>;
- Définir les options d'ingénierie Statemachine
- Spécifier les définitions de classe de collection pour les connecteurs d'association

## Accéder

| | |
|---|---|
| Ruban | Paramètres > Modèle > Options > Ingénierie du code source > C# |

## Options

| Option | Action |
|---|---|
| Options pour le modèle actuel | Type l'extension de fichier par défaut à appliquer lors de la génération du code source C# et une liste de toutes les classes de collection supplémentaires que vous souhaitez définir. |
| Ingénierie Statemachine | Dans les champs valeur , utilisez les flèches déroulantes pour définir les options sur True ou False ; ces options s'appliquent uniquement à la génération de code à partir de modèles Statemachine dans le modèle actuel :<br>• « Utiliser la nouvelle Statemachine Gabarit » : définissez sur True pour utiliser les gabarits de génération de code d' Enterprise Architect Release 11 et versions ultérieures, définissez sur False pour appliquer les gabarits hérités d'EASL<br>• « Générer un code de trace » : définissez sur True pour générer un code de trace, False pour l'omettre |
| Cours de collection | Cliquez sur ce bouton pour ouvrir la dialogue « Classes de collection pour les rôles d'association », à travers laquelle vous spécifiez les définitions de classes de collection pour les connecteurs d'association. |

## Notes

- Ces options affectent tous les utilisateurs du modèle actuel ; cependant, elles ne s'appliquent pas aux autres modèles

# Options C++ - Utilisateur

Si vous avez l'intention de générer du code C++ à partir de votre modèle, vous pouvez configurer les options de génération de code à l'aide de la page « Spécifications C++ » de la dialogue « Préférences ».

## Accéder

| | |
|---|---|
| Ruban | Démarrer > Apparence > Préférences > Préférences > Ingénierie du code source > C++ |
| Raccourcis Clavier | Ctrl+F9 |

## Options

| Option | Action |
|---|---|
| Désactiver la langue | Laissez cette case à cocher décochée pour support la génération de code C++.<br>Sélectionnez cette option pour désactiver support du code C++. |
| Options pour l'utilisateur actuel | Dans les champs valeur , spécifiez les options qui s'appliquent sous votre propre ID d'utilisateur dans tous les modèles auxquels vous accédez :<br>• Le type d'attribut par défaut à créer<br>• Si Namespaces doivent être générés lors de la génération de classes C++<br>• Quel style appliquer lors de la génération et du traitement des commentaires pour C++<br>• S'il faut générer des commentaires pour les méthodes C++ de la déclaration ou procéder à une rétro-ingénierie des commentaires à partir de la déclaration<br>• Faut-il générer des commentaires pour les méthodes C++ de l'implémentation ou procéder à une rétro-ingénierie des commentaires à partir de l'implémentation ?<br>• S'il faut mettre à jour les commentaires lors de la régénération du code à partir du modèle<br>• S'il faut mettre à jour le fichier d'implémentation lors de la régénération du code à partir du modèle<br>• L'emplacement du répertoire de code source par défaut (cliquez sur le bouton  )<br>• Les extensions de fichier par défaut à lire lors de l'importation d'un répertoire de code C++<br>• L' Éditeur de Code à utiliser (cliquez sur le bouton  )<br>• Le chemin de recherche du fichier d'implémentation par rapport au chemin du fichier d'en-tête |

# Options C++ - Modèle

Si vous avez l'intention de générer du code C++ à partir de votre modèle, vous pouvez configurer les options de génération de code spécifiques au modèle à l'aide de la page « Spécifications C++ » de la dialogue « Gérer les options Modèle » pour :

- Indiquez la version de C++ à générer ; cela contrôle l'ensemble des gabarits utilisés et la manière dont les propriétés sont créées
- Spécifier le type de référence par défaut utilisé lorsqu'un type est spécifié par référence
- Spécifier les extensions de fichier par défaut
- Spécifier les préfixes Get/Set par défaut
- Spécifier les définitions de classe de collection pour les connecteurs d'association
- Définir des classes de collection supplémentaires - pour définir des classes de collection personnalisées, qui peuvent être de simples substitutions (telles que CArray<#TYPE#>) ou un mélange d'autres chaînes et substitutions (telles que Cmap<CString,LPCTSTR,#TYPE#*,#TYPE#*>) ; ces classes de collection sont définies par défaut :
  - CArray<#TYPE#>;CMap<CString,LPCTSTR,#TYPE#*,#TYPE#*>;
- Définir les options d'ingénierie Statemachine

## Accéder

| | |
|---|---|
| Ruban | Paramètres > Modèle > Options > Ingénierie du code source > C++ |

## Options

| Option | Action |
|---|---|
| Options pour le modèle actuel | Dans les champs valeur , spécifiez les options qui affectent tous les utilisateurs du modèle actuel :<br>• La version de C++ que vous utilisez (qui détermine les gabarits à utiliser lors de la génération de code)<br>• Le type de référence par défaut à utiliser lors de la création de propriétés pour les attributs C++ par référence<br>• Les extensions de fichier source et d'en-tête par défaut pour les fichiers de code<br>• Le préfixe « Get » par défaut<br>• Le préfixe « Set » par défaut<br>• Les classes de collection supplémentaires |
| Options d'ingénierie Statemachine | Dans les champs valeur , utilisez les flèches déroulantes pour définir les options sur True ou False ; ces options s'appliquent uniquement à la génération de code à partir de modèles Statemachine dans le modèle actuel :<br>• « Utiliser la nouvelle Statemachine Gabarit » : définissez sur True pour utiliser les gabarits de génération de code d' Enterprise Architect Release 11 et versions ultérieures, définissez sur False pour appliquer les gabarits hérités d'EASL<br>• « Générer un code de trace » : définissez sur True pour générer un code de trace, False pour l'omettre |

| | |
|---|---|
| Cours de collection | Cliquez sur ce bouton pour ouvrir la dialogue « Classes de collection pour les rôles d'association », à travers laquelle vous spécifiez les définitions de classes de collection pour les connecteurs d'association. |

## Notes

- Ces options affectent tous les utilisateurs du modèle actuel ; cependant, elles ne s'appliquent pas aux autres modèles

# Options Delphi - Utilisateur

Si vous avez l'intention de générer du code Delphi à partir de votre modèle, vous pouvez configurer les options de génération de code à l'aide de la page « Spécifications Delphi » de la dialogue « Préférences » pour :

- Définir le type d'attribut par défaut
- Indiquer un répertoire source par défaut
- Définir l'éditeur de code par défaut à utiliser pour modifier le code source Delphi

## Accéder

| | |
|---|---|
| Ruban | Démarrer > Apparence > Préférences > Préférences > Ingénierie du code source > Delphi |
| Raccourcis Clavier | Ctrl+F9 |

## Options

| Option | Action |
|---|---|
| Désactiver la langue | Laissez cette case à cocher décochée pour support la génération de code Delphi.<br>Sélectionnez cette option pour désactiver support du code Delphi. |
| Options pour l'utilisateur actuel | Spécifie les options utilisées pour l'utilisateur actuel ; ces options s'appliquent à tous les modèles auxquels l'utilisateur accède. |

# Options Delphi - Modèle

Si vous avez l'intention de générer du code Delphi à partir de votre modèle, vous pouvez configurer les options de génération de code spécifiques au modèle à l'aide de la page « Spécifications Delphi » de la dialogue « Gérer les options Modèle » pour :

- Spécifier les extensions de fichier par défaut (en-tête et source)
- Spécifier les définitions de classe de collection pour les connecteurs d'association

## Accéder

| | |
|---|---|
| Ruban | Paramètres > Modèle > Options > Ingénierie du code source > Delphi |

## Options

| Option | Action |
|---|---|
| Options pour le modèle actuel | Type l’extension de fichier par défaut à appliquer lors de la génération du code source Delphi. |
| Cours de collection | Cliquez sur ce bouton pour ouvrir la dialogue « Classes de collection pour les rôles d'association », à travers laquelle vous spécifiez les définitions de classes de collection pour les connecteurs d'association. |

## Notes

- Ces options affectent tous les utilisateurs du modèle actuel ; cependant, elles ne s'appliquent pas aux autres modèles

# Propriétés Delphi

Enterprise Architect offre support complet pour les propriétés Delphi. Celles-ci sont implémentées sous forme de Valeur Étiquetés , avec un éditeur de propriétés spécialisé pour aider à créer et modifier les propriétés de classe. En utilisant l'option de menu contextuel de l'élément « Visibilité Fonctionnalité », vous pouvez afficher le compartiment « étiquettes » qui contient les propriétés. Les classes Delphi importées avec des propriétés ont cette fonctionnalité automatiquement rendue visible pour votre commodité.

## Activer manuellement l'éditeur de propriétés

- Dans la classe sélectionnée, définissez le langage de génération de code sur « Delphi »
- Cliquez-droit sur la Classe et sélectionnez 'Delphi Propriétés ' pour ouvrir l'éditeur

En utilisant l'éditeur Delphi Propriétés , vous pouvez créer des propriétés rapidement et simplement ; à partir de là, vous pouvez :

- Modifier le nom et la portée (seuls les publics et les publiés sont actuellement pris en charge)
- Modifier le type de propriété (la liste déroulante inclut toutes les classes définies dans le projet)
- Définissez les informations de lecture et d'écriture (les listes déroulantes contiennent tous les attributs et opérations de la classe actuelle ; vous pouvez également saisir du texte libre)
- Réglez « Stocké » sur Vrai ou Faux
- Définir les informations sur les outils
- Définir la valeur par défaut, si elle existe

## Notes

- Lorsque vous utilisez la dialogue « Créer une propriété » à partir de l'écran « Attribut », le système génère une paire de fonctions Get et Set avec la définition de propriété requise en tant que Valeur Étiquetés ; vous pouvez modifier manuellement ces Valeur Étiquetés si nécessaire
- Les propriétés publiques sont affichées avec un préfixe « + » et publiées avec un « ^ »
- Lors de la création d'une propriété dans la dialogue « Créer une implémentation de propriété » (accessible via la dialogue « Attributes »), vous pouvez définir la portée sur « Publié » si le type de propriété est Delphi
- Seuls « Public » et « Publié » sont pris en charge
- Si vous modifiez le nom d'une propriété et effectuez une ingénierie directe, une nouvelle propriété est ajoutée, mais vous devez supprimer manuellement l'ancienne du fichier source

# Options Java - Utilisateur

Si vous avez l'intention de générer du code Java à partir de votre modèle, vous pouvez configurer les options de génération de code à l'aide de la page « Spécifications Java » de la dialogue « Préférences ».

## Accéder

| | |
|---|---|
| Ruban | Démarrer > Apparence > Préférences > Préférences > Ingénierie du code source > Java |
| Raccourcis Clavier | Ctrl+F9 |

## Options

| Option | Action |
|---|---|
| Désactiver la langue | Laissez cette case à cocher décochée pour support la génération de code Java.<br>Cochez cette case pour désactiver support du code Java. |
| Options pour l'utilisateur actuel | Dans les champs valeur , spécifiez les options qui s'appliquent sous votre propre ID d'utilisateur dans tous les modèles auxquels vous accédez ; le :<br>• Type d'attribut par défaut à créer (sélectionnez dans la liste déroulante)<br>• Emplacement du répertoire de code source par défaut (cliquez sur le bouton  )<br>• Éditeur de Code à utiliser (cliquez sur le bouton  ) |

# Options Java - Modèle

Si vous avez l'intention de générer du code Java à partir de votre modèle, vous pouvez configurer les options de génération de code spécifiques au modèle à l'aide de la page « Spécifications Java » de la dialogue « Gérer les options Modèle » pour :

- Spécifiez l'extension de fichier par défaut
- Spécifier un préfixe « Get » par défaut
- Spécifier un préfixe « Set » par défaut
- Définir les options d'ingénierie Statemachine
- Spécifier les définitions de classe de collection pour les connecteurs d'association
- Définir des classes de collection supplémentaires - pour définir des classes de collection personnalisées, qui peuvent être de simples substitutions (telles que CArray<#TYPE#>) ou un mélange d'autres chaînes et substitutions (telles que Cmap<CString,LPCTSTR,#TYPE#*,#TYPE#*>) ; ces classes de collection sont définies par défaut :
  - HashSet<#TYPE#>;Map< String ,#TYPE#>;

## Accéder

| | |
|---|---|
| Ruban | Paramètres > Modèle > Options > Ingénierie du code source > Java |

## Options

| Option | Action |
|---|---|
| Options pour le modèle actuel | Dans les champs valeur , spécifiez les options qui affectent tous les utilisateurs du modèle actuel ; les :<br>• Extension de fichier par défaut pour les fichiers de code<br>• Les préfixes Get et Set par défaut<br>• Les classes de collection par défaut et supplémentaires |
| Ingénierie Statemachine | Dans les champs valeur , utilisez les flèches déroulantes pour définir les options sur True ou False ; ces options s'appliquent uniquement à la génération de code à partir de modèles Statemachine dans le modèle actuel :<br>• « Utiliser la nouvelle Statemachine Gabarit » : définissez sur True pour utiliser les gabarits de génération de code d' Enterprise Architect Release 11 et versions ultérieures, définissez sur False pour appliquer les gabarits hérités d'EASL<br>• « Générer un code de trace » : définissez sur True pour générer un code de trace, False pour l'omettre |
| Cours de collection | Cliquez sur ce bouton pour ouvrir la dialogue « Classes de collection pour les rôles d'association », à travers laquelle vous spécifiez les définitions de classes de collection pour les connecteurs d'association. |

## Notes

- Ces options affectent tous les utilisateurs du modèle actuel ; cependant, elles ne s'appliquent pas aux autres modèles

# Options MySQL - Utilisateur

Si vous avez l'intention de générer du code MySQL à partir de votre modèle, vous pouvez configurer les options de génération de code à l'aide de la page « MySQL » de la dialogue « Préférences » pour :

- Spécifier un type d’attribut par défaut
- Spécifier un répertoire source par défaut
- Spécifier les extensions de nom de fichier pour les fichiers à importer
- Spécifier un éditeur pour modifier le code
- Spécifier un propriétaire par défaut

## Accéder

| | |
|---|---|
| Ruban | Démarrer > Apparence > Préférences > Préférences > Ingénierie du code source > MySQL |
| Raccourcis Clavier | Ctrl+F9 |

## Options

| Option | Action |
|---|---|
| Désactiver la langue | Laissez cette case à cocher décochée pour support la génération de code MySQL.<br>Sélectionnez cette option pour désactiver support du code MySQL. |
| Options pour l'utilisateur actuel | Spécifie les options utilisées pour l'utilisateur actuel ; ces options s'appliquent à tous les modèles auxquels l'utilisateur accède. |

# Options MySQL - Modèle

Si vous avez l'intention de générer du code MySQL à partir de votre modèle, vous pouvez configurer les options de génération de code spécifiques au modèle à l'aide de la page « MySQL » de la dialogue « Gérer les options Modèle » pour :

- Spécifier l'extension de fichier par défaut
- Spécifier les définitions de classe de collection pour les connecteurs d'association

## Accéder

| | |
|---|---|
| Ruban | Paramètres > Modèle > Options > Ingénierie du code source > MySQL |

## Options

| Option | Action |
|---|---|
| Options pour le modèle actuel | Type l’extension de fichier par défaut à appliquer lors de la génération du code source MySQL. |
| Cours de collection | Cliquez sur ce bouton pour ouvrir la dialogue « Classes de collection pour les rôles d'association », à travers laquelle vous spécifiez les définitions de classes de collection pour les connecteurs d'association. |

## Notes

- Ces options affectent tous les utilisateurs du modèle actuel ; cependant, elles ne s'appliquent pas aux autres modèles

# Options PHP - Utilisateur

Si vous avez l'intention de générer du code PHP à partir de votre modèle, vous pouvez configurer les options de génération de code à l'aide de la page « Spécifications PHP » de la dialogue « Préférences » pour :

- Définissez une liste d'extensions séparées par des points-virgules à examiner lors de l'importation d'un code de répertoire pour PHP
- Définir un répertoire par défaut pour ouvrir et enregistrer le code source PHP
- Spécifiez l'éditeur par défaut à utiliser lors de l'édition du code PHP

## Accéder

| | |
|---|---|
| Ruban | Démarrer > Apparence > Préférences > Préférences > Ingénierie du code source > PHP |
| Raccourcis Clavier | Ctrl+F9 \| Ingénierie du code source \| PHP |

## Options

| Option | Action |
|---|---|
| Désactiver la langue | Laissez cette case à cocher décochée pour support la génération de code PHP.<br>Sélectionnez cette option pour désactiver support du code PHP. |
| Options pour l'utilisateur actuel | Spécifie les options utilisées pour l'utilisateur actuel ; ces options s'appliquent à tous les modèles auxquels l'utilisateur accède. |

# Options PHP - Modèle

Si vous avez l'intention de générer du code PHP à partir de votre modèle, vous pouvez configurer les options de génération de code spécifiques au modèle à l'aide de la page « Spécifications PHP » de la dialogue « Gérer les options Modèle » pour :

- Spécifiez la version PHP par défaut à générer
- Définir l'extension de fichier par défaut
- Spécifier un préfixe « Get » par défaut
- Spécifier un préfixe « Set » par défaut

## Accéder

| | |
|---|---|
| Ruban | Paramètres > Modèle > Options > Ingénierie du code source > PHP |

## Options

| Option | Action |
|---|---|
| Options pour le modèle actuel | Type la version PHP par défaut, l'extension de fichier par défaut à appliquer lors de la génération du code source PHP et les préfixes par défaut « Get » et « Set ». |

## Notes

- Ces options affectent tous les utilisateurs du modèle actuel ; cependant, elles ne s'appliquent pas aux autres modèles

# Options Python - Utilisateur

Si vous avez l'intention de générer du code Python à partir de votre modèle, vous pouvez configurer les options de génération de code à l'aide de la page « Spécifications Python » de la dialogue « Préférences » pour :

- Spécifiez le répertoire source par défaut à utiliser
- Spécifiez l'éditeur par défaut utilisé pour écrire et modifier le code Python

## Accéder

| | |
|---|---|
| Ruban | Démarrer > Apparence > Préférences > Préférences > Ingénierie du code source > Python |
| Raccourcis Clavier | Ctrl+F9 |

## Options

| Option | Action |
|---|---|
| Désactiver la langue | Laissez cette case à cocher décochée pour support la génération de code Python.<br>Sélectionnez cette option pour désactiver support du code Python. |
| Options pour l'utilisateur actuel | Spécifie les options utilisées pour l'utilisateur actuel ; ces options s'appliquent à tous les modèles auxquels l'utilisateur accède. |

# Options Python - Modèle

Si vous avez l'intention de générer du code Python à partir de votre modèle, vous pouvez configurer les options de génération de code spécifiques au modèle à l'aide de la page « Spécifications Python » de la dialogue « Gérer les options Modèle » pour :

- Spécifiez l'extension de fichier par défaut

## Accéder

| | |
|---|---|
| Ruban | Paramètres > Modèle > Options > Ingénierie du code source > Python |

## Options

| Option | Action |
|---|---|
| Options pour le modèle actuel | Type l’extension de fichier par défaut à appliquer lors de la génération du code source Python. |

## Notes

- Ces options affectent tous les utilisateurs du modèle actuel ; cependant, elles ne s'appliquent pas aux autres modèles

# Options SystemC - Utilisateur

Si vous avez l'intention de générer du code SystemC à partir de votre modèle, vous pouvez configurer les options de génération de code à l'aide de la page « SystemC » de la dialogue « Préférences » pour :

- Spécifier un répertoire source par défaut
- Spécifier un éditeur pour modifier le code

## Accéder

| | |
|---|---|
| Ruban | Démarrer > Apparence > Préférences > Préférences > Ingénierie du code source > SystemC |
| Raccourcis Clavier | Ctrl+F9 |

## Options

| Option | Action |
|---|---|
| Désactiver la langue | Laissez cette case à cocher décochée pour support la génération de code SystemC.<br>Sélectionnez cette option pour désactiver support du code SystemC. |
| Options pour l'utilisateur actuel | Spécifie les options utilisées pour l'utilisateur actuel ; ces options s'appliquent à tous les modèles auxquels l'utilisateur accède. |

# Options SystemC - Modèle

Si vous avez l'intention de générer du code SystemC à partir de votre modèle, vous pouvez configurer les options de génération de code spécifiques au modèle à l'aide de la page « SystemC » de la dialogue « Gérer les options Modèle » pour :

- Spécifier l'extension de fichier par défaut
- Spécifier les définitions de classe de collection pour les connecteurs d'association

## Accéder

| | |
|---|---|
| Ruban | Paramètres > Modèle > Options > Ingénierie du code source > SystemC |

## Options

| Option | Action |
|---|---|
| Options pour le modèle actuel | Type l’extension de fichier par défaut à appliquer lors de la génération du code source SystemC. |
| Cours de collection | Cliquez sur ce bouton pour ouvrir la dialogue « Classes de collection pour les rôles d'association », à travers laquelle vous spécifiez les définitions de classes de collection pour les connecteurs d'association. |

## Notes

- Ces options affectent tous les utilisateurs du modèle actuel ; cependant, elles ne s'appliquent pas aux autres modèles

# Options Teradata - Utilisateur

Si vous avez l'intention de générer du code Teradata à partir de votre modèle, vous pouvez configurer les options de génération de code à l'aide de la page « Teradata » de la dialogue « Préférences » pour :

- Spécifier un type d'attribut par défaut
- Spécifier un répertoire source par défaut
- Spécifier un éditeur pour modifier le code

## Accéder

| | |
|---|---|
| Ruban | Démarrer > Apparence > Préférences > Préférences > Ingénierie du code source > Teradata |
| Raccourcis Clavier | Ctrl+F9 |

## Options

| Option | Action |
|---|---|
| Désactiver la langue | Laissez cette case à cocher décochée pour support la génération de code Teradata.<br>Sélectionnez cette option pour désactiver support du code Teradata. |
| Options pour l'utilisateur actuel | Spécifie les options utilisées pour l'utilisateur actuel ; ces options s'appliquent à tous les modèles auxquels l'utilisateur accède. |

# Options Teradata - Modèle

Si vous avez l'intention de générer du code Teradata à partir de votre modèle, vous pouvez configurer les options de génération de code spécifiques au modèle à l'aide de la page « Teradata » de la dialogue « Gérer les options Modèle » pour :

- Spécifiez l'extension de fichier par défaut
- Spécifier les définitions de classe de collection pour les connecteurs d'association

## Accéder

| | |
|---|---|
| Ruban | Paramètres > Modèle > Options > Ingénierie du code source > Teradata |

## Options

| Option | Action |
|---|---|
| Options pour le modèle actuel | Type l’extension de fichier par défaut à appliquer lors de la génération du code source Teradata. |
| Cours de collection | Cliquez sur ce bouton pour ouvrir la dialogue « Classes de collection pour les rôles d'association », à travers laquelle vous spécifiez les définitions de classes de collection pour les connecteurs d'association. |

## Notes

- Ces options affectent tous les utilisateurs du modèle actuel ; cependant, elles ne s'appliquent pas aux autres modèles

# Options VB.NET - Utilisateur

Si vous avez l'intention de générer du code VB.NET à partir de votre modèle, vous pouvez configurer les options de génération de code à l'aide de la page « Spécifications VB.NET » de la dialogue « Préférences » pour :

- Spécifier le type d’attribut par défaut
- Indiquer s'il faut générer des espaces de noms
- Spécifier un répertoire source par défaut
- Spécifier un éditeur pour modifier le code

## Accéder

| | |
|---|---|
| Ruban | Démarrer > Apparence > Préférences > Préférences > Ingénierie du code source > VB.Net |
| Raccourcis Clavier | Ctrl+F9 |

## Options

| Option | Action |
|---|---|
| Désactiver la langue | Laissez cette case à cocher décochée pour support la génération de code VB.NET.<br>Sélectionnez cette option pour désactiver support du code VB.NET. |
| Options pour l'utilisateur actuel | Spécifie les options utilisées pour l'utilisateur actuel ; ces options s'appliquent à tous les modèles auxquels l'utilisateur accède. |

# Options VB.NET - Modèle

Si vous avez l'intention de générer du code VB.NET à partir de votre modèle, vous pouvez configurer les options de génération de code spécifiques au modèle à l'aide de la page « Spécifications VB.Net » de la dialogue « Gérer les options Modèle » pour :

- Spécifiez l'extension de fichier par défaut
- Spécifier les définitions de classe de collection pour les connecteurs d'association

## Accéder

| | |
|---|---|
| Ruban | Paramètres > Modèle > Options > Ingénierie du code source > VB.Net |

## Options

| Option | Action |
|---|---|
| Options pour le modèle actuel | Type l’extension de fichier par défaut à appliquer lors de la génération du code source VB.Net. |
| Cours de collection | Cliquez sur ce bouton pour ouvrir la dialogue « Classes de collection pour les rôles d'association », à travers laquelle vous spécifiez les définitions de classes de collection pour les connecteurs d'association. |

## Notes

- Ces options affectent tous les utilisateurs du modèle actuel ; cependant, elles ne s'appliquent pas aux autres modèles

# Options Verilog - Utilisateur

Si vous avez l'intention de générer du code Verilog à partir de votre modèle, vous pouvez configurer les options de génération de code à l'aide de la page « Verilog » de la dialogue « Préférences » pour :

- Spécifier un répertoire source par défaut
- Spécifier un éditeur pour modifier le code

## Accéder

| | |
|---|---|
| Ruban | Démarrer > Apparence > Préférences > Préférences > Ingénierie du code source > Verilog |
| Raccourcis Clavier | Ctrl+F9 |

## Options

| Option | Action |
|---|---|
| Désactiver la langue | Laissez cette case à cocher décochée pour support la génération de code Verilog.<br>Sélectionnez cette option pour désactiver support du code Verilog. |
| Options pour l'utilisateur actuel | Spécifie les options utilisées pour l'utilisateur actuel ; ces options s'appliquent à tous les modèles auxquels l'utilisateur accède. |

# Options Verilog - Modèle

Si vous avez l'intention de générer du code Verilog à partir de votre modèle, vous pouvez configurer les options de génération de code spécifiques au modèle à l'aide de la page « Verilog » de la dialogue « Gérer les options Modèle » pour :

- Spécifier l'extension de fichier par défaut
- Spécifier les définitions de classe de collection pour les connecteurs d'association

## Accéder

| | |
|---|---|
| Ruban | Paramètres > Modèle > Options > Ingénierie du code source > Verilog |

## Options

| Option | Action |
|---|---|
| Options pour le modèle actuel | Type l’extension de fichier par défaut à appliquer lors de la génération du code source Verilog. |
| Cours de collection | Cliquez sur ce bouton pour ouvrir la dialogue « Classes de collection pour les rôles d'association », à travers laquelle vous spécifiez les définitions de classes de collection pour les connecteurs d'association. |

## Notes

- Ces options affectent tous les utilisateurs du modèle actuel ; cependant, elles ne s'appliquent pas aux autres modèles

# Options VHDL - Utilisateur

Si vous avez l'intention de générer du code VHDL à partir de votre modèle, vous pouvez configurer les options de génération de code à l'aide de la page « VHDL » de la dialogue « Préférences » pour :

- Spécifier un répertoire source par défaut
- Spécifier un éditeur pour modifier le code

## Accéder

| | |
|---|---|
| Ruban | Démarrer > Apparence > Préférences > Préférences > Ingénierie du code source > VHDL |
| Raccourcis Clavier | Ctrl+F9 |

## Options

| Option | Action |
|---|---|
| Désactiver la langue | Laissez cette case à cocher décochée pour support la génération de code VHDL.<br>Sélectionnez cette option pour désactiver support du code VHDL. |
| Options pour l'utilisateur actuel | Spécifie les options utilisées pour l'utilisateur actuel ; ces options s'appliquent à tous les modèles auxquels l'utilisateur accède. |

# Options VHDL - Modèle

Si vous avez l'intention de générer du code VHDL à partir de votre modèle, vous pouvez configurer les options de génération de code spécifiques au modèle à l'aide de la page « VHDL » de la dialogue « Gérer les options Modèle » pour :

- Spécifier l'extension de fichier par défaut
- Spécifier les définitions de classe de collection pour les connecteurs d'association

## Accéder

| | |
|---|---|
| Ruban | Paramètres > Modèle > Options > Ingénierie du code source > VHDL |

## Options

| Option | Action |
|---|---|
| Options pour le modèle actuel | Type l’extension de fichier par défaut à appliquer lors de la génération du code source VHDL. |
| Cours de collection | Cliquez sur ce bouton pour ouvrir la dialogue « Classes de collection pour les rôles d'association », à travers laquelle vous spécifiez les définitions de classes de collection pour les connecteurs d'association. |

## Notes

- Ces options affectent tous les utilisateurs du modèle actuel ; cependant, elles ne s'appliquent pas aux autres modèles

# Options Visual Basic - Utilisateur

Si vous avez l'intention de générer du code Visual Basic à partir de votre modèle, vous pouvez configurer les options de génération de code à l'aide de la page « Spécifications VB » de la dialogue « Préférences » pour :

- Spécifier le type d'attribut par défaut
- Définir le répertoire source par défaut
- Définir les extensions de fichier pour rechercher les fichiers de code à importer
- Définir l'éditeur par défaut à utiliser pour éditer le code source

## Accéder

| | |
|---|---|
| Ruban | Démarrer > Apparence > Préférences > Préférences > Ingénierie du code source > Visual Basic |
| Raccourcis Clavier | Ctrl+F9 |

## Options

| Option | Action |
|---|---|
| Désactiver la langue | Laissez cette case à cocher décochée pour support la génération de code Visual Basic.<br>Sélectionnez cette option pour désactiver support du code Visual Basic. |
| Options pour l'utilisateur actuel | Spécifie les options utilisées pour l'utilisation actuelle ; ces options s'appliquent à tous les modèles auxquels l'utilisateur accède. |

# Options Visual Basic - Modèle

Si vous avez l'intention de générer du code Visual Basic à partir de votre modèle, vous pouvez configurer les options de génération de code spécifiques au modèle à l'aide de la page « Spécifications VB » de la dialogue « Gérer les options Modèle » pour :

- Spécifiez la version par défaut de Visual Basic à générer
- Indiquer l'extension de fichier par défaut lors de la lecture/écriture
- Indiquez le mode de transaction Microsoft Transaction Server (MTS) pour les objets MTS
- Spécifiez si une classe utilise une utilisation multiple (vrai ou faux)
- Spécifier si une classe utilise la propriété Persistable
- Indiquer les comportements de liaison de données et de source de données
- Définir l'espace de noms global
- Définir l'attribut Exposed
- Indiquez si l'attribut Créable est Vrai ou Faux
- Spécifier les définitions de classe de collection pour les connecteurs d'association

## Accéder

| | |
|---|---|
| Ruban | Paramètres > Modèle > Options > Ingénierie du code source > Visual Basic |

## Options

| Option | Action |
|---|---|
| Options pour le modèle actuel | Type l'extension de fichier par défaut à appliquer lors de la génération du code source Visual Basic, puis cliquez sur la flèche déroulante dans chacun des autres champs et sélectionnez la valeur appropriée. |
| Cours de collection | Cliquez sur ce bouton pour ouvrir la dialogue « Classes de collection pour les rôles d'association », à travers laquelle vous spécifiez les définitions de classes de collection pour les connecteurs d'association. |

## Notes

- Ces options affectent tous les utilisateurs du modèle actuel ; cependant, elles ne s'appliquent pas aux autres modèles

# Options linguistiques MDG Technologie

Si vous avez chargé une MDG Technologie qui spécifie un module de code dans votre dossier *Sparx Systems* > *EA* > *MDG Technologies* , la langue est incluse dans la liste « Ingénierie du code source » de la dialogue « Préférences ». La langue n'est répertoriée dans la dialogue « Préférences » que si un fichier MDG Technologie l'utilise réellement dans votre modèle.

## Accéder

| | |
|---|---|
| Ruban | Démarrer > Apparence > Préférences > Préférences > Ingénierie du code source > MDG |
| Raccourcis Clavier | Ctrl+F9 |

## Options

| Champ | Action |
|---|---|
| Extension par défaut | Extension par défaut pour les fichiers sources générés ; affichée si l'option est dans la technologie.<br>Ceci est enregistré par projet. |
| Importer des extensions de fichiers | Dossier par défaut à partir duquel importer les fichiers sources ; affiché si la technologie supporte les espaces de noms.<br>Ceci est enregistré une fois pour tous les projets. |
| Générer Namespaces | Indique si les espaces de noms sont générés ou non. |
| Répertoire source par défaut | Le répertoire par défaut pour enregistrer les fichiers sources générés.<br>Ceci est toujours affiché. |
| Éditeur | Indique l'éditeur utilisé pour éditer les fichiers sources. |
| Type att | Indique le type d'attribut par défaut. |

## Notes

- Ces options sont définies dans la technologie à l'intérieur de l' étiquette <CodeOptions> d'un module de code, comme indiqué :
  <CodeOption name="DefaultExtension">.rb</CodeOption>

# Options de réinitialisation

Enterprise Architect stocke certaines des options d'une classe lors de sa création initiale. Certaines sont globales ; par exemple, $LinkClass est stocké lors de la création initiale de la classe. Ainsi, dans les classes existantes, la modification globale dans la dialogue « Préférences » ne sera pas automatiquement prise en compte. Vous devez modifier les options de la classe existante.

## Modifier les options pour une seule classe

| Étape | Action |
|---|---|
| 1 | Cliquez sur la classe à modifier et sélectionnez l'option de ruban « Développer > Code source > Générer > Générer un élément unique ».<br>La dialogue ' Générer Code' s'affiche. |
| 2 | Cliquez sur le bouton Avancé.<br>La dialogue « Options Object » s'affiche. |
| 3 | Cliquez sur l'option « Attributes / Opérations ». |
| 4 | Modifiez les options et cliquez sur le bouton Fermer pour appliquer les modifications. |

## Modifier les options pour toutes les classes d'un Paquetage

| Étape | Action |
|---|---|
| 1 | Cliquez sur Paquetage dans la fenêtre Navigateur et sélectionnez l'option de ruban « Développer > Préférences > Options > Réinitialiser la langue source ».<br>La dialogue « Gérer la génération de code » s'affiche. |
| 2 | Dans le champ « Où se trouve la langue : », cliquez sur la flèche déroulante et sélectionnez la langue que vous souhaitez modifier. |
| 3 | Dans le champ « Convertir en : », cliquez sur la flèche déroulante et sélectionnez la langue vers laquelle vous souhaitez changer. |
| 4 | Cochez la case en regard de chaque option à appliquer aux éléments de classe modifiés dans le Paquetage :<br>• Effacer les noms de fichiers pour générer du code<br>• Réinitialiser les options par défaut sur chaque classe<br>• Traiter Paquetages enfants sous le Paquetage sélectionné |
| 5 | Cliquez sur le bouton OK pour appliquer les modifications. |

# Classes de collection d'ensembles

À l'aide d' Enterprise Architect , vous pouvez définir des classes de collection pour générer du code à partir de connecteurs d'association où le rôle cible a un paramètre de multiplicité supérieur à 1.

## Tâches

| Tâche | Détail |
|---|---|
| Définition des classes de collection | Dans la section « Ingénierie du code source » de la dialogue « Gérer les options Modèle » (sélectionnez l'option de ruban « Paramètres > Modèle > Options > Ingénierie du code source »), sur chaque page de langue, cliquez sur le bouton Classes de collection.<br>La dialogue « Classes de collection pour les rôles d'association » s'affiche. Dans cette dialogue , vous pouvez définir :<br>• La classe de collection par défaut pour les rôles 1..*<br>• La classe de collection ordonnée à utiliser pour les rôles 1..*<br>• La classe de collection qualifiée à utiliser pour les rôles 1..* |
| Définition des classes de collection pour une classe spécifique | Les classes de collection spécifiques à la classe peuvent être définies en cliquant sur le bouton Classes de collection dans la dialogue « Propriétés » de la classe de l'élément. |
| Priorité de génération de code | Lorsque Enterprise Architect génère du code pour un connecteur qui a un rôle de multiplicité > 1 :<br>1. Si le qualificateur est défini, utilisez la collection qualifiée :<br>- pour la classe si définie<br>- sinon utiliser la collection qualifiée en langage de code<br>2. Si l'option « Ordre » est définie, utilisez la collection ordonnée :<br>- pour la classe si définie<br>- sinon utiliser la collection ordonnée du langage du code<br>3. Sinon, utilisez la collection par défaut :<br>- pour la classe si définie<br>- sinon utilisez la collection par défaut du langage de code |
| Utilisation Marqueurs | Vous pouvez inclure le marqueur #TYPE# dans le nom de la collection ; Enterprise Architect le remplace par le nom de la classe collectée au moment de la génération de la source (par exemple, Vector<#TYPE#> deviendrait Vector<foo>).<br>À l'inverse, lors de la rétro-ingénierie, un connecteur d'association est également créé si une entrée correspondante (par exemple, foo si foo est trouvé dans le modèle) est définie comme une classe de collection. |
| Classes de collection supplémentaires | Des classes de collection supplémentaires peuvent être définies dans les pages d'options de langage spécifiques au modèle pour C# , C++ et Java. |
| Type de membre | Dans l'onglet « Rôle(s) » de la dialogue « Propriétés » de l'Association (accessible depuis le menu contextuel cliquez-droit de n'importe quelle Association), il existe un champ « Type de membre » pour chacun des rôles source et cible.<br>Si vous définissez cette option, la valeur que vous entrez remplace toutes les |

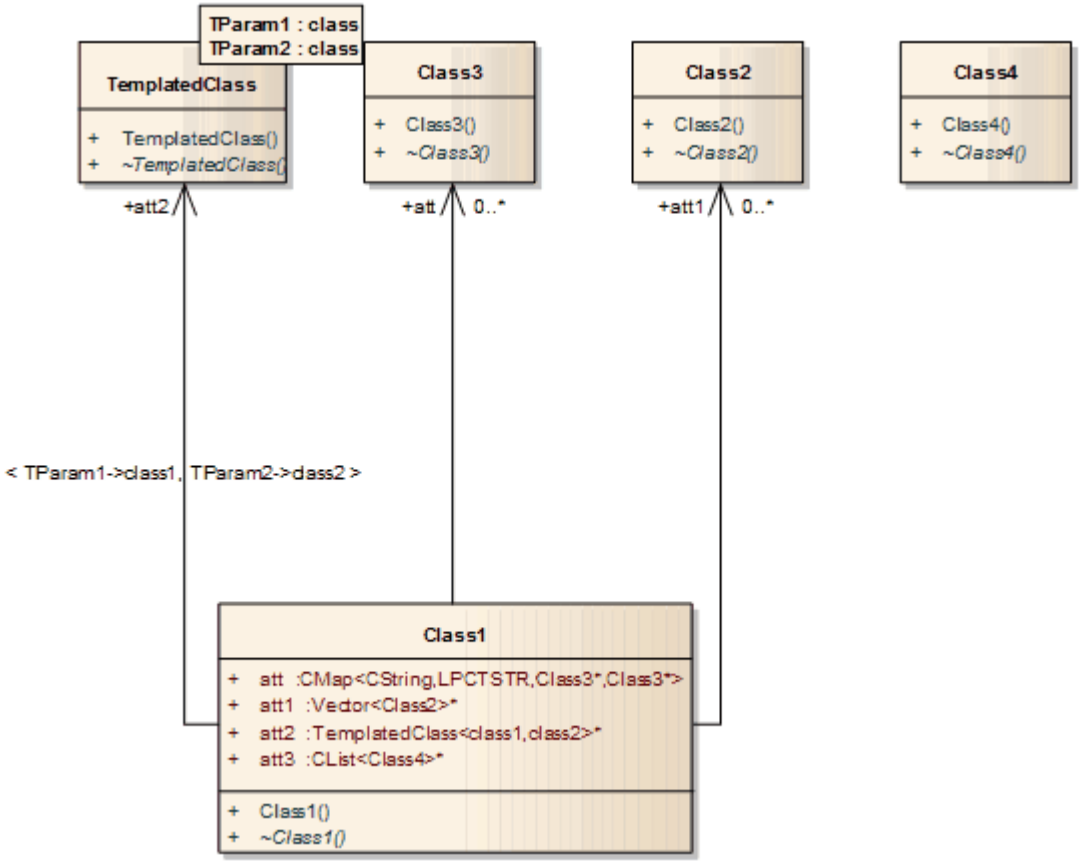
| | |
|---|---|
| | options répertoriées. |

# Exemple d'utilisation des classes de collection

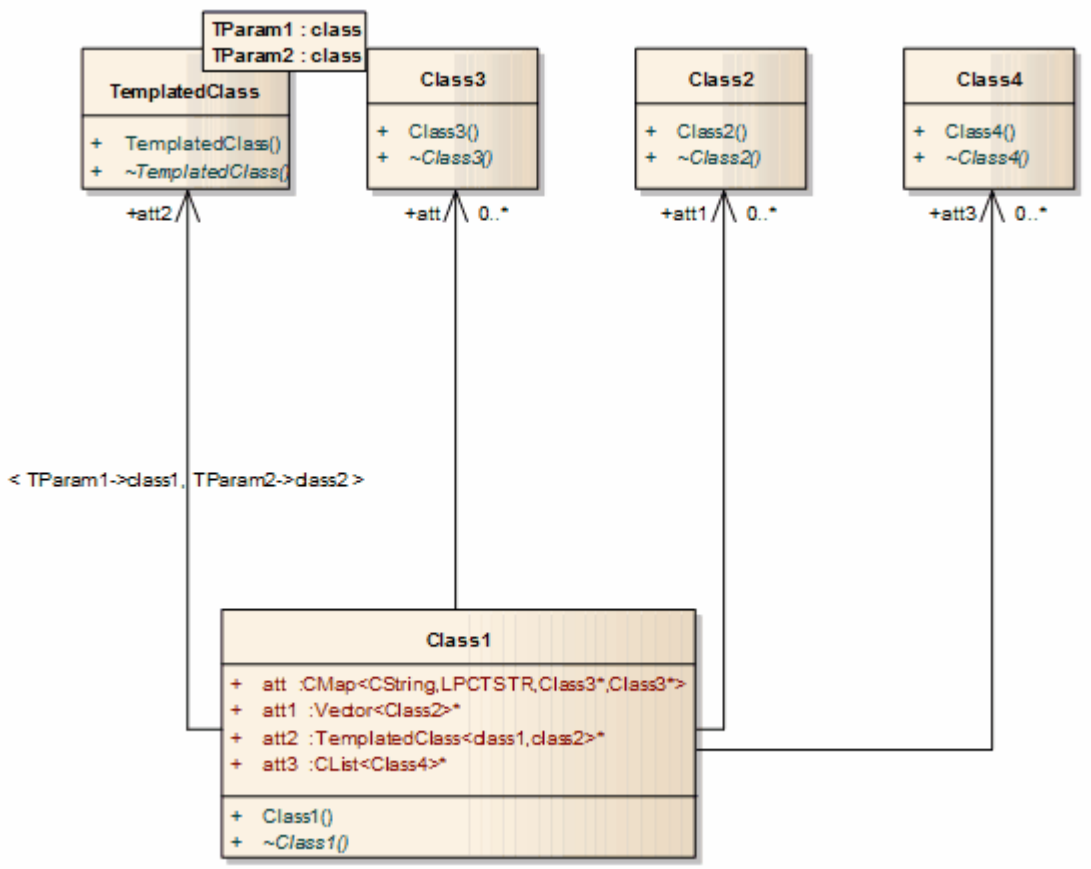
Considérez ce code source :

```
classe Classe1
{
publique:
Classe1();
virtuel ~Class1();
CMap<CString,LPCTSTR,Class3*,Class3*> att;
Vecteur<Classe2> *att1;
TemplatedClass<classe1,classe2> *att2;
CList<Class4> *att3;
};
classe Classe2
{
publique:
Classe2();
virtuel ~Class2();
};
classe Classe3
{
publique:
Classe3();
virtuel ~Class3();
};
classe Classe4
{
publique:
Classe4();
virtuel ~Class4();
};
modèle<classe TParam1, classe TParam2>
classe TemplatedClass
{
publique:
Classe modèle() {
}
virtuel ~TemplatedClass() {
}
};
```

Si ce code est importé dans le système avec les options d'importation par défaut, ce diagramme est généré :




Si, toutefois, vous entrez la valeur « CList<#Type#> » dans le champ « Classes de collection supplémentaires » dans la page d'options de langage spécifiques au modèle ( C# , Java, C++), un connecteur d'association est également créé pour la classe 4 :


TParam1 : class
TParam2 : class
TemplatedClass
+ TemplatedClass()
+ ~TemplatedClass()
Class3
+ Class3()
+ ~Class3()
Class2
+ Class2()
+ ~Class2()
Class4
+ Class4()
+ ~Class4()
+att2
+att
0..*
+att1
0..*
+att3
0..*
< TParam1->class1, TParam2->class2 >
Class1
+ att :CMap<CString,LPCTSTR,Class3*,Class3*>
+ att1 :Vector<Class2>*
+ att2 :TemplatedClass<class1,class2>*
+ att3 :CList<Class4>*
+ Class1()
+ ~Class1()

# Chemins locaux

Lorsqu'une équipe de développeurs travaille sur le même modèle Enterprise Architect , chaque développeur peut stocker sa version du code source dans son système de fichiers local, mais pas toujours au même emplacement que ses collègues développeurs. Pour gérer ce scénario dans Enterprise Architect , vous pouvez définir des chemins locaux pour chaque utilisateur, dans la dialogue « Chemins locaux ».

Vous pouvez utiliser des chemins locaux pour générer du code et de la rétro-ingénierie, ainsi que dans Contrôle de Version , pour développer des schémas XML et générer des documents et des rapports Web.

Les chemins locaux peuvent prendre un peu de temps à configurer, mais si vous souhaitez travailler en collaboration sur la source et le modèle simultanément, l'effort en vaut la peine.

Par exemple, si :

- Le développeur A stocke ses fichiers .java dans un répertoire C:\Java\Source, tandis que le développeur B stocke les siens dans D:\Source, et
- Les deux développeurs souhaitent générer et rétroconcevoir le même modèle Enterprise Architect situé sur un lecteur réseau partagé (ou répliqué)

Le développeur A peut définir un chemin local de :

JAVA_SOURCE = "C:\Java\Source"

Toutes les classes générées et stockées dans le projet Enterprise Architect sont stockées comme :

%JAVA_SOURCE%\<xxx.java>

Le développeur B définit un chemin local comme :

JAVA_SOURCE = "D:\Source"

Désormais, Enterprise Architect stocke tous les fichiers Java dans ces répertoires comme suit :

%JAVA_SOURCE%\<nom de fichier>

Sur la machine de chaque développeur, le nom de fichier est étendu à la version locale correcte.

## Accéder

| | |
|---|---|
| Ruban | Développer > Code source > Options > Configurer les chemins locaux |

# Dialogue sur les chemins locaux

À l'aide de la dialogue « Chemins locaux », vous pouvez configurer des chemins locaux pour un seul utilisateur sur une machine particulière. Pour une description de l'utilisation des chemins locaux, consultez la rubrique *Chemins locaux* .

## Accéder

| | |
|---|---|
| Ruban | Développer > Code source > Options > Configurer les chemins locaux |

## Options

| Option | Action |
|---|---|
| Chemin | Type ou recherchez dans votre navigateur le chemin du répertoire local dans le système de fichiers (par exemple, d:\java\source). |
| ID | Type l' ID partagé qui remplace le chemin local (par exemple, JAVA_SRC). |
| Type | Cliquez sur la flèche déroulante et sélectionnez le type de chemin à appliquer (par exemple, Java). |
| Chemins relatifs | Répertorie les chemins actuellement définis pour le modèle, la valeur par défaut étant le plus récent en haut.<br>Si vous souhaitez modifier la séquence des chemins dans la liste, cliquez sur un chemin et utilisez les boutons  pour déplacer le chemin vers le haut ou vers le bas d'une position dans la liste. |
| Appliquer le chemin | Cliquez sur un chemin dans la liste « Chemins relatifs » et cliquez sur ce bouton pour mettre à jour tous les noms de chemin complets existants dans le modèle avec le nom de chemin relatif partagé. Par exemple :<br>d:\java\source\main.java pourrait devenir %JAVA_SRC%\main.java |
| Développer le chemin | Cliquez sur un chemin dans la liste « Chemins relatifs » et cliquez sur ce bouton pour supprimer le chemin relatif et le remplacer par le nom du chemin complet (l'effet inverse du bouton Appliquer le chemin). |
| Nouveau | Cliquez sur ce bouton pour effacer les champs de données afin de pouvoir définir un autre chemin local. |
| Sauvegarder | Lorsque vous avez défini un chemin local, cliquez sur ce bouton pour l'enregistrer et l'ajouter à la liste « Chemins relatifs ». |
| Supprimer | Cliquez sur un chemin dans la liste « Chemins relatifs » et cliquez sur ce bouton pour supprimer complètement le chemin de la liste. |
| Fermer | Cliquez sur ce bouton pour fermer le dialogue et enregistrer les modifications |

| | |
|---|---|
| | apportées à la liste. |

## Notes

- Vous pouvez également configurer un lien hypertexte (pour une commande Enterprise Architect ) sur un diagramme pour accéder à la dialogue « Chemins locaux » pour changer, mettre à jour ou développer votre chemin local actuel.
- Si l'acte d'étendre ou d'appliquer un chemin pour un fichier lié crée un enregistrement en double, le processus ignorera cet enregistrement et affichera un message à la fin du processus

# Macros de langage

Lors de la rétro-ingénierie d'un langage tel que C++, vous pouvez trouver des directives de préprocesseur dispersées dans le code. Cela peut faciliter la gestion du code, mais peut entraver l'analyse du langage C++ sous-jacent.

Pour remédier à cela, vous pouvez inclure un nombre quelconque de définitions de macros, qui sont ignorées pendant la phase d'analyse de la rétro-ingénierie. Il est toujours préférable, si vous disposez de facilité , de prétraiter le code en utilisant d'abord le compilateur approprié ; de cette façon, les définitions et définitions de macros complexes sont étendues et peuvent être facilement analysées. Si vous ne disposez pas de facilité , cette option constitue un substitut pratique.

## Accéder

| | |
|---|---|
| Ruban | Paramètres > Données de référence > Paramètres > Macros du préprocesseur ou<br>Développer > Code source > Options > Configurer > Définir les macros du préprocesseur |

## Définir une macro

| Étape | Action |
|---|---|
| 1 | Sélectionnez l'option de menu « Macros du préprocesseur ».<br>La dialogue « Macros de langage » s'affiche. |
| 2 | Cliquez sur le bouton Ajouter nouveau. |
| 3 | Entrez les détails de votre macro. |
| 4 | Cliquez sur le bouton OK . |

## Macros intégrées dans les déclarations

Les macros sont parfois utilisées dans la déclaration de classes et d'opérations, comme dans ces exemples :

classe __declspec Foo

{

int __declspec Barre( int p);

};

Si declspec est défini comme une macro C++, comme indiqué, la classe et l'opération importées contiennent une Valeur Étiquetée appelée DeclMacro1 avec valeur __declspec (les macros suivantes seraient définies comme DeclMacro2, DeclMacro3 et ainsi de suite).

Lors de l'ingénierie prospective, ces Valeur Étiquetés sont utilisées pour régénérer les macros dans le code.

## Définir des macros complexes

Il est parfois utile de définir des règles pour des macros complexes pouvant s'étendre sur plusieurs lignes ; Enterprise Architect ignore toute la section de code définie par la règle.

De telles macros peuvent être définies dans Enterprise Architect comme dans ces deux exemples ; les deux types peuvent être combinés dans une seule définition.

**Bloc macros**

DEBUT_PARTIE_INTERFACE ^ FIN_PARTIE_INTERFACE

Le symbole ^ représente le corps de la macro - cela permet de passer d'une macro à une autre ; les espaces entourant le symbole ^ sont obligatoires.

**Macros de fonctions**

RTTI_EMULATION()

Enterprise Architect ignore le jeton, y compris tout ce qui se trouve à l'intérieur des parenthèses.

Les macros de fonction peuvent également inclure le corps de la fonction :

RTTI_EMULATION() {}

Dans ce cas, Enterprise Architect ignore le jeton, y compris tout ce qui se trouve entre parenthèses et entre accolades. Note que si la macro de fonction inclut le corps de la fonction, elle ne peut pas être combinée avec une macro Bloc .

## Notes

- Vous pouvez transporter ces définitions de macro de langage (ou macro de préprocesseur) entre les modèles, en utilisant les options « Paramètres > Modèle > Transférer > Exporter les données de référence » et « Importer les données de référence » ; les macros sont exportées sous forme de liste de macros.

# Développement de langages de programmation

Vous pouvez utiliser une gamme de langages de programmation établis dans Enterprise Architect , mais si ceux-ci ne sont pas adaptés à vos besoins, vous pouvez développer le vôtre. Vous l'appliquerez ensuite à vos modèles via une MDG Technologie que vous pourriez développer uniquement à cette fin, ou à des fins plus larges. Après avoir développé le langage, vous pouvez également écrire gabarits de transformation MDA pour convertir un Modèle indépendant de la plate-forme ou un modèle dans un autre langage en un modèle pour votre nouveau langage, ou vice-versa.

## Accéder

| | |
|---|---|
| Ruban | Développer > Code source > Options > Modifier le code Gabarits |
| Raccourcis Clavier | Ctrl+Maj+P |

## Développer un langage de programmation

| Étape | Description |
|---|---|
| 1 | Dans l'éditeur Code Gabarit , cliquez sur le bouton Nouveau langage et, dans la dialogue « Types de données des langages de programmation », cliquez sur le bouton Ajouter un produit.<br>Saisissez le nom de votre nouveau langage de programmation et définissez les types de données correspondants. Vous ne pouvez pas accéder au nouveau langage dans l'éditeur Code Gabarit tant qu'au moins un type de données n'a pas été ajouté au langage. |
| 2 | Après avoir défini tous les types de données dont vous avez besoin, cliquez sur le bouton Fermer, sélectionnez la langue dans le champ « Langue » de l'éditeur de code Gabarit et commencez à éditer ou à créer les gabarits de code pour la nouvelle langue.<br>Les gabarits du code définissent comment le système doit fonctionner :<br>• Ingénierie de code avancée de vos modèles dans le nouveau langage<br>• Génération de code Comportementale (si cela est approprié) |
| 3 | Si vous le préférez, vous pouvez également définir des options de code source pour votre nouveau langage. Il s'agit de paramètres supplémentaires pour le langage qui ne sont pas fournis par les types de données ou gabarits de code et qui aident à définir la manière dont le système gère ce langage lors de la génération et de la rétro-ingénierie du code.<br>Les options de code sont mises à disposition de vos modèles uniquement via une MDG Technologie . |
| 4 | Définir une grammaire pour votre langue est une étape facultative qui offre deux avantages principaux :<br>• Rétro-ingénierie du code existant dans votre modèle<br>• Synchronisation lors de la génération de code afin que les modifications apportées au fichier depuis sa dernière génération ne soient pas perdues.<br>Pour accéder à l'éditeur de grammaire, sélectionnez l'option de ruban « Développer > Code source > Éditeur de grammaire ». |
| 5 | Si vous souhaitez effectuer des transformations MDA vers (ou depuis) votre nouveau langage de programmation, vous pouvez également modifier et créer gabarits de transformation pour celui-ci. Le |

| | |
|---|---|
| | processus de création gabarits de transformation est très similaire à celui de création gabarits de code. |
| 6 | Après avoir créé les types de données, gabarits de code, les options de code, la grammaire et gabarits de transformation pour votre nouveau langage, vous pouvez les incorporer et les distribuer dans un MDG Technologie . |

# Cadre de code Gabarit

Lorsque vous utilisez Enterprise Architect pour générer du code à partir d'un modèle ou pour transformer le modèle, le système fait référence au Code Gabarit Framework (CTF) pour les paramètres qui définissent comment il doit :

- Ingénierie avancée d'un modèle UML
- Générer du code Comportementale
- Effectuer une transformation Model Driven Architecture (MDA)
- Générer du DDL en modélisation de base de données

Une gamme de gabarits standards est disponible pour la génération directe de code et pour la transformation ; si vous ne souhaitez pas utiliser les configurations CTF standards, vous pouvez les personnaliser pour répondre à vos besoins.

## CTF Gabarits

| Type Gabarit | Détail |
|---|---|
| Code Gabarits | Lorsque vous effectuez l'ingénierie directe d'un modèle de classe, les gabarits de code définissent la manière dont le code squelettique doit être généré pour un langage de programmation donné. Les gabarits d'un langage sont automatiquement associés au langage.<br>Les gabarits sont écrits sous forme de texte brut avec une syntaxe qui partage certains aspects des langages de balisage et des langages de script. |
| Transformation du Modèle Gabarits | Transformation du Modèle Gabarits fournissent une méthode entièrement configurable pour définir la manière dont les transformations Model Driven Architecture (MDA) convertissent les éléments et les fragments de modèle d'un domaine à un autre.<br>Ce processus est à deux niveaux. Il crée un langage intermédiaire (qui peut être visualisé pour le débogage) qui est ensuite traité pour créer les objets. |
| Gabarits de génération de code Comportementale | Enterprise Architect supporte la génération de code définissable par l'utilisateur des modèles Comportementale UML .<br>Ceci applique le framework Code Gabarit standard mais inclut des macros de génération de code spécifiques Bibliothèque Simulation Enterprise Architect (EASL). |
| DDL Gabarits | Gabarits DDL sont très similaires aux gabarits de génération de code, mais ils ont été étendus pour support la génération DDL avec leur propre ensemble de gabarits de base, de macros, de macros de fonctions et d'options gabarit . |

# Code Gabarit Personnalisation

Enterprise Architect vous aide à générer du code source à partir de modèles UML pour une large gamme de langages de programmation. gabarits standards (mappings) sont fournis prêts à l'emploi, mais vous pouvez personnaliser la manière dont le code est généré en utilisant le framework Code Gabarit (CTF) pratique et flexible. Ce framework sophistiqué vous permet de personnaliser chaque détail de la manière dont le code est généré, y compris la facilité de création de nouveaux gabarits pour les langages non pris en charge dans le produit de base. Par exemple, JavaScript ne fait pas partie des langages pris en charge, mais une série de gabarits peut être écrite rapidement pour générer JavaScript à partir de modèles UML . Dans ces cas, gabarits existants servent de point de départ et de référence utiles pour les nouveaux langages.

Le framework gabarit de code fournit également le mécanisme de génération de modèles comportementaux et est utilisé pour les gabarits de transformation.

## Fonctionnalités

| Fonctionnalité | Détail |
|---|---|
| Gabarits par défaut | Les codes par défaut Gabarits sont intégrés à Enterprise Architect pour les langages pris en charge par l'ingénierie directe. |
| Éditeur de code Gabarit | Un éditeur de code Gabarit est fourni pour créer et maintenir des code Gabarits définis par l'utilisateur. |
| Code de personnalisation Gabarits | Descriptions de la syntaxe gabarit et des macros et fonctions que vous pouvez utiliser pour contrôler les effets des gabarits . |
| Synchroniser le code | Un sous-ensemble du Code Gabarits par défaut pour synchroniser le code. |

# Coder et transformer Gabarits

gabarits gabarits de code et de transformation ( Transformation du Modèle ) définissent la manière dont le système doit générer ou transformer du code dans l'un ou l'autre des langages de programmation supporte par Enterprise Architect . Chaque langage dispose d'une large gamme de gabarits de base, chacun définissant la manière dont une structure de code particulière est générée. Vous pouvez utiliser ces gabarits de base tels quels, ou vous pouvez personnaliser et ajouter des gabarits pour mieux support votre utilisation des langages standard, ou d'autres langages que vous pourriez définir pour le système. Vous pouvez révision , mettre à jour et créer gabarits via l'éditeur Gabarit de code ou l'éditeur Gabarit de transformation.

L'ordre dans lequel les gabarits de base sont listés dans les deux éditeurs est lié à l'ordre hiérarchique des objets et de leurs parties à traiter. Les appels sont effectués de certains gabarits de base vers d'autres, et vous pouvez ajouter d'autres appels à la fois aux gabarits de base et à vos propres gabarits personnalisés. Par défaut, le gabarit du fichier est le point de départ d'un processus de génération de code via les gabarits ; un fichier est constitué de classes qui peuvent contenir Attributes et des opérations.

## Accéder

| | |
|---|---|
| Ruban | Développer > Code source > Options > Modifier le code Gabarits<br>Conception > Paquetage > Transformation > Transformation Gabarits |
| Raccourcis Clavier | Ctrl+Maj+P (Génération de code Gabarits )<br>Ctrl+Alt+H ( Gabarits de transformation MDA) |

## Application des Gabarits

| Action | Détail |
|---|---|
| Appel Gabarits | Dans n'importe quel gabarit , vous pouvez appeler d'autres gabarits en utilisant %TemplateName%. Les signes de pourcentage (%) englobants indiquent une macro.<br>Vous utiliseriez ceci pour un seul appel au gabarit ClassBody, %ClassBody%, comme indiqué :<br>% liste = "Nom du modèle" @separator= " \n" @indent= " " %<br>La macro %list effectue un passage itératif sur tous les objets dans la portée du gabarit actuel et appelle le TemplateName pour chacun d'eux :<br>% liste = "ClassBody" @separator= " \n" @indent= " " %<br>Après la génération ou la transformation, chaque macro est substituée pour produire la sortie générée ; pour un langage tel que C++, le résultat du traitement de ce gabarit pourrait être :<br>/**<br>* Ceci est un exemple note de cours générée à l'aide gabarits de code<br>* @auteur Sparx Systems<br>*/<br>classe ClasseA : publique ClasseB<br>{ |

| | |
|---|---|
| | ...<br>}<br> |
| Exécution du Code Gabarits | Chaque gabarit peut agir uniquement sur un type d'élément particulier ; par exemple, le gabarit ClassNotes agit uniquement sur les éléments de classe et d'interface UML .<br>L'élément à partir duquel le code est en cours de génération est dit dans la portée ; si l'élément dans la portée est stéréotypé, le système recherche un gabarit qui a été défini pour ce stéréotype. Si un gabarit spécialisé est trouvé, il est exécuté ; sinon, l'implémentation par défaut du gabarit de base est utilisée.<br>Gabarits sont traités séquentiellement, ligne par ligne, en remplaçant chaque macro par sa valeur de texte sous-jacente du modèle. |
| Transférer Gabarits entre les projets | Si vous modifiez un gabarit de génération ou de transformation de code de base, ou créez un gabarit personnalisé, vous pouvez les copier d'un projet à un autre en tant que données de référence. |

# Base Gabarits

Le framework Code Gabarit se compose d'un certain nombre de gabarits de base. Chaque gabarit de base transforme des aspects particuliers de l' UML en parties correspondantes de langages orientés objet.

Les gabarits de base forment une hiérarchie, qui varie légèrement selon les différents langages de programmation. Dans une hiérarchie gabarit typique propre à un langage tel que C# ou Java (qui n'ont pas de fichiers d'en-tête), les gabarits peuvent être modélisés comme des classes, mais sont généralement simplement du texte brut. Cette hiérarchie serait légèrement plus compliquée pour des langages tels que C++ et Delphi, qui ont gabarits d'implémentation distincts.

Chacun des gabarits de base doit être spécialisé pour être utile dans l'ingénierie de code ; en particulier, chaque gabarit est spécialisé pour les langages pris en charge (ou « produits »). Par exemple, il existe un gabarit ClassBody défini pour C++, un autre pour C# , un autre pour Java, et ainsi de suite ; en spécialisant les gabarits , vous pouvez adapter le code généré pour l'entité UML correspondante.

Une fois les gabarits de base spécialisés pour une langue donnée, ils peuvent être encore plus spécialisés en fonction de :

- Un stéréotype de classe, ou
- Un stéréotype de fonctionnalité (où la fonctionnalité peut être une opération ou un attribut)

Ce type de spécialisation permet, par exemple, à une opération C# stéréotypée comme « propriété » d'avoir un gabarit de corps d'opération différent d'une opération ordinaire ; le gabarit de corps d'opération peut alors être davantage spécialisé, en fonction du stéréotype de classe.

## gabarits de base utilisés dans le CTF

| Gabarit | Description |
|---|---|
| Attribut | Un gabarit de niveau supérieur pour générer des variables membres à partir d'attributs UML . |
| Déclaration d'attribut | Utilisé par l'attribut gabarit pour générer une déclaration de variable membre. |
| Notes sur les attributs | Utilisé par l'attribut gabarit pour générer notes de variables membres. |
| Classe | Un gabarit de niveau supérieur pour générer des classes à partir de classes UML . |
| Base de classe | Utilisé par le gabarit de classe pour générer un nom de classe de base dans la liste d'héritage d'une classe dérivée, où la classe de base n'existe pas dans le modèle. |
| Corps de la classe | Utilisé par la classe gabarit pour générer le corps d'une classe. |
| Déclaration de classe | Utilisé par le gabarit de classe pour générer la déclaration d'une classe. |
| Interface de classe | Utilisé par le gabarit de classe pour générer un nom d'interface dans la liste d'héritage d'une classe dérivée, où l'interface n'existe pas dans le modèle. |
| Notes de cours | Utilisé par le gabarit de classe pour générer les notes de classe. |
| Déposer | Un gabarit de niveau supérieur pour générer le fichier source.<br>Pour les langages tels que C++, cela correspond au fichier d'en-tête. |
| Section d'importation | Utilisé dans le gabarit du fichier pour générer des dépendances externes. |
| | |

| | |
|---|---|
| Attribut lié | Un gabarit de niveau supérieur pour générer des attributs dérivés d'associations UML . |
| Notes sur les attributs liés | Utilisé par le gabarit d'attribut lié pour générer les notes d'attribut. |
| Déclaration d'attribut lié | Utilisé par le gabarit d'attribut lié pour générer la déclaration d'attribut. |
| Base de classe liée | Utilisé par le gabarit de classe pour générer un nom de classe de base dans la liste d'héritage d'une classe dérivée, pour un élément de classe dans le modèle qui est un parent de la classe actuelle. |
| Interface de classe liée | Utilisé par le gabarit de classe pour générer un nom d'interface dans la liste d'héritage d'une classe dérivée, pour un élément d'interface dans le modèle qui est un parent de la classe actuelle. |
| Namespace | Un gabarit de niveau supérieur pour générer des espaces de noms à partir de Paquetages UML (bien que tous les langages n'aient pas d'espaces de noms, ce gabarit peut être utilisé pour générer une construction équivalente, comme Paquetages en Java). |
| Corps Namespace | Utilisé par le gabarit Namespace pour générer le corps d'un espace de noms. |
| Déclaration Namespace | Utilisé par le gabarit Namespace pour générer la déclaration d'espace de noms. |
| Opération | Un gabarit de niveau supérieur pour générer des opérations à partir des opérations d'une classe UML . |
| Opération Corps | Utilisé par le gabarit d'opération pour générer le corps d'une opération UML . |
| Déclaration d'opération | Utilisé par le gabarit d'opération pour générer la déclaration d'opération. |
| Notes d'opération | Utilisé par le gabarit d'opération pour générer la documentation d'une opération. |
| Paramètre | Utilisé par le gabarit de déclaration d'opération pour générer des paramètres. |

## Gabarits pour générer du code pour les langages avec des sections d'interface et d'implémentation séparées

| Gabarit | Description |
|---|---|
| Implémentation de classe | Un gabarit de niveau supérieur pour générer l'implémentation d'une classe. |
| Corps de classe Impl | Utilisé par le gabarit de classe Impl pour générer l'implémentation des membres de la classe. |
| Fichier Imp | Un gabarit de niveau supérieur pour générer le fichier d'implémentation. |
| Notes de fichier Impl | Utilisé par le gabarit File Impl pour générer notes dans le fichier source. |
| | |

| | |
|---|---|
| Section d'importation Imp | Utilisé par le gabarit File Impl pour générer des dépendances externes. |
| Opération Impl | Un gabarit de niveau supérieur pour générer des opérations à partir des opérations d'une classe UML . |
| Opération Body Impl | Utilisé par le gabarit d'opération pour générer le corps d'une opération UML . |
| Déclaration d'opération Impl | Utilisé par le gabarit d'opération pour générer la déclaration d'opération. |
| Notes d'opération Impl | Utilisé par le gabarit d'opération pour générer la documentation d'une opération. |

# Génération et transformation de code d'exportation Gabarits

Il est possible d'exporter gabarits de génération et de transformation de code de votre modèle vers un fichier .xml. Vous pouvez ensuite importer ce fichier - et donc les gabarits - dans d'autres modèles, en tant que données de référence. Vous pouvez exporter gabarits personnalisés, qui incluent ceux que vous ou d'autres utilisateurs avez créés et mis à jour, et gabarits de base (standard) qui ont été personnalisés. Vous n'avez pas besoin d'exporter gabarits de base qui n'ont pas été modifiés, car ils sont disponibles dans chaque installation d' Enterprise Architect .

## Accéder

| | |
|---|---|
| Ruban | Paramètres > Modèle > Transfert > Exporter les données de référence |

## Exporter un gabarit de Génération de Code ou gabarit de Transformation

| Étape | Action |
|---|---|
| 1 | Dans la dialogue « Exporter les données de référence », dans la liste « Nom », sélectionnez les gabarits à exporter.<br>La liste comprend tous gabarits de génération ou de transformation de code standard qui ont été modifiés, ainsi que tous gabarits personnalisés que vous avez créés ou modifiés.<br>Vous pouvez sélectionner un ou plusieurs gabarits à exporter vers un seul fichier XML, en appuyant sur Ctrl ou Maj lorsque vous cliquez sur les noms gabarit . |
| 2 | Cliquez sur le bouton Exporter. |
| 3 | Lorsque vous y êtes invité, entrez un nom de fichier valide avec une extension .xml. |
| 4 | Cliquez sur le bouton Enregistrer et sur le bouton OK .<br>Cela exporte le(s) gabarit (s) vers le fichier ; vous pouvez utiliser n'importe quel visualiseur de texte ou XML pour examiner le fichier. |

# Génération et transformation de code d'importation Gabarits

Si vous avez exporté gabarits de génération et/ou de transformation de code à partir d'un modèle Enterprise Architect , vous pouvez les importer dans d'autres modèles Enterprise Architect en tant que données de référence.

## Accéder

| | |
|---|---|
| Ruban | Paramètres > Modèle > Transfert > Importer les données de référence |

## Génération et/ou transformation de code d'importation Gabarits

| Étape | Action |
|---|---|
| 1 | Dans la dialogue « Importer des données de référence », cliquez sur le bouton Sélectionner un fichier et accédez au fichier .xml contenant les gabarits de génération ou de transformation de code requis. |
| 2 | Sélectionnez le nom d'un ou plusieurs jeux de données gabarit et cliquez sur le bouton Importer. |

# Synchroniser le code

Enterprise Architect utilise gabarits de code lors de la synchronisation directe de ces langages de programmation :

- ActionScript
- C
- C++
- C#
- Delphes
- Java
- PHP
- Python
- VB
- VB.Net

Trois types de changements peuvent survenir dans la source lorsqu'elle est synchronisée avec le modèle UML :

- Les sections existantes sont synchronisées : par exemple, le type de retour dans une déclaration d'opération est mis à jour
- De nouvelles sections sont ajoutées aux fonctionnalités existantes : par exemple, Notes sont ajoutées à une déclaration de classe là où il n'y en avait pas auparavant
- De nouvelles fonctionnalités et éléments sont ajoutés : par exemple, une nouvelle opération est ajoutée à une classe

Chacun de ces changements a un effet différent sur le CTF et doit être géré différemment par Enterprise Architect , comme décrit dans ces rubriques :

- *Synchroniser les sections existantes*
- *Ajouter de nouvelles sections aux Fonctionnalités existantes*
- *Ajouter de nouvelles Fonctionnalités et éléments*

## Sections de code pouvant être synchronisées

Seul un sous-ensemble des gabarits de base CTF est utilisé lors de la synchronisation. Ce sous-ensemble correspond aux sections distinctes qu'Enterprise Enterprise Architect reconnaît dans le code source.

| Code Gabarit | Section du code |
|---|---|
| Notes de cours | Commentaires précédant la déclaration de classe. |
| Déclaration de classe | Jusqu'aux parents de la classe inclus. |
| Notes sur les attributs | Commentaires précédant une déclaration d'attribut. |
| Déclaration d'attribut | Jusqu'au caractère final inclus. |
| Notes d'opération | Commentaires précédant une déclaration d'opération. |
| Notes d'opération Impl | Quant Notes d'opération. |
| Déclaration d'opération | Jusqu'au caractère final inclus. |
| Déclaration d'opération | |

| | |
|---|---|
| Impl | Jusqu'au caractère final inclus. |
| Opération Corps | Tout ce qui se trouve entre et y compris les accolades. |
| Opération Body Impl | Quant à l'Opération Corps. |

# Synchroniser les sections existantes

Lorsqu'une section existante dans le code source diffère du résultat généré par le gabarit correspondant, cette section est remplacée.

Considérez, par exemple, cette déclaration de classe C++ :

(asm) classe A : public B

Supposons maintenant que vous ajoutiez une relation d'héritage de la classe A à la classe C ; la déclaration de classe entière serait remplacée par quelque chose ressemblant à ceci :

(asm) classe A : public B, public C

# Ajouter de nouvelles sections

Ces sections peuvent être ajoutées aux fonctionnalités existantes dans le code source, en tant que nouvelles sections :

- Notes de cours
- Notes sur les attributs
- Notes d'opération
- Notes d'opération Impl
- Opération Corps
- Opération Body Impl

Supposons que, dans cet exemple, la classe A n'avait aucune note lorsque vous avez généré le code à l'origine :

(asm) classe A : public B, public C

Si vous spécifiez maintenant une note dans le modèle pour la classe A, Enterprise Architect tente d'ajouter la nouvelle note à partir du modèle lors de la synchronisation, en exécutant le gabarit Class Notes .

Pour faire de la place à l'insertion de la nouvelle section, vous pouvez spécifier la quantité d'espace blanc à ajouter à la section via des macros de synchronisation.

# Ajouter de nouvelles Fonctionnalités et éléments

Ces fonctionnalités et éléments peuvent être ajoutés au code source lors de la synchronisation :

- Attributes
- Classes internes
- Opérations

Ils sont ajoutés en exécutant les gabarits pertinents pour chaque nouvel élément ou fonctionnalité du modèle.

Enterprise Architect tente de préserver l'indentation appropriée des nouvelles fonctionnalités dans le code, en trouvant les indentations spécifiées dans les macros de liste de la classe ; pour les langages qui utilisent des espaces de noms, la macro « synchNamespaceBodyIndent » est disponible.

Les classes définies dans un espace de noms (non global) sont indentées en fonction de la valeur définie pour cette macro, lors de la synchronisation.

La valeur est ignorée :

- Pour les classes définies dans un Paquetage configuré comme espace de noms racine, ou
- Si l'option « Générer Namespaces » est définie sur False dans la page de langue appropriée ( C# , C++ ou VB.Net) dans la dialogue « Préférences » (' Démarrer > Apparence > Préférences > Préférences > Ingénierie du code source > <langue>')

# L'éditeur Code Gabarit

L'éditeur de code Gabarit fournit les facilités de l' Éditeur de Code Common, y compris Intelli-sense pour les différentes macros. Pour plus d'informations sur Intelli-sense et l' Éditeur de Code Common, consultez la rubrique *Édition du code source* .

## Accéder

| | |
|---|---|
| Ruban | Développer > Code source > Options > Modifier le code Gabarits |
| Raccourcis Clavier | Ctrl+Maj+P |

## Options

| Option | Action |
|---|---|
| Langue | Sélectionnez le langage de programmation. |
| Nouvelle langue | Affichez la dialogue « Types de données des langages de programmation », qui vous permet d'inclure des langages de programmation autres que ceux pris en charge par Enterprise Architect , pour lesquels créer ou modifier gabarits de code. |
| Gabarit | Affiche le contenu du gabarit actif et ouvre l'éditeur de modification gabarits . |
| Gabarits | Lister les gabarits du code de base ; le gabarit actif est mis en surbrillance.<br>Le champ « Modifié » indique si vous avez modifié le gabarit par défaut pour la langue actuelle. |
| Les stéréotypes supplantent | Lister les gabarits stéréotypés, pour le gabarit de base actif.<br>Le champ « Modifié » indique si vous avez modifié un gabarit stéréotypé par défaut. |
| Ajouter un nouveau Gabarit personnalisé | Invoquer une dialogue pour créer un gabarit stéréotypé personnalisé. |
| Ajouter un nouveau remplacement stéréotypé | Appeler une dialogue pour ajouter un gabarit stéréotypé, pour le gabarit de base actuellement sélectionné. |
| Obtenir Gabarit par défaut | Mettre à jour l'affichage de l'éditeur avec la version par défaut du gabarit actif. |
| Sauvegarder | Remplacer les gabarits actifs par le contenu de l'éditeur. |
| Supprimer | Si vous avez remplacé le gabarit actif, le remplacement est supprimé et remplacé par le gabarit de code par défaut correspondant. |

## Notes

- Gabarits de code modifiés et définis par l'utilisateur peuvent être importés et exportés en tant que données de référence (voir la rubrique *Partage des données de référence* ) ; les gabarits définis pour chaque langue sont indiqués dans la dialogue « Exporter les données de référence » par le nom de la langue avec le suffixe _Code_Templates - si aucun gabarits n'existe pour une langue, il n'y a pas d'entrée pour la langue dans le dialogue

# Créer un nouveau Gabarit personnalisé

La dialogue Créer un nouveau Gabarit personnalisé offre la possibilité de créer un gabarit personnalisé pour le langage de programmation ou le système de gestion de base de données (SGBD) actuel, en fonction des informations modifiées avec l'éditeur de code Gabarit .

Lorsque cette dialogue est chargée, il vous sera demandé de saisir une valeur pour Type Gabarit et le nom Gabarit . Pour enregistrer un nouveau gabarit Type et le nom sont tous deux requis.

## Options

| Option | Action |
|---|---|
| Type Gabarit | Choisissez le type de Gabarit pour le nouveau Custom Gabarit . |
| Nom de Gabarit | Entrez un nom pour le nouveau Gabarit personnalisé. |
| OK | Enregistrez les détails du nouveau Custom Gabarit . |
| Annuler | Fermez la dialogue Créer un nouveau Gabarit personnalisé et perdez toutes les modifications non enregistrées. |

## Note :

Tous gabarits de type " <none> » sont traités comme des fonctions. Par conséquent, Enterprise Architect supprimera automatiquement tous les caractères d'espace saisis dans le nom.

# Syntaxe du code Gabarit

Les Code Gabarits sont écrits à l'aide de l'éditeur Code Gabarit d' Enterprise Architect . L'éditeur Code Gabarit supporte la coloration syntaxique du langage Code Gabarit Framework.

## Éléments de syntaxe

| Éléments | Détail |
|---|---|
| Constructions de base | Gabarits peuvent contenir :<br>• Texte littéral<br>• Variables<br>• Macros<br>• Appels vers d'autres gabarits |
| Commentaires | Si vous souhaitez ajouter des commentaires aux gabarits , utilisez la commande :<br>$COMMENT="texte"<br>où « texte » est le texte du commentaire ; celui-ci doit être placé entre guillemets.<br>La commande est sensible à la casse et doit être saisie en majuscules. |

# Texte littéral

Tout texte dans un gabarit donné qui ne fait pas partie d'une macro ou d'une définition/référence de variable est considéré comme du texte littéral. À l'exception des lignes vides, qui sont ignorées, le texte littéral est directement substitué à partir du gabarit dans le code généré.

Considérez cet extrait du gabarit de déclaration de classe Java :

$bases = "Base"

classe % className % $bases

Sur la dernière ligne, le mot « class », y compris l'espace qui suit, serait traité comme du texte littéral et donc pour une classe nommée « foo », renverrait la sortie :

classe fooBase

Une ligne vide suivant la variable $bases n'aurait aucun effet sur la sortie.

## Insertion de caractères système :

Les caractères %, $, " et \ ont une signification particulière dans la syntaxe gabarit et ne peuvent pas toujours être utilisés comme texte littéral. Si ces caractères doivent être générés à partir des gabarits , ils peuvent être reproduits en toute sécurité à l'aide de ces macros de substitution directe :

| Macro | Action |
|---|---|
| %dl% | Produire un caractère $ littéral. |
| %pc% | Produire un caractère % littéral. |
| %qt% | Produire un caractère " littéral. |
| %sl% | Produire un caractère \ littéral |

## Notes

Les opérateurs de conjonction String (« + », « + = ») ne sont pas obligatoires mais peuvent être utilisés

# Variables

Les variables Gabarit offrent un moyen pratique de stocker et de récupérer des données dans un gabarit . Cette section explique comment les variables sont définies et référencées.

## Définitions des variables

Les définitions de variables prennent la forme de base :

$<nom> = <valeur>

où <nom> peut être n'importe quelle séquence alphanumérique et <valeur> est dérivé d'une macro ou d'une autre variable.

Un exemple simple de définition serait :

$foo = %className%

Les variables peuvent être définies à l'aide de valeurs provenant de :

- Substitution, fonction ou macros de liste
- Littéraux String , placés entre guillemets doubles
- Références variables

## Règles de définition

Ces règles s'appliquent aux définitions de variables :

- Les variables ont une portée globale dans le gabarit dans lequel elles sont définies et ne sont pas accessibles aux autres gabarits
- Chaque variable doit être définie au début d'une ligne, sans aucun espace intermédiaire
- Les variables sont indiquées en préfixant le nom avec $, comme dans $foo
- Les variables n'ont pas besoin d'être déclarées avant d'être définies
- Les variables doivent être définies à l'aide de l'opérateur d'affectation (=) ou de l'opérateur d'addition-affectation (+=)
- Plusieurs termes peuvent être combinés dans une seule définition à l'aide de l'opérateur d'addition (+)

## Exemples

Utilisation d'une macro de substitution :

$foo = %opTag:"bar"%

En utilisant une string littérale :

$foo = "barre"

En utilisant une autre variable :

$foo = $bar

Utilisation d'une macro de liste :

$ops = %list="Opération" @separator="\n\n" @indent="\t"%

En utilisant l'opérateur d'addition-affectation (+=) :

$body += %list="Opération" @separator="\n\n" @indent="\t"%

Cette définition est équivalente à :

$body = $body + %list="Opération" @separator="\n\n" @indent="\t"%

Utilisation de plusieurs termes :

$templateArgs = %list="Paramètre de classe" @séparateur=", "%

$template = "modèle< " + $templateArgs + "> "

## Références variables

Les valeurs des variables peuvent être récupérées en utilisant une référence de la forme :

$<nom>

où <nom> peut être une variable précédemment définie.

Des références variables peuvent être utilisées :

- Dans le cadre d'une macro, comme l'argument d'une macro de fonction
- En tant que terme dans une définition de variable
- En tant que substitution directe de la valeur de la variable dans la sortie

Il est légal de référencer une variable avant qu'elle ne soit définie. Dans ce cas, la variable est supposée contenir une string vide valeur : " "

## Références variables - Exemple 1

Utilisation de variables dans le cadre d'une macro. Ceci est un extrait du gabarit par défaut de C++ ClassNotes.

$wrapLen = %genOptWrapComment%

$style = %genOptCPPCommentStyle% (Définir des variables pour stocker les options de style et de longueur d'habillage)

%if $style == "XML.NET"% (Référence à $style dans le cadre d'une condition)

%XML_COMMENT($wrapLen)%

%autre%

%CSTYLE_COMMENT($wrapLen)% (Référence à $wrapLen comme argument de la macro de fonction)

%finSi%

## Références de variables - Exemple 2

Utilisation de références de variables dans le cadre d'une définition de variable.

$foo = "foo" (Définir nos variables)

$bar = "barre"

$foobar = $foo + $bar ($foobar contient maintenant la valeur foobar)

## Références de variables - Exemple 3

Substitution de valeurs de variables dans la sortie.

$bases=%classInherits% (Stocker le résultat du gabarit ClassInherits dans $bases)

Classe %className%$bases (affiche maintenant la valeur de $bases après le nom de la classe)

# Macros

Les macros permettent d'accéder aux champs d'éléments du modèle UML et sont également utilisées pour structurer la sortie générée. Toutes les macros sont entourées de signes de pourcentage (%), comme indiqué :

%<nom de la macro>%

En général, les macros (y compris les délimiteurs %) remplacent le texte littéral dans la sortie. Par exemple, considérez cet élément du gabarit de déclaration de classe :

... classe %className% ...

La macro de substitution de champ, %className%, entraînerait la substitution du nom de classe actuel dans la sortie. Ainsi, si la classe générée était nommée Foo, la sortie serait :

... classe Foo ...

Le CTF contient un certain nombre de types de macros :

- Template Substitution Macros
- Field Substitution Macros
- Substitution Examples
- Attribute Field Substitution Macros
- Class Field Substitution Macros
- Code Generation Option Field Substitution Macros
- Connector Field Substitution Macros
- Constraint Field Substitution Macros
- Effort Field Substitution Macros
- File Field Substitution Macros
- File Import Field Substitution Macros
- Link Field Substitution Macros
- Linked File Field Substitution Macros
- Metric Field Substitution Macros
- Operation Field Substitution Macros
- Package Field Substitution Macros
- Parameter Field Substitution Macros
- Problem Field Substitution Macros
- Requirement Field Substitution Macros
- Resource Field Substitution Macros
- Risk Field Substitution Macros
- Scenario Field Substitution Macros
- Tagged Value Substitution Macros
- Template Parameter Substitution Macros
- Test Field Substitution Macros
- Function Macros
- Control Macros
- List Macro
- Branching Macros
- Synchronization Macros

- The Processing Instruction (PI) Macro
- EASL Code Generation Macros

# Macros de substitution Gabarit

Les macros de substitution Gabarit correspondent aux gabarits de base et aboutissent à l'exécution du gabarit nommé. Par convention, les macros gabarit sont nommées selon la casse Pascal.

Structure : %<Nom du modèle>%

où <TemplateName> peut être l'un des gabarits répertoriés dans cette rubrique.

Lorsqu'un gabarit est référencé à partir d'un autre gabarit , il est généré par rapport aux éléments actuellement concernés. Le gabarit spécifique est sélectionné en fonction des stéréotypes des éléments concernés.

Comme indiqué précédemment, il existe une hiérarchie implicite entre les différents gabarits . Il convient de prendre certaines précautions afin de préserver une hiérarchie raisonnable des références gabarit . Par exemple, il n'est pas judicieux d'utiliser la macro %ClassInherits% dans l'un des gabarits Attribute ou Operation. Inversement, les gabarits Operation et Attribute sont conçus pour être utilisés dans le gabarit ClassBody.

## Macros de substitution Gabarit dans le CTF

- Attribut
- Déclaration d'attribut
- Déclaration d'attributImpl
- Notes sur les attributs
- Classe
- Base de classe
- Corps de classe
- ClasseBodyImpl
- Déclaration de classe
- Déclaration de classeImpl
- ClasseImpl
- La classe hérite
- Interface de classe
- Notes de cours
- Paramètre de classe
- Déposer
- FichierImpl
- Section d'importation
- ImportationSectionImpl
- Classe intérieure
- Implémentation de la classe intérieure
- Attribut lié
- Déclaration d'attribut lié
- Notes sur les attributs liés
- Base de classe liée
- Interface de classe liée
- Namespace
- Espace de nomsCorps

- Déclaration d'espace de noms
- Espace de nomsImpl
- Opération
- OpérationCorps
- OpérationBodyImpl
- Déclaration d'opération
- OpérationDéclarationImpl
- OpérationImpl
- Notes d'opération
- Paramètre

# Macros de substitution de champs

Les macros de substitution de champs permettent d'accéder aux données de votre modèle. Elles permettent notamment d'accéder aux champs de données de :

- Paquetages
- Cours
- Attributes
- Opérations et
- Paramètres

Les macros de substitution de champ sont nommées selon la casse Camel. Par convention, la macro est préfixée par une forme abrégée de l'élément de modèle correspondant. Par exemple, les macros liées aux attributs commencent par att, comme dans la macro %attName%, pour accéder au nom de l'attribut concerné.

Les macros qui représentent des cases à cocher renvoient une valeur de T si la case est sélectionnée. Sinon, la valeur est vide.

Ce tableau répertorie un petit nombre de macros de substitution de champs de projet. Les macros spécifiques au type sont répertoriées dans les sous-rubriques de cette section *Macros de substitution de champs* .

## Macros de projet

| Nom de la macro | Description |
|---|---|
| Date et heure | L'heure actuelle au format : JJ-MMM-AAAA HH:MM:SS AM/PM. |
| eaGUID | Un GUID unique pour cette génération. |
| Version ea | Version du programme (située dans la dialogue « À propos Enterprise Architect » en sélectionnant « Démarrer > Aide > Aide > À propos d'EA »). |

# Exemples de substitution

Les macros de substitution de champ peuvent être utilisées de deux manières :

- Substitution directe ou
- Substitution conditionnelle

## Substitution directe

Ce formulaire remplace directement la valeur correspondante de l'élément concerné dans la sortie.

Structure : %<macroName>%

Où <macroName> peut être l'une des macros répertoriées dans les tableaux Macros de substitution de champ.

## Exemples

- %Nom de classe%
- %opName%
- %attName%

## Substitution conditionnelle

Cette forme de macro permet de réaliser des substitutions alternatives en fonction de la valeur de la macro.

Structure : %<macroName> (== "<texte> " ) ? <subTrue> (: <subFalse>) %

Où:

- () indique que les valeurs entre parenthèses sont facultatives
- <text> est une string représentant une valeur possible pour la macro
- <subTrue> et <subFalse> peuvent être une combinaison de chaînes entre guillemets et du mot-clé valeur ; lorsque la valeur est utilisée, elle est remplacée par la valeur de la macro dans la sortie

## Exemples

- %classAbstract=="T" ? "pure" : " "%
- %opStereotype=="opérateur" ? "opérateur" : " "%
- %paramDefault != " " ? " = " valeur : " "%

Ces trois exemples ne génèrent aucun résultat si la condition échoue. Dans ce cas, la condition False peut être omise, ce qui donne cette utilisation :

- %classAbstract=="T" ? "pur"%
- %opStereotype=="opérateur" ? "opérateur"%
- %paramDefault != " " ? " = "valeur%

Le troisième exemple des deux blocs montre une comparaison vérifiant une valeur non vide ou une existence. Ce test peut également être omis.

- %paramDefault ? " = " valeur : " "%

- %paramDefault ? " = " valeur%

Tous ces exemples contenant paramDefault sont équivalents. Si le paramètre concerné avait une valeur par défaut de 10, la sortie de chacun d'eux serait normalement :

= 10

## Notes

- Dans une macro de substitution conditionnelle, tout espace blanc suivant <macroName> est ignoré ; si un espace blanc est requis dans la sortie, il doit être inclus dans les chaînes de substitution entre guillemets

# Macros de substitution de champs d'attributs

Ce tableau répertorie chacune des macros de substitution de champ d'attribut.

Les macros de substitution de champs sont nommées selon la casse Camel. Les macros qui représentent des cases à cocher renvoient une valeur de « T » si la case est sélectionnée. Sinon, la valeur est vide.

## Macros d'attributs

| Nom de la macro | Description |
|---|---|
| attAlias | dialogue « Attributes » : Alias. |
| attAutoriser les doublons | dialogue « Détails Attributes » : case à cocher « Autoriser les doublons ». |
| attClassifierGUID | Le GUID unique pour le classificateur de l'attribut actuel. |
| attCollection | dialogue « Détails Attributes » : case à cocher « L'attribut est une collection ». |
| attConst | Boîte dialogue « Attributes » : case à cocher « Const ». |
| attContainerType | dialogue « Détails Attributes » : Type de conteneur. |
| attConfinement | dialogue « Attributes » : Confinement. |
| attDérivé | dialogue « Attributes » : case à cocher « Dérivé ». |
| attGUID | Le GUID unique pour l'attribut actuel. |
| attInitial | dialogue « Attributes » : Initial. |
| attIsEnumLiteral | dialogue « Attributes » : case à cocher « Est littéral ». |
| ID d'identification d'attirance | Boîte de dialogue « Détails Attributes » : case à cocher « isID ». |
| attLongueur | dialogue « Colonne » : Durée. |
| attLimite inférieure | dialogue « Détails Attributes » : limite inférieure. |
| Nom d'utilisateur | dialogue « Attributes » : Nom. |
| attNotes | dialogue « Attributes » : Notes . |
| attOrderedMultiplicité | dialogue « Détails Attributes » : case à cocher « Multiplicité ordonnée ». |
| attProperty | dialogue « Attributes » : case à cocher « Propriété ». |
| attQualType | Le type d'attribut qualifié par le chemin de l'espace de noms (en cas de génération d'espaces de noms) et le chemin du classificateur (délimité par des points). Si le |

| | |
|---|---|
| | classificateur d'attribut n'a pas été défini, est équivalent à la macro attType. |
| attScope | dialogue « Attributes » : Portée. |
| attStatique | Boîte de dialogue « Attributes » : case à cocher « Statique ». |
| attStéréotype | dialogue « Attributes » : Stéréotype. |
| attType | dialogue « Attributes » : Type . |
| attLimite supérieure | dialogue « Détails Attributes » : limite supérieure. |
| attVolatile | Boîte de dialogue « Détails Attributes » : case à cocher « Transient ». |

# Macros de substitution de champs de classe

Ce tableau fournit une liste de méthodes permettant d'accéder à chaque propriété de classe disponible dans les gabarits de génération et de transformation de code.

Les macros de substitution de champs sont nommées selon la casse Camel. Les macros qui représentent des cases à cocher renvoient une valeur de « T » si la case est sélectionnée. Sinon, la valeur est vide.

## Macros de classe

| Nom de la macro | Description |
|---|---|
| élémentType | Le type d'élément : Interface ou Classe. |
| Résumé de la classe | dialogue Classe ' Propriétés ' : case à cocher 'Résumé' (onglet 'Détails'). |
| classeAlias | dialogue de la classe ' Propriétés ' : champ 'Alias'. |
| Arguments de classe | dialogue de classe « Détail » : C++ Gabarits : Arguments. |
| classeAuteur | dialogue de la classe ' Propriétés ' : champ 'Auteur'. |
| nom de base de la classe | dialogue « Hiérarchie Type » : Nom de la classe (à utiliser lorsqu'il n'existe aucun connecteur entre les classes enfant et de base). |
| classeBaseScope | La portée de l'héritage en tant qu'ingénierie inverse. (À utiliser lorsqu'il n'existe aucun connecteur entre les classes enfant et de base.) |
| classeBaseVirtual | La propriété virtuelle de l'héritage telle qu'elle a été conçue à rebours. (À utiliser lorsqu'il n'existe aucun connecteur entre les classes enfant et de base.) |
| classeComplexité | dialogue de la classe ' Propriétés ' : champ 'Complexité'. |
| classeCréé | La date et l'heure de création de la classe. |
| classeGUID | Le GUID unique pour la classe actuelle. |
| classeHasConstructor | Examine la liste des méthodes de l' object actuel et, selon les conventions du langage actuel, renvoie T si l'une d'elles est un constructeur par défaut. Généralement utilisé avec la macro genOptGenConstructor. |
| classeHasCopyConstructor | Examine la liste des méthodes de l' object actuel et, selon les conventions du langage actuel, renvoie T si l'une d'elles est un constructeur de copie. Généralement utilisé avec la macro genOptGenCopyConstructor. |
| classeHasDestructor | Examine la liste des méthodes de l' object actuel et, selon les conventions du langage actuel, renvoie T si l'une d'elles est un destructeur. Généralement utilisé avec la macro genOptGenDestructor. |
| classeHasParent | Vrai, si la classe dans la portée a une ou plusieurs classes de base. |
| | |

| | |
|---|---|
| classeHasStereotype | Vrai, si la classe dans la portée a un stéréotype qui correspond à un nom de stéréotype (que vous pouvez éventuellement spécifier comme entièrement qualifié). Il vérifie donc tous les stéréotypes qu'une classe possède et renvoie « T » si l'un d'entre eux correspond au stéréotype spécifié ou à une spécialisation de celui-ci. Par exemple :<br>• %classHasStereotype:"block"% renverra 'T' pour toute classe stéréotypée par bloc de n'importe quelle version SysML, y compris associationBlock<br>• %classHasStereotype:"SysML1.4::block"% correspondra spécifiquement aux versions SysML 1.4<br>Comparez ceci avec la classe Stereotype, plus tard. |
| Importations de classe | dialogue « Code Gen » : Importations. |
| classeIsActive | dialogue de classe « Avancé » : case à cocher « Est Actif ». |
| classeIsAssociationClass | Vrai, si l'Association est un connecteur AssociationClass. |
| classeIsInstanciated | Vrai, si la classe est une classe gabarit instanciée. |
| classeIsLeaf | dialogue de classe « Avancé » : case à cocher « Est une feuille ». |
| classeIsRoot | dialogue de classe « Avancé » : case à cocher « Est root ». |
| classeIsSpecification | dialogue de classe « Avancé » : case à cocher « Est-ce Spécification ». |
| classeMots-clés | Boîte dialogue Classe ' Propriétés ' : champ 'Mots clés'. |
| classeLangue | dialogue de la classe ' Propriétés ' : champ 'Langue'. |
| classeMacros | Une liste séparée par des espaces de macros définies pour la classe. |
| classeModifié | La date et l'heure de la dernière modification de la classe. |
| classeMultiplicité | dialogue de classe « Avancé » : Multiplicité. |
| nom de la classe | Boîte dialogue Classe ' Propriétés ' : champ 'Nom'. |
| Notes de cours | dialogue Classe ' Propriétés ' : champ ' Note '. |
| classeParamDefault | dialogue « Détail » de la classe. |
| nom du paramètre de classe | dialogue « Détail » de la classe. |
| classeParamType | dialogue « Détail » de la classe. |
| classePersistance | dialogue Classe ' Propriétés ' : Champ 'Persistance' (onglet 'Détails') |
| classePhase | Boîte dialogue Classe ' Propriétés ' : champ 'Phase'. |
| nomQualClasse | Le nom de la classe préfixé par ses classes externes. Les noms de classe sont séparés par deux points (::). |
| | |

| | |
|---|---|
| portée de la classe | dialogue Classe ' Propriétés ' : champ 'Scope'. |
| classeStéréotype | dialogue « Propriétés » de la classe : champ « Stéréotype ». Récupère le nom du premier stéréotype appliqué à la classe. Lorsqu'il est utilisé dans une comparaison, il vérifie si ce premier stéréotype correspond exactement à une string .<br>Par exemple : %classStereotype=="enumeration" ? "enum" : "class"%<br>Comparez ceci avec classHasStereotype, plus tôt. |
| Statut de la classe | Boîte dialogue Classe ' Propriétés ' : champ 'Statut'. |
| classeVersion | Boîte dialogue Classe ' Propriétés ' : champ 'Version'. |

# Option de génération de code Macros de substitution de champ

Les macros de substitution de champ d'option de génération de code fonctionnent sur les options de génération de code source définies dans les pages « Ingénierie du code source » de :

- dialogue « Préférences » (' Démarrer > Apparence > Préférences > Préférences > Ingénierie du code source') pour les options spécifiques à l'utilisateur, ou
- dialogue « Gérer les options Modèle » (« Paramètres > Modèle > Options ») pour les options spécifiques au modèle

Pour plus d'informations sur la division des options, consultez la rubrique *Options d'ingénierie du code source* .

Les macros de substitution de champ sont nommées selon la casse Camel. Les macros qui représentent des cases à cocher renvoient une valeur de « T » si la case est sélectionnée. Sinon, la valeur est vide. Ce tableau répertorie chacune des macros de substitution de champ d'option de génération de code.

## Macros d'options de génération de code

| Nom de la macro | Description |
|---|---|
| genOptActionScriptVersion | Page des spécifications ActionScript : version par défaut. |
| genOptCDefaultAttributeType | Page Spécifications C : Type d'attribut par défaut. |
| genOptCGenMethodNotesInBody | Page Spécifications C : Notes de méthode en cours d'implémentation. |
| genOptCGenMethodNotesInHeader | Page Spécifications C : Notes de méthode dans l'en-tête. |
| Notes de synchronisation genOptC | Page Spécifications C : Synchroniser Notes dans la génération. |
| fichier genOptCSynchCFile | Page Spécifications C : Synchroniser le fichier d'implémentation dans la génération. |
| genOptCDefaultSourceDirectory | Page des spécifications C : Répertoire source par défaut. |
| genOptCNamespaceDelimiter | Page de spécifications C : délimiteur Namespace . |
| genOptCOperationRefParam | Page Spécifications C : Référence comme paramètre de fonctionnement. |
| genOptCOperationRefParamStyle | Page Spécifications C : Style de paramètre de référence. |
| genOptCOperationRefParamName | Page Spécifications C : Nom du paramètre de référence. |

| | |
|---|---|
| genOptCConstructorName | Page Spécifications C : nom du constructeur par défaut. |
| genOptCDestructorName | Page des spécifications C : nom du destructeur par défaut. |
| genOptCPPCommentStyle | Page des spécifications C++ : Style de commentaire. |
| genOptCPPDefaultAttributeType | Page des spécifications C++ : Type d'attribut par défaut. |
| genOptCPPDefaultReferenceType | Page des spécifications C++ : Type de référence par défaut. |
| genOptCPPDefaultSourceDirectory | Page des spécifications C++ : Répertoire source par défaut. |
| genOptCPPGenMethodNotesInHeader | Page des spécifications C++ : case à cocher « Notes de méthode dans l'en-tête ». |
| genOptCPPGenMethodNotesInBody | Page des spécifications C++ : case à cocher Notes de méthode dans le corps. |
| genOptCPPGetPrefix | Page des spécifications C++ : obtenir le préfixe. |
| Extension de titre genOptCPPH | Page des spécifications C++ : extension d'en-tête. |
| genOptCPPSetPrefix | Page des spécifications C++ : définir le préfixe. |
| Extension de source genOptCPP | Page des spécifications C++ : extension de source. |
| Notes de synchronisation genOptCPPS | Page des spécifications C++ : Synchroniser Notes . |
| genOptCPPSynchCPPFile | Page des spécifications C++ : Synchroniser le fichier CPP. |
| genOptCSDefaultAttributeType | Page des spécifications C# : Type d'attribut par défaut. |
| Extension de source genOptCS | Page des spécifications C# : extension de fichier par défaut. |
| genOptCSGenDispose | Page Spécifications C# : Générer Dispose. |
| genOptCSGenFinalizer | Page Spécifications C# : Générer Finalizer. |
| genOptCSGenNamespace | Page Spécifications C# : Générer Namespace . |
| genOptCSDefaultSourceDirectory | Page des spécifications C# : Répertoire source par défaut. |
| genOptDefaultAssocAttNa | Page d'ingénierie du code source : nom par défaut pour l'attribut associé. |

| | |
|---|---|
| me | |
| genOptDefaultConstructorScope | Page Durées de vie Object : visibilité du constructeur par défaut. |
| genOptDefaultCopyConstructorScope | Page Durées de vie Object : visibilité du constructeur de copie par défaut. |
| Base de données genOptDefault | Page Éditeurs de Code : Base de données par défaut. |
| genOptDefaultDestructorScope | Page Durées de vie Object : Visibilité du constructeur du destructeur par défaut. |
| genOptGenCapitalizedProprietés | Page « Ingénierie du code source » : case à cocher « Mettre en majuscules les noms d'attributs pour Propriétés ». |
| genOptGenComments | Page 'Ingénierie du Code Source' : case à cocher 'Commentaires - Générer '. |
| genOptGenConstructeur | Page Durées de vie Object : case à cocher « Générer un constructeur ». |
| genOptGenConstructorInline | Page Durées de vie Object : case à cocher « Constructeur en ligne ». |
| genOptGenCopyConstructeur | Page Durées de vie Object : case à cocher « Générer un constructeur de copie ». |
| genOptGenCopyConstructorInline | Page Durées de vie Object : case à cocher « Copier le constructeur en ligne ». |
| genOptGenDestructeur | Page Durées de vie Object : case à cocher « Générer un destructeur ». |
| genOptGenDestructorInline | Page Durées de vie Object : case à cocher « Destructeur en ligne ». |
| genOptGenDestructorVirtuel | Page Durées de vie Object : case à cocher « Destructeur virtuel ». |
| genOptGenImplementedInterfaceOps | Page « Génération de code » : case à cocher « Générer des méthodes pour les interfaces implémentées ». |
| genOptGenPrefixBoolProperties | Page « Ingénierie du code source » : case à cocher « Utiliser « Est » pour la propriété booléenne Get() ». |
| genOptGenRoleNames | Page « Ingénierie du code source » : case à cocher « Générer automatiquement les noms de rôle lors de la création du code ». |
| genOptGenUnspecAssocDir | Page « Ingénierie du code source » : case à cocher « Ne pas générer de membres lorsque la direction de l'association n'est pas spécifiée ». |
| genOptJavaDefaultAttributeType | Page des spécifications Java : type d’attribut par défaut. |
| | |

| | |
|---|---|
| genOptJavaGetPrefix | Page des spécifications Java : obtenir le préfixe. |
| genOptJavaDefaultSource Directory | Page des spécifications Java : répertoire source par défaut. |
| genOptJavaSetPrefix | Page des spécifications Java : définir le préfixe. |
| Extension source genOptJava | Page des spécifications Java : extension du code source. |
| genOptPHPDefaultSource Directory | Page des spécifications PHP : Répertoire source par défaut. |
| genOptPHPGetPrefix | Page de spécifications PHP : obtenir le préfixe. |
| genOptPHPSetPrefix | Page de spécifications PHP : définir le préfixe. |
| Extension de source genOptPHP | Page de spécifications PHP : extension de fichier par défaut. |
| Version genOptPHP | Page des spécifications PHP : Version PHP. |
| Préfixe de propriété genOpt | Page « Ingénierie du code source » : supprimez les préfixes lors de la génération des propriétés Get/Set. |
| genOptVBMultiUse | Page Spécifications VB : case à cocher « Multi-usage ». |
| genOptVBPersistable | Page Spécifications VB : case à cocher « Persistant ». |
| Comportement de liaison de données genOptVB | Page des spécifications VB : case à cocher « Comportement de liaison de données ». |
| Comportement de genOptVBDataSource | Page Spécifications VB : case à cocher « Comportement de la source de données ». |
| genOptVBGlobal | Page des spécifications VB : case à cocher « Espace de noms global ». |
| genOptVBCréable | Page des spécifications VB : case à cocher « Créable ». |
| genOptVBExposed | Page Spécifications VB : case à cocher « Exposé ». |
| genOptVBMTS | Page des spécifications VB : mode de transaction MTS. |
| genOptVBNetGenNamespace | Page Spécifications VB.Net : Générer Namespace . |
| Version genOptVB | Page des spécifications VB : version par défaut. |
| Commentaire genOptWrap | Page « Ingénierie du code source » : longueur d'encapsulation pour les lignes de commentaires. |

# Macros de substitution de champs de connecteur

Ce tableau répertorie chacune des macros de substitution de champ de connecteur.

Les macros de substitution de champs sont nommées selon la casse Camel. Les macros qui représentent des cases à cocher renvoient une valeur de « T » si la case est sélectionnée. Sinon, la valeur est vide.

## Macros de connecteur

| Nom de la macro | Description |
|---|---|
| connecteurAlias | dialogue ' Propriétés ' du connecteur : champ 'Alias'. |
| connecteurAssociationClassElemGUID | Le GUID de l'élément de classe d'association du connecteur. |
| connecteurAssociationClassElemName | Le nom de l'élément de classe d'association du connecteur. |
| connecteurDestAccess | dialogue ' Propriétés ' du connecteur, onglet 'Rôle cible' : Accès. |
| connecteurDestAggregation | dialogue ' Propriétés ' du connecteur, onglet 'Rôle cible' : Agrégation. |
| connecteurDestAlias | dialogue ' Propriétés ' du connecteur, onglet 'Rôle cible' : Alias. |
| connecteurDestAllowDuplicates | dialogue « Propriétés » du connecteur, onglet « Rôle cible » : case à cocher « Autoriser les doublons ». |
| connecteurDestChangeable | dialogue ' Propriétés ' du connecteur, onglet 'Rôle cible' : Modifiable. |
| connecteurDestConstraint | dialogue ' Propriétés ' du connecteur, onglet 'Rôle cible' : Contrainte(s). |
| connecteurDestContainment | dialogue ' Propriétés ' du connecteur, onglet 'Rôle cible' : Confinement. |
| connecteurDestDerived | dialogue ' Propriétés ' du connecteur, onglet 'Rôle cible' : case à cocher 'Dérivé'. |
| connecteurDestDerivedUnion | dialogue ' Propriétés ' du connecteur, onglet 'Rôle cible' : case à cocher ' DerivedUnion '. |
| connecteurDestElem* | Un ensemble de macros qui accèdent à une propriété de l'élément à l'extrémité cible d'un connecteur. Le * (astérisque) est un caractère générique qui correspond à n'importe quelle macro de substitution de classe dans la liste des macros de classe. Par exemple :<br>• connecteurDestElemAlias (classAlias)<br>• connecteurDestElemAuthor (classAuthor) |
| connecteurDestElemType | Le type d'élément de l'élément de destination du connecteur. (Séparé des macros connectorDestElem* car il n'y a pas de macro de substitution classType.) |
| | |

| | |
|---|---|
| connecteurDestFeature* | Un ensemble de macros qui accèdent à une propriété de la fonctionnalité à l'extrémité cible d'un connecteur. Le * (astérisque) est un caractère générique qui correspond à toute macro de substitution d'attribut ou d'opération dans la liste des macros d'attribut ou d'opération, en fonction du connectorDestFeatureType.<br>Par exemple:<br>• connectorDestFeatureReturnClassifierGUID - GUID du classificateur de retour d'une opération<br>• connectorDestFeatureContainment - confinement d'un attribut |
| connecteurDestFeatureType | Le type de fonctionnalité de destination du connecteur.<br>• connectorDestFeatureType="Attribut" ou "Opération" |
| connecteurDestMemberType | dialogue ' Propriétés ' du connecteur, onglet 'Rôle cible' : Type de membre . |
| connecteurDestMultiplicité | dialogue ' Propriétés ' du connecteur, onglet 'Rôle cible' : Multiplicité. |
| connecteurDestNavigabilité | dialogue ' Propriétés ' du connecteur, onglet 'Rôle cible' : Navigabilité. |
| connecteurDestNotes | dialogue ' Propriétés ' du connecteur, onglet 'Rôle cible' : Notes sur le rôle . |
| connecteurDestOrdered | dialogue ' Propriétés ' du connecteur, onglet 'Rôle cible' : case à cocher 'Ordonné'. |
| connecteurDestOwned | dialogue ' Propriétés ' du connecteur, onglet 'Rôle cible' : case à cocher 'Possédé'. |
| connecteurDestQualifier | dialogue ' Propriétés ' du connecteur, onglet 'Rôle cible' : Qualificateur(s). |
| connecteurDestRole | dialogue ' Propriétés ' du connecteur, onglet 'Rôle cible' : Rôle. |
| connecteurDestScope | dialogue ' Propriétés ' du connecteur, onglet 'Rôle cible' : Périmètre cible. |
| connecteurDestStereotype | Connecteur dialogue ' Propriétés ', onglet ' Rôle cible ' : Stéréotype. |
| connecteurDirection | Propriétés du Connecteur : Direction. |
| connecteurEffet | dialogue « Contraintes de transition » : champ « Effet ». |
| connecteurGuard | Boîtes de dialogue « Flux Object » et « Contraintes de transition » : champ « Garde ». |
| connecteurGUID | Le GUID unique pour le connecteur actuel. |
| connecteurIsAssociationClass | Vrai, si le connecteur est un connecteur AssociationClass. |
| nom du connecteur | Propriétés du Connecteur : Nom. |
| connecteurNotes | Propriétés du Connecteur : Notes . |
| connecteurSourceAccess | dialogue ' Propriétés ' du connecteur, onglet 'Rôle source' : Accès. |
| | |

| | |
|---|---|
| connecteurSourceAggregation | dialogue ' Propriétés ' du connecteur, onglet 'Rôle source' : Agrégation. |
| connecteurSourceAlias | dialogue ' Propriétés ' du connecteur, onglet 'Rôle source' : Alias. |
| connecteurSourceAllowDuplicates | dialogue ' Propriétés ' du connecteur, onglet 'Rôle source' : case à cocher Autoriser les doublons. |
| connecteurSourceChangeable | dialogue ' Propriétés ' du connecteur, onglet 'Rôle source' : Modifiable. |
| connecteurSourceContraint | dialogue ' Propriétés ' du connecteur, onglet 'Rôle source' : Contrainte(s). |
| connecteurSourceContainment | dialogue ' Propriétés ' du connecteur, onglet 'Rôle source' : Confinement. |
| connecteurSourceDérivé | dialogue ' Propriétés ' du connecteur, onglet 'Rôle source' : case à cocher 'Dérivé'. |
| connecteurSourceDerivedUnion | dialogue ' Propriétés ' du connecteur, onglet 'Rôle source' : case à cocher ' DerivedUnion '. |
| connecteurSourceElem* | Un ensemble de macros qui accèdent à une propriété de l'élément à l'extrémité source d'un connecteur. Le * (astérisque) est un caractère générique qui correspond à n'importe quelle macro de substitution de classe dans la liste des macros de classe. Par exemple :<br>• connecteurSourceElemAlias (classAlias)<br>• connecteurSourceElemAuthor (classAuthor) |
| connecteurSourceElemType | Le type d'élément de l'élément source du connecteur. (Séparé des macros connectorSourceElem* car il n'y a pas de macro de substitution classType.) |
| connecteurSourceFonctionnalité* | Un ensemble de macros qui accèdent à une propriété de la fonctionnalité à l'extrémité source d'un connecteur. Le * (astérisque) est un caractère générique qui correspond à toute macro de substitution d'attribut ou d'opération dans la liste des macros d'attribut ou des macros d'opération, en fonction du connectorSourceFeatureType. Par exemple :<br>• connectorSourceFeatureCode - Code de l'opération<br>• connectorSourceFeatureInitial - Initiale de l'attribut |
| connecteurSourceFeatureType | Le type de fonctionnalité source du connecteur.<br>• connectorSourceFeatureType="Attribut" ou "Opération" |
| connecteurSourceMemberType | dialogue ' Propriétés ' du connecteur, onglet 'Rôle source' : Type de membre . |
| connecteurSourceMultiplicité | dialogue ' Propriétés ' du connecteur, onglet 'Rôle source' : Multiplicité. |
| connecteurSourceNavigabilité | dialogue ' Propriétés ' du connecteur, onglet 'Rôle source' : Navigabilité. |
| connecteurSourceNotes | dialogue ' Propriétés ' du connecteur, onglet 'Rôle source' : Notes sur le rôle . |
| | |

| | |
|---|---|
| connecteurSourceCommandé | dialogue ' Propriétés ' du connecteur, onglet 'Rôle source' : case à cocher 'Ordonné'. |
| connecteurSourcePropriété | dialogue ' Propriétés ' du connecteur, onglet 'Rôle source' : case à cocher 'Détenu'. |
| connecteurSourceQualifier | dialogue ' Propriétés ' du connecteur, onglet 'Rôle source' : Qualificateur(s). |
| connecteurSourceRole | dialogue ' Propriétés ' du connecteur, onglet 'Rôle source' : Rôle. |
| connecteurSourceScope | dialogue ' Propriétés ' du connecteur, onglet 'Rôle source' : Périmètre cible. |
| connecteurSourceStéréotype | dialogue ' Propriétés ' du connecteur, onglet 'Rôle source' : Stéréotype. |
| connecteurStéréotype | dialogue ' Propriétés ' du connecteur : champ 'Stéréotype'. |
| connecteurTrigger | dialogue 'Contraintes de Transition' : champ ' Déclencheur '. |
| Type de connecteur | Le type de connecteur ; par exemple, Association ou Généralisation. |
| connecteurPoids | dialogue « Contraintes de flux Object » : champ « Poids ». |

# Macros de substitution de champs de contraintes

Ce tableau répertorie chacune des macros de substitution de champ « Contrainte ».

Les macros de substitution de champs sont nommées selon la casse Camel. Les macros qui représentent des cases à cocher renvoient une valeur de « T » si la case est sélectionnée. Sinon, la valeur est vide.

## Macros de contraintes

| Nom de la macro | Description |
|---|---|
| nom de contrainte | dialogue « Classe », onglet « Contraintes » : Nom. |
| Notes sur les contraintes | dialogue « Classe », onglet « Contraintes » : Notes . |
| Statut de contrainte | dialogue « Classe », onglet « Contraintes » : Statut. |
| type de contrainte | dialogue 'Classe', onglet 'Contraintes' : Type . |
| contraintePoids | dialogue 'Classe', onglet 'Contraintes' : touches de classement (main en haut/en bas). |

# Macros de substitution de champs d'effort

Ce tableau répertorie chacune des macros de substitution de champ « Effort ».

Les macros de substitution de champs sont nommées selon la casse Camel. Les macros qui représentent des cases à cocher renvoient une valeur de « T » si la case est sélectionnée. Sinon, la valeur est vide.

## Macros d'effort

| Nom de la macro | Description |
|---|---|
| effortNom | Fenêtre d'effort : Effort. |
| effortNotes | Fenêtre d'effort : Notes (sans étiquette). |
| effortTemps | Fenêtre d'effort : Temps. |
| effortType | Fenêtre d'effort : Type . |

# Macros de substitution de champs de fichiers

Ce tableau répertorie chacune des macros de substitution de champ de fichier.

Les macros de substitution de champs sont nommées selon la casse Camel. Les macros qui représentent des cases à cocher renvoient une valeur de « T » si la case est sélectionnée. Sinon, la valeur est vide.

## Macros de fichiers

| Nom de la macro | Description |
|---|---|
| extension de fichier | L'extension du type de fichier du fichier en cours de génération. |
| nom de fichier | Le nom du fichier en cours de génération. |
| nom_fichierImpl | Le nom du fichier d'implémentation pour cette génération, le cas échéant. |
| en-têtes de fichier | dialogue « Code Gen » : en-têtes. |
| Importations de fichiers | dialogue « Code Gen » : Importations. Pour les langues prises en charge, cela inclut également les dépendances dérivées de ces types de relations :<br>• Agrégation<br>• Association<br>• Classificateur d'attributs<br>• Type de retour de la méthode<br>• Classificateur de paramètres de méthode<br>• Généralisation<br>• Réalisation (à l'interface)<br>• Liaison Gabarit (C++)<br>• Dépendance |
| chemin du fichier | Le chemin complet du fichier en cours de génération. |
| chemin du fichierImpl | Le chemin complet du fichier d'implémentation pour cette génération, le cas échéant. |

# Macros de substitution de champs d'importation de fichiers

Ce tableau répertorie chacune des macros de substitution de champ d'importation de fichier.

Les macros de substitution de champs sont nommées selon la casse Camel. Les macros qui représentent des cases à cocher renvoient une valeur de T si la case est sélectionnée. Sinon, la valeur est vide.

## Macros d'importation de fichiers

| Nom de la macro | Description |
|---|---|
| Nom de la classe d'importation | Le nom de la classe en cours d'importation. |
| importer le nom du fichier | Le nom de fichier de la classe en cours d'importation. |
| importerFilePath | Le chemin complet de la classe en cours d'importation. |
| importerDeL'agrégation | T si la classe possède un connecteur d'agrégation vers une classe dans ce fichier, F sinon. |
| importerDeL'Association | T si la classe a un connecteur d'association vers une classe dans ce fichier, F sinon. |
| importerDeAtt | T si un attribut d'une Classe dans le fichier courant est du type de cette Classe, F sinon. |
| importerDeDépendance | T si la classe a un connecteur de dépendance vers une classe dans ce fichier, F sinon. |
| importerDeGénéralités | T si la classe a un connecteur de généralisation vers une classe dans ce fichier, F sinon. |
| importerDeMeth | T si le type de retour d'une méthode d'une classe dans le fichier courant est le type de cette classe, F sinon. |
| importerDeParam | T si un paramètre de méthode d'une classe dans le fichier courant est du type de cette classe ; sinon F. |
| importerDePropertyType | T si la classe possède une propriété (Partie/Port) typant une autre classe, F sinon. |
| importerDeRéalisation | T si la classe possède un connecteur de réalisation vers une classe dans ce fichier, F sinon. |
| importerFromTemplateBinding | T si la classe a un connecteur TemplateBinding vers une classe dans ce fichier, F sinon. |
| importerDansFichier | T si la classe est dans le fichier courant, F sinon. |
| importerPackagePath | Le chemin Paquetage avec un séparateur « . » de la classe en cours d'importation. |

| | |
|---|---|
| ImporterRelativeFilePath | Le chemin d'accès relatif au fichier de la classe en cours d'importation à partir du chemin d'accès au fichier en cours de génération. |

# Macros de substitution de champs de liens

Si vous souhaitez donner accès aux données concernant les connecteurs du modèle, en particulier les associations et les généralisations, vous pouvez utiliser les macros « Substitution de champ de lien ». Les noms des macros sont en majuscules Camel. Les macros qui représentent des cases à cocher renvoient une valeur de « T » si la case est sélectionnée ; sinon, la valeur est vide.

## Liens macros

| Nom de la macro | Description/Résultat |
|---|---|
| lienAttAccess | dialogue ' Propriétés ' de l'association, Rôle cible : champ 'Accès'. |
| lienAttAggregation | Association dialogue ' Propriétés ', Rôle Source ou Cible : Agrégation. |
| lienAttCollectionClass | La collection appropriée à l'attribut lié dans la portée. |
| lienAttContainment | Association ' Propriétés ' dialogue , Rôle cible : Confinement. |
| lienAttName | dialogue 'Association Propriétés ' : Cible. |
| lienAttNotes | dialogue ' Propriétés ' de l'association, Rôle cible : Notes sur le rôle . |
| lienAttOwnedByAssociation | Vrai, si la case à cocher « Propriété » sur la page « Rôle(s) » de la dialogue « Propriétés » de l'Association n'est pas sélectionnée. |
| lienAttOwnedByClass | Vrai, si la case à cocher « Propriété » sur la page « Rôle(s) » de la dialogue « Propriétés » de l'Association est sélectionnée. |
| lienAttQualName | La cible de l'association qualifiée par le chemin de l'espace de noms (si vous générez des espaces de noms) et le chemin du classificateur (délimité par des points). |
| lienAttRole | dialogue ' Propriétés ' de l'association, Rôle cible : Rôle. |
| lienAttRoleAlias | dialogue 'Association Propriétés Rôle Cible' : Alias |
| lienAttStereotype | Association dialogue ' Propriétés ', Rôle cible : Stéréotype. |
| lienAttTargetScope | dialogue ' Propriétés ' de l'association, Rôle cible : Périmètre cible. |
| lienCarte | Lien dialogue ' Propriétés ', Rôle cible : Multiplicité. |
| lienGUID | Le GUID unique pour le connecteur actuel. |
| lienIsAssociationClass | Vrai, si l'Association est un connecteur AssociationClass. |
| lienEstLié | Renvoie T si des TemplateBindings sont spécifiés sur le connecteur. |
| lienParamSubs | Renvoie une liste séparée par des virgules des arguments spécifiés. |

| | |
|---|---|
| lienParentName | Généralisation dialogue ' Propriétés ' : champ 'Cible'. |
| lienParentQualName | La cible de généralisation qualifiée par le chemin de l'espace de noms (si vous générez des espaces de noms) et le chemin du classificateur (délimité par des points). |
| lienStéréotype | Le stéréotype du connecteur actuel. |
| lienHéritage Virtuel | Généralisation dialogue ' Propriétés ' : champ 'Héritage Virtuel'. |

# Macros de substitution de champs de fichiers liés

Ce tableau répertorie chacune des macros de substitution de champ « Fichier lié ».

Les macros de substitution de champs sont nommées selon la casse Camel. Les macros qui représentent des cases à cocher renvoient une valeur de « T » si la case est sélectionnée. Sinon, la valeur est vide.

## Macros de fichiers liés

| Nom de la macro | Description |
|---|---|
| fichier liéDernière écriture | dialogue Classe ' Propriétés ' : onglet 'Fichiers', champ 'Dernière écriture'. |
| Notes de fichiers liés | dialogue Classe ' Propriétés ' : onglet ' Fichiers ', champ ' Notes '. |
| chemin du fichier lié | dialogue Classe ' Propriétés ' : onglet 'Fichiers', champ 'Chemin du fichier'. |
| Taille du fichier lié | dialogue Classe ' Propriétés ' : onglet 'Fichiers', champ 'Taille'. |
| Type de fichier lié | dialogue Classe ' Propriétés ' : onglet ' Fichiers ', champ ' Type '. |

# Macros de substitution de champs métriques

Ce tableau répertorie chacune des macros de substitution de champ métrique.

Les macros de substitution de champs sont nommées selon la casse Camel. Les macros qui représentent des cases à cocher renvoient une valeur de « T » si la case est sélectionnée. Sinon, la valeur est vide.

## Macros métriques

| Nom de la macro | Description |
|---|---|
| nom métrique | Écran Métriques : champ « Métrique ». |
| Notes métriques | Écran Métriques : champ ( Notes ). |
| type métrique | Écran métriques : champ « Type ». |
| Poids métrique | Écran de métriques : champ « Poids ». |

# Macros de substitution de champs d'opération

Les macros 'Substitution de champs d'opération' permettent d'accéder aux données concernant les opérations dans le modèle. Les noms des macros sont en majuscules Camel. Les macros qui représentent des cases à cocher renvoient une valeur de 'T' si la case est sélectionnée ; sinon la valeur est vide.

## Macros de substitution de champs d'opération

| Nom de la macro | Description/Résultat |
|---|---|
| opRésumé | dialogue « Opération » : case à cocher « Virtuel ». |
| opAlias | dialogue « Opération » : Alias. |
| comportementop | dialogue « Opération Comportement » : Comportement. |
| Code d'opération | dialogue « Opération Comportement » : Code de comportement. |
| OpConcurrence | dialogue « Opération » : Concurrence. |
| opConst | dialogue « Opération » : case à cocher « Const ». |
| GUID d'opération | Le GUID unique pour l'opération en cours. |
| opHasSelfRefParam | Analyse la liste des paramètres dans l'opération en cours, renvoyant « T » si un type est la référence de classe (cela peut être ClassA* ou ClassA&, selon la valeur de la macro de substitution de champ d'option de génération de code genOptCOperationRefParamStyle). |
| opImplMacros | Une liste séparée par des espaces de macros définies dans l'implémentation de cette opération. |
| opIsRequête | Boîte de dialogue « Opération » : case à cocher « IsQuery ». |
| macros opérationnelles | Une liste séparée par des espaces de macros définies dans la déclaration pour cette opération. |
| nom de l'opération | dialogue « Opération » : Nom. |
| Notes d'opération | dialogue « Opération » : Notes . |
| opPure | dialogue « Opération » : case à cocher « Pure ». |
| opReturnArray | Boîte de dialogue « Opération » : case à cocher « Renvoyer le tableau ». |
| opReturnClassifierGUID | Le GUID unique pour le classificateur de l'opération en cours. |
| opReturnQualType | Le type de retour de l'opération qualifié par le chemin de l'espace de noms (en cas de génération d'espaces de noms) et le chemin du classificateur (délimité par des points). Si le classificateur de type de retour n'a pas été défini, il est équivalent à la |

| | |
|---|---|
| | macro opReturnType. |
| opReturnType | dialogue « Opération » : Type de retour . |
| portée opérationnelle | dialogue « Opération » : Portée. |
| opStatique | dialogue « Opération » : case à cocher « Statique ». |
| opStéréotype | dialogue « Opération » : Stéréotype. |
| opSynchronisé | dialogue « Opération » : case à cocher « Synchronisé ». |

# Macros de substitution de champs de Paquetage

Ce tableau répertorie les macros de substitution de champ Paquetage .

Les macros de substitution de champs sont nommées selon la casse Camel. Les macros qui représentent des cases à cocher renvoient une valeur de « T » si la case est sélectionnée. Sinon, la valeur est vide.

## Paquetage de macros

| Nom de la macro | Description |
|---|---|
| packageRésumé | dialogue ' Paquetage ' : Résumé. |
| Alias de paquet | dialogue ' Paquetage ' : Alias. |
| packageAuteur | dialogue ' Paquetage ' : Auteur. |
| Complexité du package | dialogue ' Paquetage ' : Complexité. |
| packageGUID | Le GUID unique pour le Paquetage actuel. |
| packageMots-clés | dialogue ' Paquetage ' : Mots-clés. |
| packageLangue | dialogue ' Paquetage ' : Langue. |
| nom du package | dialogue ' Paquetage ' : Nom. |
| chemin du paquet | La string représentant la hiérarchie des Paquetages , pour la classe concernée. Chaque nom Paquetage est séparé par un point (.). |
| paquetPhase | dialogue ' Paquetage ' : Phase. |
| portée du paquet | dialogue ' Paquetage ' : Portée. |
| Statut du paquet | dialogue ' Paquetage ' : Statut. |
| paquetStéréotype | dialogue ' Paquetage ' : Stéréotype. |
| Version du paquet | dialogue ' Paquetage ' : Version. |

# Macros de substitution de champs de paramètres

Ce tableau répertorie chacune des macros de substitution de champ de paramètre.

Les macros de substitution de champs sont nommées selon la casse Camel. Les macros qui représentent des cases à cocher renvoient une valeur de « T » si la case est sélectionnée. Sinon, la valeur est vide.

## Macros de paramètres

| Nom de la macro | Description |
|---|---|
| paramètreClassifierGUID | Le GUID unique pour le classificateur du paramètre actuel. |
| paramètre par défaut | Opération dialogue « Paramètres » : champ « Par défaut ». |
| paramètreFixé | Boîte de dialogue « Paramètres » : case à cocher « Fixe ». |
| paramètreGUID | Le GUID unique pour le paramètre actuel. |
| paramètreIsEnum | Vrai, si le paramètre utilise le mot-clé enum (C++). |
| paramètreKind | Opération dialogue « Paramètres » : champ « Type ». |
| nom_paramètre | Opération dialogue « Paramètres » : champ « Nom ». |
| Notes de paramètre | Opération dialogue « Paramètres » : champ « Notes ». |
| paramètreQualType | Le type de paramètre qualifié par le chemin de l'espace de noms (en cas de génération d'espaces de noms) et le chemin du classificateur (délimité par des points). Si le classificateur de paramètres n'a pas été défini, est équivalent à la macro paramType. |
| paramètreType | Opération dialogue « Paramètres » : champ « Type ». |

# Macros de substitution de champs problématiques

Ce tableau répertorie chacune des macros de substitution de champ Problème.

Les macros de substitution de champs sont nommées selon la casse Camel. Les macros qui représentent des cases à cocher renvoient une valeur de « T » si la case est sélectionnée. Sinon, la valeur est vide.

## Problèmes de macros

| Nom de la macro | Description |
|---|---|
| problèmeTerminé par | dialogue « Maintenance », onglet « Problèmes d'éléments » : Terminé par. |
| problèmeCompletedDate | dialogue « Maintenance », onglet « Problèmes d'élément » : Terminé. |
| problèmeHistoire | dialogue « Maintenance », onglet « Problèmes d'éléments » : Historique. |
| nom du problème | dialogue « Maintenance », onglet « Problèmes d'élément » : Nom. |
| ProblèmeNotes | dialogue « Maintenance », onglet « Problèmes d'éléments » : Description. |
| problèmePriorité | dialogue « Maintenance », onglet « Problèmes d'éléments » : Priorité. |
| problèmeSoulevé par | dialogue « Maintenance », onglet « Problèmes d'élément » : soulevé par. |
| problèmeRaisedDate | dialogue « Maintenance », onglet « Problèmes d'élément » : surélevé. |
| Statut du problème | dialogue « Maintenance », onglet « Problèmes d'élément » : Statut. |
| problèmeVersion | dialogue « Maintenance », onglet « Problèmes d'élément » : Version. |

# Macros de substitution de champs d'exigences

Ce tableau répertorie chacune des macros de substitution de champ d'exigence avec une description du résultat.

Les macros de substitution de champs sont nommées selon la casse Camel. Les macros qui représentent des cases à cocher renvoient une valeur de « T » si la case est sélectionnée. Sinon, la valeur est vide.

## Macros d'exigences

| Nom de la macro | Description |
|---|---|
| exigenceDifficulté | dialogue ' Propriétés ' : Onglet 'Exiger' : Difficulté. |
| exigenceDernière mise à jour | dialogue ' Propriétés ' : Onglet 'Exiger' : Dernière mise à jour. |
| exigenceNom | dialogue ' Propriétés ' : Onglet 'Exiger' : Brève description. |
| ExigenceRemarques | dialogue ' Propriétés ' : Onglet 'Exiger' : Notes . |
| exigencePriorité | dialogue ' Propriétés ' : Onglet 'Exiger' : Priorité. |
| exigenceStatut | dialogue ' Propriétés ' : Onglet 'Exiger' : Statut. |
| exigenceType | Boîte dialogue ' Propriétés ' : Onglet 'Exiger' : Type . |

# Macros de substitution de champs de ressources

Ce tableau répertorie chacune des macros de substitution de champ de ressource.

Les macros de substitution de champs sont nommées selon la casse Camel. Les macros qui représentent des cases à cocher renvoient une valeur de « T » si la case est sélectionnée. Sinon, la valeur est vide.

## Macros de ressources

| Nom de la macro | Description |
|---|---|
| ressourceAllocatedTime | Fenêtre Allocation des ressources : Alloué Time. |
| Date de fin de la ressource | Fenêtre d'allocation des ressources : date de fin. |
| ressourceExpectedTime | Fenêtre d'allocation des ressources : temps prévu. |
| ressourceTemps dépensé | Fenêtre d'allocation des ressources : temps dépensé. |
| Histoire des ressources | Fenêtre d'allocation des ressources : Historique. |
| nom de la ressource | Fenêtre d'allocation des ressources : Ressource. |
| Notes sur les ressources | Fenêtre d'allocation des ressources : description. |
| ressourcePourcentage terminé | Fenêtre d'allocation des ressources : Terminée (%). |
| rôle de ressource | Fenêtre d'allocation des ressources : Rôle. |
| Date de début de la ressource | Fenêtre Allocation des ressources : Date Démarrer . |

# Macros de substitution de champs de risque

Ce tableau répertorie chacune des macros de substitution de champ de risque.

Les macros de substitution de champs sont nommées selon la casse Camel. Les macros qui représentent des cases à cocher renvoient une valeur de « T » si la case est sélectionnée. Sinon, la valeur est vide.

## Macro-risques

| Nom de la macro | Description |
|---|---|
| nom du risque | Fenêtre Risques : Risque. |
| Notes sur les risques | Fenêtre Risques : ( Notes ). |
| Type de risque | Fenêtre Risques : Type . |
| risquePoids | Fenêtre Risques : Poids. |

# Macros de substitution de champs de scénario

Ce tableau répertorie chacune des macros de substitution de champ de scénario avec une description du résultat.

Les macros de substitution de champs sont nommées selon la casse Camel. Les macros qui représentent des cases à cocher renvoient une valeur de « T » si la case est sélectionnée. Sinon, la valeur est vide.

## Macros de scénario

| Nom de la macro | Description |
|---|---|
| scénarioGUID | L' ID unique d'un scénario. Identifie le scénario sans ambiguïté dans un modèle. |
| nom du scénario | dialogue ' Propriétés ', onglet 'Scénario' : Scénario. |
| Notes de scénario | Boîte dialogue ' Propriétés ', onglet 'Scénario' : ( Notes ). |
| Type de scénario | Boîte dialogue ' Propriétés ', onglet 'Scénario' : Type . |

# Macros de substitution Valeur Étiquetée

Les macros Valeur Étiquetée sont une forme spéciale de macros de substitution de champ, qui donnent accès aux étiquettes d'éléments et aux Valeur Étiquetés correspondantes. Ils peuvent être utilisés de deux manières :

- Substitution directe
- Substitution conditionnelle

## Substitution directe

Cette forme de macro remplace directement la valeur de l' étiquette nommée dans la sortie.

Structure : %<macroName>:"<tagName>"%

<macroName> peut être l'un des suivants :

- Étiquette d'att
- Étiquette de classe
- connecteurDestElemTag
- connecteurDestTag
- connecteurSourceElemTag
- connecteurSourceTag
- connecteurTag
- lienAttTag
- lienTag
- balise d'op
- PaquetTag
- paramètreTag

Cela correspond aux étiquettes pour les attributs, les classes, les opérations, Paquetages , les paramètres, les connecteurs avec les deux extrémités, les éléments aux deux extrémités des connecteurs et les connecteurs incluant l'extrémité de l'attribut.

<tagName> est une string représentant le nom étiquette spécifique.

## Exemple

%opTag:"attribut"%

## Substitution conditionnelle

Cette forme de macro imite la substitution conditionnelle définie pour les macros de substitution de champ.

Structure : %<macroName>:"<tagName> " (== "<test> " ) ? <subTrue> (: <subFalse>) %

Note :

- <macroName> et <tagName> sont tels que définis ici
- (<texte>) indique que <texte> est facultatif
- <test> est une string représentant une valeur possible pour la macro

- <subTrue> et <subFalse> peuvent être une combinaison de chaînes entre guillemets et du mot-clé valeur ; lorsque la valeur est utilisée, elle est remplacée par la valeur de la macro dans la sortie

## Exemples

%opTag : "opInline" ? "en ligne" : " "%

%opTag:"opInline" ? "inline"%

%classTag:"unsafe" == "true" ? "unsafe" : " "%

%classTag:"unsafe" == "true" ? "unsafe"%

Les macros Valeur Étiquetée utilisent la même convention de dénomination que les macros de substitution de champs.

# Macros de substitution de paramètres Gabarit

Si vous souhaitez donner accès dans un gabarit de transformation aux données concernant la transformation de la substitution de paramètres de liaison d'un connecteur Gabarit Binding dans le modèle, vous pouvez utiliser les macros de substitution de paramètres Gabarit . Les noms des macros sont en majuscules Camel. Les macros qui représentent des cases à cocher renvoient une valeur de 'T' si la case est sélectionnée ; sinon, la valeur est vide.

## Macros de substitution de paramètres Gabarit

| Nom de la macro | Description |
|---|---|
| paramètreSubstitutionFormelle | dialogue « Propriétés de liaison Gabarit , onglet « Paramètre de liaison », panneau « Substitution(s) de paramètres » : Nom formel du paramètre Gabarit . |
| paramètreSubstitutionActuel | dialogue « Propriétés de liaison Gabarit , onglet « Paramètre de liaison », panneau « Substitution(s) de paramètre » : nom/expression du paramètre réel. |
| paramètreSubstitutionActualClassifier | dialogue ' Gabarit Binding Propriétés ', onglet 'Paramètre de liaison', panneau 'Substitution(s) de paramètres' : classificateur de paramètres réel. |

# Macros de substitution de champs Test

Ce tableau répertorie chacune des macros de substitution de champ Test avec une description du résultat.

Les macros de substitution de champs sont nommées selon la casse Camel. Les macros qui représentent des cases à cocher renvoient une valeur de « T » si la case est sélectionnée. Sinon, la valeur est vide.

## Test les macros

| Nom de la macro | Description |
|---|---|
| Critères d'acceptation du test | dialogue Tester : Critères d'acceptation. |
| testVérifié par | Fenêtre Cas Test : Vérifié par. |
| Date d'exécution du test | Fenêtre Cas Test : Dernier Exécuter . |
| classe de test | Fenêtre Cas Test : Classe Test (le type de test défini : Unité, Intégration, Système, Acceptation, Inspection, Scénario) |
| testEntrée | dialogue Tester : Entrée. |
| nom du test | Fenêtre Cas Test : Test . |
| Notes de test | Fenêtre Cas Test : Description. |
| Résultats du test | dialogue Tester : Résultats. |
| testExécuté par | Fenêtre Cas Test : Exécuter par. (Les valeurs sont dérivées des définitions de l'auteur du projet dans la dialogue « Personnes » - « Paramètres > Données de référence > Types Modèle > Personnes > Auteurs du projet ».) |
| Statut du test | Fenêtre Cas Test : État. |
| Type de test | Fenêtre Cas Test : Type . |

# Macros de fonctions

Les macros de fonction sont un moyen pratique de manipuler et de formater divers éléments de données. Chaque macro de fonction renvoie une string de résultat. Il existe deux manières principales d'utiliser les résultats des macros de fonction :

- Substitution directe de la string renvoyée dans la sortie, par exemple : %TO_LOWER(attName)%
- Stockage de la string renvoyée dans le cadre d'une définition de variable telle que : $name = %TO_LOWER(attName)%

Les macros de fonction peuvent prendre des paramètres, qui peuvent être transmis aux macros comme suit :

- Littéraux String , placés entre guillemets doubles
- Macros de substitution directe sans les signes de pourcentage qui les entourent
- Références variables
- Littéraux numériques

Plusieurs paramètres sont transmis à l'aide d'une liste séparée par des virgules.

Les macros de fonction sont nommées selon le style Tout en majuscules, comme dans :

%CONVERT_SCOPE(opScope)%

Les macros de fonctions disponibles sont décrites ici. Les paramètres sont indiqués par des crochets, comme dans :

FONCTION_NOM([param]).

## CONVERTIR_SCOPE([umlScope])

À utiliser avec les langues prises en charge pour convertir [umlScope] en mot-clé de portée approprié pour la langue en cours de génération. Ce tableau montre la conversion de [umlScope] par rapport à la langue donnée.

| Langue | Conversions |
|---|---|
| C++ | Paquetage ==> public<br>Public ==> public<br>Privé ==> privé<br>Protégé ==> protégé |
| C# | Paquetage ==> interne<br>Public ==> public<br>Privé ==> privé<br>Protégé ==> protégé |
| Delphes | Paquetage ==> protégé<br>Public ==> public<br>Privé ==> privé<br>Protégé ==> protégé |
| Java | Paquetage ==> {vide}<br>Public ==> public<br>Privé ==> privé<br>Protégé ==> protégé |
| | |

| | |
|---|---|
| PHP | Paquetage ==> public<br>Public ==> public<br>Privé ==> privé<br>Protégé ==> protégé |
| VB | Paquetage ==> Protégé<br>Public ==> Public<br>Privé ==> Privé<br>Protégé ==> Protégé |
| VB .Net | Paquetage ==> Ami<br>Public ==> Public<br>Privé ==> Privé<br>Protégé ==> Protégé |

## COLLECTION_CLASS([langue])

Donne la classe de collection appropriée pour la langue spécifiée pour l'attribut lié actuel.

## CSTYLE_COMMENT([longueur_enveloppe])

Convertit les notes de l'élément actuellement dans la portée en commentaires simples de style C, en utilisant /* et */.

## DELPHI_PROPERTIES([portée], [separator] , [retrait])

Génère une propriété Delphi.

## DELPHI_COMMENT([wrap_length])

Convertit les notes de l'élément actuellement concerné en commentaires Delphi.

## EXEC_ADD_IN(, [nom_fonction],, ...,)

Appelle une fonction Add-In Enterprise Architect , qui peut renvoyer une string de résultat.

[addin_name] et [function_name] spécifient les noms du Add-In et de la fonction à appeler.

Les paramètres de la fonction Add-In peuvent être spécifiés via les paramètres [prm_1] à [prm_n].

$result = %EXEC_ADD_IN("MonAddin", "ProcessOperation", classGUID, opGUID)%

Toute fonction appelée par la macro EXEC_ADD_IN doit avoir deux paramètres : un object EA.Repository et un tableau Variant contenant tous les paramètres supplémentaires de l'appel EXEC_ADD_IN. Le type de retour doit être Variant.

Fonction publique ProcessOperation( Référentiel As EA.Repository , args As Variant) As Variant

## RECHERCHER([src], [sous-chaîne])

Position de la première instance de [subString] dans [src] ; -1 si aucune.

## OBTENIR_ALIGNEMENT()

Renvoie une string dans laquelle tout le texte de la ligne de sortie actuelle est converti en espaces et tabulations.

## JAVADOC_COMMENT([wrap_length])

Convertit les notes de l'élément actuellement dans la portée en commentaires de style javadoc.

## GAUCHE([src], [compte])

Les [count] premiers caractères de [src].

## LONGUEUR([src])

Longueur de [src]. Renvoie une string .

## MATH_ADD(x,y) MATH_MULT(x,y) et MATH_SUB(x,y)

Dans un gabarit de code ou gabarit DDL, ces trois macros exécutent respectivement les fonctions mathématiques de :

- Addition (x+y)
- Multiplication (x*y) et
- Soustraction (xy)

Les arguments x et y peuvent être des entiers ou des variables, ou une combinaison des deux. Considérez ces exemples, tels qu'ils sont utilisés dans un gabarit « Class » pour la génération de code C++ :

- $a = %MATH_ADD(3,4)%
- $b = %MATH_SUB(10,3)%
- $c = %MATH_MULT(2,3)%
- $d = %MATH_ADD($a,$b)%
- $e = %MATH_SUB($b,$c)%
- $f = %MATH_MULT($a,$b)%
- $g = %MATH_MULT($a,10)%
- $h = %MATH_MULT(10,$b)%

Ceux-ci calculent, dans la même séquence, à :

- a = 3 + 4 = $a

- b = 10 - 3 = $b
- c = 2 * 3 = $c
- d = a + b = $d
- e = b - c = $e
- f = a * b = $f
- g = a * 10 = $g
- h = 10 * b = $h

Lorsque le code est généré, le fichier .h (pour C++) contient ces chaînes correspondantes :

- a = 3 + 4 = 7
- b = 10 - 3 = 7
- c = 2 * 3 = 6
- d = a + b = 14
- e = b - c = 1
- f = a * b = 49
- g = a * 10 = 70
- h = 10 * b = 70

## MID([src], [début]) MID([src], [début], [nombre])

Sous-chaîne de [src] commençant à [start] et incluant [count] caractères. Lorsque [count] est omis, le reste de la string est inclus.

## PI([option], [ valeur ], {[option], [ valeur ]})

Ensembles le PI pour le gabarit actuel à [ valeur ]. Les valeurs valides pour [ valeur ] sont :

- " \n"
- " "
- « «
- « »

<option> contrôle le moment où le nouveau PI prend effet. Les valeurs valides pour <option> sont :

- I, Immédiat : le nouveau PI est généré avant la prochaine ligne gabarit non vide
- N, Suivant : le nouveau PI est généré après la prochaine ligne gabarit non vide

Plusieurs paires d'options sont autorisées dans un appel. Un exemple de situation où cette fonction serait utilisée est celle où un mot clé est toujours sur une nouvelle ligne, comme illustré ici :

%PI=" "%

%classAbstract ? "abstrait"%

%si classTag:"macro" != " "%

%PI("Je", " \n", "N", " " )%

%classTag:"macro"%

%finSi%

classe

%Nom de classe%

Pour plus de détails, voir *la macro Instruction de traitement (PI)* .

## PROCESS_END_OBJECT([nom_modèle])

Permet aux classes qui sont une classe plus éloignée de la classe de base, d'être transformées en objets (tels que des attributs, des opérations, Paquetages , des paramètres et des colonnes) de la classe de base. [template_name] fait référence au gabarit de travail qui stocke temporairement les données.

## SUPPRIMER LES DUPLICATS([source], [separator] )

Où [source] est une liste séparée par [separator] ; cela supprime toutes les chaînes en double ou vides.

## REMPLACER([ string ], [ancien], [nouveau])

Remplace toutes les occurrences de [old] par [new] dans la string donnée <string>.

## RESOLVE_OP_NAME()

Résout les conflits dans les noms d'interface lorsque deux interfaces de méthode ont le même nom.

## RESOLVE_QUALIFIED_TYPE() RESOLVE_QUALIFIED_TYPE( [separator] ) RESOLVE_QUALIFIED_TYPE( [separator] , [par défaut])

Génère un type qualifié pour l'attribut actuel, l'attribut lié, le parent lié, l'opération ou le paramètre. Permet la spécification d'un séparateur autre que. et une valeur par défaut lorsque certaines valeur sont requises.

## DROITE([src], [compte])

Les [count] derniers caractères de [src].

## TO_LOWER([ string ])

Convertit [ string ] en minuscules.

## TO_UPPER([ string ])

Convertit [ string ] en majuscules.

## TRIM([ string ]) TRIM([ string ], [trimChars])

Supprime les espaces blancs de début et de fin de [ string ]. Si [trimChars] est spécifié, tous les caractères de début et de fin de l'ensemble de <trimChars> sont supprimés.

## TRIM_LEFT([ string ]) TRIM_LEFT([ string ], [trimChars])

Supprime les caractères de début spécifiés de <string>.

## TRIM_RIGHT([ string ]) TRIM_RIGHT([ string ], [trimChars])

Supprime les caractères de fin spécifiés de <string>.

## VB_COMMENT([longueur_d'enveloppement])

Convertit les notes de l'élément actuellement dans la portée en commentaires de style Visual Basic.

## WRAP_COMMENT([commentaire], [wrap_length], [indent], [start_string])

Encapsule le texte [commentaire] à une largeur [wrap_length] en plaçant [indent] et [start_string] au début de chaque ligne.

$comportement = %WRAP_COMMENT(opBehavior, "40", " " , " // " )%

<wrap_length> doit toujours être passé sous forme de string , même si WRAP_COMMENT traite ce paramètre comme un integer .

## WRAP_LINES([texte], [longueur_enveloppe], [chaîne_début] {, [chaîne_fin] })

Encapsule [texte] comme indiqué pour être [wrap_length], en ajoutant [start_string] au début de chaque ligne et [end_string] à la fin de la ligne si cela est spécifié.

## XML_COMMENT([wrap_length])

Convertit les notes de l'élément actuellement concerné en commentaires de style XML.

# Contrôle des macros

Les macros de contrôle sont utilisées pour contrôler le traitement et le formatage des gabarits . Les types de macros de contrôle de base incluent :

- La macro de liste, pour générer plusieurs fonctionnalités d'éléments, telles que des attributs et des opérations
- Les macros de ramification, qui forment des constructions if-then-else pour exécuter conditionnellement des parties d'un gabarit
- La macro PI pour formater les nouvelles lignes dans la sortie, qui prend effet à partir de la prochaine ligne non vide
- Une macro de fonction PI qui permet de définir PI sur une variable et ajoute la possibilité de définir le PI qui est généré avant la ligne suivante
- Les macros de synchronisation

En général, les macros de contrôle sont nommées selon la casse Camel.

# Liste des macros

Si vous devez parcourir en boucle ou itérer sur un ensemble d'objets contenus dans ou sous l' object actuel, vous pouvez le faire à l'aide de la macro *%list* . Cette macro effectue un passage itératif sur tous les objets dans la portée du gabarit actuel et appelle un autre gabarit pour traiter chacun d'eux.

La structure de base est la suivante :

%list=<Nom du modèle> @separator=<chaîne> @indent=<chaîne> (<conditions>) %

où <string> est une string littérale entre guillemets et <TemplateName> peut être l'un de ces noms gabarit :

- Attribut
- AttributImpl
- Classe
- Base de classe
- ClasseImpl
- Initialiseur de classe
- Interface de classe
- Contrainte
- Gabarit personnalisé ( gabarits personnalisés vous permettent de définir vos propres gabarits )
- Effort
- Classe intérieure
- Implémentation de la classe intérieure
- Fichier lié
- Métrique
- Namespace
- Opération
- OpérationImpl
- Paramètre
- Problème
- Exigence
- Ressource
- Risque
- Scénario
- Test

<conditions> est facultatif et ressemble aux conditions des instructions « if » et « elseIf ».

## Exemple

Dans une transformation de classe, la classe peut contenir plusieurs attributs ; cet exemple appelle la transformation d'attribut et génère le résultat du traitement de la transformation pour chaque attribut de la classe concernée. La liste résultante sépare ses éléments par une seule nouvelle ligne et les indente respectivement de deux espaces. Si la classe concernée avait des attributs stéréotypés, ils seraient générés à l'aide du gabarit spécialisé approprié.

%list="Attribut" @separator="\n" @indent=" "%

L'attribut séparateur, désigné par @separator, spécifie l'espace qui doit être utilisé entre les éléments de la liste, à l'exclusion du dernier élément de la liste.

L'attribut indent, indiqué par @indent, spécifie l'espace par lequel chaque ligne de la sortie générée doit être indentée.

## Cas particuliers

Il y a quelques cas particuliers à prendre en compte lors de l'utilisation de la macro %list :

- Si le gabarit d'attribut est utilisé comme argument de la macro %list, cela génère également des attributs dérivés des associations en exécutant le gabarit LinkedAttribute approprié
- Si le gabarit ClassBase est utilisé comme argument de la macro %list, cela génère également des bases de classe dérivées des liens dans le modèle en exécutant le gabarit LinkedClassBase approprié
- Si le gabarit ClassInterface est utilisé comme argument de la macro %list, cela génère également des bases de classe dérivées des liens dans le modèle en exécutant le gabarit LinkedClassInterface approprié
- Si InnerClass ou InnerClassImpl est utilisé comme argument de la macro %list, ces classes sont générées à l'aide des gabarits Class et ClassImpl respectivement ; ces arguments indiquent que les gabarits doivent être traités en fonction des classes internes de la classe concernée

# Branchement des macros

Les macros de ramification fournissent des constructions if-then-else. Le CTF supporte une forme limitée de ramification via ces macros :

- si
- sinonSi
- autre
- finSi
- endTemplate (qui sort du gabarit actuel)

La structure de base des macros if et elseIf est :

%si <test> <opérateur> <test>%

où <opérateur> peut être l'un des éléments suivants :

- ==
- !=
- < (comparaison mathématique, inférieur à)
- > (comparaison mathématique, supérieur à)
- <= (comparaison mathématique, inférieur ou égal à)
- >= (comparaison mathématique, supérieur ou égal à)

et <test> peut être l'un des suivants :

- une string littérale, entourée de guillemets doubles
- une macro de substitution directe, sans les signes de pourcentage qui l'entourent
- une référence variable

Note que si vous utilisez l'un des opérateurs de comparaison mathématiques, <test> doit être un nombre décimal au format string .

Les branches peuvent être imbriquées et plusieurs conditions peuvent être spécifiées à l'aide de l'une des méthodes suivantes :

- et, ou
- ou

Lors de la spécification de plusieurs conditions, « et » et « ou » ont le même ordre de priorité et les conditions sont traitées de gauche à droite.

Si les instructions conditionnelles sur les chaînes sont sensibles à la casse, « une String » n'est pas égale à « une CHAÎNE ». Par conséquent, dans certaines situations, il est préférable de définir la variable $str=TO_LOWER(variable) ou TO_UPPER(variable) puis de comparer à un cas spécifique.

Les macros ne sont pas prises en charge dans les instructions conditionnelles. Il est préférable d'affecter les résultats d'une macro ( string ) à une variable, puis d'utiliser la variable dans la comparaison.

$fldType = % TO_LOWER ($paramètre1)%

$COMMENT = "Utilisez les 4 premiers caractères pour les types de champs Date et Heure"

$fldType4 = % GAUCHE ($fldType, 4)%

%si $fldType4 == "date"%

Date et heure

%endif%

Cela prend un paramètre de valeur « Datetime », « DATETIME » ou « Date » et renvoie « Datetime ».

Les macros endif ou endTemplate doivent être utilisées pour signifier la fin d'une branche. De plus, la macro endTemplate provoque le retour immédiat du gabarit , si la branche correspondante est en cours d'exécution.

## Exemple 1

%si elemType == "Interface"%

;

%autre%

%OpérationBody%

%finSi%

Dans ce cas:

- Si l'élément Type est « Interface », un point-virgule est renvoyé
- Si l'elemType n'est pas "Interface", un gabarit appelé Operation Body est appelé

## Exemple 2

$bases="ClassBase"

$interfaces=""%

%si $bases != " " et $interfaces != ""%

:$bases, $interfaces

%elseIf $bases != ""%

:$bases

%elseIf $interfaces != ""%

: $interfaces

%finSi%

Dans ce cas, le texte renvoyé est ':ClassBase'.

## Conditions utilisant une valeur booléenne

Lors de la configuration d'une ramification à l'aide de conditions impliquant une case à cocher système (champs booléens), comme Attribute.Static (attStatic), l'instruction conditionnelle serait écrite comme suit :

%si attStatic == "T"%

Par exemple:

% si attCollection == "T" ou attOrderedMultiplicity == "T" %

% fin du modèle %

# Macros de synchronisation

Les macros de synchronisation sont utilisées pour fournir des indications de formatage à Enterprise Architect lors de l'insertion de nouvelles sections dans le code source, lors de la synchronisation en aval. Les valeurs des macros de synchronisation doivent être définies dans les gabarits du fichier.

La structure de configuration des macros de synchronisation est :

%<nom>=<valeur>%

où <nom> peut être l'une des macros répertoriées ici et <valeur> est une string littérale entourée de guillemets doubles.

## Macros de synchronisation

| Nom de la macro | Description |
|---|---|
| Espace de synchronisationNewClassNotes | Espace à ajouter à une nouvelle note de cours. valeur par défaut : \n. |
| synchNewAttributeNotesSpace | Espace à ajouter à une nouvelle note d'attribut. valeur par défaut : \n. |
| synchNewOperationNotesSpace | Espace à ajouter à une nouvelle note d'opération. valeur par défaut : \n. |
| synchNouvelleOpérationBodySpace | Espace à ajouter au corps d'une nouvelle opération. valeur par défaut : \n. |
| synchNamespaceBodyIndent | Indentation appliquée aux classes dans les espaces de noms non globaux. valeur par défaut : \t. |

# La macro d'instructions de traitement (PI)

La macro PI (Processing Instruction) fournit un moyen de définir le texte séparateur à insérer entre les morceaux de code (qui représentent des entités) générés à l'aide d'un gabarit .

La structure de définition de l'instruction de traitement est :

%PI=<valeur>%

Dans cette structure, <value> est une string littérale entourée de guillemets doubles, avec ces options :

- " \n" - Nouvelle ligne (par défaut)
- " " - Espace
- " \" - Onglet
- " " - Nul

Par défaut, le PI est configuré pour générer une nouvelle ligne (\n) pour chaque substitution non vide, ce comportement peut être modifié en réinitialisant la macro PI. Par exemple, la déclaration d'attribut d'une classe dans un code VB simple serait générée en une seule ligne (sans nouvelle ligne). Ces propriétés sont dérivées des propriétés d'attribut de classe dans le modèle pour générer, par exemple :

Format d'impression privé constant comme String = "Portrait"

Le gabarit pour générer ceci commence par le PI défini sur un espace plutôt que sur une nouvelle ligne :

% PI = " " %

% CONVERT_SCOPE (attScope)%

%finSi %

% si attConst == "T" %

Const

%finSi %

Lors de la transformation, attscope renvoie le mot-clé VB « Private » et attConst renvoie « Const » sur la même ligne espacée d'un seul espace (conformément à l'exemple de définition VB Class.Attribute précédent).

Alternativement, lors de la génération d'une classe, vous pouvez souhaiter que la déclaration de classe, les notes et le corps de la classe soient tous séparés par des lignes doubles. Dans ce cas, le %PI est défini sur '/n/n' pour renvoyer un espacement de ligne double :

% PI = " \n\n" %

% Déclaration de classe %

% Notes de cours %

% Corps de classe %

## Caractéristiques de l'IP

- Les lignes vides n'ont aucun effet sur la sortie
- Toute ligne contenant une macro qui produit un résultat vide n'entraîne pas de séparateur PI (espace/nouvelle ligne)
- La dernière entrée ne renvoie pas de PI ; par exemple, %Classbody% n'a pas de double ligne ajoutée après le corps

# Macros de génération de code pour Statemachines Exécutables

Les gabarits répertoriés ici sont disponibles via l'éditeur de code Gabarit (l'option de ruban « Développer > Code source > Options > Modifier le code Gabarits » ); sélectionnez « STM_C++_Structured » dans le champ « Langue ».

Les gabarits sont structurés comme indiqué :

StmContextStateMachineEnum

StmStateMachineEnum

StmContextStateEnum

StmAllStateEnum

StmContextTransitionEnum

StmTransitionEnum

StmContextEntryEnum

StmAllEntryEnum

Chaîne de machine StmContextStateToEnum

Chaîne de machine StmStateToEnum

StmContextStateEnumToString

StmStateEnumToString

StmContextTransitionEnumToString

StmTransitionEnumToString

Nom de l'état du contexte Stm vers Guid

Nom de l'état Stm vers Guid

Nom de la transition StmContextVersGuid

Nom de transition Stm vers Guid

Définition du contexte Stm

StmStateMachineEnum

StmAllStateEnum

StmTransitionEnum

StmAllEntryEnum

Initialisation de la variable StmAllRegion

État de Stm avec événement différé

Événement différé Stm
Mappage de processus de transition Stm
Procédure de transition Stm
Sortie de transition Stm
Entrée de transition Stm
Transition sortante de la cible Stm
État de la sous-machine parent de la cible Stm
Mappage de StmStateProc
Procédure d'état de Stm
Entrée d'état Stm
Transition sortante Stm
Entrée de référence StmConnectionPoint
StmParameterizedInitial
StmSubMachineInitial
StmRegionInitial
StmRegionDésactiver
Procédure de sortie de l'état de Stm
Transition d'état Stm
Événement StmState
Transition déclenchée par StmState
StmStateCompletionTransition
État de la transition entrante de StmState
État de transition sortante de StmState
Événement StmSubmachineStateExit
Transition sortante StmVertex
Événement de sortie de référence de point de connexion Stm
Événement StmStateExit
Transition sortante StmVertex
Variable StmAllRegion
Chaîne de machine StmStateToEnum
Exécution de StmStateMachine
Données initiales de l'état Stm
Entrée de machine à état stm
Transition sortante Stm
StmStateMachineRunInitial
StmStateMachineInitial
Exécution de StmStateMachine

Gestionnaire de contexte Stm

Gestionnaire de simulation Stm
Déclaration d'instance de contexte Stm

Instance de contexte Stm
État d'exécution de la variable de contexte Stm
Association d'instances de contexte Stm
StmContextInstanceClear

Proxy d'événement Stm
Enumération StmSignal
StmContextJoinEventEnum
Enumération d'événements StmJoin
Enumération d'événements Stm
Définition de StmSignal
Attribution d'attributs de signal Stm
Attribut StmSignal
Initialisation du signal Stm
Chaîne d'événements StmEventStringToEnum
StmEventEnumToString
Nom de l'événement Stm à Guid

Gestionnaire de console Stm
Déclaration d'instance de contexte Stm
Instance de contexte Stm
État d'exécution de la variable de contexte Stm
Association d'instances de contexte Stm
StmContextInstanceClear

StmStateMachineStrongToEnum

StmInitialForTransition

Transition sortante StmVertext

Événement StmSend

StmBroadcastEvent

Référence de contexte Stm

## Signal et événement

| Nom de la macro | Description |
|---|---|
| stmEventEnum | Le nom de l'événement avec le préfixe « ENUM_ », tout en majuscules. |

| | |
|---|---|
| Guide des événements Stm | Le GUID de l'événement. |
| Nom de l'événement stm | Le nom de l'événement avec les espaces et les astérisques supprimés. |
| Variable d'événement stm | Le nom de l'événement avec le préfixe « m_ » en minuscule. |
| Événement stmIsSignal | Est « T » si l'élément est un SignalEvent. |
| stmSignalEnum | Le nom du signal avec le préfixe « ENUM_ », tout en majuscules. |
| stmSignalFirstEvent | Le nom de l'événement avec le préfixe « ENUM_ », tout en majuscules. |
| stmSignalGuid | Le GUID du signal. |
| stmSignalNom | Le nom du signal avec les espaces et les astérisques supprimés. |
| Variable de signal stm | Le nom du signal avec le préfixe « m_ » en minuscule. |
| stmTriggerName | Transition Propriétés : Le nom du Déclencheur . |
| Spécifications de stmTrigger | Propriétés de Transition : La spécification du Déclencheur . |
| Type de déclencheur stm | Propriétés de Transition : Le type du Déclencheur . |

## Contexte

| Nom de la macro | Description |
|---|---|
| stmContextName | Le nom de la classe avec les espaces et les astérisques supprimés. |
| stmContextQualName | Le nom qualifié de la classe pour laquelle le code est généré. |
| stmContextVariableName | |
| stmContextFileName | Le nom du fichier de sortie pour la classe pour laquelle le code est généré. |

## Écriture de l'état d'exécution Object dans l'initialisation Statemachine

| Nom de la macro | Description |
|---|---|
| stmContextVariableRunstateName | |
| stmContextVariableRunstat | |

| | |
|---|---|
| eValue | |
| stmContextHasStatemachine | Est « T » si le contexte actuel possède une ou plusieurs Statemachines . |
| stmHasHistoryPattern | Est « T » si la Statemachine a un Motif d'historique. |
| stmHasTerminatePattern | Est « T » si la Statemachine a un Motif de terminaison. |
| stmHasDeferredEventPattern | Est « T » si la Statemachine a un Motif d'événement différé. |
| stmHasSubmachinePattern | Est « T » si la Statemachine a un Motif de sous-machine. |
| stmHasOrthogonalPattern | Est « T » si la Statemachine a un Motif orthogonal. |

## Statemachine

| Nom de la macro | Description |
|---|---|
| stmStatemachineName | Le nom de la Statemachine avec les astérisques et les espaces supprimés. |
| stmStatemachineEnum | Le nom de la Statemachine plus « ENUM_ » plus le nom de la Statemachine en majuscules. |
| stmStatemachineGuid | Le GUID de l'élément Statemachine . |
| stmStateCount | Le nombre d'éléments State dans la Statemachine . |
| stmSubmachineInitialCount | Le nombre d'éléments initiaux dans l'élément State de la sous-machine. |
| stmStatemachineHasSubmachineState | Est « T » si la Statemachine possède au moins un State de sous-machine. |
| stmStatemachineInitialCount | Le nombre d'éléments initiaux dans la Statemachine . |

## Région

| Nom de la macro | Description |
|---|---|
| stmRegionEnum | Le nom de la région State plus « ENUM_ » plus le nom de la région State en majuscules. |
| stmRegionFQName | Le nom complet de la région State . |

| | |
|---|---|
| stmRegionName | Le nom de la région State avec les espaces et les astérisques supprimés. |
| Variable de région stm | Le nom de la région State avec le préfixe « m_ » en minuscule. |
| Variable stmRegionFQ | Le nom complet de la région State avec le préfixe « m_ » en minuscule. |
| stmRegionGuid | Le GUID de la région. |
| stmRegionInitial | |
| stmRegionIsOwnedByStateMachine | Est « T » si la région appartient à une Statemachine . |

## Transition

| Nom de la macro | Description |
|---|---|
| stmTransitionEnum | Le nom de la Transition avec le préfixe « ENUM_ », plus le nom de la Transition en majuscules. |
| Guide de transition stm | Le GUID de la Transition. |
| stmTransitionName | Le nom de la Transition avec les espaces et les astérisques supprimés. |
| stmTransitionSourceGuid | Le GUID de l'élément source dans la transition. |
| stmTransitionTargetGuid | Le GUID de l'élément cible dans la transition. |
| Variable de transition stm | Le nom de la Transition avec le préfixe 'm_' en minuscule. |
| Variable source de transition stm | |
| Variable cible de transition stm | |
| Variable stmTransitionFQ | |
| stmSourceVertexEnum | Le nom du sommet source de la transition plus « _ENUM » plus le nom du sommet source de la transition en majuscules. |
| stmTargetVertexEnum | Le nom du sommet cible de la transition plus « _ENUM » plus le nom du sommet cible de la transition en majuscules. |
| stmSourceIsInitial | Est « T » si la source de la transition est une initiale. |
| stmSourceIsState | Est « T » si la source de la transition est un State . |
| | |

| | |
|---|---|
| stmSourceIsEntryPoint | La valeur est « T » si la source de la transition est un point d'entrée. |
| stmSourceIsExitPoint | La valeur est « T » si la source de la transition est un point de sortie. |
| stmSourceIsFork | C'est « T » si la source de la transition est un Fork. |
| stmSourceIsJoin | Est « T » si la source de la transition est un élément Join. |
| stmTargetIsFinalState | Est « T » si la cible de la transition est un élément State final. |
| stmTargetIsExitPoint | Est « T » si la cible de la transition est un élément de point de sortie. |
| stmTargetIsState | Est « T » si la cible de la transition est un élément State . |
| stmTargetIsChoice | La valeur est « T » si la cible de la transition est un élément de choix. |
| stmTargetIsJunction | Est « T » si la cible de la transition est un élément de jonction. |
| stmTargetIsEntryPoint | Est « T » si la cible de la transition est un élément de point d'entrée. |
| stmTargetIsConnectionPointReference | Est « T » si la cible de la transition est un élément de référence de point de connexion. |
| stmTargetIsFork | Est « T » si la cible de la transition est un élément Fork. |
| stmTargetIsJoin | Est « T » si la cible de la transition est un élément de jointure. |
| Effet de transition stm | L'effet de la transition. |
| stmTransitionGuard | La Garde de la Transition. |
| Type de transition stm | Le type ou la sorte de transition. |
| stmTargetInitialTransition | |
| stmTargetIsSubmachineState | Est « T » si la cible de la transition est un State de sous-machine. |
| stmSourceStateEnum | Le nom de l'état source de la transition avec le préfixe « _ENUM » en majuscule. |
| stmTargetStateEnum | Le nom de l'état cible de la transition, avec le préfixe « _ENUM » en majuscule. |
| stmTargetVertexFQName | Le nom entièrement qualifié du sommet cible de la transition. |
| stmTargetIsDeepHistory | Est « T » si la cible de la transition est un State d'histoire profonde. |
| stmTargetIsShallowHistory | Est « T » si la cible de la transition est un State d'histoire superficielle. |
| stmTargetIsTerminate | Est « T » si la cible de la transition est un élément Terminate. |
| stmParentIsStateMachine | Est « T » si le sommet est un point d'entrée ou un point de sortie, ou si le conteneur est une Statemachine . |

| | |
|---|---|
| stmSourceParentStateEnum | |
| stmTargetParentStateEnum | |
| stmTargetSubmachineEnum | |
| Index de la région cible stm | |
| stmIsSelfTransition | Est « T » si la source de la transition est la même que sa cible. |
| stmHistoriquePropriétaireRégionInitialTransition | |
| stmDefaultHistoryTransition | |

## Sommet et State

| Nom de la macro | Description |
|---|---|
| stmVertexName | Le nom du Vertex. |
| stmStateName | Le nom de l' State . |
| stmVertexGuid | Le GUID du sommet. |
| stmVertexFQName | Le nom complet du Vertex. |
| stmStateFQName | Le nom complet de l' State . |
| stmVertexType | Le type du sommet ; l'un des éléments suivants : « State », « FinalState », « Pseudostate », « ConnectionPointReference » ou « » (vide). |
| stmPseudo-étatKind | Le type de pseudo-état ; l'un des éléments suivants : « initial », « deepHistory », « shallowHistory », « join », « fork », « junction », « choice », « entryPoint », « exitPoint » ou « terminate ». |
| stmPseudo-étatNom | Le nom du pseudo-État. |
| Variable stmPseudostate | Le nom du pseudo-État avec le préfixe « m_ » en minuscule. |
| stmPseudostateÉtatNom de la machine | Le nom de la Statemachine pseudo-état. |
| stmPseudostateStateMachineVariable | Le nom de la Statemachine pseudo-état avec le préfixe « m_ » en minuscule. |
| | |

| | |
|---|---|
| Variable stmVertex | Le nom du Vertex avec le préfixe 'm_' en minuscule. |
| stmVertexEnum | Le nom du sommet plus '_ENUM' plus le nom du sommet en majuscules. |
| stmStateEnum | Le nom de l' State plus « _ENUM » plus le nom de l' State en majuscules. |
| stmConnectionPointReferenceStateName | Le nom de la référence du point de connexion. |
| Variable d'état de référence du point de connexion stm | Le nom de la référence du point de connexion avec le préfixe « m_ » en minuscules. |
| stmConnectionPointReferenceEntryCount | |
| stmParameterizedInitialCount | |
| stmInitialCountForTransition | |
| Variable d'état stm | Le nom de l' State avec le préfixe « m_ » en minuscule. |
| comportement de stmStateEntry | Le comportement défini pour une opération Action « entrée » pour un State (le texte de l'onglet « Comportement » pour l'opération Action « entrée » dans la fenêtre Fonctionnalités de l'élément). |
| Code d'entrée stmState | Le code initial défini pour une opération Action « entrée » pour un State (le texte de l'opération Action « entrée » dans l'onglet « Code » du comportement). |
| stmStateDoBehavior | Le comportement défini pour une opération Action « do » pour un State (le texte de l'onglet « Comportement » pour l'opération Action « do » dans la fenêtre Fonctionnalités de l'élément). |
| stmStateDoCode | Le code initial défini pour une opération Action « do » pour un State (le texte de l'opération Action « do » dans l'onglet « Code » du comportement). |
| comportement de sortie stmState | Le comportement défini pour une opération Action « quitter » pour un State (le texte de l'onglet « Comportement » pour l'opération Action « quitter » dans la fenêtre Fonctionnalités de l'élément). |
| stmStateExitCode | Le code initial défini pour une opération Action « sortie » pour un State (le texte de l'opération Action « sortie » dans l'onglet « Code » du comportement). |
| stmStateSubmachineName | Le nom de la sous-machine. |
| Variable de sous-machine stmState | Le nom de la sous-machine avec le préfixe « m_ » en minuscule. |
| stmStateIsFinal | Est « T » si l’ State est un FinalState. |
| stmStateIsSubmachineState | Est 'T' si l' State est un State de sous-machine (page ' Propriétés ' \| Avancé \| propriété 'isSubmachineState'). |
| | |

| | |
|---|---|
| stmSubMachineEnum | Le nom de la sous-machine suivi de « _ENUM » plus le nom de la sous-machine en majuscules. |
| stmStateHasChildrenToJoin | |
| stmStateIsTransitionTarget | |
| stmCeciEstLaSource | |
| stmThisIsSourceState | |
| stmStateParentIsSubmachine | Est « T » si le conteneur de State est une Statemachine . |
| stmStateContainerMatchTransitionContainer | |
| stmVertexRegionIndex | |
| stmÉtatRégionNombre | Le nombre de régions dans l' State . |
| stmStateInitialCount | Le nombre d'éléments initiaux dans la Statemachine . |
| Variable stmVertexContainer | |
| stmVertexParentEnum | |
| stmStateHasUnGuardedCompletionTransition | |
| stmStateEventHasUnGuardedTransition | |
| stmInitialTransition | |

## Association d'instances

| Nom de la macro | Description |
|---|---|
| Nom de l'instance source stm | |
| stmTargetInstanceName | |
| stmSourceRoleName | |
| stmTargetRoleName | |

# Macros de génération de code EASL

Enterprise Architect fournit un certain nombre de macros de génération de code Enterprise Architect Simulation Bibliothèque (EASL) pour générer du code à partir de modèles comportementaux. Il s'agit de :

- EASL_INIT
- EASL_OBTENIR
- Liste EASLListe et
- EASL_FIN

## EASL_INIT

La macro EASL_INIT permet d'initialiser un modèle de comportement EASL. La génération du code du modèle de comportement dépend de ce modèle.

| Aspect | Description |
|---|---|
| Syntaxe | %EASL_INIT(<<GUID>>)%<br>où:<br>• <<GUID>> est le GUID de l' Object (généralement un élément de classe) qui est le propriétaire du modèle de comportement |

## EASL_OBTENIR

La macro EASL_GET permet de récupérer une propriété ou une collection d'un object EASL. Les objets EASL et les propriétés et collections de chaque object sont identifiés dans les rubriques *Collections EASL* et *Propriétés EASL* .

| Aspect | Description |
|---|---|
| Syntaxe | $result = %EASL_GET(<<Propriété>>, <<ID du propriétaire>>, <<Nom>>)%<br>où:<br>• <<Property>> est l'un des éléments suivants : « Property », « Collection », « At », « Count » ou « IndexOf »<br>• <<OwnerID>> est l' ID de l' object propriétaire pour lequel la propriété/collection doit être récupérée<br>• <<Nom>> est le nom de la propriété ou de la collection à laquelle on accède<br>• $result est la valeur renvoyée ; c'est « » si ce n'est pas une propriété valide<br>Si <<Propriété>> est :<br>• « At », alors <<OwnerID>> est l' ID d'une collection et <<Name>> est l'index dans la collection pour laquelle l'élément doit être récupéré<br>• « Count », puis << Owner ID >> est l' ID d'une collection et << Name >> n'est pas utilisé ; il récupérera le numéro d'élément dans la collection<br>• "IndexOf", puis <<Owner ID>> est l' ID d'une collection et <<Name>> est l' ID de l'élément dans la collection ; il récupérera l'index (format string ) de l'élément dans la collection |
| Exemple | $sPropName = %EASL_GET("Propriété", $context, "Nom")% |

## Liste EASL

La macro EASLList est utilisée pour restituer chaque object d'une collection EASL en utilisant le gabarit approprié.

| Aspect | Description |
|---|---|
| Syntaxe | $result = %EASLList=<<Nom du modèle>> @separator=<<Séparateur>> @indent=<<indent>> @owner=<<Identifiant du propriétaire>> @collection=<<NomDeLaCollection>> @option1=<<OPTION1>> @option2=<<OPTION2>>..... @optionN=<<OPTIONN>>%<br>où:<br>• <<TemplateName>> est le nom de tout gabarit de modèle comportemental ou gabarit personnalisé<br>• <<Separator>> est un séparateur de liste (tel que « \n »)<br>• <<indent>> est une indentation à appliquer au résultat<br>• <<OwnedID>> est l' ID de l' object qui contient la collection requise<br>• <<CollectionName>> est le nom de la collection requise<br>• <<OPTION1>...<<OPTION99>> sont des options diverses qui peuvent être passées sur le gabarit ; chaque option est donnée comme paramètre d'entrée supplémentaire au gabarit<br>• $result est la valeur résultante ; c'est « » s'il ne s'agit pas d'une collection valide |
| Exemple | $sStates = %EASLList=" State " @separator="\n" @indent="\t" @owner=$StateMachineGUID @collection=" States " @option=$sOption% |

## EASL_FIN

La macro EASL_END est utilisée pour libérer le modèle de comportement EASL.

| Aspect | Description |
|---|---|
| Syntaxe | %EASL_END% |

## Comportementale Modèle Gabarits

- Action
- Affectation Action
- Pause Action
- Appel Action
- Action Créer
- Action Détruire
- Action si

- Boucle Action
- Action Opaque
- Action parallèle
- Action RaiseEvent
- Action RaiseException
- Interrupteur Action
- Comportement
- Comportement corporel
- Déclaration de comportement
- Paramètre de comportement
- Appeler l'argument
- Décision Action
- Condition Décision
- Logique Décision
- Tableau de Décision
- Garde
- Déclaration de propriété
- Notes sur la propriété
- Object de propriété
- State
- Rappel State
- State Dénombrer
- State Nom énuméré
- Statemachine
- Historique Statemachine
- Transition
- Effet de transition
- Déclencheur

# Collections EASL

Cette rubrique répertorie les collections EASL pour chacun des objets EASL, tels que récupérés par la macro de génération de code EASL Code Generation Macros .

## Action

| Nom de la collection | Description |
|---|---|
| Arguments | Les arguments de Action . |
| Sous-actions | Les sous-actions de l' Action . |

## Comportement

| Nom de la collection | Description |
|---|---|
| Actes | Les actions du comportement. |
| Nœuds | Les nœuds du comportement. |
| Paramètres | Les paramètres du comportement. |
| Variables | Les variables du comportement. |

## Classificateur

| Nom de la collection | Description |
|---|---|
| Toutes les machines d'état | Toutes Statemachines pour le classificateur. |
| Propriétés asynchrones | Les propriétés asynchrones du classificateur. |
| Déclencheurs asynchrones | Les déclencheurs asynchrones du Classificateur. |
| Comportements | Les comportements du Classificateur. |
| Propriétés | Les propriétés du classificateur. |
| Propriétés chronométrées | Les propriétés chronométrées du classificateur. |
| Déclencheurs temporisés | Les déclencheurs temporisés du Classifier. |
| | |

| | |
|---|---|
| Déclencheurs | Tous déclencheurs du Classificateur. |

## Construction

| Nom de la collection | Description |
|---|---|
| Tous les enfants | Les enfants du Construct. |
| Dépendances du client | Les dépendances du client sur le Construct. |
| Stéréotypes | Les stéréotypes du Construct. |
| Dépendances des fournisseurs | Les dépendances du fournisseur sur le Construct. |

## Nœud

| Nom de la collection | Description |
|---|---|
| Bords entrants | Les bords entrants du nœud. |
| Bords sortants | Les bords sortants du nœud. |
| Sous-nœuds | Les sous-nœuds du nœud. |

## State

| Nom de la collection | Description |
|---|---|
| Comportements à faire | Les comportements de State . |
| Comportements d'entrée | Les comportements d'entrée de State . |
| Comportements de sortie | Les comportements de sortie de State . |

## Statemachine

| Nom de la collection | Description |
|---|---|

| | |
|---|---|
| Tous les États finaux | Les States finaux de Statemachine . |
| Tous les États | Tous States au sein de la Statemachine , y compris ceux au sein States de sous-machine. |
| Transitions dérivées | Les transitions dérivées de la Statemachine avec l'effet valide associé. |
| States | Les States dans la Statemachine . |
| Transitions | Les transitions au sein de la Statemachine . |
| Sommets | Les sommets de la Statemachine . |

## Transition

| Nom de la collection | Description |
|---|---|
| Effets | Les effets de la transition. |
| Gardes | Les gardes de la Transition. |
| Déclencheurs | Les déclencheurs de la Transition . |

## Déclencheur

| Nom de la collection | Description |
|---|---|
| Transitions déclenchées | Les transitions déclenchées associées au Déclencheur . |

## Sommet

| Nom de la collection | Description |
|---|---|
| Transitions sortantes dérivées | Les transitions sortantes dérivées du sommet après avoir traversé les pseudo-nœuds. |
| Transitions entrantes | Les transitions entrantes du Vertex. |
| Transitions sortantes | Les transitions sortantes du Vertex. |

# Propriétés EASL

Cette rubrique répertorie les propriétés EASL pour chacun des objets EASL, telles que récupérées par la macro de génération de code EASL Code Generation Macros .

## Action

| Nom de la propriété | Description |
|---|---|
| Comportement | Le comportement associé à Action ( Action de comportement d'appel ou Action d'opération d'appel). |
| Corps | Le corps de Action . |
| Contexte | Le contexte de l' Action . |
| Garde | La garde de Action . |
| EstFinal | Une vérification pour savoir si l'action est une Action finale. |
| Est gardé | Une vérification pour savoir si l'action est une Action gardée. |
| EstInitial | Une vérification pour savoir si l'action est une Action initiale. |
| Gentil | Le genre d' Action . |
| Suivant | L'action suivante de Action . |
| Nœud | Le nœud associé à Action dans le graphique. |

## Argument

| Nom de la propriété | Description |
|---|---|
| Paramètre | L' ID du paramètre associé à l'argument. |
| Valeur | La valeur par défaut de l'argument. |

## Comportement

| Nom de la propriété | Description |
|---|---|
| Action initiale | L'action initiale du comportement. |

| | |
|---|---|
| est en lecture seule | Le isReadOnly du comportement. |
| estSingleExecution | L'exécution est unique du comportement. |
| Gentil | Le type de comportement. |
| Type de retour | Le type de retour du comportement. |
| Spécification | La spécification du comportement. |

## Appeler l'événement

| Nom de la propriété | Description |
|---|---|
| Opération | Le fonctionnement du CallEvent. |

## ChangeEvent

| Nom de la propriété | Description |
|---|---|
| Changer d'expression | L'expression de changement du ChangeEvent. |

## Classificateur

| Nom de la propriété | Description |
|---|---|
| A des comportements | Une vérification pour savoir si le classificateur possède des modèles comportementaux (activité et interaction). |
| Langue | Le langage du classificateur. |
| Statemachine | La Statemachine du classificateur. |

## Condition

| Nom de la propriété | Description |
|---|---|
| | |

| | |
|---|---|
| Expression | L'expression de la condition. |
| Inférieur | La valeur inférieure de la Condition. |
| Supérieur | La valeur supérieure de la Condition. |

## Construction

| Nom de la propriété | Description |
|---|---|
| Obtenir une valeur marquée | La Valeur Étiquetée de la Propriété. |
| Le stéréotype est-il appliqué ? | Une vérification pour savoir si un stéréotype particulier est appliqué à la propriété. |
| Notes | Notes sur la propriété. |
| Type UML | Le type UML de la propriété. |
| Visibilité | La visibilité de la Propriété. |

## Bord

| Nom de la propriété | Description |
|---|---|
| Depuis | L' ID du nœud à partir duquel l'Edge provient. |
| À | L' ID du nœud sur lequel l'Edge est ciblé. |

## Objet événement

| Nom de la propriété | Description |
|---|---|
| Type d'événement | Le type d'événement de l' Object événement. |

## Exemple

| Nom de la propriété | Description |
|---|---|
| Classificateur | Le classificateur de l'instance. |
| Valeur | La valeur de l'instance. |

## Paramètre

| Nom de la propriété | Description |
|---|---|
| Direction | La direction du paramètre. |
| Type | Le type du paramètre. |
| Valeur | La valeur du paramètre. |

## Primitif

| Nom de la propriété | Description |
|---|---|
| Nom FQ | Le nom FQ du Primitif. |
| ID | L' ID du Primitif. |
| Nom | Le nom du Primitif. |
| ObjectType | Le type object du Primitive. |
| Mère | L'IDParent du Primitive. |

## Objet de propriété

| Nom de la propriété | Description |
|---|---|
| Taille de la limite | La taille limite du PropertyObject (s'il s'agit d'une collection). |
| ClassificateurStereoType | Le stéréotype du classificateur du PropertyObject. |
| Est-ce queAsynchProp | Une vérification pour savoir si le PropertyObject est une propriété asynchrone. |
| EstCollection | Une vérification pour savoir si le PropertyObject est une collection. |

| | |
|---|---|
| IsOrdered | Une vérification pour savoir si le PropertyObject est ordonné (s'il s'agit d'une collection). |
| EstTimedProp | Une vérification pour savoir si le PropertyObject est une propriété temporisée. |
| Gentil | Le type de PropertyObject. |
| Valeur inférieure | La valeur inférieure du PropertyObject (s'il s'agit d'une collection). |
| Type | Le type de PropertyObject. |
| Valeur supérieure | La valeur supérieure du PropertyObject (s'il s'agit d'une collection). |
| Valeur | La valeur de PropertyObject. |

## SignalEvent

| Nom de la propriété | Description |
|---|---|
| Signal | Le signal du SignalEvent. |

## State

| Nom de la propriété | Description |
|---|---|
| A une sous-machine | Une vérification pour savoir si l' State est un état de sous-machine. |
| Est-ce que l'état final | Un contrôle pour savoir si l’ State est un état final. |
| Sous-machine | Obtenir l' ID de la sous-machine contenue dans l' State (le cas échéant). |

## Statemachine

| Nom de la propriété | Description |
|---|---|
| A un état de sous-machine | Une vérification pour savoir si la Statemachine possède un état de sous-machine. |
| État initial | L'état initial de la Statemachine . |
| État de la sous-machine | L' State de la sous-machine de Statemachine . |

## Événement temporel

| Nom de la propriété | Description |
|---|---|
| Quand | La propriété « quand » de TimeEvent. |

## Transition

| Nom de la propriété | Description |
|---|---|
| A un effet | Une vérification pour savoir si la transition a un effet valide. |
| IsDerived | Une vérification pour savoir si la transition est une transition dérivée. |
| Est-ce que Transcend | Vérification permettant de savoir si la transition transcende d'une Statemachine ( State de sous-machine) à une autre. |
| Est déclenché | Une vérification pour savoir si la transition est déclenchée. |
| Source | La source de la Transition. |
| Cible | L'objectif de la Transition. |

## Déclencheur

| Nom de la propriété | Description |
|---|---|
| État de destination asynchrone | L'état de destination asynchrone du Déclencheur (s'il s'agit d'un déclencheur asynchrone). |
| Propriété dépendante | L' ID de la propriété associée au Déclencheur . |
| Événement | L'événement du Déclencheur . |
| Nom | Le nom du Déclencheur . |
| Type | Le type du Déclencheur . |

## Sommet

| Nom de la propriété | Description |
| --- | --- |
| EstHistoire | Une vérification pour savoir si le sommet est un état d'historique. |
| Est-ce un pseudo-état ? | Une vérification pour savoir si le sommet est un pseudo-état. |
| Type d'état pseudo | Le type de pseudo-état du Vertex. |

# Appelez Gabarits depuis Gabarits

En utilisant des appels de fonction avec des paramètres, vous pouvez appeler gabarits à partir d'autres gabarits , qu'il s'agisse gabarits standards ou gabarits définis par l'utilisateur créés dans votre projet. De plus, gabarits appelés peuvent renvoyer une valeur et peuvent être appelés de manière récursive.

## Exemples

Une instruction d'appel renvoyant un paramètre à une variable :

$sSource = %StateEnumeratedName($Source)%

Une instruction d'appel à un gabarit qui a des paramètres :

%RuleTask($GUID, $index)%

En utilisant l'instruction $parameter dans le gabarit appelé :

$GUID = $paramètre1

$index = $paramètre2

Gabarits support les appels récursifs, tels que cet appel récursif sur le gabarit RuleTask :

$GUID = $paramètre1

$index = $paramètre2

% PI = " " %

$nul = "Initialiser object condition et action"

$count = %BR_GET("Nombre de règles")%

% si $count == " " ou $count == $index %

%ComputeRulet($GUID)%

\n

% fin du modèle %

%Règle($index)%

\n

$index = %MATH_ADD($index, "1")%

%RuleTask($GUID, $index)%

# L'éditeur Code Gabarit dans le développement des OMD

Ces rubriques décrivent comment utiliser la fenêtre de l'éditeur Code Gabarit pour créer gabarits personnalisés :

- Create Custom Templates
- Customize Base Templates
- Add New Stereotyped Templates

L'éditeur de code Gabarit fournit les facilités de l' Éditeur de Code Common, notamment Intelli-sense pour les macros gabarit de génération de code. Pour plus d'informations sur Intelli-sense et l' Éditeur de Code Common, consultez la rubrique *Édition du code source* .

# Créer Gabarits personnalisés

Enterprise Architect propose une large gamme de gabarits qui définissent la manière dont les éléments de code sont générés. Si ceux-ci ne sont pas suffisants pour vos besoins (par exemple, si vous souhaitez générer du code dans un langage non pris en charge par Enterprise Architect ), vous pouvez créer gabarits personnalisés entièrement nouveaux. Vous pouvez également ajouter des remplacements de stéréotypes à vos gabarits personnalisés ; par exemple, vous pouvez répertorier tous vos paramètres et leurs notes dans vos notes de méthode.

## Accéder

| | |
|---|---|
| Ruban | Développer > Code source > Options > Modifier le code Gabarits<br>Conception > Paquetage > Transformation > Transformation Gabarits |
| Raccourcis Clavier | Ctrl+Maj+P ( gabarits de génération de code)<br>Ctrl+Alt+H ( gabarits de transformation MDA ) |

## Créer gabarits personnalisés à l'aide de l'éditeur Code Gabarits

| Étape | Description |
|---|---|
| 1 | Dans le champ « Langue », cliquez sur la flèche déroulante et sélectionnez le langage de programmation approprié. |
| 2 | Cliquez sur le bouton Ajouter un nouveau Gabarit personnalisé.<br>La dialogue « Créer un nouveau Gabarit personnalisé » s'affiche. |
| 3 | Dans le champ « Type Gabarit », cliquez sur la flèche déroulante et sélectionnez l' object modélisation approprié.<br>L'option '<None>' nécessite un traitement spécial ; elle permet la définition d'une macro de fonction qui ne s'applique réellement à aucun des types, mais doit être appelée en tant que fonction pour définir les variables $parameter1, $parameter2 et ainsi de suite pour chaque valeur transmise. |
| 4 | Dans le champ « Nom Gabarit », saisissez un nom approprié.<br>Cliquez sur le bouton OK . |
| 5 | Dans l'onglet 'Code Gabarits Editor', le nouveau gabarit est inclus dans la liste ' Gabarits ', avec la valeur 'Oui' dans le champ 'Modifié'.<br>Le gabarit est appelé < Gabarit Type>__< Gabarit Name>.<br>Note le double caractère de soulignement entre le type gabarit et le nom gabarit . |
| 6 | Sélectionnez le gabarit dans la liste Gabarits et modifiez le contenu du champ Gabarit pour répondre à vos besoins. |
| 7 | Cliquez sur le bouton Enregistrer.<br>Ceci stocke le nouveau gabarit , qui est maintenant disponible dans la liste des gabarits à utiliser. Vous |

| | |
|---|---|
| | pouvez également ajouter une substitution de stéréotype au gabarit , si nécessaire. |

## Notes

- Pour une langue personnalisée, vous devez définir le gabarit du fichier afin qu'il puisse appeler les gabarits de la section d'importation, Namespace et de la classe, ainsi que tout autre gabarits que vous jugez applicable.

# Personnaliser Gabarits de base

Enterprise Architect propose une large gamme de gabarits qui définissent la manière dont les éléments de code sont générés. Si vous souhaitez modifier la manière dont un élément de code est généré, vous pouvez personnaliser les gabarits fournis par le système. Vos modifications peuvent concerner l'effet du gabarit lui-même ou ses appels à d'autres gabarits . Vous pouvez également ajouter des remplacements de stéréotypes à vos gabarits personnalisés ; par exemple, vous pouvez répertorier tous vos paramètres et leurs notes dans vos notes de méthode.

Lorsque vous personnalisez un gabarit (de base) fourni par le système, vous créez en réalité une copie du gabarit qui est utilisée de préférence à l'original. Toutes les modifications ultérieures sont apportées à cette copie et le gabarit de base d'origine est masqué. Si vous supprimez ensuite la copie, elle ne peut plus remplacer l'original, qui est alors réutilisé.

## Accéder

| | |
|---|---|
| Ruban | Développer > Code source > Options > Modifier le code Gabarits |
| Raccourcis Clavier | Ctrl+Maj+P |

## Personnaliser un gabarit de base

| Étape | Description |
|---|---|
| 1 | Dans l'éditeur de code Gabarit , dans le champ « Langue », cliquez sur la flèche déroulante et sélectionnez le langage de programmation pour lequel vous souhaitez personnaliser les gabarits de base. |
| 2 | Dans la liste Gabarits , cliquez sur le gabarit de base à éditer. |
| 3 | Mettre à jour le gabarit . |
| 4 | Cliquez sur le bouton Enregistrer pour enregistrer vos modifications. |
| 5 | Répétez les étapes 2 à 4 pour chacun des gabarits de base pertinents que vous souhaitez personnaliser. |
| 6 | Si vous préférez, ajoutez une ou plusieurs substitutions de stéréotypes à l'un des gabarits . |

# Ajouter de nouveaux Gabarits stéréotypés

Il est parfois utile de définir un gabarit de génération de code spécifique à utiliser avec des éléments d'un stéréotype donné. Cela permet de générer un code différent pour les éléments, en fonction de leur stéréotype. Enterprise Architect fournit des gabarits par défaut, qui ont été spécialisés pour les stéréotypes couramment utilisés dans les langages pris en charge. Par exemple, le gabarit « Operation Body » pour C# a été spécialisé pour le stéréotype de propriété, de sorte qu'il génère automatiquement ses méthodes constitutives « get » et « set ». Vous pouvez remplacer les gabarits stéréotypés par défaut comme décrit dans la rubrique *Remplacer Gabarits par défaut* . De plus, vous pouvez définir gabarits pour vos propres stéréotypes, comme décrit ici.

## Accéder

| | |
|---|---|
| Ruban | Développer > Code source > Options > Modifier le code Gabarits |
| Raccourcis Clavier | Ctrl+Maj+P |

## Ajouter un nouveau gabarit stéréotypé à l'aide de l'éditeur Code Gabarit

| Étape | Description |
|---|---|
| 1 | Sélectionnez la langue appropriée dans la liste Langue. |
| 2 | Sélectionnez l'un des gabarits de base, dans la liste Gabarits . |
| 3 | Cliquez sur le bouton « Ajouter un nouveau remplacement stéréotypé ».<br>La dialogue « Nouveau remplacement Gabarit » s'affiche. |
| 4 | Sélectionnez la Fonctionnalité et/ou le stéréotype de Classe requis.<br>Cliquez sur le bouton OK . |
| 5 | Le nouveau gabarit stéréotypé s'affiche dans la liste des remplacements de stéréotypes, marqué comme modifié. |
| 6 | Effectuez les modifications requises dans l'éditeur de code Gabarits . |
| 7 | Cliquez sur le bouton Enregistrer pour stocker le nouveau gabarit stéréotypé dans le fichier projet.<br>Enterprise Architect peut maintenant utiliser le gabarit stéréotypé lors de la génération de code pour les éléments de ce stéréotype. |

## Notes

- Les stéréotypes de classe et fonctionnalité peuvent être combinés pour fournir un niveau de spécialisation supplémentaire pour fonctionnalités ; par exemple, si les propriétés doivent être générées différemment lorsque la

classe a un stéréotype MyStereotype, alors la propriété et MyStereotype doivent être spécifiés dans la dialogue New Gabarit Override

# Remplacer Gabarits par défaut

Enterprise Architect dispose d'un ensemble de gabarits de génération de code intégrés ou par défaut. L'éditeur Gabarits de code vous permet de modifier ces gabarits par défaut, personnalisant ainsi la manière dont Enterprise Architect génère le code. Vous pouvez choisir de modifier tout ou partie des gabarits de base pour obtenir le style de codage requis.

Tous gabarits que vous avez remplacés sont stockés dans le fichier projet. Lors de la génération de code, Enterprise Architect vérifie d'abord si un gabarit a été modifié et, si tel est le cas, utilise ce gabarit . Sinon, le gabarit par défaut approprié est utilisé.

## Accéder

| | |
|---|---|
| Ruban | Développer > Code source > Options > Modifier le code Gabarits |
| Raccourcis Clavier | Ctrl+Maj+P |

## Référence

Remplacez un gabarit de génération de code par défaut à l'aide de l'éditeur Gabarits de code.

Lors de la génération de code, Enterprise Architect utilise maintenant le gabarit de remplacement au lieu du gabarit par défaut.

| Champ/Bouton | Description |
|---|---|
| Langue | Sélectionnez la langue appropriée dans la liste. |
| Gabarits | Sélectionnez l’un des gabarits de base dans la liste. |
| Les stéréotypes supplantent | Si le gabarit de base comporte des remplacements stéréotypés, vous pouvez en sélectionner un dans la liste. |
| <Autres domaines> | Apportez toutes les autres modifications nécessaires. |
| Sauvegarder | Cliquez sur ce bouton pour enregistrer la version modifiée du gabarit dans le fichier projet. Le gabarit est marqué comme modifié. |

# Cadre grammatical

Enterprise Architect fournit support de l'ingénierie inverse pour un certain nombre de langages de programmation courants. Cependant, si le langage que vous utilisez n'est pas pris en charge, vous pouvez écrire votre propre grammaire à l'aide de l'éditeur de grammaire intégré. Vous pouvez ensuite incorporer la grammaire dans une MDG Technologie pour fournir à la fois support de l'ingénierie inverse et de la synchronisation de code pour votre langage cible.

Le framework permettant d'écrire une grammaire et de l'importer dans Enterprise Architect est le complément direct du framework Code Gabarit . Alors que gabarits de code servent à convertir un modèle en une forme textuelle, les grammaires sont nécessaires pour convertir un texte en modèle. Les deux sont nécessaires pour synchroniser les modifications dans vos fichiers sources.

Un exemple de fichier source de langue et un exemple de grammaire pour cette langue sont fournis dans le répertoire Exemples de code, auquel vous pouvez accéder à partir de votre répertoire d'installation (l'emplacement par défaut est C:\Program Files\ Sparx Systems \EA). Deux autres fichiers de grammaire sont également fournis, illustrant des aspects spécifiques du développement de grammaires.

## Composants

| Composant | Description |
|---|---|
| Grammaire Syntaxe | Les grammaires définissent la manière dont un texte doit être découpé en une structure, ce qui est nécessaire lorsque vous convertissez du code en une représentation UML . Au niveau le plus simple, une grammaire est constituée d'instructions permettant de découper une entrée pour former une structure.<br>Enterprise Architect utilise une variante du formulaire Backus-Naur (nBNF) pour inclure des instructions de traitement, dont l'exécution renvoie des informations structurées à partir des résultats analysés sous la forme d'un arbre de syntaxe abstraite (AST), qui est utilisé pour générer une représentation UML . |
| Éditeur de grammaire | L'éditeur de grammaire est un éditeur intégré que vous pouvez utiliser pour ouvrir, modifier, valider et enregistrer des fichiers de grammaire. |
| Débogage de la grammaire | Vous pouvez déboguer les fichiers de grammaire que vous créez en utilisant deux facilités :<br>• L' Parser , qui génère l'AST pour la grammaire<br>• Le profileur, qui analyse également la grammaire et génère l'AST mais qui expose le chemin de profilage pour montrer exactement ce qui s'est passé à chaque étape du processus |

# Grammaire Syntaxe

Les grammaires définissent la manière dont un texte doit être décomposé en une structure, ce qui est exactement ce dont vous avez besoin lorsque vous convertissez du code en une représentation UML . Au niveau le plus simple, une grammaire n'est qu'une série d'instructions permettant de décomposer une entrée pour former une structure. Enterprise Architect utilise une variante de la forme Backus-Naur (BNF) pour exprimer une grammaire de manière à lui permettre de convertir le texte en une représentation UML . Ce que la grammaire d' Enterprise Architect offre par rapport à une BNF pure est l'ajout d'instructions de traitement, qui permettent de renvoyer des informations structurées à partir des résultats analysés sous la forme d'un arbre de syntaxe abstrait (AST). Une fois l'AST terminé, Enterprise Architect le traitera pour produire un modèle UML .

## Syntaxe

| Syntaxe | Détail |
|---|---|
| Commentaires | Les commentaires ont la même forme que dans de nombreux langages de programmation.<br><br>// Vous pouvez commenter la fin d'une ligne en ajoutant deux /.<br>/* Vous pouvez commenter plusieurs lignes en ajoutant un / suivi d'un *.<br>Le commentaire se termine lorsque vous ajoutez un * suivi d'un /. */ |
| Instructions | Les instructions précisent les détails clés du fonctionnement de la grammaire. Elles sont généralement incluses en haut de la grammaire et ressemblent aux appels de fonction dans la plupart des langages de programmation. |
| Règles | Les règles constituent le corps d'une grammaire. Une règle peut avoir une ou plusieurs définitions séparées par des délimiteurs de type pipe (\|).<br>Pour qu'une règle soit validée, une seule définition complète doit être validée. Les règles se terminent par le caractère point-virgule (;). |
| Définitions | Une définition est l'un des chemins que peut emprunter une règle. Chaque définition est composée d'un ou plusieurs termes. |
| Listes de définitions | Une liste de définitions correspond à un ou plusieurs ensembles de termes. Ceux-ci seront évalués dans l'ordre jusqu'à ce que l'un d'eux réussisse. Si aucun ne réussit, la règle qui le contient échoue. Chaque paire de définitions est séparée par un caractère \|.<br>Il s'agit d'une règle simple comportant trois définitions :<br><greeting> ::= "bonjour" \| "salut" \| ["bonjour"] "matin"; |
| Termes | Un terme peut être une référence à une règle, une valeur spécifique, une plage de valeurs, une sous-règle ou une commande. |
| Commandes | Comme les instructions, les commandes ressemblent à des appels de fonction. Elles remplissent deux fonctions principales :<br>• Traiter les jetons d'une manière spécifique ou<br>• Pour fournir un résultat à l'appelant |

# Instructions de grammaire

Les instructions précisent les détails clés du fonctionnement de la grammaire. Elles sont généralement incluses en haut de la grammaire et ressemblent aux appels de fonction dans la plupart des langages de programmation.

## Instructions

| Instruction | Description |
|---|---|
| sensible aux majuscules et minuscules() | L'une de ces deux instructions doit spécifier si la correspondance des jetons doit être sensible à la casse ou non. Par exemple, les langages de la famille BASIC ne sont pas sensibles à la casse, tandis que les langages de la famille C le sont. |
| caseInsensitive() | |
| délimiteurs(DelimiterRule: Expression) | L'instruction delimiters indique à l'analyseur lexical quelle règle utiliser pour la découverte des délimiteurs. Les délimiteurs sont utilisés lors de l'analyse des mots-clés et peuvent être définis comme les caractères qui peuvent être utilisés immédiatement avant ou après les mots-clés de la langue. |
| lex(TokenRule: Expression) | L'instruction lex indique à l'analyseur lexical le nom de la règle racine à utiliser pour son analyse. |
| analyser(RootRule: Expression)<br>analyser(RootRule : Expression, SkipRule : Expression) | L'instruction d'analyse indique à l'analyseur le nom de la règle racine à utiliser pour son traitement. Le deuxième argument facultatif spécifie une règle de saut (ou d'échappement), qui est généralement utilisée pour gérer les commentaires. |

# Règles de grammaire

Les règles sont exécuter pour décomposer le texte en structures. Une règle est composée d'une ou plusieurs définitions, chacune d'elles étant composée d'un ou plusieurs termes.

## Types de règles

| Règle | Description |
|---|---|
| Règles nommées | Un nom suivi d'une liste de définitions. Par exemple :<br><règle> ::= <terme1> <terme2> \| "-" <terme1>; |
| Règles en ligne | À l'intérieur d'une définition, une règle est définie entre parenthèses. Celles-ci agissent exactement de la même manière que s'il s'agissait d'une règle nommée appelée par un terme. Par exemple :<br><règle> ::= (<en ligne>); |
| Règles facultatives | À l'intérieur d'une définition, une règle définie entre crochets. Cette règle réussit même si le contenu échoue. Par exemple :<br><règle> ::= [<inline>]; |
| Règles répétitives | À l'intérieur d'une définition, un terme suivi d'un signe plus. Cette règle correspond à la règle interne une ou plusieurs fois. Par exemple :<br><règle> ::= <en ligne>+;<br>règle ::= (<terme1> <terme2>)+; |
| Règles répétitives facultatives | À l'intérieur d'une définition, une règle suivie d'une étoile. Cette règle correspond à la règle interne zéro fois ou plus, ce qui signifie qu'elle réussit même si la règle interne ne réussit jamais. Par exemple :<br><règle> ::= <inline>*;<br>règle ::= (<terme1> <terme2>)*; |

# Termes de grammaire

Les termes identifient où les jetons sont consommés.

## Types de termes

| Type | Description |
|---|---|
| Termes concrets | Chaînes entre guillemets.<br>Par exemple, « classe » |
| Caractères Unicode | Un terme réservé au lexer, ayant le préfixe U+0x suivi d'un nombre hexadécimal.<br>Par exemple : U+0x1234 |
| Gammes | Un terme réservé à l'analyseur lexical, correspondant à n'importe quel caractère entre les deux caractères spécifiés.<br>Par exemple, « a »..« z » ou U+0x1234..U+2345 |
| Références | Le nom d'une autre règle, entre crochets angulaires. Le jeton correspondra si cette règle réussit.<br>Par exemple, <anotherRule> |
| Commandes | Un appel à une commande spécifique. |

# Commandes de grammaire

Les commandes, comme les instructions, ressemblent à des appels de fonction. Elles remplissent deux fonctions principales :

- Traiter les jetons d'une manière spécifique ou
- Pour fournir un résultat à l'appelant

## Commandes

| Commande | Description |
|---|---|
| attribut(Nom : String , valeur : expression) | Crée un attribut sur le nœud AST actuel. L'attribut sera créé avec le nom spécifié dans la source de grammaire et recevra la valeur de tous les jetons consommés dans le cadre de l'exécution de l'expression de valeur.<br>Cette commande produit les attributs du nœud AST sur lesquels Enterprise Architect opère dans l'ingénierie de code. |
| attributeEx(Nom : String )<br>attributeEx(Nom : String , Valeur : String ) | Crée un attribut sur le nœud AST actuel sans consommer de jetons. L'attribut sera créé avec le même nom que celui spécifié dans la source de grammaire et avec une valeur vide ou la valeur spécifiée par l'argument Value facultatif.<br>Cette commande produit les attributs du nœud AST sur lesquels Enterprise Architect opère dans l'ingénierie de code. |
| except(Cible : Expression, Exception : Expression) | Consomme les données d'entrée qui correspondent à l'expression cible, mais échoue sur les données qui correspondent à l'expression d'exception. Son fonctionnement est assez similaire à celui de la commande skip, mais exactement à l'opposé de celle-ci. |
| échouer() | Fait échouer l'analyseur sur la règle actuelle, y compris toutes les définitions restantes. |
| mots-clés() | Correspond à n'importe quelle string littérale utilisée comme terme de grammaire ; c'est-à-dire que si vous entrez une string explicite que vous recherchez, elle devient un mot clé. |
| mapLeft(Cible : Expression, Commun : Expression) | Consomme les données d'entrée qui correspondent à l'expression cible, puis consomme les données d'entrée qui correspondent à l'expression commune.<br>Les nœuds et Attributes AST générés à partir de l'expression commune sont ensuite copiés dans chacun des nœuds de niveau supérieur générés par l'expression cible.<br>Les expressions cibles et communes doivent être analysées avec succès pour que cette commande réussisse. |
| mapRight(Commun : Expression, Cible : Expression) | Consomme les données d'entrée qui correspondent à l'expression commune, puis consomme les données d'entrée qui correspondent à l'expression cible.<br>Les nœuds et Attributes AST générés à partir de l'expression commune sont ensuite copiés dans chacun des nœuds de niveau supérieur générés par l'expression cible. |

| | |
|---|---|
| | Les expressions communes et cibles doivent être analysées avec succès pour que cette commande réussisse. |
| nœud (Nom : String , cible : expression) | Crée un nœud AST sous le nœud AST actuel (les nœuds sur lesquels Enterprise Architect opère dans l'ingénierie de code). Le nœud sera créé avec le nom spécifié dans la source de grammaire. |
| preProcess(Cible : Expression) | Évalue une expression et utilise ces données prétraitées dans plusieurs définitions. Cette fonction est particulièrement utile dans l'analyse d'expressions, où la même expression de gauche sera évaluée par rapport à un certain nombre d'opérateurs. Cette commande réduit le travail que l'analyseur doit effectuer pour que cela se produise. |
| sauter(Cible : Expression)<br>skip(Cible : Expression, Échap : Expression) | Consomme des données d'entrée (caractères lors de l'analyse lexicale et jetons lors de l'analyse) jusqu'à ce que l'expression « Target » soit mise en correspondance. L'expression facultative « Escape » peut être utilisée pour gérer des instances telles que des guillemets échappés dans des chaînes. |
| skipBalanced(Origine : Expression, Cible : Expression)<br>skipBalanced(Origine : Expression, Cible : Expression, Échappement : Expression) | Consomme des données d'entrée (caractères ou jetons) jusqu'à ce que l'expression « Target » corresponde et que le niveau d'imbrication atteigne zéro. Si l'expression « Origin » correspond pendant ce processus, le niveau d'imbrication est augmenté. Si l'expression « Target » correspond, le niveau d'imbrication est diminué. Lorsque le niveau d'imbrication atteint zéro, la commande se termine avec succès. Une expression « Escape » facultative peut être fournie. |
| sauterEOF() | Consomme toutes les données restantes (caractères ou jetons) jusqu'à la fin du fichier. |
| jeton (cible : expression) | Crée un jeton lors de l'analyse lexicale pour le traitement lors de l'analyse. La valeur du jeton sera la valeur de tous les caractères consommés suite à l'exécution de l'expression cible. |
| avertissement() | Insère un avertissement dans l'AST résultant. |

# Nœuds AST

Lors de la définition d'une grammaire, vous utiliserez des nœuds AST et des attributs de nœud AST qui peuvent être reconnus dans l'ingénierie de code dans Enterprise Architect , dans les résultats AST renvoyés par les commandes attribute, attributeEx et node. Les nœuds et les attributs sont identifiés dans ces tableaux . Tous les autres seront ignorés dans l'ingénierie de code.

## Noeud FILE

Le nœud FILE représente un fichier. Il n'est associé à rien, mais contient toutes les informations requises.

| Multiplicité / Noeuds | Description |
|---|---|
| 0..* / PAQUETAGE | Voir *le nœud PACKAGE* . |
| 0..* / CLASSE | Voir *le nœud CLASS* . |
| 0..* / IMPORT | Le nœud qui représente l'espace de noms/ Paquetage ou équivalent importé. L'attribut 'NAME' du nœud sera le nom de l'espace de noms/ Paquetage ou équivalent importé. |
| 0..* / COMMENT | Les étiquettes de champ faisant partie d'une règle de saut seront au niveau racine ; le générateur de code recherche les commentaires de ce type par position par rapport au nœud. |
| 0..1 / INSÉRER_POSITION | Cela donne la position où les nouvelles classes, Paquetages et implémentations de méthodes peuvent être insérés dans le fichier. S'il n'est pas trouvé, le générateur de code insère automatiquement les nouveaux éléments immédiatement après que le dernier ait été trouvé dans le code. |

## Noeud PACKAGE

Le nœud PACKAGE correspond à un espace de noms ou équivalent dans le fichier. Lors de l'importation avec « paquetage par espace de noms », Enterprise Architect crée un Paquetage directement sous l'importation pour cela et place toutes les classes à l'intérieur. Lorsque vous n'importez pas d'espaces de noms, Enterprise Architect recherche des classes sous ce point, mais ne fait rien avec ce nœud.

De plus, si vous générez avec les espaces de noms activés (voir les rubriques d'aide sur *les options de code* pour les langages génériques), une classe générée ne correspondra pas à une classe dans le code, sauf si elles se trouvent sous la même structure Paquetage .

Contenu dans les nœuds : FICHIER

| Multiplicité / Noeuds | Description |
|---|---|
| 1 / NOM | Voir *le nœud NAME* . |
| 0..* / CLASSE | Voir *le nœud CLASS* . |
| 0..* / PAQUETAGE | Le nœud Paquetage enfant. |
| 0..1 / | Donne la position où le corps Paquetage s'ouvre. Cela peut également être utilisé |

| | |
|---|---|
| POSITION_OUVERTE | comme position d'insertion. |
| 0..1 / INSÉRER_POSITION | Indique la position à laquelle les nouvelles classes et Paquetages peuvent être insérés dans le fichier. S'il n'est pas trouvé, le générateur de code insère automatiquement les nouveaux éléments immédiatement après que le dernier ait été trouvé dans le code. |
| 0..1 / SUPPRIMER | Empêche l'indentation lors de l'insertion dans ce Paquetage . |

## Noeud CLASSE/INTERFACE

Le nœud CLASS (ou INTERFACE) est le plus important dans la génération de code. Il est introduit sous forme d'objets Class (ou Interface).

Voir *la classe DECLARATION* et *la classe BODY* .

Contenu dans les nœuds : FILE, PACKAGE, Class BODY

## Déclaration de CLASSE

Contenu dans les nœuds : CLASSE/INTERFACE

| Multiplicité / Noeuds | Description |
|---|---|
| 1 / NOM | Voir *le nœud NOM* . |
| 0..* / PARENT | Voir *le nœud PARENT* . |
| 0..* / ÉTIQUETTE | Voir *le nœud TAG* . |
| 0..1 / DESCRIPTION | Voir *le nœud DESCRIPTION* . |
| 1 / NOM | Le nom de la classe. S'il existe un nœud NAME, celui-ci écrasera cet attribut. |
| 0..1 / PORTÉE | La portée UML de la classe - publique, privée, protégée ou Paquetage . |
| 0..1 / RÉSUMÉ | Si présent, indique qu'il s'agit d'une classe abstraite. |
| 0..1 / VERSION | La version de la classe. |
| 0..1 / STÉRÉOTYPE | Le stéréotype qu'Enterprise Enterprise Architect doit attribuer à la classe. Cela ne prend pas support les stéréotypes multiples. |
| 0..1 / FEUILLE D'ÎLE | Si présent, indique qu'il s'agit d'une classe feuille/finale/scellée qui ne peut être héritée par aucune sous-classe. |
| 0..1 / MULTIPLICITÉ | Si présent, représente la multiplicité de la classe. |
| 0..1 / LANGUE | En règle générale, vous n'avez pas besoin de définir cette option. |
| | |

| | |
|---|---|
| 0..1 / REMARQUE | Généralement non utilisé car il est abordé par les commentaires au-dessus de la classe. |
| 0..1 / ALIAS | S'il est présent, représente l'alias de tout identifiant, tel qu'un Namespace , une classe ou une variable. |
| 0..* / MACRO | Ajoute une Valeur Étiquetée numérotée qu'Enterprise Enterprise Architect peut utiliser pour round -retour entre les macros. |

## Noeud BODY de classe

Contenu dans les nœuds : CLASSE/INTERFACE

| Multiplicité / Noeuds | Description |
|---|---|
| 0..* / MÉTHODE | Voir *le nœud METHODE* . |
| 0..* / ATTRIBUT | Voir *le nœud ATTRIBUTE* . |
| 0..* / CHAMP | Voir *le nœud FIELD* . |
| 0..* / CLASSE | Voir *le nœud CLASS* . |
| 0..* / PORTÉE | Voir *le nœud SCOPE* . |
| 0..* / PROPRIÉTÉ | Ce nœud représente la définition de la propriété dans le corps de la classe. |
| 0..* / ÉTIQUETTE | Voir *le nœud TAG* . |
| 0..* / PARENT | Voir *le nœud PARENT* . |
| 0..1 / POSITION_OUVERTE | Indique la position à laquelle le corps de la classe s'ouvre. Cela peut également être utilisé comme position d'insertion. |
| 0..1 / INSÉRER_POSITION | Indique la position à laquelle les nouveaux membres de la classe peuvent être insérés dans le fichier. S'il n'est pas trouvé, le générateur de code insère automatiquement les nouveaux éléments immédiatement après que le dernier ait été trouvé dans le code. |

## Nœud SCOPE

Il s'agit d'une fonctionnalité facultative pour les langages ressemblant à C++ qui ont des blocs qui spécifient la portée des éléments. Le langage doit avoir un nom spécifié qui est utilisé pour la portée de tous les éléments du Bloc . À tous les autres égards, il se comporte de manière identique au nœud Class BODY.

Contenu dans les nœuds : classe BODY

| Multiplicité / Noeuds | Description |
|---|---|
| | |

| | |
|---|---|
| 1 / NOM | Utilisé comme portée pour toutes les méthodes et attributs contenus dans la portée. |

## Noeud METHODE

Contenu dans les nœuds : classe BODY, SCOPE

| Multiplicité / Noeuds | Description |
|---|---|
| 1 / Méthode DECLARATION | Voir *la méthode DECLARATION Node* . |

## Méthode DECLARATION Node

Contenu dans les nœuds : METHODE

| Multiplicité / Noeuds | Description |
|---|---|
| 0..1 / TYPE | Voir *TYPE Node* . |
| 0..* / PARAMÈTRE | Voir *le nœud PARAMETER* . |
| 0..* / ÉTIQUETTE | Voir *TAG NODE* . |
| 0..1 / DESCRIPTION | Voir *le nœud DESCRIPTION* . |
| 0..1 / MULTI PARAMÈTRE | Prend en charge le style de déclaration de liste de paramètres de Delphi. C'est l'équivalent de FIELD. |
| 1 / NOM | Le nom de la méthode. |
| 0..1 / TYPE | Le type de retour de la méthode. |
| 0..1 / PORTÉE | La portée UML de la méthode - Public, Privé, Protégé ou Paquetage . |
| 0..1 / RÉSUMÉ | Si présent, indique que la méthode est abstraite. |
| 0..1 / STÉRÉOTYPE | Le stéréotype qu'Enterprise Enterprise Architect doit attribuer à la méthode. Cette option ne prend pas support les stéréotypes multiples. |
| 0..1 / STATIQUE | Si présent, indique que la méthode est statique. |
| 0..1 / CONST ou CONSTANTE | Si présent, indique que la méthode est constante. |
| 0..1 / PUR | Si présent, indique que la méthode est une méthode pure. |
| 0..1 / QUESTIONNAIRE | Si présent, indique que la méthode est en mode requête/lecture seule. |
| | |

| | |
|---|---|
| 0..1 / TABLEAU | Si présent, indique que le type de méthode (type de retour) est un tableau. |
| 0..1 / SYNCHRONISÉ | Si présent, indique que la méthode est une méthode synchronisée. |
| 0..* / MACRO | La macro spécifiée dans la déclaration de méthode. |
| 0..1 / CSHARPIMPLEMENTS | Spécifie un comportement spécial pour C# . |
| 0..1 / COMPORTEMENT | Fournit support pour Aspect J, en utilisant le comportement. |
| 0..1 / AFFICHAGE DU COMPORTEMENT | Fournit support pour Aspect J, en utilisant le comportement, et affiche le comportement rétro-conçu sur le diagramme . |

## Nœud ATTRIBUT

Contenu dans les nœuds : classe BODY, SCOPE

| Multiplicité / Noeuds | Description |
|---|---|
| 1 / TYPE | Voir *TYPE Node* . |
| 0..* / ÉTIQUETTE | Voir *le nœud TAG* . |
| 0..1 / DESCRIPTION | Voir *le nœud DESCRIPTION* . |
| 1 / NOM | Le nom de l'attribut. |
| 0..1 / TYPE | Le type de l'attribut. |
| 0..1 / PORTÉE | La portée UML de l'attribut - Public, Privé, Protégé ou Paquetage . |
| 0..1 / PAR DÉFAUT | La valeur par défaut de l'attribut. |
| 0..1 / CONTAINER ou ARRAY | Si présent, indique le conteneur de l'attribut. |
| 0..1 / CONFINEMENT | Référence ou valeur . |
| 0..1 / STÉRÉOTYPE | Le stéréotype qu'Enterprise Enterprise Architect doit attribuer à l'attribut. Cette option ne prend pas support les stéréotypes multiples. |
| 0..1 / STATIQUE | Si présent, indique qu'il s'agit d'un attribut statique. |
| 0..1 / CONST ou CONSTANTE | Si présent, indique qu'il s'agit d'un attribut constant. |
| 0..1 / COMMANDÉ | Si présent, indique que l'attribut ( valeur ) est ordonné. |
| 0..1 / LIMITE BASSE | Si présent, représente la bordure inférieure de la valeur de l'attribut. |

| | |
|---|---|
| 0..1 / LIMITE HAUTE | Si présent, représente la bordure supérieure de la valeur de l'attribut. |
| 0..1 / TRANSITOIRE ou VOLATILE | Si présent, indique que l'attribut est transitoire ou volatil. |

## Nœud FIELD

Un champ correspond à plusieurs déclarations d'attributs en une seule. Tout ce qui n'est pas défini dans les déclarateurs mais défini dans le champ lui-même sera défini pour chaque déclarateur. Tout ce qui est pris en charge dans un attribut est pris en charge dans le champ. Si aucun déclarateur n'est trouvé, cela fonctionne de la même manière qu'un attribut.

Contenu dans les nœuds : classe BODY, SCOPE

| Multiplicité / Noeuds | Description |
|---|---|
| 0..* / DÉCLARATEUR | Voir *le nœud ATTRIBUTE* . |

## Nœud PARAMETRE

Contenu dans les nœuds : méthode DECLARATION, TEMPLATE

| Multiplicité / Noeuds | Description |
|---|---|
| 1 / TYPE | Voir *TYPE Node* . |
| 0..* / ÉTIQUETTE | Voir *le nœud TAG* . |
| 0..1 / DESCRIPTION | Voir *le nœud DESCRIPTION* . |
| 0..1 / NOM | Le nom du paramètre. |
| 0..1 / TYPE | Le type du paramètre. |
| 0..1 / TYPE | Devrait être dedans, dehors, dehors ou revenir. |
| 0..1 / PAR DÉFAUT | La valeur par défaut du paramètre. |
| 0..1 / FIXE | Si présent, indique que le paramètre est fixe/constant. |
| 0..1 / TABLEAU | Si présent, indique que le type de paramètre est un tableau. |

## Nœud NOM

Contenu dans les nœuds : PACKAGE, classe DECLARATION

| Multiplicité / Noeuds | Description |
|---|---|
| 1 / NOM | La partie nom. |
| 0..* / QUALIFICATION | La partie qualificative. |
| 0..* / PARTIE DU NOM | Une alternative à l'utilisation de NAME et QUALIFIER. Une string de valeurs, toutes sauf la dernière étant considérées comme des qualificateurs. La dernière est considérée comme le nom. |

## Nœud TYPE

Contenu dans les nœuds : méthode DECLARATION, ATTRIBUTE, PARAMETER

| Multiplicité / Noeuds | Description |
|---|---|
| 0..1 / MODÈLE | L'ensemble du texte du gabarit est le nom du type.<br>Utilisé uniquement si NAME n'est pas défini.<br>Voir *le nœud TEMPLATE* . |
| 1 / NOM | La partie nom. |
| 0..* / QUALIFICATION | La partie qualificative. |
| 0..* / PARTIE DU NOM | Une alternative à l'utilisation de NAME et QUALIFIER. Une string de valeurs, toutes sauf la dernière étant considérées comme des qualificateurs. La dernière est considérée comme le nom. |

## Nœud MODÈLE

Contenu dans les nœuds : TYPE

| Multiplicité / Noeuds | Description |
|---|---|
| 0..* / PARAMÈTRE | Voir *le nœud PARAMETER* . |
| 1 / NOM | |

## Nœud PARENT

Contenu dans les nœuds : classe DÉCLARATION

| Multiplicité / Noeuds | Description |
|---|---|
| 0..1 / TYPE | A la valeur Parent, Implements ou VirtualP. |

| | |
|---|---|
| 1 / NOM | La partie nom du parent. |
| 0..* / QUALIFICATION | La partie qualificative du parent. |
| 0..* / PARTIE DU NOM | Une alternative à l'utilisation de NAME et QUALIFIER. Une string de valeurs, toutes sauf la dernière étant considérées comme des qualificateurs. La dernière est considérée comme le nom. |
| 0..1 / INSTANTIATION | Si présent, indique l'instanciation d'un paramètre gabarit . |

## Nœud TAG

Contenu dans les nœuds : classe DECLARATION, méthode DECLARATION, ATTRIBUT, PARAMETER

| Multiplicité / Noeuds | Description |
|---|---|
| 1 / NOM | Le nom de la Valeur Étiquetée (l' Étiquette ). |
| 0..* / VALEUR | La valeur de la Valeur Étiquetée . |
| 0..1 / MÉMO | S'il est présent, indique que le type de la Valeur Étiquetée est <memo>. |
| 0..1 / NOMÉMO | Si présent, indique que le type de la Valeur Étiquetée n'est pas <memo>. |
| 0..1 / GROUP | S'il est présent, indique que la valeur est un groupe Valeur Étiquetée . |

## DESCRIPTION Noeud

Contenu dans les nœuds : classe DECLARATION, méthode DECLARATION, ATTRIBUT, PARAMETER

| Multiplicité / Noeuds | Description |
|---|---|
| 0..* / VALEUR | Le texte qu'Enterprise Enterprise Architect doit attribuer à la Note . |

# Édition des grammaires

Si vous devez écrire et modifier une grammaire pour du code importé dans un nouveau langage de programmation, vous pouvez le faire à l'aide de l'éditeur de grammaire intégré.

## Accéder

| | |
|---|---|
| Ruban | Développer > Code source > Éditeur de grammaire |

## Créer et modifier la grammaire

| Champ/Bouton | Action |
|---|---|
| Grammaire ouverte | Affichez un navigateur via lequel vous pouvez localiser et ouvrir le fichier contenant la grammaire que vous souhaitez modifier. |
| Récent | Les grammaires récemment utilisées peuvent être rapidement accessibles à l'aide de cette zone de liste déroulante. |
| Sauvegarder | Enregistrer le fichier actuel. |
| Enregistrer sous | Enregistre une copie du fichier actuel |
| Valider la grammaire | La validation grammaticale va exécuter une série de tests sur la grammaire actuelle pour s'assurer de sa validité. Des erreurs et des avertissements seront affichés vous informant à la fois des erreurs qui rendront la grammaire inutilisable et des conditions dans lesquelles vous pourriez obtenir des résultats inattendus. |
| Aide | Afficher cette rubrique d'aide. |

## Options Menu Contexte

| Champ/Bouton | Action |
|---|---|
| Ouvrir le fichier | Affichez un navigateur via lequel vous pouvez localiser et ouvrir le fichier contenant la grammaire que vous souhaitez modifier. |
| Valider | La validation grammaticale va exécuter une série de tests sur la grammaire actuelle pour s'assurer de sa validité. Des erreurs et des avertissements seront affichés vous informant à la fois des erreurs qui rendront la grammaire inutilisable et des conditions dans lesquelles vous pourriez obtenir des résultats inattendus. |
| Langue | L'éditeur de grammaire utilise par défaut la forme Backus-Naur normale (nBNF). |

| | |
|---|---|
| | L'option mBNF est également disponible. |
| Numéros de ligne | Activez ou désactivez les numéros de ligne dans l'éditeur de grammaire. |

# Analyse des résultats AST

L'arbre de syntaxe abstraite (AST) est le code qu'Enterprise Enterprise Architect voit lorsqu'il traite une grammaire.

Vous analysez le texte dans la moitié inférieure de la fenêtre de l'éditeur de grammaire et révision ce qui est affiché comme résultat. Vous pouvez soit ouvrir un fichier, soit coller du texte. Si vous avez collé du texte qui correspond à quelque chose qui ne peut pas apparaître au niveau du fichier (comme les paramètres d'opération), vous pouvez sélectionner une règle alternative à utiliser comme point de départ. L'analyse commencera alors à partir de cette règle.

## Accéder

| | |
|---|---|
| Ruban | Développer > Code source > Éditeur de grammaire > Débogueur de grammaire > Résultats AST |

## Options de la barre d'outils

| Option | Action |
|---|---|
| Ouvrir le fichier | Ouvrez un exemple de fichier d’entrée pour effectuer un test. |
| Récent | Les fichiers sources récemment ouverts peuvent être sélectionnés dans cette zone de liste déroulante. |
| Analyser | Effectuez l'opération d'analyse. Si l'analyse réussit, l'onglet « Résultats AST » contiendra l'AST résultant. |
| Sélectionner une règle | Cette liste déroulante vous permet de sélectionner une règle racine alternative pour le traitement de votre source d'échantillon. |
| Aide | Afficher cette rubrique d'aide. |

# Profilage de l'analyse grammaticale

Lorsque vous analysez une grammaire que vous avez créée, elle peut présenter des erreurs que vous ne pouvez pas diagnostiquer immédiatement. Pour vous aider à résoudre ces erreurs, vous pouvez révision le processus suivi par l'analyseur pour générer l'AST que vous pouvez voir, à l'aide du profileur de grammaire.

Vous analysez à nouveau le texte dans la moitié inférieure de la fenêtre de l'éditeur de grammaire, mais cette fois, l'arborescence affiche chaque règle que l'analyseur a tenté d'appliquer, où il est arrivé et s'il a réussi ou non. Les règles d'ouverture d'un fichier, de collage d'un fichier et de définition de la règle de départ restent les mêmes.

## Accéder

| | |
|---|---|
| Ruban | Développer > Code source > Éditeur de grammaire > Débogueur de grammaire > Résultats du profileur |

## Options de la barre d'outils

| Option | Action |
|---|---|
| Ouvrir le fichier | Affichez un navigateur via lequel vous pouvez localiser et ouvrir le fichier contenant la grammaire que vous souhaitez modifier. |
| Analyser | Effectuez l'opération d'analyse. Si l'analyse réussit, l'onglet « Résultats AST » contiendra l'AST résultant et l'onglet « Résultats du profil » contiendra des informations de débogage concernant le chemin emprunté par l'analyseur dans votre grammaire. Les données de profil sont extrêmement utiles lors du débogage d'une nouvelle grammaire. |
| Sélectionner une règle | Si vous souhaitez utiliser une règle racine différente pour traiter votre source d'échantillon, cliquez sur la flèche déroulante et sélectionnez la règle alternative. |
| Aide | Afficher cette rubrique d'aide. |

## Notes

- Étant donné que le profilage peut prendre beaucoup de temps pour les fichiers volumineux, l'onglet « Résultats du profil » n'est pas rempli si vous n'affichez pas cet onglet lorsque vous commencez l'analyse

# Éditeur de macros

L'éditeur de macros permet à l'utilisateur de compléter la grammaire avec une liste de mots-clés et de règles pour exclure les macros lors des opérations d'analyse grammaticale. La liste de définitions de macros est particulièrement utile lors du développement de grammaires pour des langages qui support les macros tels que C++. Elle évite la nécessité de décrire ces règles dans la grammaire elle-même et peut être utilisée avec plusieurs grammaires.

Cette fonctionnalité est disponible à partir de la version 14.1 Enterprise Architect .

## Accéder

| | |
|---|---|
| Ruban | Développer > Code source > Éditeur de grammaire > Éditeur de macros |

## Édition des macros

| | |
|---|---|
| Ouvrir le fichier | Ouvrir une liste de définitions de macros existante |
| Récent | Les listes de définitions de macros récemment ouvertes peuvent être sélectionnées à partir de cette zone de liste déroulante |
| Sauvegarder | Enregistre les modifications apportées à la liste de définitions de macros ouverte |
| Enregistrer sous | Enregistre une copie de la liste de définitions de macros existante |
| Valider | Valide la grammaire de la liste de définitions de macro |

# Exemples de grammaires

Le répertoire Exemples de code configuré par le programme d'installation Enterprise Architect contient un exemple de grammaire que vous pouvez charger dans l'éditeur de grammaire pour révision , et dans le Débogueur de grammaire pour analyser et profiler.

L'exemple de grammaire se compose de deux fichiers :

- test.ssl - un fichier source de langage d'exemple simple, dans le style de C, et
- ssl.nbnf - une grammaire pour le langage d'exemple simple

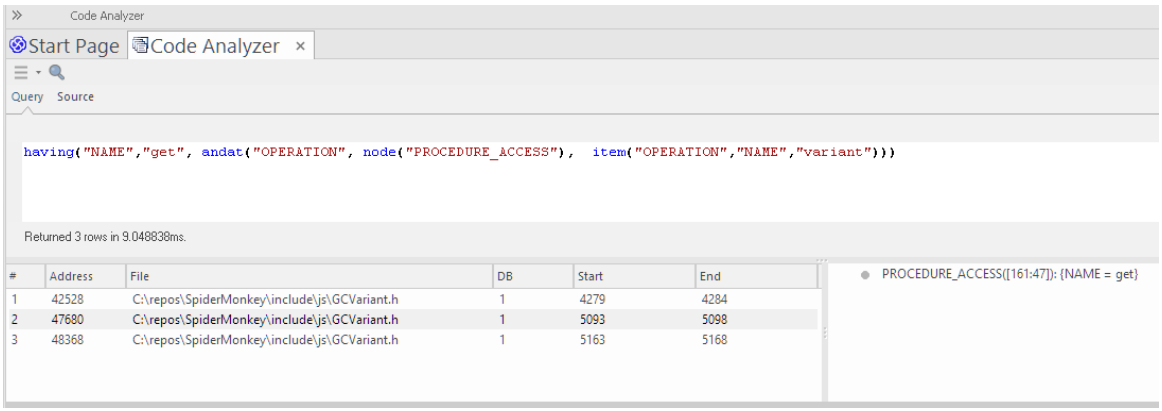
L'exemple illustre :

- Tokenisation (en utilisant le Lexer)
- Création d'un Paquetage
- Création d'une classe ou d'une interface
- Création d'un attribut
- Création d'une opération (avec paramètres)
- Importer des commentaires

Le répertoire Exemples de code contient également deux autres fichiers de grammaire que vous pouvez examiner :

- Exemple d'expressions.nBNF - cela illustre comment l'analyse des expressions est configurée et traitée, avec un texte de commentaire détaillé fournissant des explications
- Exemple CSV.nBNF - un exemple de grammaire pour le traitement des fichiers CSV

# Analyseur de code




L'analyseur de code est un outil essentiel pour quiconque travaille quotidiennement avec du code source.

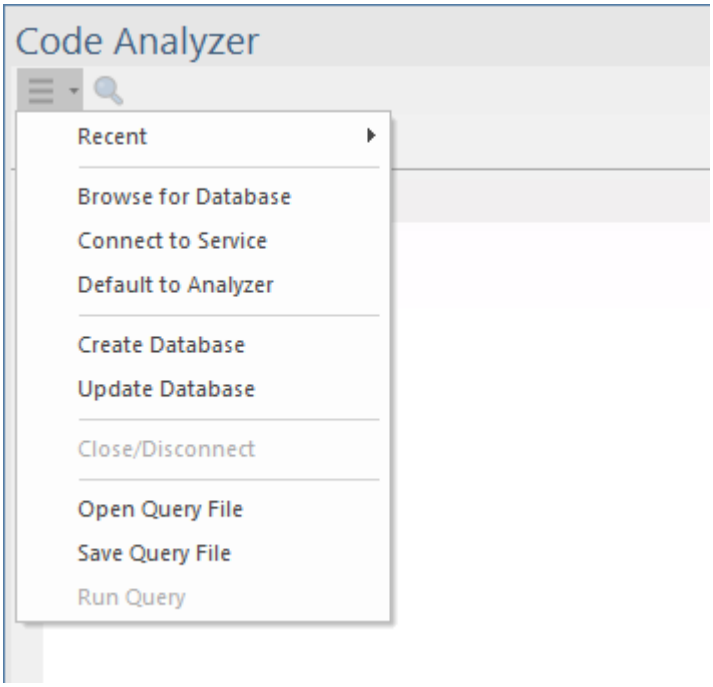
Il peut effectuer des requêtes très complexes sur des référentiels de code source à une vitesse fulgurante, que ce soit en local ou sur un service cloud Sparx Intel. Les requêtes sont composées à l'aide d'un langage de haut niveau développé par Sparx System. Le langage utilise un vocabulaire restreint mais expressif qui s'apprend facilement et permet d'interroger les métriques de code beaucoup plus rapidement que les méthodes conventionnelles.

## Accéder

| | |
|---|---|
| Ruban | Développer > Code source > Analyseur de code |

## Menu de l'analyseur de code

Le menu de l'analyseur de code s'affiche lorsque vous cliquez sur l'icône  dans le coin supérieur gauche de la fenêtre.




Le menu fournit diverses commandes pour les activités associées à l'utilisation de Code Analyzer, notamment le choix d'une base de données Code Miner à utiliser, la mise à jour de la base de données Code Miner et l'ouverture d'un fichier Query pour modification.

Ce tableau décrit chacune des commandes du menu.

| Commande | Description |
|---|---|
| Récent | Affiche un sous-menu qui fournit une liste des connexions récentes aux services et aux fichiers de base de données locaux . |
| Parcourir la base de données | Affiche une dialogue « sélecteur de fichiers », vous permettant de rechercher une base de données Code Miner sur votre machine. |
| Se connecter au service | Affiche la dialogue « Connexion à la base de données Code Miner , dans laquelle vous spécifiez les détails de connexion pour une (liste de) services de base de données Code Miner . |
| Par défaut sur Analyzer | La sélection de cette option entraîne la connexion automatique de Code Analyzer au service Code Miner configuré pour le script Analyseur d'Exécution actif, lorsque Code Analyzer est démarré. |
| Créer une base de données | Affiche la dialogue « Créer une base de données Code Miner », qui vous permet de créer une base de données Code Miner à partir d'un référentiel de code source dans le système de fichiers. |
| Mettre à jour la base de données | Affiche la dialogue « Mise à jour de la base de données Code Miner , qui vous permet d'effectuer une mise à jour incrémentielle d'une base de données Code Miner existante, afin d'intégrer les modifications récentes apportées aux fichiers de code source. |
| Fermer/Déconnecter | Ferme ou se déconnecte de la bibliothèque ou du service de base de données Code Miner . |
| Ouvrir le fichier Query | Affiche une dialogue « Ouvrir un fichier » vous permettant de choisir un fichier de requête mFQL dans le système de fichiers. |

| | |
|---|---|
| Enregistrer le fichier Query | Affiche une dialogue « Enregistrer le fichier » vous permettant d'enregistrer la requête mFQL actuelle dans un fichier nommé . |
| Exécuter Query | Exécute la requête entière ou le contenu sélectionné de la requête saisi dans l'éditeur d'onglet « Query ».<br>Raccourci F6. |

## Avant d'utiliser l'analyseur

Avant de pouvoir utiliser Code Analyzer, vous devez d'abord créer une base de données Code Miner ou en localiser une existante à laquelle Code Analyzer peut accéder. La création d'une base de données Code Miner est résumée ici, ou vous pouvez lire une description détaillée dans la rubrique d'aide *Création d'une nouvelle base de données Code Miner* .

En fonction de l'emplacement de la bibliothèque que vous utiliserez, vous devrez soit :

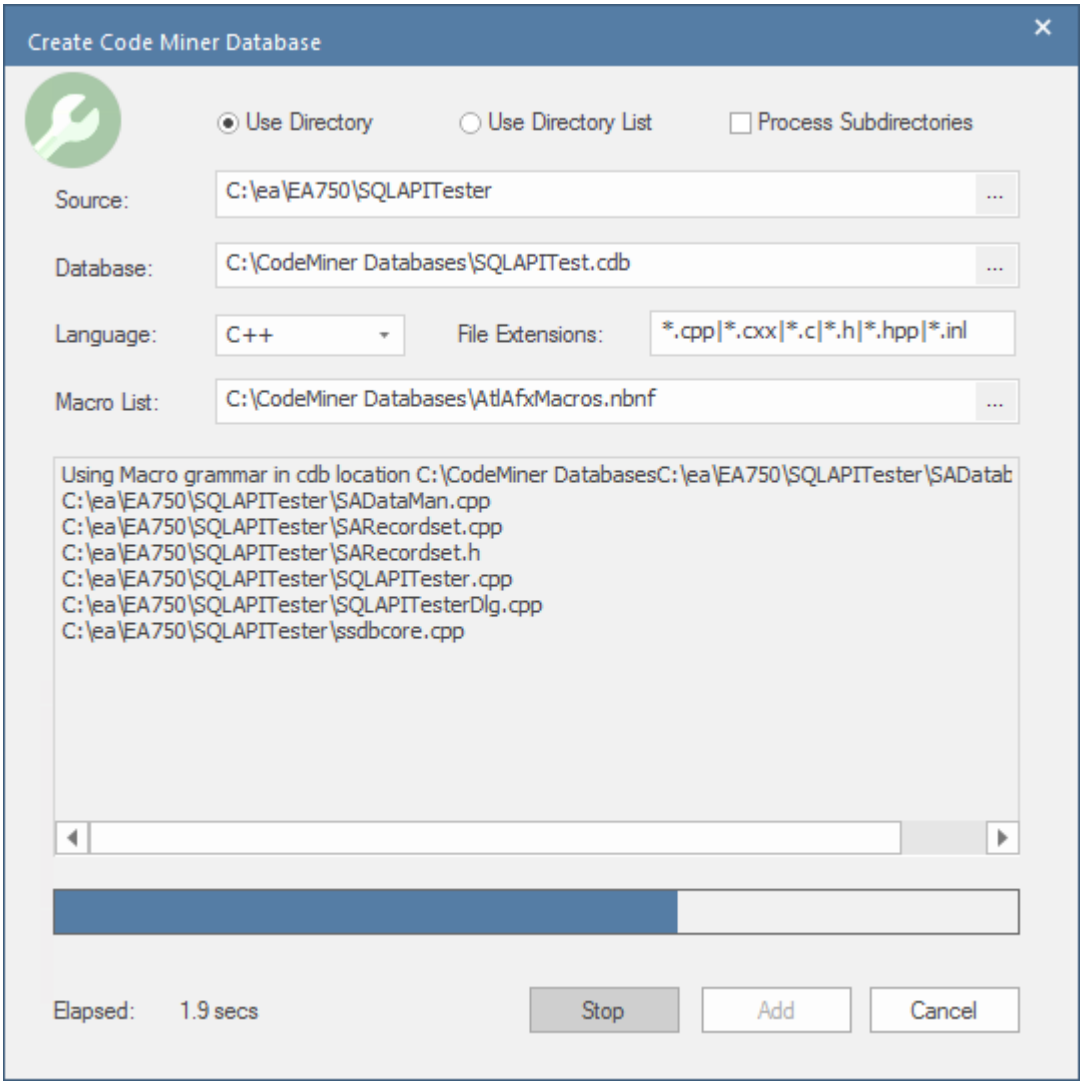
- Sélectionnez un fichier de bibliothèque Code Miner à utiliser, ou
- Connectez-vous à un service qui héberge une base de données Code Miner .

Une fois ces étapes terminées, vous êtes prêt à commencer à écrire et à exécuter des requêtes dans l’analyseur de code.

## Création d'une base de données Code Miner

Les bases de données Code Miner sont créées à partir de référentiels de code source. Le processus est similaire à la compilation de code, en utilisant la grammaire du langage pour analyser les fichiers individuels.

Il existe deux types de build : complet et incrémental. Le build complet initial peut prendre un certain temps, mais les builds incrémentales suivantes sont incroyablement rapides.




**Utiliser un répertoire comme entrée**

Vous pouvez sélectionner un seul dossier comme racine du code source que vous souhaitez compiler. Avec cette option, vous pouvez choisir d'inclure des sous-répertoires

**Utilisation d'une liste de répertoires**

Parfois, vous souhaitez utiliser plusieurs projets, mais tous les projets ne se trouvent pas dans un seul répertoire. Dans ce cas, vous pouvez créer un fichier texte qui répertorie le chemin d'accès complet à chaque dossier que vous souhaitez inclure et spécifier ce fichier texte dans le champ « Source ». Chaque chemin d'accès au répertoire doit être répertorié sur une ligne distincte.

c:\mesprojets\project1\tools\scintilla

c:\mesprojets\project2\src

d:\mylibs\lib1\src

Si vous souhaitez traiter de manière récursive les sous-répertoires d'un répertoire, faites précéder le chemin d'un point d'exclamation comme ceci :

!d:\mylibs\lib1\src

Toute ligne commençant par un caractère # est traitée comme un commentaire.

# inclure scintilla

c:\mesprojets\project1\tools\scintilla

**Langue**

Dans ce champ, vous spécifiez la langue utilisée dans le code source à partir duquel cette base de données Code Miner est construite.

Les langages disponibles sont : C++, C# , Java, XML, MDGTechnology et Custom.

**Liste des macros**

Lorsque le langage sélectionné est « C++ », le champ de sélection « Liste des macros » s'affiche. Pour C++, le succès et la profondeur des informations compilées dans la base de données peuvent être inextricablement liés à l'utilisation de macros. Ce champ peut être utilisé pour sélectionner un fichier de macro nBNF qui sera utilisé comme composant grammatical auxiliaire pour la compilation.

Par défaut, le fichier macro est celui qui se trouve dans le dossier d'installation Enterprise Architect . Vous êtes libre de modifier ou d'étendre le contenu de ce fichier pour l'adapter à vos besoins, par exemple lorsque vous devez corriger des erreurs signalées dans le fichier log de compilation.

**Grammaire**

Sparx Systems a développé des grammaires pour tous les langages répertoriés dans la liste déroulante : C++, C# , Java, XML et également MDGTechnology. Pour ces langages, un fichier de grammaire intégré est utilisé.

Il existe également une option permettant de sélectionner une langue « Personnalisée ». Lorsque « Personnalisée » est sélectionné, le champ « Grammaire » s'affiche. Ce champ permet de spécifier un fichier contenant la grammaire de votre langue personnalisée. Code Miner utilisera ensuite cette grammaire pour analyser le code source écrit dans cette langue.

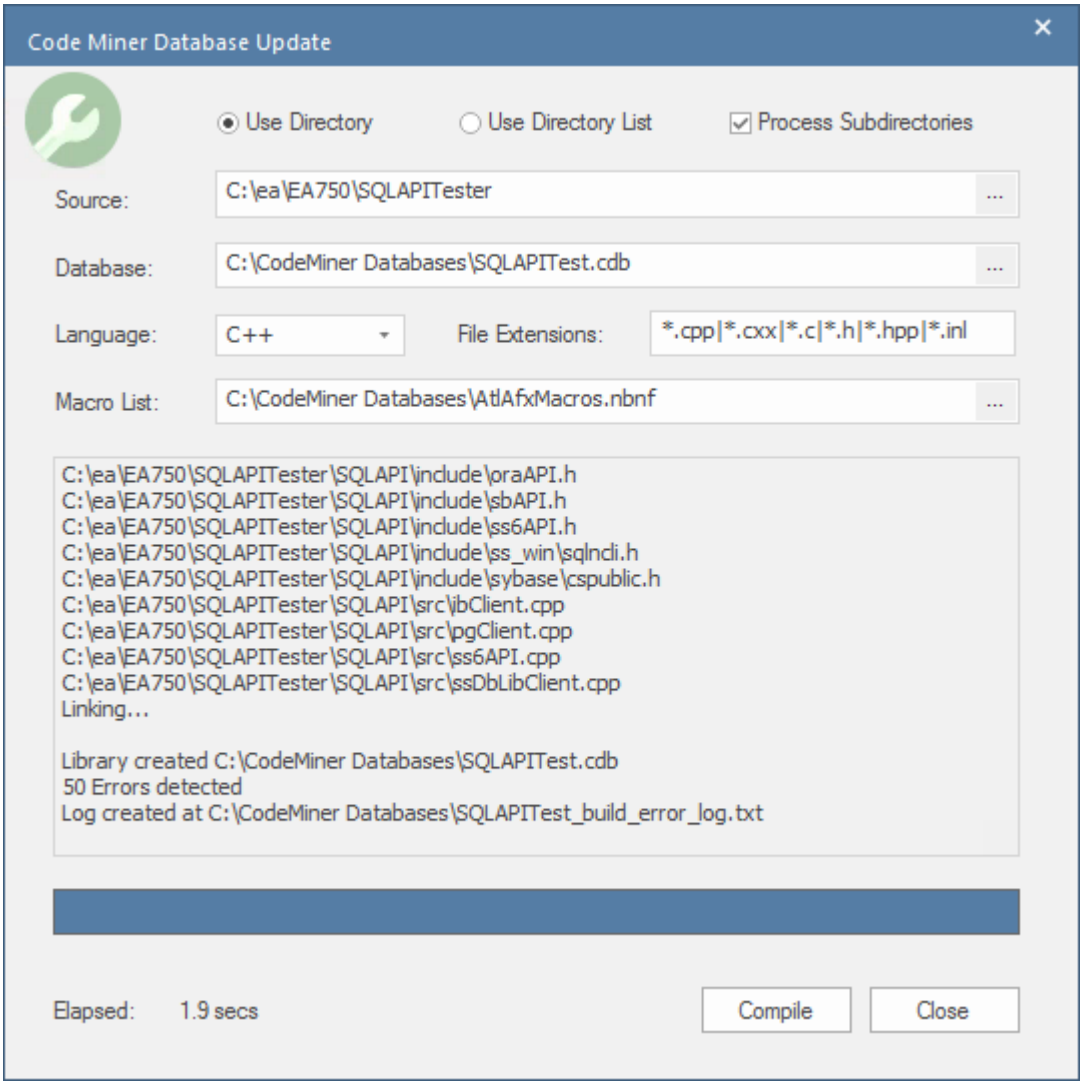
Les utilisateurs qui développent un langage personnalisé devront spécifier les règles grammaticales de ce langage et les enregistrer dans un fichier nBNF. L'éditeur de grammaire d' Enterprise Architect est conçu spécifiquement à cet effet.

Le Help Topic *Grammar Framework* fournit des informations détaillées sur la rédaction d'une grammaire nBNF.

## Mise à jour d'une base de données Code Miner

De temps en temps, vous souhaiterez mettre à jour votre base de données Code Miner . Généralement, lorsque vous avez apporté des modifications à votre code source, mais également après avoir mis à jour un fichier de grammaire ou étendu un fichier de macro.

Le processus de mise à jour d'une base de données est très similaire à la création d'une nouvelle base de données, mais plus rapide car vous ne partez pas de zéro. Choisissez simplement l'option de menu « Mettre à jour la base de données ». La dialogue « Mise à jour de la base de données Code Miner » s'affiche. Les champs de saisie seront renseignés avec les valeurs de la dernière version. Procédez comme pour « Créer une base de données Code Miner ».



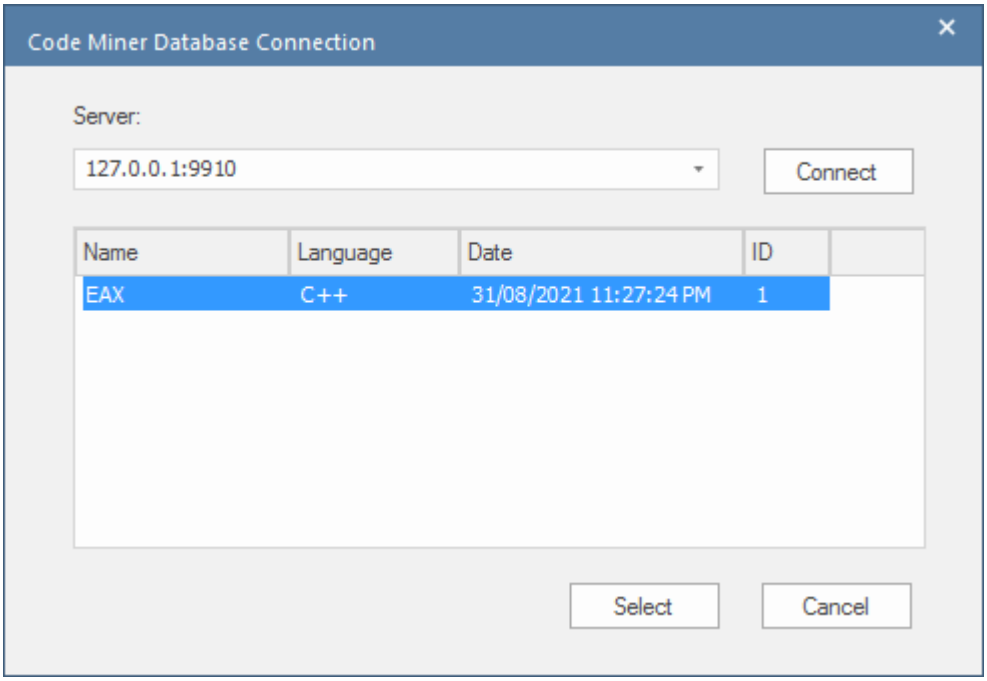

## Sélection d'un fichier de base de données Code Miner

Si vous choisissez d'utiliser un fichier de bibliothèque pour votre base de données Code Miner , choisissez l'option de menu « Rechercher une base de données ». Cela affichera un « Sélecteur de fichiers », dans lequel vous pourrez rechercher et sélectionner un fichier *.cdb.

## Connexion à un service

Lors de la connexion à un service, le dialogue répertorie toutes les bases de données hébergées par le service.

Vous pouvez choisir de sélectionner une base de données individuelle dans la liste, ou simplement cliquer sur le bouton Sélectionner, auquel cas les requêtes seront exécutées sur toutes les bases de données répertoriées par le service.

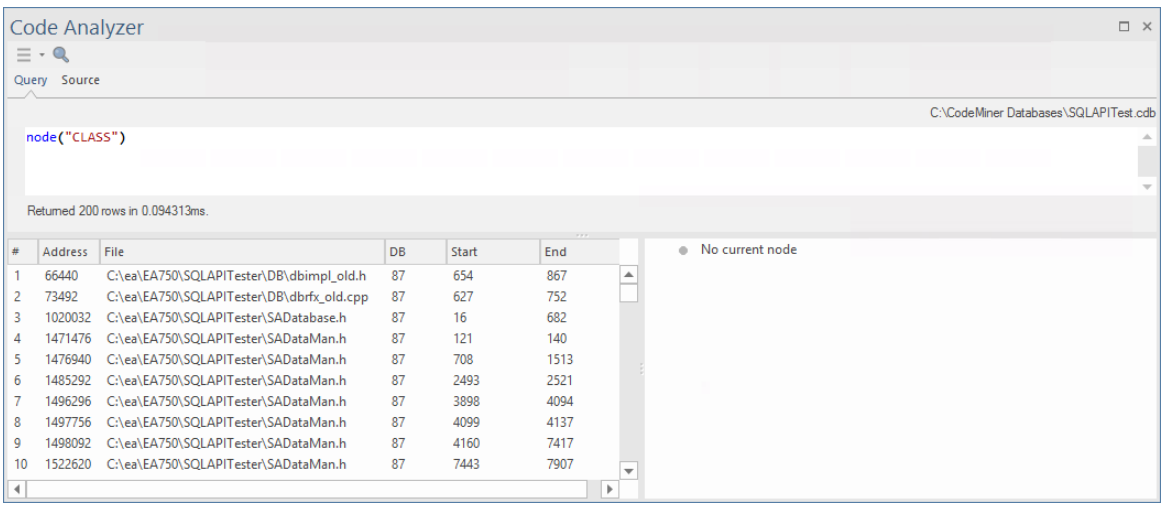



## Exécution de requêtes

Une fois connecté à une base de données Code Miner , vous êtes prêt à commencer à exécuter des requêtes.

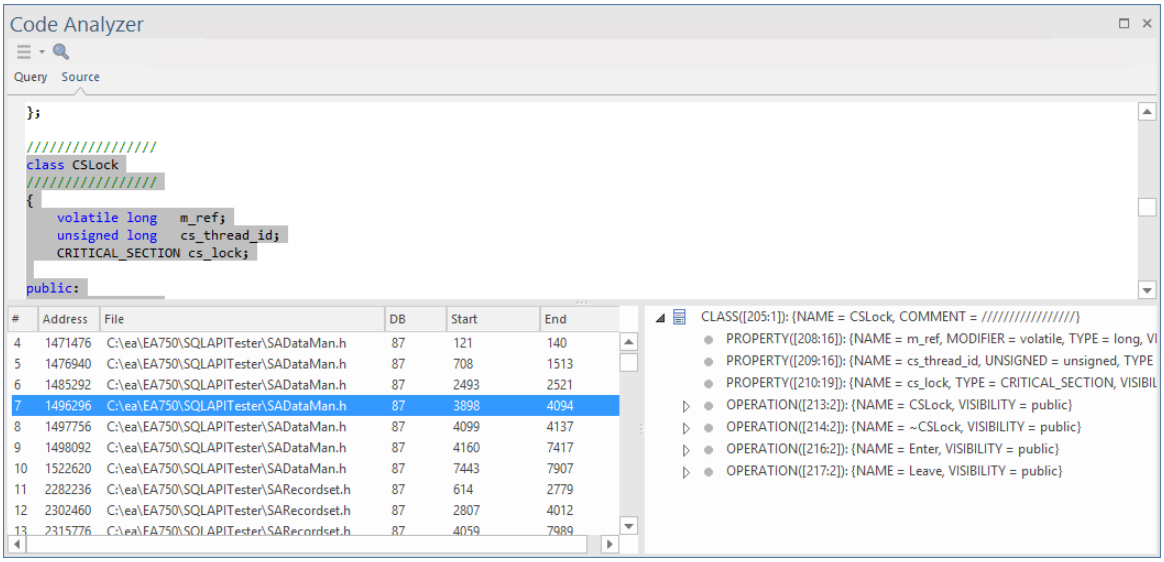
Pour exécuter une requête, sélectionnez l'onglet Query dans la fenêtre Code Analyzer, saisissez votre requête, puis cliquez sur l'icône pour exécuter la requête.

Dans cet exemple, nous avons exécuter un nœud de requête simple ( « CLASS » ), qui renverra tous les nœuds « Class » trouvés dans la base de données Code Miner .

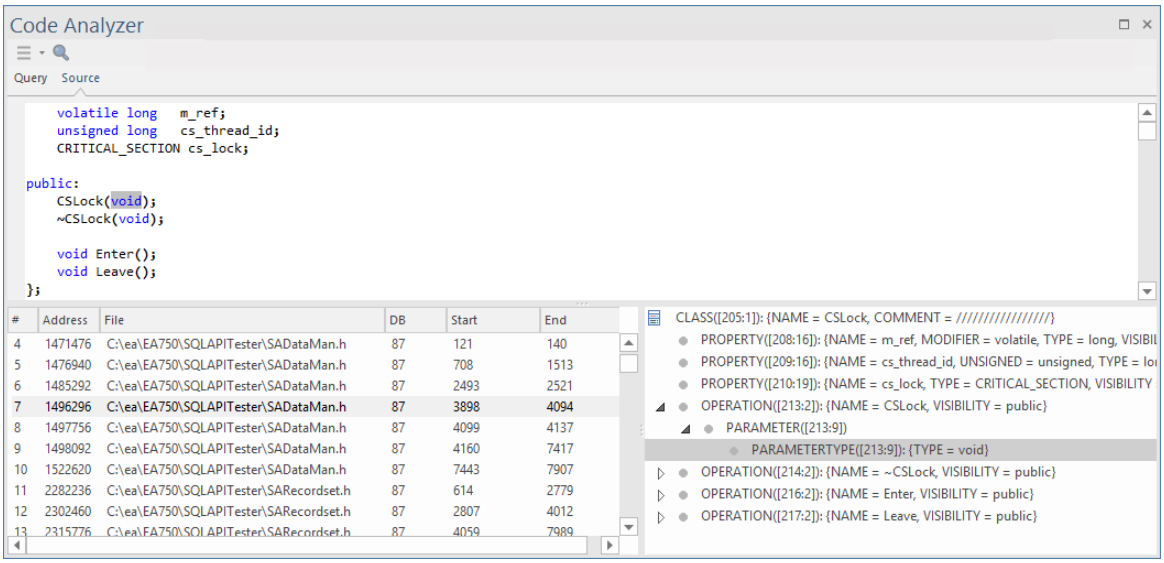



| # | Address | File | DB | Start | End |
|---|---|---|---|---|---|
| 1 | 66440 | C:\ea\EA750\SQLAPITester\DB\dbimpl_old.h | 87 | 654 | 867 |
| 2 | 73492 | C:\ea\EA750\SQLAPITester\DB\dbrfx_old.cpp | 87 | 627 | 752 |
| 3 | 1020032 | C:\ea\EA750\SQLAPITester\SADatabase.h | 87 | 16 | 682 |
| 4 | 1471476 | C:\ea\EA750\SQLAPITester\SADataMan.h | 87 | 121 | 140 |
| 5 | 1476940 | C:\ea\EA750\SQLAPITester\SADataMan.h | 87 | 708 | 1513 |
| 6 | 1485292 | C:\ea\EA750\SQLAPITester\SADataMan.h | 87 | 2493 | 2521 |
| 7 | 1496296 | C:\ea\EA750\SQLAPITester\SADataMan.h | 87 | 3898 | 4094 |
| 8 | 1497756 | C:\ea\EA750\SQLAPITester\SADataMan.h | 87 | 4099 | 4137 |
| 9 | 1498092 | C:\ea\EA750\SQLAPITester\SADataMan.h | 87 | 4160 | 7417 |
| 10 | 1522620 | C:\ea\EA750\SQLAPITester\SADataMan.h | 87 | 7443 | 7907 |

En sélectionnant un résultat dans le panneau inférieur gauche, l'onglet « Source » est activé et affiche le code source correspondant au nœud sélectionné. Les détails de ce nœud de classe sont affichés dans le panneau inférieur droit.




La sélection d'un élément de détail dans le panneau inférieur droit entraîne un rétrécissement de la sélection dans le code source, comme indiqué ici.




## Exemple Query - Intersection

As an example, this mFQL query finds all the classes that have an operation named GetOption.

```
andat( "CLASS", item("OPERATION", "NAME", "GetOption"), node("CLASS"))
```

This clause returns a set of operations for which the 'NAME' value is "GetOption":

```
item("OPERATION", "NAME", "GetOption")
```

This clause returns a set of all Class nodes:

```
node("CLASS")
```

Formal syntax:

```
andat( string:rule, set:left, set:right)
```

'andat' takes the set of operations (left), applies the rule "CLASS" (only include rows that have a CLASS parent), then intersects that set with the set of all known classes (right). If the intersection succeeds, the operation node is added to the result set, otherwise it is excluded.

## Le langage Query - mFQL

Le langage de requête utilisé avec Code Analyzer est décrit en détail dans la rubrique d'aide *Code Miner Query Language (mFQL)* .

Une brève description et quelques exemples sont également présentés ici.

Le langage mFQL est basé sur des ensembles. Chaque instruction fonctionne en utilisant les différents types d'opérations sur les ensembles, dont il n'existe que quelques-uns.

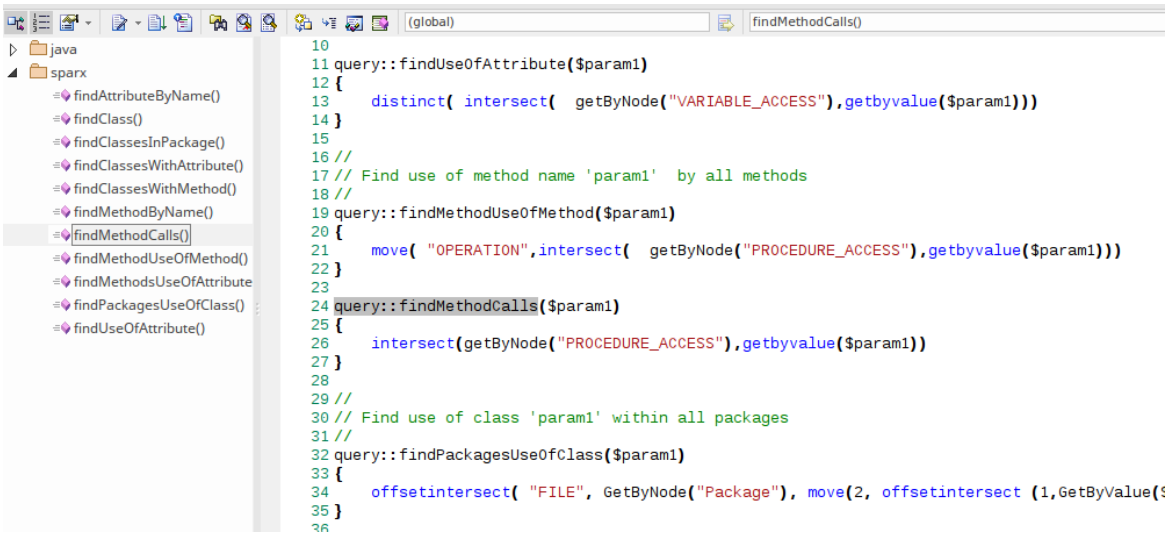
# Cadre Code Miner

Le système Code Miner offre un accès rapide et complet aux informations du code source existant. En analysant l'ensemble du code source et en stockant l'arbre de syntaxe abstraite qui en résulte dans une base de données optimisée pour la lecture, le système offre un accès complet à tous les aspects du code source d'origine, dans un format compréhensible par la machine.

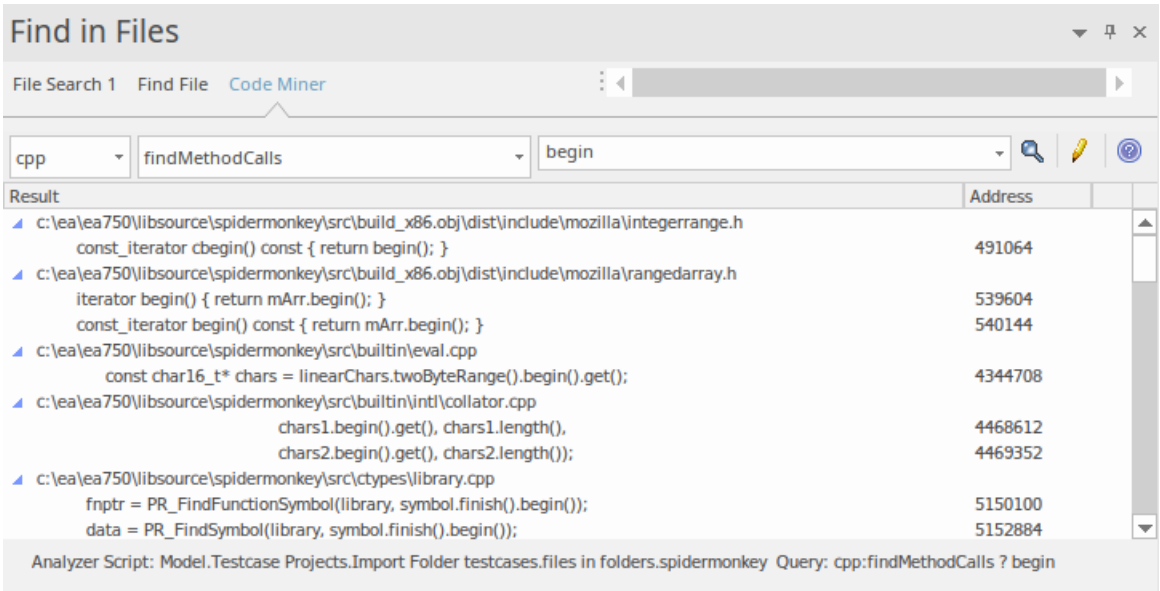
L'objectif principal du système est de permettre un accès rapide et efficace aux données cachées dans le code source. De grands efforts ont été faits pour garantir des performances maximales, tout en fournissant les interfaces les plus simples possibles. Le système peut ainsi être utilisé pour analyser la structure du programme, calculer des métriques, tracer des relations et même effectuer du refactoring.

Les informations des bases de données Code Miner sont récupérées à l'aide de requêtes écrites dans le langage Query Code Miner (mFQL) de Code Miner . Le langage lui-même est relativement simple et fournit un petit nombre de commandes. Aussi simple que soit le langage, il supporte les requêtes de taille et de complexité arbitraires. La conception offre des performances extrêmes pour toutes les requêtes, grandes et petites.

Cette fonctionnalité est disponible à partir de la version 14.1 Enterprise Architect .




L'analyseur de code d' Enterprise Architect , ses outils de recherche et les fonctionnalités Intelli-sense de ses éditeurs de code utilisent tous les informations extraites de ces bases de données.

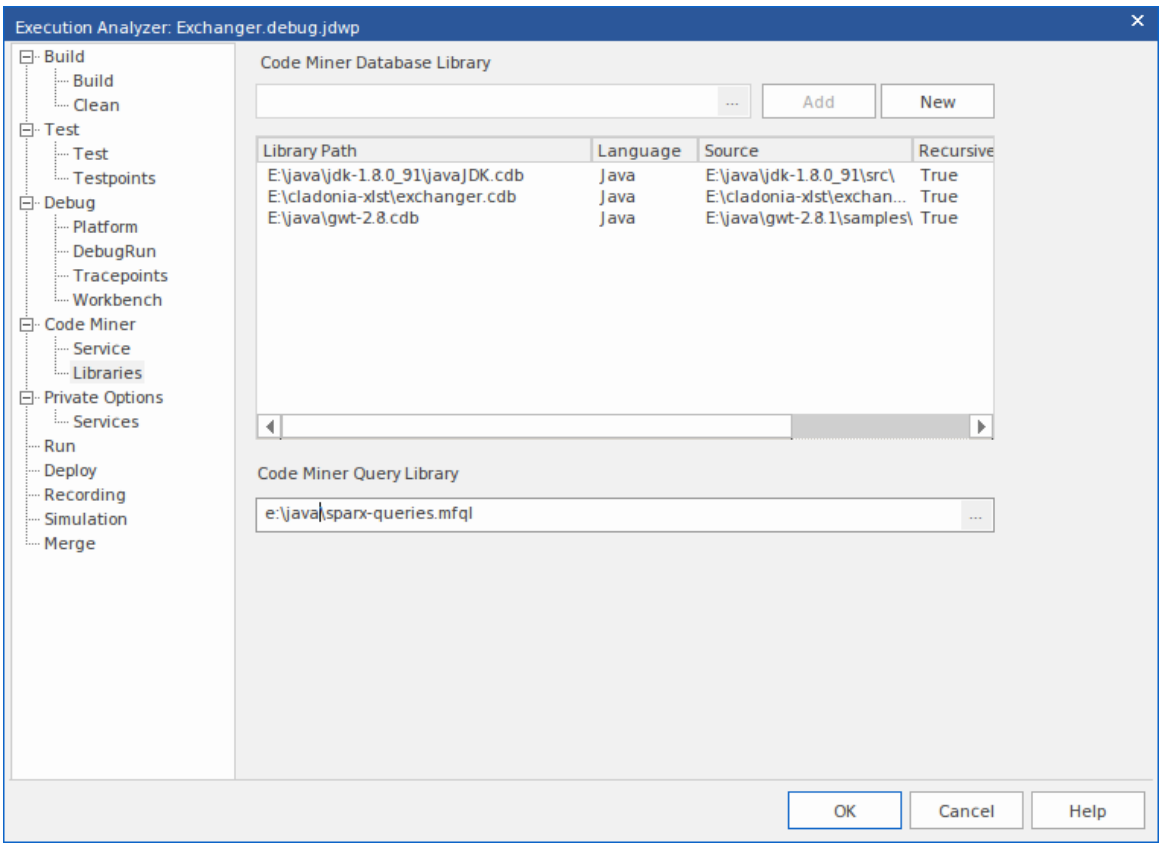



Le script d'analyse actuellement actif, ainsi que les paramètres de requête, sont indiqués en bas de la page « Code Miner » de l'outil de recherche.

# Bibliothèques Code Miner

Les bibliothèques Code Miner sont gérées dans Enterprise Architect à l'aide de l' Éditeur de Script Analyseur . Ces bibliothèques sont une collection de bases de données Code Miner , dont une existe normalement pour chaque framework ou projet. L' Éditeur de Script Analyseur permet de créer de nouvelles bases de données et d'ajouter, de mettre à jour ou de supprimer des bases de données existantes. Ensemble, ces bases de données forment la Bibliothèque Code Miner utilisée par les fonctionnalités Code Analyzer et Intelli-sense d' Enterprise Architect . La bibliothèque peut être utilisée localement ou déployée sur un emplacement de serveur où elle peut servir plusieurs clients. Vous sélectionnez le scénario à utiliser sur la page « Sparx Intel Service » du script Analyzer.

Cette fonctionnalité est disponible à partir de la version 14.1 Enterprise Architect .




## Accéder

Dans la fenêtre Analyseur d'Exécution , localisez et double-cliquez sur le script requis - la dialogue de l'éditeur de script s'affiche. Dans cette dialogue , sélectionnez la page ' Code Miner > Librairies'.

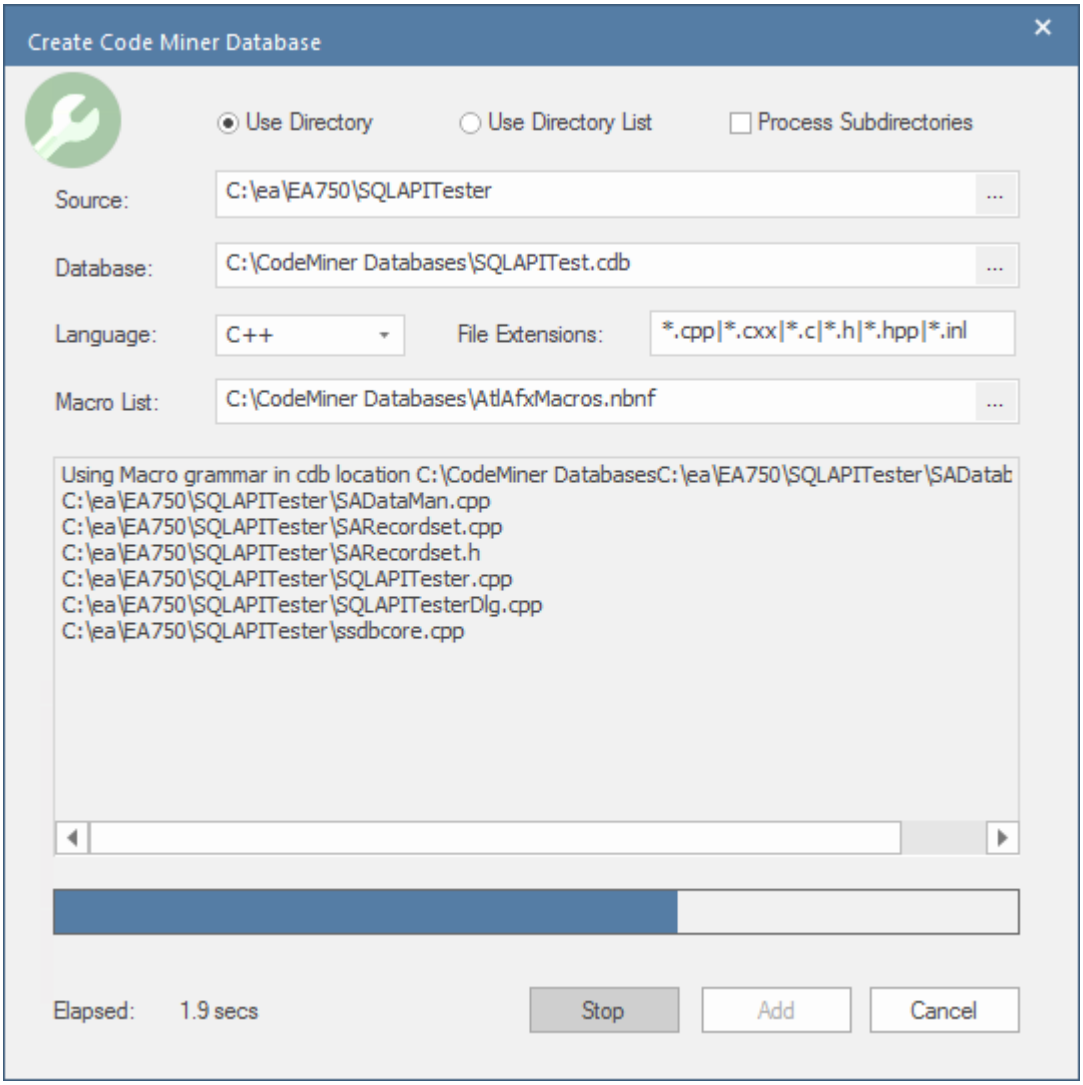
| | |
|---|---|
| Ruban | Exécuter > Outils > Analyseur, ou<br>Develop > Source Code > Analyseur d'Exécution > Edit Scripts d'Analyseur |

## Créer une nouvelle base de données

Sur la page ' Code Miner | Libraries' de l' Éditeur de Script d'Analyzer, cliquez sur le bouton 'Nouveau' pour créer une nouvelle base de données.

Dans la dialogue « Créer une base de données Code Miner », spécifiez le(s) dossier(s) contenant le code source du projet,

sélectionnez le langage de programmation et entrez le chemin de destination de la base de données Code Miner . Lorsque vous cliquez sur le bouton « Compiler », les détails de la compilation s'affichent dans la fenêtre log .




Une fois le processus terminé, cliquez sur le bouton « Ajouter » pour ajouter la base de données nouvellement créée à la bibliothèque.

Pour des informations détaillées sur la création de nouvelles bases de données, veuillez consulter la rubrique d'aide *Création d'une nouvelle base de données Code Miner* .

## Ajout d'une base de données existante

Sélectionnez une base de données Code Miner existante à l'aide du bouton de sélection " ... " dans le champ du chemin de la base de données (les bases de données Code Miner ont l'extension de fichier .CDB), puis cliquez sur le bouton Ajouter. Les détails sur la base de données sont répertoriés dans la bibliothèque. Les informations présentées affichent la grammaire du langage de programmation utilisée pour créer la base de données. Le chemin de la base de code analysé pendant la création est également affiché et indique si le processus d'analyse a été appliqué de manière récursive via des sous-répertoires.

## Mise à jour d'une base de données

De temps en temps, lorsque vous mettez à jour le code source d'un projet, vous souhaiterez mettre à jour la base de

données Code Miner créée à partir de ce code source.

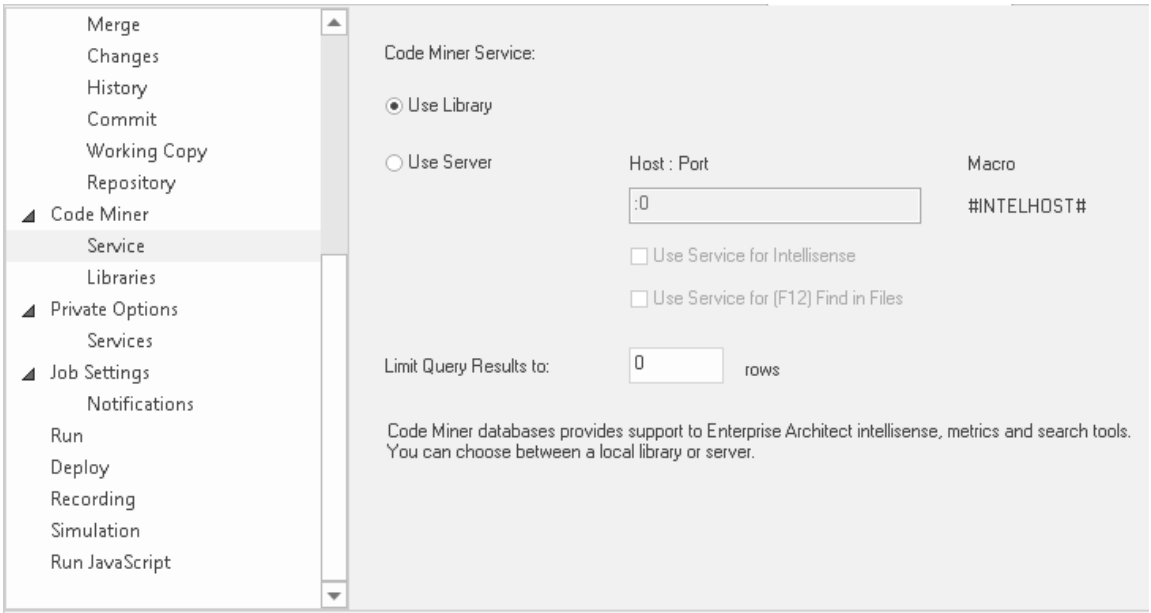
Pour mettre à jour une seule base de données Code Miner , sélectionnez-la dans la liste, cliquez-droit et choisissez « Mettre à jour la dialogue » dans son menu contextuel. Une dialogue similaire à celle « Créer une base de données » s'affiche. Cliquez sur le bouton « Compiler », Code Miner recréera la base de données à partir de la base de code mise à jour.

## Suppression d'une base de données

Pour supprimer une seule base de données Code Miner , sélectionnez-la dans la liste et choisissez « Supprimer la sélection » dans son menu contextuel.

## Configuration Enterprise Architect pour utiliser une Bibliothèque Code Miner

Dans un script d'analyse Enterprise Architect , choisissez la page « Sparx Intel Service » et sélectionnez « Utiliser Bibliothèque ». Enterprise Architect extrait ensuite ses informations Intelli-sense des bases de données répertoriées dans la section « Bibliothèques » du script d'analyse actuellement actif.



# Création d'une nouvelle base de données Code Miner

L'analyseur de code d' Enterprise Architect , les fonctionnalités Intelli-sense de ses éditeurs de code et ses outils de recherche utilisent tous les bases de données Code Miner .

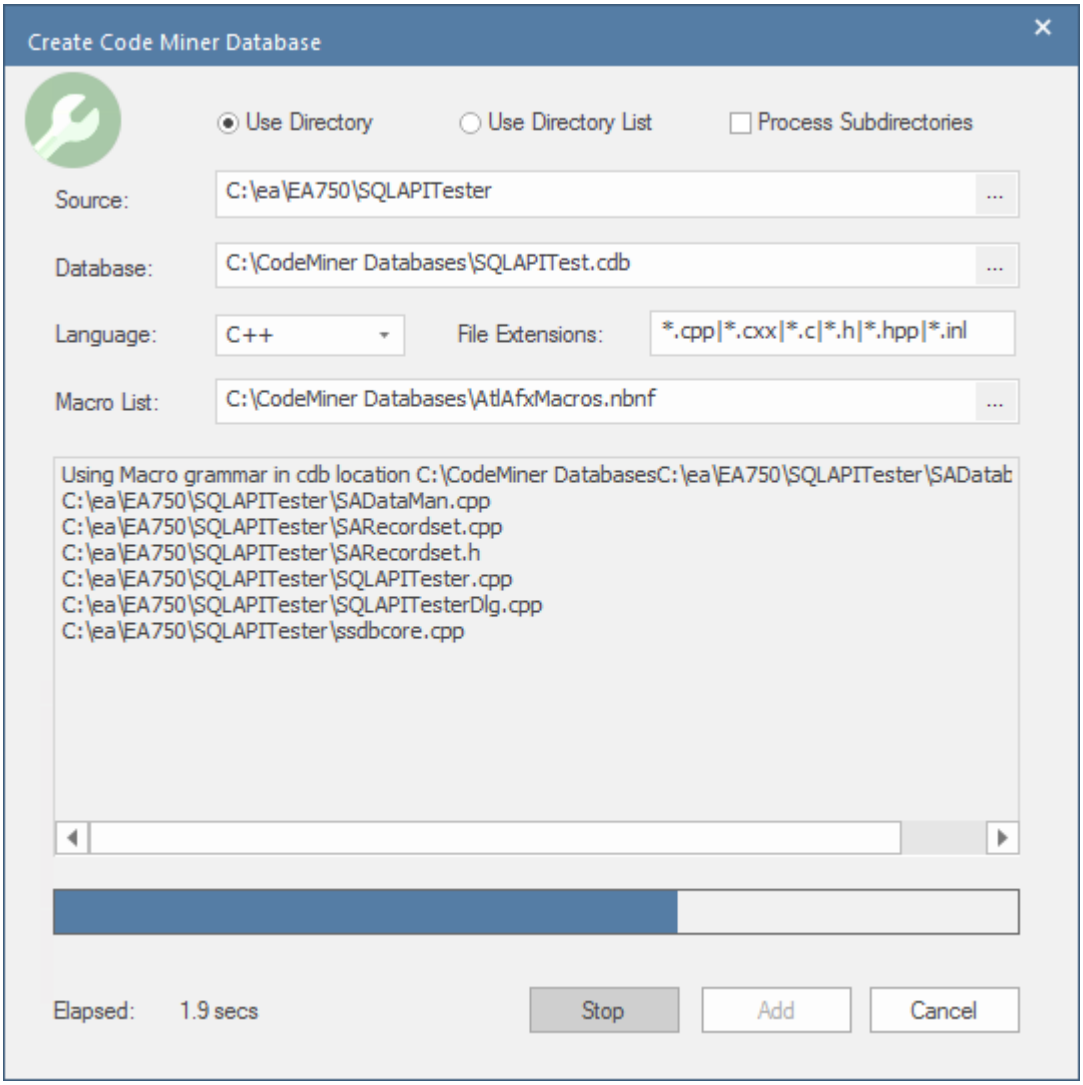
Une base de données Code Miner est créée en analysant les fichiers de code source selon les règles grammaticales du langage sélectionné et en stockant l'arbre de syntaxe abstrait résultant dans une base de données optimisée pour la lecture. Une ou plusieurs bases de données peuvent être combinées pour former une Bibliothèque Code Miner .

## Accéder

| | |
|---|---|
| Fenêtre de l'analyseur de code | Dans la fenêtre Code Analyzer, cliquez sur le bouton de menu,  , dans la barre d'outils, puis choisissez l'option de menu « Créer une base de données ». |
| Exécution Éditeur de Script Analyseur | Avec la fenêtre Éditeur de Script de l' Analyseur d'Exécution ouverte, sélectionnez la page ' Code Miner > Librairies', puis cliquez sur le bouton 'Créer'. |

## Créer Dialogue de base de données Code Miner

La dialogue « Créer une base de données Code Miner » permet de lancer le processus d'analyse des fichiers de code source pour créer une base de données Code Miner . Dans le dialogue , vous spécifiez une plage d'entrées utilisées par le processus, telles que le dossier de code source, le fichier de langue et de liste de macros, ainsi que le nom du fichier de sortie. Les champs dialogue sont décrits dans le tableau présenté ci-dessous.




| Champ | Description |
|---|---|
| Utiliser le répertoire | Sélectionnez cette option lorsque tous les fichiers sources à traiter résident dans un même répertoire.<br>Lorsque cette option est sélectionnée, la case à cocher « Traiter les sous-répertoires » est activée. |
| Utiliser la liste des répertoires | Sélectionnez cette option lorsque le code source de votre projet réside dans plusieurs répertoires distincts. Dans ce cas, utilisez le champ « Source » pour spécifier un fichier contenant une liste de répertoires contenant le code source à traiter. |
| Sous-répertoires de processus | Cette case à cocher est activée lorsque l'option « Utiliser le répertoire » est sélectionnée. Lorsque cette option est sélectionnée, le fichier de code source résidant dans l'un des sous-répertoires du répertoire « Source » spécifié sera également traité. |
| Source | Ce champ est utilisé pour spécifier le répertoire (ou les répertoires) contenant les fichiers de code source qui seront traités pour créer la base de données Code Miner .<br>Lorsque l'option « Utiliser le répertoire » est sélectionnée, ce champ est utilisé pour spécifier le dossier racine dans lequel rechercher les fichiers de code source. |

| | |
|---|---|
| | Lorsque l'option « Utiliser la liste de répertoires » est sélectionnée, ce champ est utilisé pour spécifier un fichier créé par l'utilisateur contenant une liste de noms de chemins d'accès aux répertoires contenant les fichiers sources à traiter. Cliquer sur le bouton  ouvre une dialogue « Sélecteur de fichiers » qui vous permet de rechercher et de choisir un fichier avec l'extension « .ssdirlist ». Pour plus d'informations, consultez la section *Fichier de liste de répertoires* ci-dessous. |
| Base de données | Ce champ spécifie le chemin d'accès complet du fichier de base de données Code Miner qui sera créé. L'extension de nom de fichier « .cdb » est utilisée pour ce fichier. |
| Langue | Il s'agit d'une liste déroulante dans laquelle vous spécifiez la langue utilisée dans les fichiers de code source en cours de traitement. Il existe un certain nombre de langues pour lesquelles Enterprise Architect fournit support « intégrée ». (Il existe des grammaires intégrées utilisées pour analyser les langues prises en charge).<br>Il existe également une option permettant de choisir une langue « personnalisée ». Si vous choisissez d'utiliser une langue personnalisée, vous devrez créer votre propre grammaire pour support l'analyse de cette langue. Lorsque l'option « Personnalisé » est sélectionnée, le champ « Fichier de grammaire » s'affiche, vous permettant de spécifier le fichier qui définit votre grammaire personnalisée. |
| Extensions de fichiers | Ce champ répertorie un certain nombre d'extensions de nom de fichier généralement associées aux fichiers de code source du langage choisi. Seuls les fichiers dont les extensions de nom de fichier correspondent à celles de la liste seront traités par l'analyseur. Vous pouvez ajouter ou supprimer des extensions de nom de fichier en fonction de vos besoins. |
| Liste des macros | Lorsque le langage sélectionné est « C++ », le champ de sélection « Liste des macros » s'affiche. Le champ Liste des macros vous permet de spécifier un fichier qui fournit une liste de macros que l'analyseur doit ignorer lorsqu'il les rencontre.<br>Pour le langage C++, les macros posent un problème à l'analyseur car elles masquent les constructions natives du langage. L'ajout du nom d'une macro au fichier de liste de macros et la mise à jour de la base de données permettent généralement d'effacer toutes les erreurs liées à cette macro.<br>Pour plus d'informations, consultez la section *Extension du fichier de liste de macros* ci-dessous. |
| Fichier de grammaire | Sparx Systems a développé des grammaires pour toutes les langues répertoriées dans la liste de sélection déroulante.<br>C++, C# , Java, XML et également MDGTechnology.<br>Il existe également une option permettant de sélectionner une langue « personnalisée ». Les utilisateurs qui développent une langue personnalisée devront spécifier des règles grammaticales pour cette langue et les enregistrer dans un fichier nBNF, afin que Code Miner puisse analyser correctement le code source écrit dans cette langue. L'éditeur de grammaire d' Enterprise Architect est conçu spécifiquement à cet effet.<br>Lorsque vous sélectionnez « Personnalisé » comme langue, vous devez ensuite spécifier le fichier de grammaire que vous avez créé pour cette langue, afin que Code Miner puisse analyser correctement votre code source.<br>Le Help Topic *Grammar Framework* fournit des informations détaillées sur la rédaction d'une grammaire nBNF. |
| Fenêtre de sortie | La fenêtre de sortie affiche la progression de l'analyse des fichiers de code source. Une fois l'analyse terminée, elle affiche également les noms du fichier de base de données et du fichier log qui ont été créés ainsi que le nombre d'erreurs rencontrées. |

| | |
|---|---|
| Bouton Compiler/Arrêter | Le bouton « Compiler » permet de démarrer l'opération de traitement. Ce bouton se transforme en bouton « Arrêter » une fois le traitement lancé, ce qui permet à l'utilisateur d'interrompre l'opération. |
| Bouton Ajouter | Une fois qu'une base de données a été compilée, le bouton « Ajouter » peut être utilisé pour ajouter cette base de données à une Bibliothèque Code Miner .<br>Plusieurs bases de données peuvent être ajoutées ensemble pour constituer une bibliothèque couvrant de nombreux projets de code source.<br>**Note** : lorsque la dialogue « Créer une base de données Code Miner » est ouverte à partir de la fenêtre Code Analyzer, le bouton « Ajouter » ne s'affiche pas. |

## Fichier de liste de répertoires

Si vous choisissez de spécifier un fichier de liste de répertoires, vous devrez créer un fichier texte simple utilisant l'extension de nom de fichier « .ssdirlist », qui répertorie le chemin d'accès complet à chaque répertoire que vous souhaitez traiter, avec un chemin par ligne. Par exemple :

c:\mesprojets\project1\tools\scintilla

c:\mesprojets\project2\src

d:\mylibs\lib1\src

Si vous souhaitez traiter de manière récursive les sous-répertoires d'un répertoire répertorié, faites précéder ce chemin d'un point d'exclamation comme ceci :

!d:\mylibs\lib1\src

Toute ligne commençant par un caractère # est traitée comme un commentaire :

# inclure scintilla

c:\mesprojets\project1\tools\scintilla

## Extension du fichier de liste de macros

Pour le langage C++, les macros posent un problème pour les grammaires car elles cachent les constructions natives du langage. L'analyseur ne peut pas effectuer de substitution sur les macros car elles sont souvent définies de manière conditionnelle et l'analyseur n'a aucune idée de l' architecture . Le fichier Macro List fournit une liste de macros que l'analyseur doit ignorer lorsqu'il les rencontre.

Lorsque vous créez une base de données Code Miner pour un référentiel de code source C++, des erreurs peuvent s'afficher. Lorsqu'une erreur se produit, utilisez le log des erreurs pour rechercher et inspecter la ligne de code à l'origine de l'erreur. Cela permet presque toujours d'identifier une macro à l'origine de l'échec de grammaire. L'ajout de ce nom à la liste des macros et la mise à jour de la base de données effaceront généralement toutes les erreurs liées à cette macro.

Par exemple, le log des erreurs affiche cette erreur :

C:\ea\EA750\SQLAPITester\SQLAPI\include\asa\sqlfuncs.h, ligne:12, col:18, Symbole inattendu ','.

Après inspection, la ligne de code à l'origine de l'erreur est la suivante :

FONCTION_INFO( externe , vide , _esqlentry_, arrêt sql, (SQLCA *))

(Il existe également de nombreuses autres lignes similaires utilisant la macro « FUNC_INFO ».)

Nous modifions donc le fichier de liste de macros par défaut, « AtxAflMacros.nbnf », en ajoutant cette ligne :

"FUNC_INFO" " ( " skipBalanced( " ( " , " ) " ) " ) " |

Cette ligne demande à l'analyseur, lorsqu'il rencontre la macro "FUNC_INFO" , d'appliquer la fonction skipBalanced( " ( " , " ) " ), qui prend deux paramètres ; dans ce cas, il s'agit des parenthèses ouvrantes et fermantes. L'analyseur reçoit donc pour instruction d'ignorer tout ce qui se trouve entre les parenthèses ouvrantes et fermantes.

Lorsque la modification apportée au fichier de liste de macros est enregistrée et que la base de données est recompilée (mise à jour), toutes les erreurs relatives à la macro « FUNC_INFO » ont été éliminées.

## Apprenez Plus

- Grammar Framework
- Code Analyzer

# Requêtes Code Miner

Code Miner queries are best considered as functions written in the Code Miner NBNF Query Language (mFQL). As such, they have unique names, can be grouped by namespace and can take one or more parameters. Queries are bundled together into one source file. This source file is identified to Enterprise Architect by naming it in your Analyzer Script.

When specified, the queries it contains are available in the Code Miner control. Parameters to these queries can be taken from selected text in a code editor, the model context or typed directly into the search field of the control.

This feature is available from Enterprise Architect Release 14.1.

```
188
189 namespace java
190 {
191 //
192 // Find all references
193 //
194 query::findByName($param1)
195 {
196     distinct(GetByValue( $param1 +))
197 }
198
199 query::findMethodByName($name)
200 {
201     move( 1, "METHOD", intersect( GetByNode("NAME"), GetByValue( $name ) )  )
202 }
203
204 query::findMethodCall($name)
205 {
206     filter( "METHOD_ACCESS", intersect(GetByNode("NAME"), GetByValue( $name )) )
207 }
208
```

This image illustrates an mFQL query from the Sparx Queries file distributed with Enterprise Architect installations. The syntax for composing an mFQL query and the mFQL language itself is described here.

## Query Syntax

The syntax for composing mFQL queries is:

*namespace*
{
    query:*name*([ $*param1 [*, $*param2 ]]*)
    {
        *mfql-expression*
    }
}

where:

- *namespace* names the collection of queries
- *name* is the 'function' name of the query
- *$param1 and $param2* are placeholders for argument substitutions at runtime
- *mfql-expression* is an mFQL expression

# Langage Query Code Miner (mFQL)

The Code Miner system provides fast and comprehensive access to the information in existing source code. By parsing all source code and storing the resulting Abstract Syntax Tree (AST) in a read-optimized database, the system provides complete access to all aspects of the original source code, in a machine understandable format.

The core goal behind the system is to provide access to the data hidden within source code in a timely and effective manner. Great pains have been taken to ensure maximal performance, while providing the simplest interfaces possible. As a result the system can be used to analyze program structure, calculate metrics, trace relationships and even perform refactoring.

## mFQL

mFQL is the query language of the Code Miner. The language itself is reasonably simple, providing a small number of commands. Simple as the language is, it supports queries of arbitrary size and complexity. The design provides extreme performance for all queries, great and small.

The language is set-based; it operates primarily on sets of abstract data obtained through discrete vertical indices. For our purposes, a set is an ordered array of numbers, each of which is a pointer to a node in the AST Store. A discrete vertical index provides a mechanism to retrieve sets by discrete value.

The language includes the three basic set-joining operations. These are 'intersect', 'union', and 'except'. The 'except' join is, more precisely, a 'symmetric difference' join. A 'complement' join can be achieved by using a short sub-query; this is detailed in the 'except' join documentation. The 'offsetIntersect' join is also discussed in detail there.

The Code Miner database provides three discrete vertical indices in its AST Store. These indices are 'node name', 'attribute name', and 'attribute value'. Each vertical index can be queried for a discrete value, which will return a set of all nodes where that value is present. The three vertical indices are queried using the functions 'getByNode', 'getByName' and 'getByValue', respectively.

Set 'traversal routines' provide mechanisms to filter sets based on patterns in the AST. The traversal routines are either destructive (move) or non-destructive (filter). Destructive traversals modify the set member values to point to the target node; non-destructive traversals ensure the target node exists. In both cases, nodes that cannot complete the traversal are removed.

Please note that all traversals in mFQL are upwards. Downwards traversals are technically complex, as a node could have any number of child nodes. Conversely, upward traversals are much simpler, with every node having zero or one parent node. For these reasons, downward traversals are not supported in the query language.

Although there are only a small number of operations in mFQL, the language is capable of expressing very finely grained and complex queries. The language is functional in design, and supports arbitrary nesting calls.

mFQL queries execute at lightning speed. The backend database was designed from the ground up for read performance. The query parser was hand optimized. Knowing that it always has pure ordered sets, the low-level code takes several shortcuts to perform joins with minimal work effort.

In order to use nBNF effectively one must possess a working knowledge of the target language, and an intimate knowledge of the grammar used to parse it.

# Le langage mFQL

This section provides a list of Code Miner NBNF Query Language (mFQL) queries with explanations and comments.

The queries shown here demonstrate different capabilities and different approaches to exploring and extracting data using mFQL and the Code Analyzer in Enterprise Architect. The mFQL queries help make the syntax human-readable and intuitive, and have been extended in Enterprise Architect to include additional functions necessary to do real things with Code Miner databases.

## The Query Language

String parameters are indicated by **string**, set parameters are indicated by **set** and number parameters are indicated by **numbers**.

**Notes**

1. Case sensitivity is defined by the case sensitivity of the language of the source code used to populate the database. If the source language is case sensitive (such as C++) all string literal parameters are case sensitive. If the source language is case insensitive (such as SQL) all string literal parameters are case insensitive.
2. Hierarchical traversals in mFQL are generally upwards. Downwards traversals are not optimal, as a node might have any number of child nodes. Upward traversals are much simpler, with every node having zero or one parent node. Downward-looking queries such as 'children' only query one level down.
3. Synonyms of some keywords are provided to better express a query intent or action in particular circumstances, and to support legacy queries. Synonyms are simple alternatives for the base function keyword. For example, 'type(str)' can be written as 'node(str)' or 'byNode(str)' or 'getByNode(str)'. The current specified version is the preferred one, with the synonyms only intended for use in exceptional circumstances.

| Statement | Description |
|---|---|
| type(value) | type(value)<br>Extracts a set based upon node name. The exact name for a node is defined by the grammar used to parse the original source. In this example, find all nodes within the database of type "CLASS".<br>type("CLASS")<br><br>*Synonyms:*<br>• node<br>• byNode<br>• getByNode |
| with(name) | with(name)<br>Searches the database for any element that has a named attribute matching the search string. The value of the attribute is ignored - this is a query for the attribute NAME only. All nodes with one or more attributes of the specified name are returned. If a single node has two attributes of the same name, one instance of that node is returned.<br>This example will find all elements in the database that have an attribute named "Type":<br>with("Type")<br><br>*Synonyms:*<br>• name |

| | |
|---|---|
| | • byName<br>• getByName |
| find(value) find([+] value [+ value] [+]) | find(value)<br>find([+] value [+ value] [+])<br>Search the database for any element having an attribute value with the provided search term. The match is case sensitive and must match the whole word. You can extract a set based upon an attribute value; when extracting nodes by attribute value, the values of all attributes for the node are considered.<br>Wildcards allow for specifying a subset of attribute values for a node. Wildcards can be used at either the beginning or end of a value specification:<br>• A leading concatenation symbol allows for any number of attributes preceding the first matched attribute<br>• A trailing concatenation symbol allows for arbitrary trailing attributes<br>In both cases, if the node would match without wildcards, it will match with them – the wildcard specifies any number of leading/trailing attributes, including none.<br>In this example, we retrieve a set of nodes that have their last two attributes being “.” and “sun”. The leading concatenation symbol specifies that any number of attributes (including none), with any value, can exist before the matched attributes, but none can follow.<br>find(+ “.” + “sun”)<br>The next example has a trailing wildcard. Any node with “com”, “.” and “sun” as the first three attributes will be returned. Any number of trailing attributes can exist.<br>find(“com” + “.” + “sun” +)<br>Both wildcards can be used together. In this example nodes with attributes with the three specified values as names, in order, regardless of leading or trailing attributes, will be returned.<br>find(+ “com” + “.” + “sun” +)<br>Example: Find all nodes in the database that have any attribute with a value of "CString":<br>find("CString")<br>Example: Find all nodes in the database with a set of attributes having these values in this order:<br>find("com" + "." + "sun")<br>*Synonyms:*<br>• value<br>• byValue<br>• getByValue |
| has(name,value) | has(name, value)<br>Finds all elements that have a named attribute with the value supplied. Unlike the intersection of 'find' and 'with', this query will only return rows with an exact name/value pair.<br>has("Type","CString") |
| having(name, value, set) | having (name, value, set)<br>Finds all elements within the supplied set that have a named attribute with the given value. Similar to 'has' but supplies a predefined input set to search. Whether to use |

| | |
|---|---|
| | 'has' or 'having' is generally determined by the kind of query structure being used, its depth and its readability.<br><br>Example 1: Find all Property elements with a name of "m_strName" that have a Type attribute of CString:<br>having("Type","CString",this("PROPERTY","NAME","m_strName"))<br><br>Example 2: Extend Example 1 to only include those that store a CString *:<br>having("Reference","*",<br>having("Type","CString",this("PROPERTY","NAME","m_strName"))) |
| this(type,name,value) | this(type, name, value)<br>Function finds one or more elements that have a matching TYPE, and WITH a named attribute having the specified VALUE.<br>Example: Find all operations named "Import Solution":<br>this("OPERATION","NAME","ImportSolution")<br><br>*Synonyms:*<br>• object<br>• item |
| like(name,like,set) | like(name, like, set)<br>like(name, like, set, caseSensitivity)<br>Finds a set of elements that have an attribute that starts with the search sub-string. Note that this is not a fully wild-carded search but is case sensitive and must be an exact match for the length of the search string.<br>caseSensitivity: specify one of:<br>• "CaseSensitive"<br>• "CaseInsensitive"<br>• "Default" - uses the code language to determine which to use<br>Note that case-insensitive searching can be slower.<br>Example: Find all Classes in the database whose NAME attribute starts with "CMapStr":<br>like("NAME","CMapStr",gettype("CLASS")) |
| contient (nom, contient, ensemble) | contient ( nom , terme_recherche , ensemble )<br>contient ( name , search_term , set , caseSensitivity )<br>Recherche un ensemble d'éléments dont l'attribut contient la sous-chaîne de recherche. Semblable à 'like' sauf que cela recherchera toute la string , pas seulement depuis le début.<br>caseSensitivity : spécifiez l'un des éléments suivants :<br>• "Sensible aux majuscules et minuscules"<br>• "Insensible à la casse"<br>• "Par défaut" - utilise le langage de code pour déterminer lequel utiliser<br>Note que la recherche insensible à la casse peut être plus lente.<br>Note que cette recherche peut être plus lente que l'utilisation de "J'aime".<br><br>Exemple : Rechercher toutes les classes dans la base de données dont l'attribut |

| | |
|---|---|
| | NAME contient "MapStr" :<br>contient ( "NOM" , "MapStr" , gettype ( "CLASSE" )) |
| and(set1,set2,...) | and(set1, set2, ...)<br>Returns the intersection of nodes between two or more sets. To be included in the final set, an element must exist in ALL the input sets.<br><br>*Synonyms:*<br>• intersect(set, set,...)<br>• {set, set, ...} |
| union(set1,set2,...) | union(set1,set2, ...)<br>Returns the distinct union of ALL nodes present in the input sets.<br><br>*Synonyms:*<br>• or(set, set ...)<br>• [set, set ...] |
| ancestor(str,set) | ancestor(str, set)<br>ancestor(num, set)<br>ancestor(num, str, set)<br>The ancestor function traverses each node in a set of a number of parent nodes, excluding any nodes that fail the traversal. The number of nodes to traverse, the name of the target node for the traversal, or both can be provided as parameters.<br>When the number of nodes is provided, but the target node name is not, any nodes with the specified number of parents will pass the traversal. Any node that runs out of parents will be dropped from the set.<br>When the name of the target is specified, but the number of nodes to traverse is not, any nodes with a parent with a matching name, at any point in the hierarchy, will pass the traversal. Any node with no matching parent is excluded.<br>When both the number of nodes and the target name are provided, only nodes that have a parent node with the specified name, at the specified offset, pass the traversal. All other nodes are removed from the set.<br><br>In this example the set hasParameter("CString","&",1) is moved up to an ancestor node named "OPERATION". If the move fails the node is dropped from the result.<br>ancestor("OPERATION",hasParameter("CString","&",1))<br>In this example the set is moved up one rung to its parent. If there is no parent, the node is dropped from the result.<br>ancestor(1,hasParameter("CString","&",1))<br>In this example the set is moved up three steps to its parent->parent->parent . If there is no such node, the node is dropped from the result.<br>ancestor(3,hasParameter("CString","&",1))<br><br>*Synonyms*:<br>• move |
| filter(str,set) | filter(str, set)<br>filter(num, set) |

| | |
|---|---|
| | filter(num, str, set)<br>The filter function is the same as the 'ancestor' function, except that it returns nodes from the original child set rather than new ancestor nodes. If a node is unable to pass the specified traversal, it is removed from the set. Nodes that pass the traversal are left in place, unmodified.<br>In this example the set hasParameter("CString","&",1) is tested for an ancestor node named "OPERATION". If the move fails the node is dropped from the result. The result set is a set of parameter types that meet the criteria.<br>filter("OPERATION",hasParameter("CString","&",1)) |
| match(NameA,setA,NameB,setB) | match(NameA,setA, NameB,setB)<br>'Match' takes two input sets and two attribute names and returns all those in 'setA' that have a matching record in 'setB', as determined by comparing the values of the named attributes 'strA' and 'strB'. That is, a 'setA' row is included if the value of attribute 'strA' in 'setA' exists in 'setB' as the value of an attribute of name 'strB'.<br>'Match' is useful for finding where one element feature is used in a different context elsewhere in the database. For example, where a unique element name or GUID is referenced by another element.<br>In this example, we match the attribute named 'TYPE' from the right set to the attribute 'NAME' in the left set. The result will be all "CLASS" type objects from the left set with NAME == TYPE(s) as specified in the right set.<br>match("NAME",type("CLASS"),"TYPE",this("PROPERTY","NAME","m_pLink")) |
| graph(targetType, targetName, linkType, linkName, start) | graph(targetType, targetName, linkType, linkName, start)<br>Find a recursive set of elements that form some kind of graph when linked by attribute pairs, in a manner similar to 'match'. The starter set is queried for all owned instances of the linkType with link Name and these are matched against a new query based on the targetType with targetName. The new set is filtered in a manner similar to 'match', and all elements in the new query that share the same NAME/VALUE pair as from the starter set are kept; all others are discarded. The resultant set is then fed back into the original set as the starter for the next iteration, with the results at each stage being added together to form the final result set.<br>Example: Return the Class hierarchy for a Class named "Car".<br>graph("CLASS","NAME","GENERALIZATION","GENERAL",this("CLASS","NAME","Car")) |
| prune(set_test,str,set_base) | prune(set_test, str, set_base)<br>prune(set_test, num, set_base)<br>For two sets of nodes, temporarily move one set UP to the named or numeric position in its ancestry and filter out any nodes that do not exist by strict intersection in the TEST set. The first set is the TEST set, the right or last set is the BASE set. The set returned is all the elements in the BASE set that, when moved to the TEST position, matched something in the TEST set. The returned nodes are the original nodes from the BASE set and are not moved up when returned.<br>Example 1 finds the set of parameter types used for operation parameters named "CustomerName" across the whole database.<br>prune(this("PARAMETER","NAME","CustomerName"),"PARAMETER",type("PARAMETERTYPE")) |

| | |
|---|---|
| | Example 2 finds all Properties of a Class named Customer, assuming the grammar used to compile the database placed the Property definition two hierarchy levels below the Class definition.<br>prune(this("CLASS","NAME","Customer"),2,type("PROPERTY")) |
| andat(str,test,base) | andat(str, base, test)<br>andat(num, base, test)<br>andat(num, str, base, test)<br>For two sets of nodes, temporarily move one set UP to the named or numeric position in its ancestry and filter out any nodes that do not exist by strict intersection in the TEST set. The first set is the TEST set, the right or last set is the BASE set. The set returned is all the elements in the BASE set that, when moved to the TEST position, matched something in the TEST set. The returned nodes are the original nodes from the BASE set and are not moved up when returned.<br>Similar to 'prune', this query supports additional options and structures the inputs in a different order to facilitate different kinds of stacked searches.<br>The 'andat' function performs both a non-destructive tree traversal and an intersect join in one operation. Each node in the left set is traversed according to parameters provided, then the result of the traversal is intersected with the right set. If the intersect passes, the original node is added to the result set. If the intersect fails, the node is excluded from the result set.<br>The traversal parameters for 'andat' are the same as for 'ancestor' and 'filter'. For more information about the traversal parameters, see the 'ancestor' function.<br>Example: For the set of all "PROPERTY" nodes in the database, move them up to a parent node of type CLASS and then intersect the result with the right hand set - in this case a CLASS named CDiagram. All nodes that pass this test are returned as PROPERTY nodes, effectively giving the set of all properties of the Class CDiagram.<br>andat("CLASS",type("PROPERTY"),this("CLASS","NAME","CDiagram"))<br>*Synonyms:*<br>• offsetIntersect<br>• offsetx |
| unique(left,right) /<br>except(left,right) | unique(left, right)<br>except(left, right)<br>Except joins return sets that contain any nodes from either set that do not appear in both sets. This join is similar to a bitwise XOR operation. In set theory, this type of join is referred to as a 'symmetric difference join'.<br>{1, 2, 3} excepted with {2, 3, 4} results in {1, 4} |
| omit(left,right) /<br>exclude(left,right) | omit(left, right)<br>exclude(left, right)<br>Exclude joins return a set that contains all nodes from the left set that do not appear in the right set. In set theory, this type of join is referred to as a 'relative complement join'.<br>{1, 2, 3} complemented with {2, 3, 4} results in {1} |
| differ(name,set,name,set) | differ(name, set, name, set)<br>Return a set of nodes that do not have a matching row in another set, using a NAME/VALUE pair from each set to match on. |

| | |
|---|---|
| | Example: This more complex example tests the complete set of Generalizations for a Class hierarchy and identifies missing or unresolved Class names in the total inheritance hierarchy. Like the 'match()' function discussed later, this function iterates over attribute name/value pairs as specified in the left and right input sets, but only includes rows in the final set where there is NO match.<br><br>differ(<br>"GENERAL",<br>children("GENERALIZATION",<br><br>graph("CLASS","NAME","GENERALIZATION","GENERAL",<br><br>this("CLASS","NAME","CMainFrame"))),<br>"NAME",<br>graph("CLASS","NAME","GENERALIZATION","GENERAL",<br>this("CLASS","NAME","CMainFrame"))<br>) |
| children(type,set) | children(type, set)<br>Return a set of child nodes of a specified type for one or more parents in the source set. For all children regardless of type, use an empty string.<br>For example, in the first query we return ALL first level children of the CMainFrame Class. In the second query we restrict the nodes returned to be of type "REGION" only.<br>children("",this("CLASS","NAME","CMainFrame"))<br>children("REGION",this("CLASS","NAME","CMainFrame")) |
| childcount(num,type,set) | childcount (num,type,set)<br>Return nodes that exactly match the number of specified children of a specified type. For example, only return operations that have 5 parameters.<br>An example usage is in specifying an exact operation signature, so we check firstly that parameter1 and parameter2 match the type we are querying for, then move those to their operation ancestor and intersect the result with the operation name "GetFromCache" we are interested in. To rule out spurious hits with operations having more than 2 parameters, we explicitly add childcount(2, ...) to ensure we only get operations that have 2 parameters.<br><br>childCount(2,"PARAMETER",<br>{<br>ancestor("OPERATION",hasParameter("CString","&",1)),<br>ancestor("OPERATION",hasParameter("CString","&",2)),<br>this("OPERATION","NAME","GetFromCache")<br>}<br>) |
| byAddress(num) | byAddress(num)<br>The byAddress function is used in applying the results of one query to another. For example, we might have a node of particular interest, and want our query to return only nodes that join (in some way) to the specified node.<br>byAddress(node: number) |

| | |
|---|---|
| | This example builds a set containing the single node related to the address specified:<br>byAddress(11256) |
| byPosition(File, Offset) | byPosition(File, Offset)<br>The byPosition function is used to return the inner-most node that covers a certain position in a file. This function is useful for locating a position in the AST based upon a file position. |
| distinct(set) | distinct(set)<br>The distinct function ensures that a set has no duplicate values. All duplicate values are excluded from the result set. |

# Extraction d'ensembles

These procedures extract sets from discrete vertical indices. There are three indices available, each with a specific extraction function. String literal parameters to these functions could be case sensitive. Case sensitivity is defined by the language of the source code used to populate the database. If the source language is case sensitive (as C++ is) all string literal parameters are case sensitive. If the source language is case insensitive (as SQL is) all string literal parameters are case insensitive.

## type

type(value: string)

Extract a set based upon a node name. The exact name for a node is defined by the grammar used to parse the original source. In this example, all nodes with the name "OPERATION" are returned.

**type("OPERATION")**

_

## with

with(value: string)

Extract a set based upon attribute name. All nodes with one or more attributes of the specified name are returned. If a single node has two attributes of the same name, one instance of that node is returned. This example returns all nodes with one or more attributes named "NAMEPART".

**with("NAMEPART")**

## find

find([+] value: string [+ value: string] [+])

Extract a set based upon an attribute value. When extracting nodes by attribute value, the value of all attributes for the node are considered. Wildcards allow for specifying a subset of attribute values for a node.

When a single value is provided, all nodes that have a single attribute with the value specified are returned. If a node has any other attributes, it is excluded. In this example, all nodes with exactly one attribute with the value of 'i' are returned.

**find("i")**

More than one value can be specified by using a concatenation symbol. When more than one value is specified, the resulting set will contain all nodes that have attributes with exactly the values specified, in the order specified. Any node with extra leading or trailing attributes is excluded. This example retrieves a set of all nodes with a set of three attributes with the values “com”, “.” and “sun”, in that order.

**find("com" + "." + "sun")**

Wildcards can be used at either the beginning or end of a value specification. A leading concatenation symbol allows for any number of attributes preceding the first matched attribute. A trailing concatenation symbol allows for arbitrary trailing attributes. In both cases, if the node would match without wildcards, it will match with them – the wildcard specifies any number of leading/trailing attributes, including none.

In this example, we retrieve a set of nodes that have their last two attributes being “.” and “sun”. The leading concatenation symbol specifies that any number of attributes (including none), with any value, can exist before the matched attributes, but none can follow.

**find(+ “.” + “sun”)**

The next example has a trailing wildcard. Any node with attributes “com”, “.” and “sun” as the first three attributes will be returned. Any number of trailing attributes can exist.

**find(“com” + “.” + “sun” +)**

Both wildcards can be used together. In this example, nodes with attributes named as the three values specified, in order, regardless of leading or trailing attributes, will be returned.

**find(+ “com” + “.” + “sun” +)**

—

# Parcours d'ensemble

## ancestor

ancestor(count: number, source: set)

ancestor(value: string, source: set)

ancestor(count: number, value: string, source: set)

The 'ancestor' function traverses each node in a set up a number of parent nodes, excluding any nodes that fail the traversal. The number of nodes to traverse, the name of the target node for the traversal, or both can be provided as parameters.

- When the number of nodes is provided, but the target node name is not, any nodes with the specified number of parents will pass the traversal; any node that runs out of parents will be dropped from the set
- When the name of the target is specified, but the number of nodes to traverse is not, nodes with a parent with a matching name at any point in the hierarchy will pass the traversal; any node with no matching parent is excluded
- When both the number of nodes and the target name are provided, only nodes that have a parent node with the specified name at the specified offset pass the traversal; all other nodes are removed from the set

It is possible - even likely - that these calls will generate sets having duplicate values. This is by design, as the concrete rules for sets do not define them as being discrete. If (as in most cases) you want your set to be discrete, use the 'distinct' function described in the *The mFQL Language* Help topic.

This sample extracts a set of all nodes named 'OPERATION', then traverses each node up one level to its immediate parent. Any 'OPERATION' node with no parent is excluded.

**ancestor(1, getByNode("OPERATION"))**

This sample extracts a set of all nodes named 'OPERATION', then traverses each node up to the first 'CLASS' parent node. Any 'OPERATION' node with no 'CLASS' parent is excluded.

**ancestor("CLASS", getByNode("OPERATION"))**

This sample extracts a set of all nodes named 'OPERATION', then traverses each node up one level to its immediate parent. If the parent node is not a 'CLASS' node, or the node fails to traverse though a lack of parent nodes, it is excluded.

**ancestor(1, "CLASS", getByNode("OPERATION"))**

–

## filter

filter(count: number, source: set)

filter(value: string, source: set)

filter(count: number, value: string, source: set)

The 'filter' function is the same as the 'ancestor' function, except that it does not modify nodes – it is non-destructive. If a node is unable to pass the specified traversal, it is removed from the set. Nodes that pass the traversal are left in place, unmodified.

It is often desirable to filter a set by the current node name. This can be used to ensure that the nodes returned from a 'with' or 'find' call are of a particular node type. This example returns all nodes with an attribute with the value of “CFoo”, where the resulting node is a “TYPE” node.

**filter(0, “TYPE”, find(“CFoo”))**

For more details on the use of the 'filter' function, see the 'ancestor' function.

—

# Joindre un ensemble

## and

and(left: set, right: set)

An 'and' join will return a set containing all nodes that exist in both the left and right set. This join is comparable to a bitwise AND operation. In set theory, this type of join is called an 'intersection'.

{1, 2, 3} intersected with {2, 3, 4} results in {2, 3}

This example returns a set that contains all nodes that have a single attribute with the name of "TYPE" and the value of "int".

```
and(
find("int"),
with("TYPE")
)
```

## union

union(left: set, right: set [, right: set])

'Union' joins return a set that includes all nodes found in either the left or the right set. This join is used to combine the results of two or more sub-queries into a single set. A 'union' join is similar to a logical OR operation. In set theory, the 'union' join is known as a union.

The 'union' join is able to operate on more than two sets. The result is a set that contains all nodes from all supplied sets. The 'union' join is the only join able to operate on more than two sets.

The result of a 'union' join is always a discrete set, unless one of the source sets contained duplicates. This means that duplicates in source sets will be preserved, but the 'union' join itself will not generate duplicates.

{1, 2, 3} unioned with {2, 3, 4} results in {1, 2, 3, 4}

This sample creates a set containing all nodes with an attribute named “TYPE” or a single attribute with the value of “int”.

```
union(
find("int"),
with("TYPE")
)
```

## except

except(left: set, right: set)

'except' joins return sets that contain any nodes from either set that do not appear in both sets. This join is similar to a bitwise XOR operation. In set theory, this type of join is referred to as a 'symmetric difference' join.

{1, 2, 3} excepted with {2, 3, 4} results in {1, 4}

For more information on the 'symmetric difference' join in set theory, see https://en.wikipedia.org/wiki/Symmetric_difference

This sample returns a set of all nodes with an attribute named "TYPE" but no single attribute with the value of "int", plus

all nodes with an attribute with the value of "int" that are not named "TYPE".

```
except(
 find("int"),
 with("TYPE")
)
```

## exclude

exclude(left: set, right: set)

'exclude' joins return a set that contains all nodes from the left set that do not appear in the right set. In set theory, this type of join is referred to as a relative complement join.

{1, 2, 3} complemented with {2, 3, 4} results in {1}

This sample returns a set of all nodes with a value of “int” that are not “TYPE” nodes:

```
Exclude(
find(“int”),
with(“TYPE”)
)
```

## andat

andat(count: number, left: set, right: set)

andat(value: string, left: set, right: set)

andat(count: number, value: string, left: set, right: set)

The andat function performs both a non-destructive tree traversal and an intersect join in one operation. Each node in the left set is traversed according to parameters provided, then the result of the traversal is intersected with the right set. If the intersect passes, the original node is added to the result set. If the intersect fails, the node is excluded from the result set.

The traversal parameters for andat are the same as for 'ancestor' and 'filter'. For more information about the traversal parameters, see the 'ancestor' function described in the *Set Traversal* Help topic.

This sample takes all “NAME” nodes, traverses them up one parent, and intersects them with a set of all “CLASS” nodes. If a “NAME” node passes both the traversal and intersect join, it is added to the result set. The result is a set of all “NAME” nodes whose immediate parent is a “CLASS” node.

```
andat(1,
type(“NAME”),
type(“CLASS”)
)
```

—

# Service Intel Sparx

Le programme de service Intel de Sparx permet aux projets de développement et aux acteurs d'obtenir des informations précieuses sur les bases de code et les cadres logiciels avec lesquels ils travaillent. Le service agit en tant que fournisseur pour les clients Enterprise Architect , permettant l'accès à Intelli-sense dans l'édition de code et à des résultats de recherche perspicaces dans les outils de recherche.

Le service Sparx Intel fait partie de l'ensemble des services Sparx Satellite. Le service peut être exécuter sur un réseau local ou Cloud exécutant Microsoft Windows . Le service Sparx Intel Satellite peut être installé en tant que service Windows ou exécuter en tant que processus autonome. Le service permet à plusieurs clients Enterprise Architect d'accéder aux mêmes informations et de les interroger à partir de nombreux domaines et cadres logiciels différents.

Cette fonctionnalité est disponible à partir de la version 16.0 Enterprise Architect

# Configuration du service Intel Sparx

Le programme SparxIntelService.exe exécute un ou plusieurs services Intel pour Enterprise Architect . Le programme se trouve dans le même dossier d'installation qu'Enterprise Enterprise Architect et utilise un fichier de configuration qui nomme les services pouvant exécuter sur la machine locale.

Dans les exemples de cette rubrique, le programme va tenter d'utiliser le fichier c:\mystuff\myservices.config. Il va rechercher un service nommé *EA* et, s'il le trouve, le démarrer.

Service d'écoute SparxIntelService.exe = EA config = c:\mystuff\myservices.config

## The Config File Format

The configuration file has this format:

```
# comment
# comment
# comment
{                   # start of service definition
   ...              # list of directives as pairs
}                   # end of service definition
{                   # start of service definition
   ...              # list of directives as pairs
}                   # end of service definition
```

Comments are indicated by the # character.

If the config directive is omitted (not recommended), the program will look for a config file of the same name as the program, in the same directory as the program.

In this example the program will attempt to use the file SparxIntelService.config in the same folder:

SparxIntelService.exe listen service:EA

| Directif | Description |
|---|---|
| nom | Lorsqu'un service est nommé sur la ligne de commande, le service avec l' attribut de nom correspondant sera démarré. |
| statut | Lorsque le statut |
| chargement paresseux | Lorsque lazyload est « true », toute base de données Code Miner sera chargée avec retard jusqu'à ce qu'une demande Intel soit adressée au service. |
| niveau de log | Définit le niveau d'informations enregistrées, sous la forme d'une combinaison de mots-clés {information, avertissement, erreur} séparés par un « \| ». Par exemple : loglevel=Information\|avertissement\|erreur |
| déconnexion | Spécifie le chemin d'accès complet du fichier log dans lequel écrire. Par exemple : logoutput =c:\logfiles\intel-service-project1. log |
| base de données | Spécifie le chemin d'accès complet de la base de données Code Miner à charger. Par exemple : |

| | |
|---|---|
| | base de données =c:\intel--service\project1.cdb<br>Plusieurs directives « base de données » sont autorisées, chacune spécifiant une base de données différente. |
| permettre | Identifie l'adresse IP autorisée à se connecter au service sur le port. Par exemple :<br>autoriser=localhost<br>autoriser=127.0.0.1<br>allow=172.160.* (les caractères génériques sont autorisés lorsque le « réseau »<br>la directive a une valeur de « réseau » ou « public »,<br>mais pas « local ») |
| réseau | Permet de restreindre les connexions de service.<br>• local - le service n'écoutera aucune connexion autre que localhost<br>• réseau - lorsqu'il est utilisé avec les directives génériques « allow », permet aux clients sur une adresse IP autorisée de se connecter<br>• pubien - permet toute connexion |
| montrer | Lorsque la valeur est « true », la fenêtre de la console du service s'affiche ; la valeur par défaut est « false ». |
| port | Le port sur lequel le service écoutera. |

## The Service Configuration Template

When choosing the 'Execute > Tools > Services > Code Miner Service > Edit Configuration File' ribbon option you display the Windows 'Save As' browser through which you can choose either the config file to open or where a file should be created.

If no config file is recorded in the registry and you specify a non-existent filename, that file is created, filled with a 'bare bones' configuration skeleton and saved. The selected/new configuration is then shown in the Enterprise Architect default editor.

The 'bare bones' template is shown here.

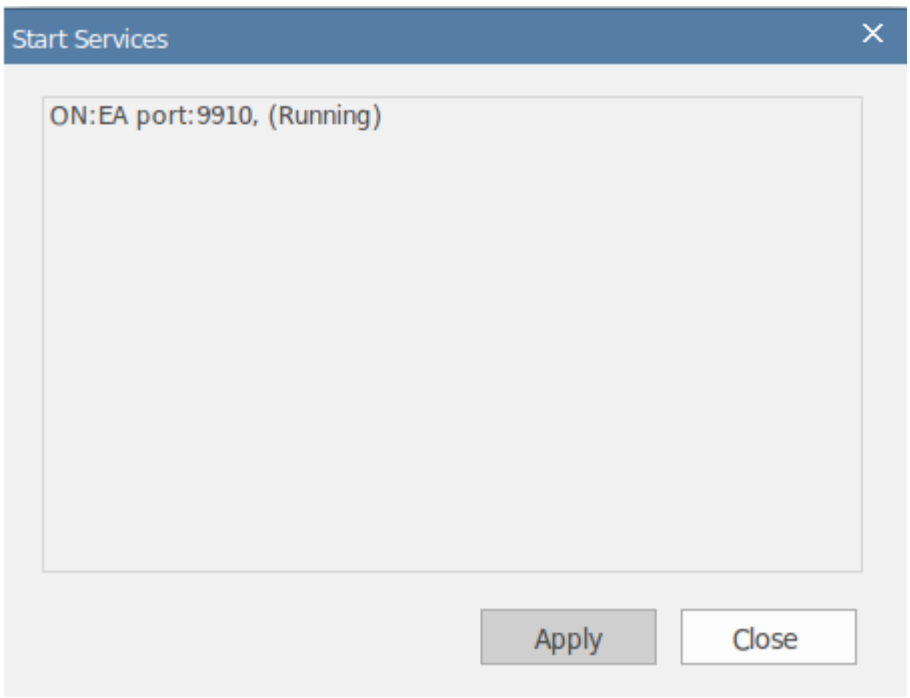
```
#-------------------------------------------------------------------------------
# Sparx Intel Service Configuration File
# -------------------------------------------------------------------------------
# This file is used to describe one or more intel services and the code miner databases that they support
# This file can be used in EA to manage a number of services on the local machine
# -------------------------------------------------------------------------------
# Service Attributes
# -------------------------------------------------------------------------------
# name                  The unique name of the service in this file
# status                "ON" - service can run, "OFF" service will never run
# lazyload                    "true" - databases are loaded n demand, "false" - databases are loaded when service
starts
# port                        Unique Port number that service will listen on and EA will connect to
```

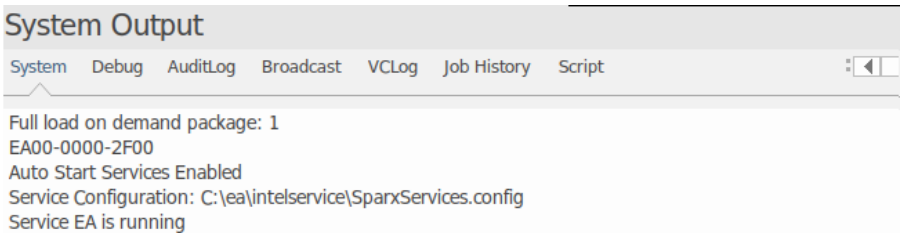
```
# network                       [optional,default=local] Restricts service to listening to locahost only (local), to a range of addresses (network) or any address (public)
# allow                         Allows a specific IP address or wildcard IP address to connect (if network is NOT local)
#                                       (There can be multiple allow directives present)
# autoupdate                    "true" - will detect updates to listed databases and reload them, "false" default, changes are not detected
# show                          [optional,default=false] shows the console window for the service
# logoutput                     [optional] The path of a log file which service can write to
# loglevel                      [optional] The levels of information logged. Combine with '|' character, e.g.: { information|warning|error }
# database                      [Required] The full path to a codeminer database which usually has the .cdb file extension
#                                       (There can be multiple database directives present)
#
# ------------------------------------------------------------------------------
# Attribute Values
# ------------------------------------------------------------------------------
#
# <string>   - text. (do not include quotes)
# <boolean>  - text, { true, false, ON, OFF }
# <path>     - fully specified file path to codeminer database
# <number>   - digits
# ------------------------------------------------------------------------------
#
{
        name=<string>,
        status=<boolean>,
        lazyload=<boolean>,
        port=<number>,
        allow=<string>,
        allow=<string>,
        network=<string>,
        autoupdate=<string>,
        show=<boolean>,
        logoutput=<string>,
        loglevel=<string>,
        database=<path>,
        database=<path>,
        database=<path>
},
{
        name=Project1,
        status=ON,
```

```
        lazyload=TRUE,
        allow=localhost,
        allow=127.0.0.1,
        port=9999,
        autoupdate=true,
        database=c:\Project1\Project1.cdb
}
```

## Options du ruban de service Intel Sparx

Lorsqu'un fichier de configuration de service existe, vous pouvez le modifier ou l'exécuter à l'aide d'un certain nombre d'options disponibles dans l'option de ruban « Exécuter > Outils > Services » dans le groupe d'options de menu Code Miner .

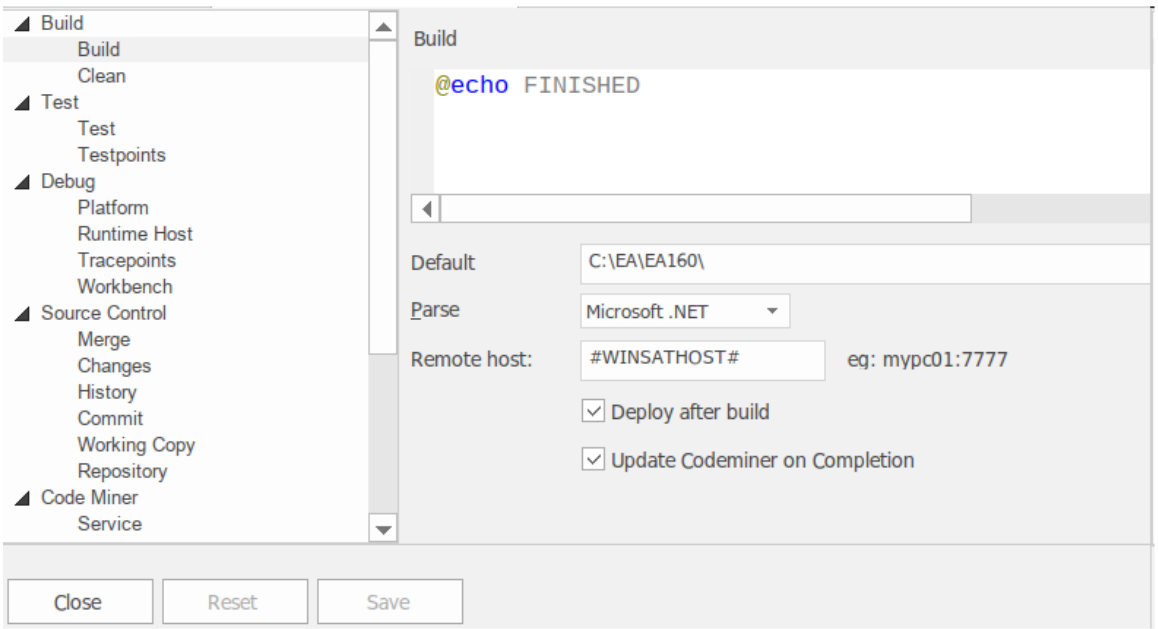
| Option | Description |
|---|---|
| Vue l'état de tous les services | (Au-dessus de toutes les catégories de service.) Cette option affiche une vue qui répertorie l'état de chaque service Enterprise Architect nommé dans le fichier de configuration actuel et son état. |
| Démarrer | Cette option lit le fichier de configuration de service actuel et démarre les services configurés pour exécuter , et arrête l'exécution des services qui ne sont pas configurés pour exécuter . Un service est configuré si :<br>1. Il est nommé dans le fichier de configuration.<br>2. Il a l'attribut status:ON.<br>  |
| Arrêtez tout | Cette option arrête tous les services en cours d'exécution. |
| | |

| | |
|---|---|
| Modifier le fichier de configuration | Cette option prompts le fichier de configuration de service à utiliser, puis ouvre ce fichier dans un éditeur de texte Enterprise Architect . Le système se souvient de l'emplacement du fichier.<br>`31 # <number>  - digits`<br>`32 # ----------`<br>`33 #`<br>`34 {`<br>`35  name=project1,`<br>`36  status=ON,`<br>`37  lazyload=true,`<br>`38  port=9910,`<br>`39  allow=localhost,`<br>`40  network=local,`<br>`41  autoupdate=true,`<br>`42  show=true,`<br>`43  logoutput=c:\My Documents\project1.txt,`<br>`44  loglevel=information\|warning\|error,`<br>`45  database=c:\My Documents\project1\project1.cdb`<br>`46 }`<br>`47` |
| Démarrer automatique avec EA | Cette option démarre automatiquement les services ayant l'attribut « status:ON » lorsque le modèle s'ouvre.<br>  <br>Les messages enregistrés dans la fenêtre de sortie système ici lorsque le modèle est ouvert indiquent que le service était déjà en cours d'exécution. |
| Arrêt automatique à la fermeture | Cette option arrête automatiquement l'exécution des services lorsque Enterprise Architect est fermé. |

# Mise à jour automatique du service Intel Sparx

Lorsque vous exécutez la commande Build pour un script Analyzer, un travail est ajouté à la file d'attente des travaux.

Si la case à cocher « Mettre à jour Codeminer à la fin » du script de construction est cochée dans l' Éditeur de Script Analyseur , une tâche supplémentaire est ajoutée au travail pour mettre à jour chacune des bases de données Codeminer répertoriées dans le script.




Les bibliothèques peuvent être vues dans la section Code Miner | Bibliothèques du script.


## Comment se déroule la tâche

La tâche de mise à jour Code Miner exécute le programme SSCodeMiner.exe avec deux arguments.

Le premier argument spécifie la base de données sur laquelle effectuer la construction incrémentielle et a la forme suivante :

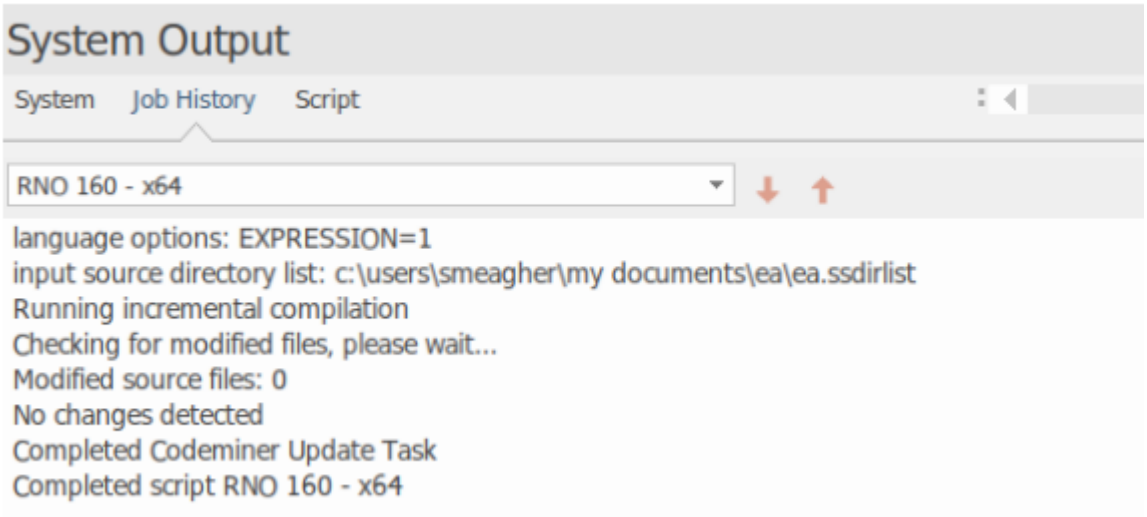
mise à jour = "c:\path\ea.cdb"

Le deuxième argument est facultatif et spécifie un fichier de grammaire de macro auxiliaire à utiliser lors de la compilation de la base de données ; il a cette forme :

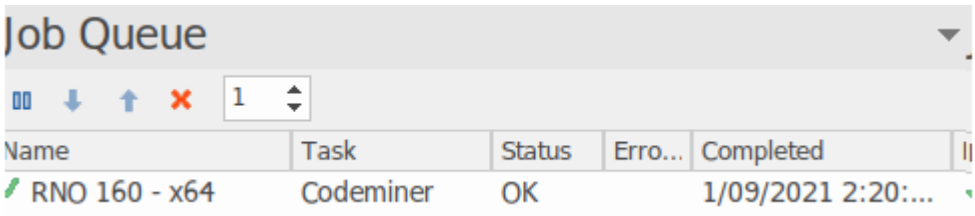
macros="c:\ea\ea160\config\CodeMiner\SparxProjectMacros.nbnf"

## Sortie du travail

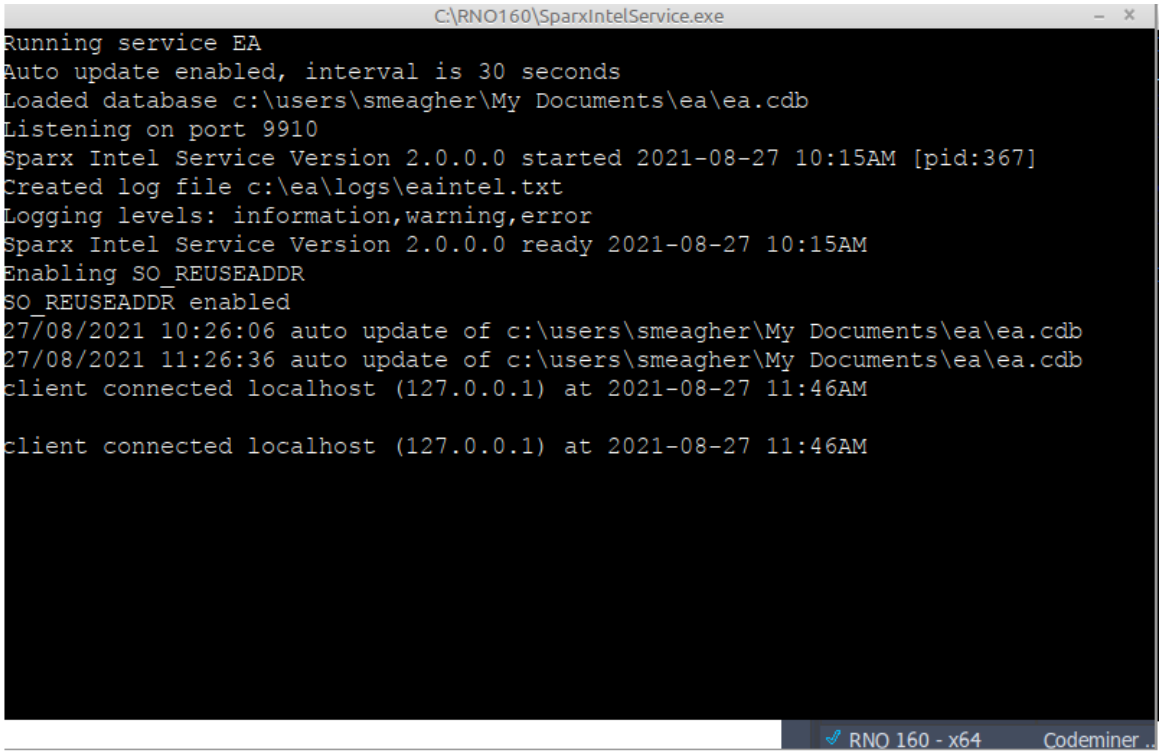
Lors de l'exécution de la tâche de mise à jour Code Miner , la sortie du processus de mise à jour SSCodeMiner.exe capturé est envoyée vers l'onglet « Historique des tâches » de la fenêtre Sortie système, sous la même forme que celle affichée lors de la mise à jour manuelle d'une base de données Code Miner dans Enterprise Architect . Dans cette illustration, nous pouvons voir que le script Analyzer RNO 160 -x64 s'est terminé avec succès.




La fenêtre de la file d'attente des tâches indique que la tâche est terminée. La dernière tâche à exécuter était la mise à jour Code Miner .




L'onglet « Historique des tâches » indique qu'aucun fichier de code source n'a été modifié. Si des modifications du code source sont détectées (c'est-à-dire que le service Code Miner a détecté une nouvelle version de ea.cdb et l'a automatiquement mise à jour), ces informations s'affichent :


C:\RNO160\SparxIntelService.exe
Running service EA
Auto update enabled, interval is 30 seconds
Loaded database c:\users\smeagher\My Documents\ea\ea.cdb
Listening on port 9910
Sparx Intel Service Version 2.0.0.0 started 2021-08-27 10:15AM [pid:367]
Created log file c:\ea\logs\eaintel.txt
Logging levels: information,warning,error
Sparx Intel Service Version 2.0.0.0 ready 2021-08-27 10:15AM
Enabling SO_REUSEADDR
SO_REUSEADDR enabled
27/08/2021 10:26:06 auto update of c:\users\smeagher\My Documents\ea\ea.cdb
27/08/2021 11:26:36 auto update of c:\users\smeagher\My Documents\ea\ea.cdb
client connected localhost (127.0.0.1) at 2021-08-27 11:46AM
client connected localhost (127.0.0.1) at 2021-08-27 11:46AM
RNO 160 - x64
Codeminer ..

# Configuration du service

## Programme de service

Le nom du programme de service est SparxintelService.exe.

## Fichier de configuration

Le service est configuré par le fichier SparxIntelService.config.

Le fichier doit être situé dans le même répertoire que le programme de service.

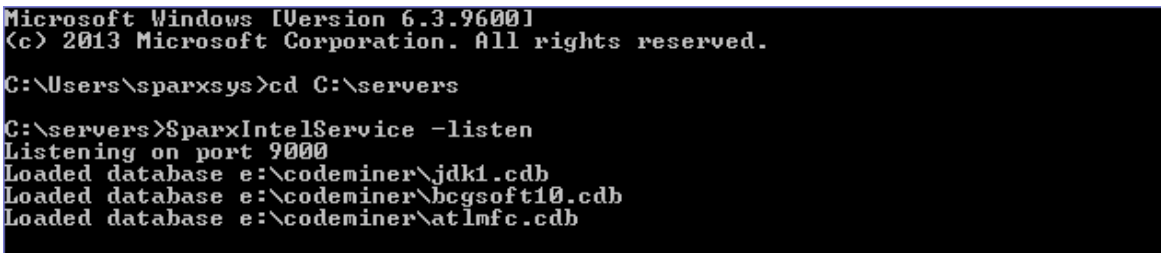
Le fichier contient un certain nombre de directives et répertorie également les bases de données Code Miner à servir.

Le fichier est lu une fois au démarrage du service.

| Directives | Description |
|---|---|
| port | Le numéro de port sur lequel le service écoutera. |
| permettre | Nomme un domaine ou une adresse IP à laquelle l'accès est autorisé : 198.* ou 127.0.0.1 |
| réseau | Les valeurs peuvent être « publiques », « réseau » ou « privées ».<br>• Utilisez « privé » lorsque les directives d'autorisation spécifient une ou plusieurs adresses IP uniques<br>• Utilisez « réseau » lorsque les directives d'autorisation spécifient un domaine générique : 198*<br>• Utilisez « public » pour autoriser tous les clients |
| base de données | Nomme le chemin d'accès physique complet d'une base de données Code Miner sur le serveur. |

## Exécution du programme de manière autonome

Depuis une console normale, entrez la commande : SparxIntelService -listen

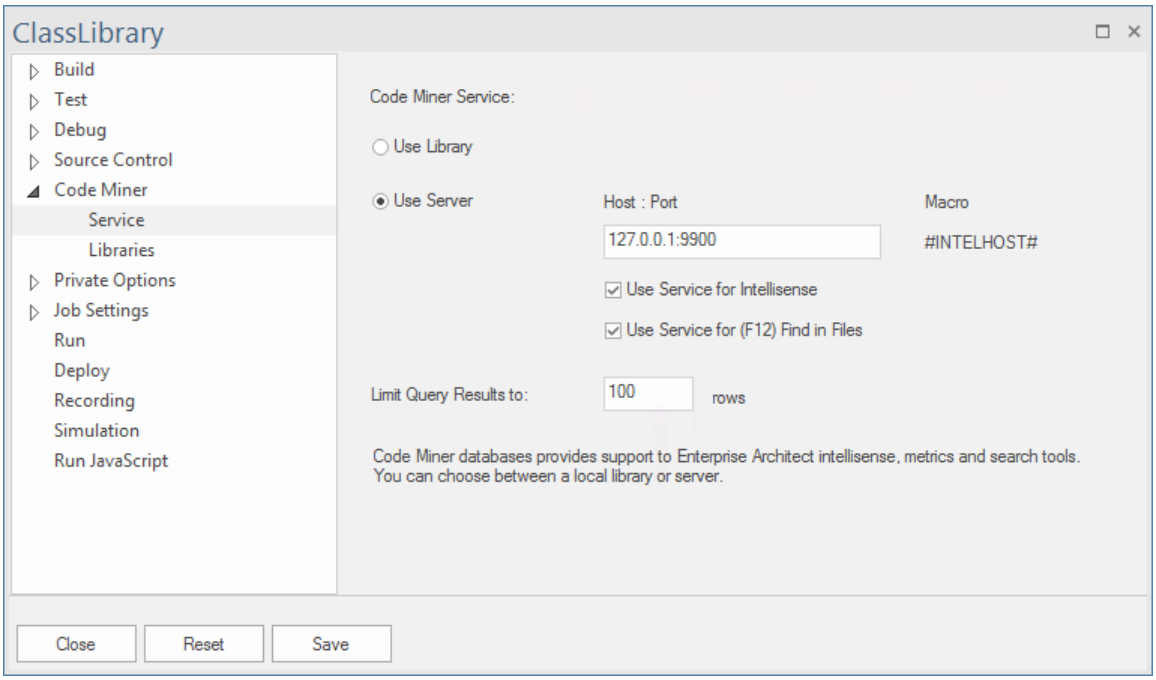



## Installation en tant que service Windows

Depuis une console d'administration, entrez la commande : SparxIntelService -install

# Configuration du client - Configuration Enterprise Architect pour utiliser un service Code Miner

Enterprise Architect utilise des composants appelés Scripts d'Analyseur pour la configuration de nombreux systèmes support . C'est ici que l'emplacement du serveur est spécifié. Cette image montre la page « Code Miner Service » d'un script.




## Accéder

| | |
|---|---|
| Ruban | Développer > Source Code > Analyseur d'Exécution > Edit Scripts d'Analyseur > Double-cliquez sur un Script > Code Miner > Service |

## Champs de configuration

| | |
|---|---|
| Utiliser le serveur | Sélectionnez ce bouton radio pour configurer le serveur Code Miner à utiliser. |
| Hôte : Port | Type le numéro du port via lequel le service fonctionnera. |
| Utiliser le service pour Intelli-sense | Cochez la case pour utiliser le service Intel pour la saisie semi-automatique des champs Intelli-sense. |
| Utiliser le service pour [F12] Rechercher dans les fichiers | Cochez la case si vous souhaitez utiliser le service au lieu de la fenêtre Rechercher dans les fichiers pour exécuter des requêtes de recherche lorsque vous appuyez sur F12. |

| | |
|---|---|
| Limiter les résultats Query aux lignes | Type le nombre de lignes de résultats de requête à afficher par page. |
| Sauvegarder | Cliquez sur ce bouton pour enregistrer les détails de configuration que vous avez saisis. |