



**ENTERPRISE ARCHITECT**

Série de Guides d'Utilisateur

# Scriptant Hybride

Author: Sparx Systems

Date: 7/11/2024

Version: 17.0

CRÉÉ AVEC  **ENTERPRISE  
ARCHITECT**

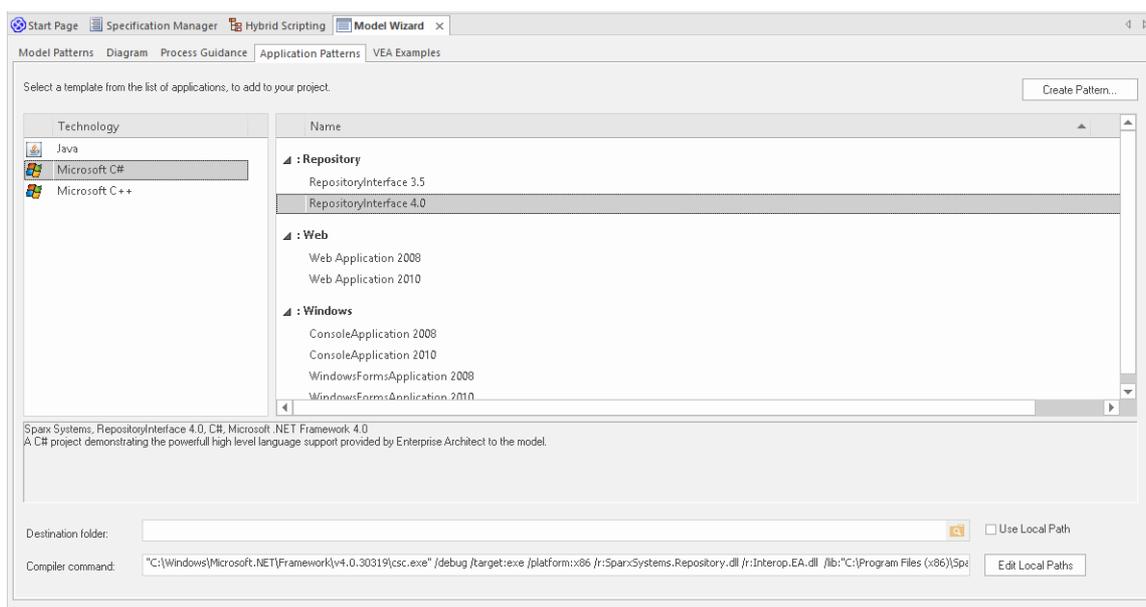
## Table des Matières

|                         |   |
|-------------------------|---|
| Scriptant Hybride ..... | 3 |
| Exemple C# .....        | 5 |
| Exemple Java .....      | 7 |

# Scriptant Hybride

Scriptant hybride étend les capacités de l'environnement de script standard aux langages de haut niveau tels que Java et C#. Scriptant hybride offre un avantage de vitesse par rapport aux scripts conventionnels et permet également aux auteurs de scripts de tirer parti des compétences existantes dans les langages de programmation courants.

## Accéder

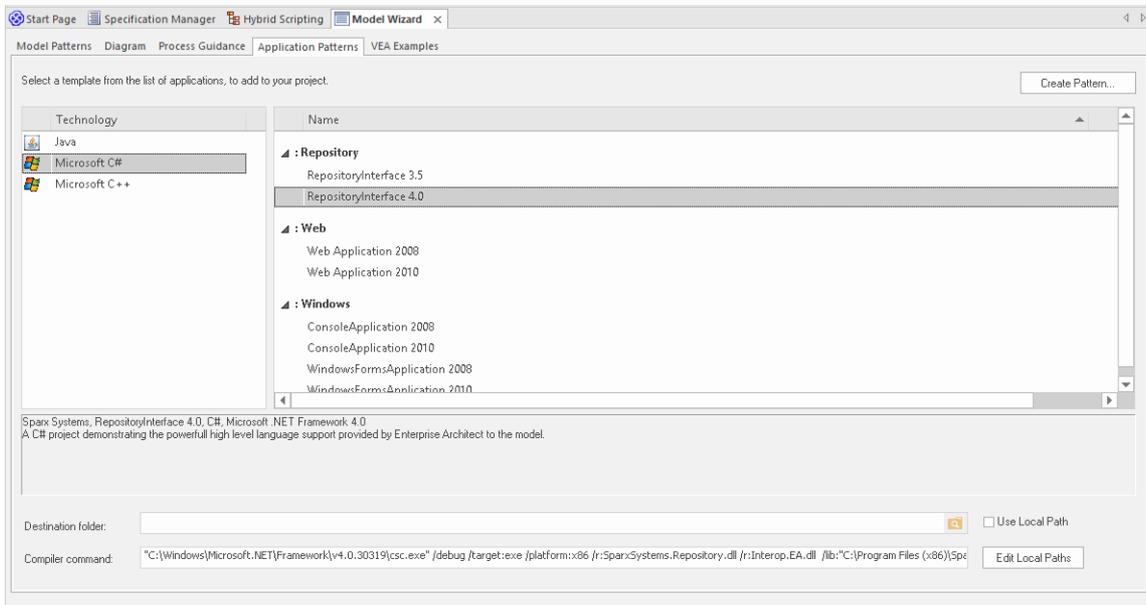


Ruban

Développer &gt; Code source &gt; Créer à partir de Motif &gt; Motifs d'application

## Fonctionnalités

- Vitesse d'exécution supérieure
- Interopérabilité améliorée
- support complète Analyseur d'Exécution Visuelle



## Exemple C#

Cet exemple de programme montre à quel point il est facile de naviguer, d'interroger et de générer des rapports sur le modèle actuel à l'aide de n'importe quel langage Microsoft .NET . Cet exemple est écrit en C# .

Lors de l'exécution, il imprimera les noms de chaque Paquetage dans le modèle que vous utilisez actuellement.

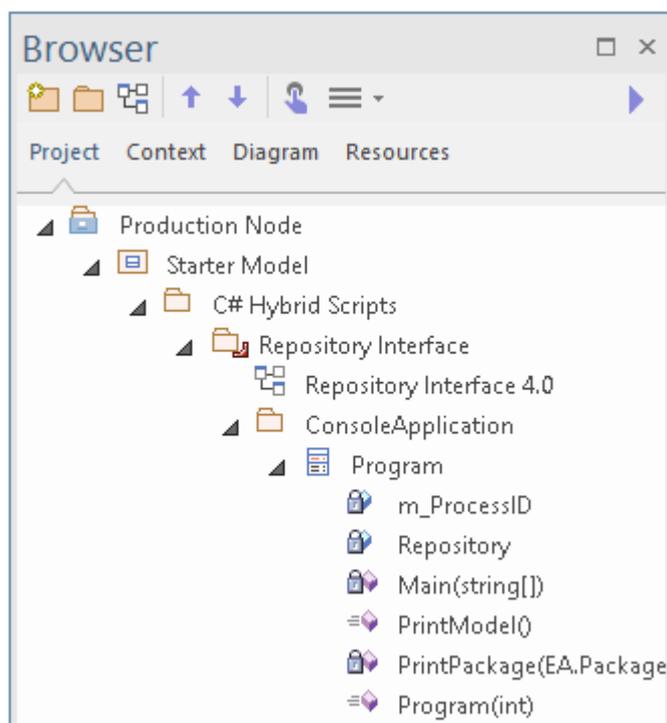
### Créer le projet

Dans la fenêtre Navigateur, sélectionnez le Paquetage dans lequel créer le gabarit, puis utilisez l'option du ruban 'Développer > Code source > Créer à partir Motif' pour afficher la fenêtre Motifs ; cliquez sur l'option 'Application Motifs'.

Depuis la page « Motifs d'application », sélectionnez le gabarit *Microsoft C# > RepositoryInterface*. (Vous pouvez choisir entre les versions 3.5 ou 4.0 du framework.) Spécifiez le dossier de destination sur le système de fichiers où le gabarit du projet sera créé, puis cliquez sur le bouton OK.

### Ouvrir le projet

Une structure Paquetage similaire à celle-ci sera créée pour vous.



Développez la structure jusqu'à localiser le diagramme *Référentiel Interface nn* et ouvrez-le.

**Overview:**  
This sample program demonstrates how easy it is to navigate, query and report on the current model using any Microsoft .NET language. This example is written in C#. When run, it will print the names of every Package in the model you are currently using.

**Framework:**  
The build uses the C# compiler from the Microsoft .NET framework.

**Version:**  
4.0

**Note:**  
The links on the right operate on the active Analyzer Script. To use these links make sure you have selected the 'Repository Interface 4.0' script. You can use this Analyzer Scripts link to do this.

 Analyzer Scripts

Build the project

⚙️

Build

Run your program

▶️

Run

Debug the program

▶️

\*DebugRun

**Program**

```

- m_ProcessID: int = 0
- Repository: EA.Repository = null

+ Main(string[]): void
+ PrintModel(): bool
- PrintPackage(EA.Package): void
+ Program(int)

```

## Construire le script

Les commandes de ce diagramme s'appliqueront à la configuration de build active. Avant de les exécuter, double-cliquez sur le lien *Scripts d'Analyseur* et cochez la case à côté de la configuration de build ' Interface Référentiel '.

## Exécuter le script

Double-cliquez sur le lien *Exécuter* pour ouvrir la console. La console s'arrêtera une fois l'exécution terminée pour que vous puissiez lire la sortie du programme ; cette sortie sera également envoyée à l'onglet « Script » de la fenêtre Sortie système. Vous pouvez modifier cela en changeant le code.

## Débugger le Script

Sélectionnez la classe « Programme » dans la fenêtre Navigateur et appuyez sur Ctrl+E pour ouvrir le code source.

Placez un Point d'Arrêt dans une des fonctions puis double-cliquez sur le lien *DebugRun* . Lorsque le Point d'Arrêt est rencontré, la ligne de code sera mise en surbrillance dans l'éditeur, comme indiqué :

- ConsoleApplication
- Program
  - Repository
  - m\_ProcessID
  - Main()
  - PrintModel()
  - PrintPackage()
  - Program()
  - Trace()

```

21         Console.WriteLine(msg);
22     }
23     public Program(int pid)
24     {
25         m_ProcessID = pid;
26         Repository = SparxSystems.Services.GetRepository(m_ProcessID);
27         Trace("Running C# Console Application AppPattern .NET 3.5");
28     }
29     private void PrintPackage(EA.Package package)
30     {
31         Trace(package.Name);
32         EA.Collection packages = package.Packages;
33         for (short ip = 0; ip < packages.Count; ip++)
34         {
35             EA.Package child = (EA.Package)packages.GetAt(ip);
36             PrintPackage(child);
37         }
38     }

```

## Exemple Java

Cet exemple de programme montre à quel point il est facile de naviguer, d'interroger et de générer des rapports sur le modèle actuel à l'aide d'un langage de haut niveau tel que Java.

Lorsqu'il exécute, il imprimera les noms de chaque Paquetage dans le modèle actuellement chargé.

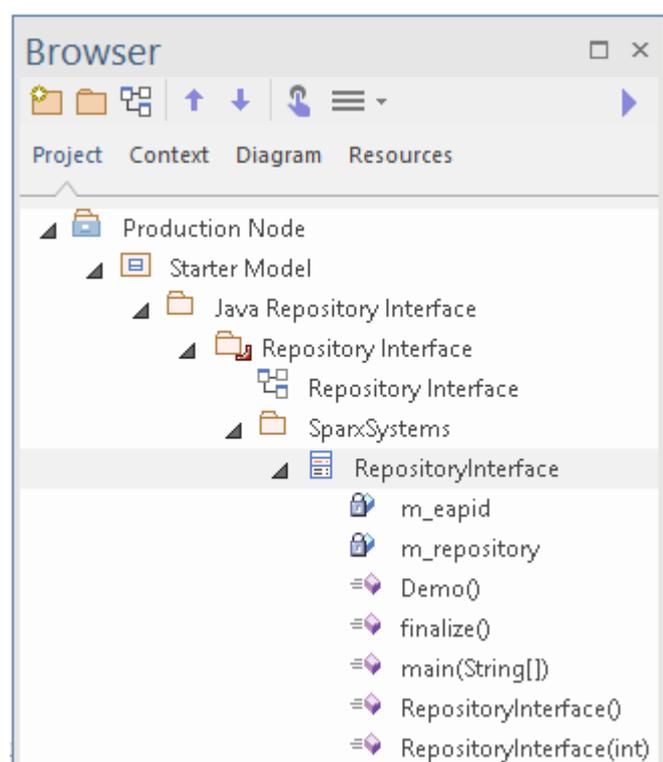
### Créer le projet

Dans la fenêtre Navigateur, sélectionnez le Paquetage dans lequel créer le gabarit, puis utilisez l'option du ruban 'Développer > Code source > Créer à partir Motif' pour afficher la fenêtre Motifs; cliquez sur l'option 'Application Motifs'.

Depuis la page 'Motifs d'application', sélectionnez le gabarit *Java > RepositoryInterface*. Spécifiez le dossier de destination sur le système de fichiers dans lequel le gabarit du projet sera créé, puis cliquez sur le bouton OK.

### Ouvrir le projet

Une structure Paquetage similaire à celle-ci sera créée pour vous.



Développez la structure jusqu'à localiser le diagramme 'Référentiel Interface' et ouvrez-le.

**Overview:**

This sample program demonstrates how easy it is to navigate, query and report on the current model using a high level language such as Java. When run, it will print the names of every Package in the currently loaded model.

**Framework:**

The build uses the compiler from the Java JDK 1.7 x86 framework.

**Version:**

1.7

**Note:**

In order to use the Build, Run and Debug links, you must first locate the 'Repository Interface' Analyzer Script generated by the wizard, and make it the active script for the model. You can use the 'Analyzer Scripts' link to do this.

 Analyzer Scripts

Build the project

→
 Build

Run your program

→
 Run

Debug the program

→
 \*DebugRun

**RepositoryInterface**

```

- m_eapid: int = 0
- m_repository: org.sparx.Repository = null
+ Demo(): void
+ finalize(): void
~ main(String[]): void
+ RepositoryInterface()
+ RepositoryInterface(int)
                    
```

## Construire le script

Les commandes du diagramme s'appliqueront sur la configuration de build active. Avant de les exécuter, double-cliquez sur le lien *Scripts d'Analyseur* et cochez la case à côté de la configuration de build ' Interface Référentiel '.

## Exécuter le script

Double-cliquez sur le lien *Exécuter* ; une console s'ouvre. La console s'arrête une fois l'exécution terminée pour que vous puissiez lire le résultat. Le résultat du programme sera également affiché dans l'onglet « Script » de la fenêtre de sortie système. Vous pouvez modifier cela en modifiant le code.

## Débugger le scénario

Sélectionnez la classe « Programme » dans la fenêtre Navigateur et appuyez sur Ctrl+E pour ouvrir le code source.

Placez un point d'arrêt dans l'une des fonctions, puis double-cliquez sur le lien *DebugRun* . Lorsque le point d'arrêt est rencontré, la ligne de code est mise en surbrillance dans l'éditeur, comme indiqué.

▲ SparxSystems

- ▲ RepositoryInterface
  - ◆ m\_eapid
  - ◆ m\_repository
  - ◆ Demo()
  - ◆ PrintPackage(org.sparx.
  - ◆ RepositoryInterface()
  - ◆ RepositoryInterface(int)
  - ◆ Trace(String)
  - ◆ finalize()
  - ◆ main(String)

```

35
36 public void Trace( String msg )
37 {
38     // You can change the System Output Tab that receives the trace messages.
39     m_repository.WriteOutput( "Script", msg, 0);
40     System.out.println( msg);
41 }
42
43 public void PrintPackage( org.sparx.Package pkg)
44 {
45     Trace( pkg.GetName());
46     Collection<org.sparx.Package> packages = pkg.GetPackages();
47     for(short i = 0; i < packages.GetCount(); i++)
48     {
49         PrintPackage(packages.GetAt(i));
50     }
51 }
                    
```

