



ENTERPRISE ARCHITECT

Série de Guides d'Utilisateur

Profilage

Author: Sparx Systems

Date: 23/11/2023

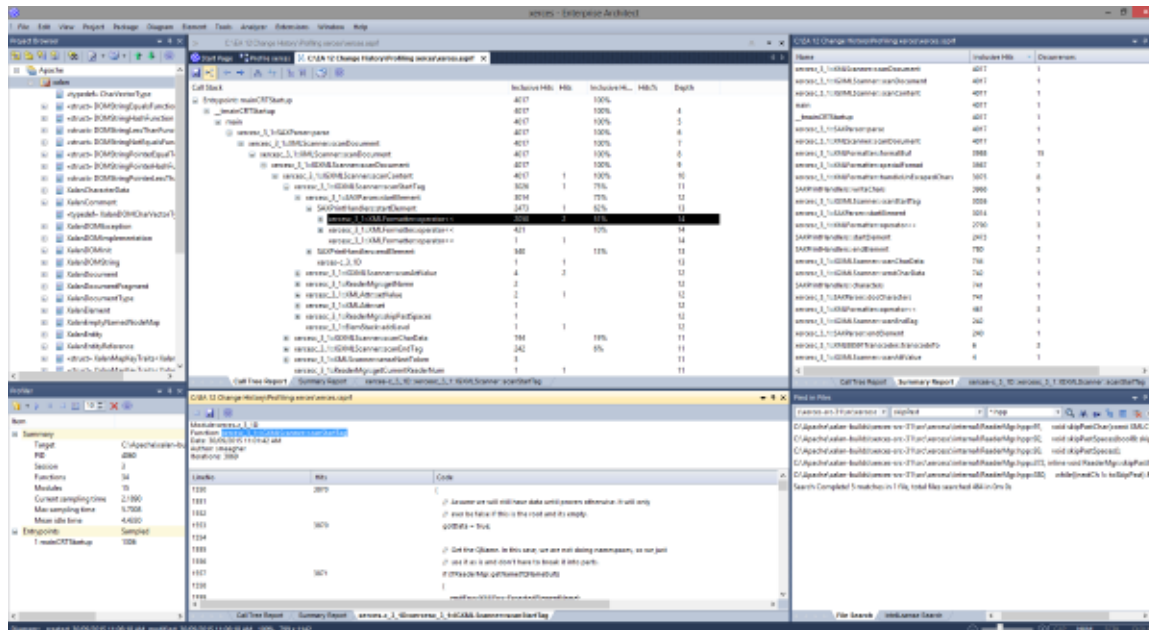
Version: 16.1

CRÉÉ AVEC  **ENTERPRISE
ARCHITECT**

Table des Matières

Profilage	3
Exigences du système	10
Démarrage	11
Graphique d'Appel	13
Profil de Pile	16
Profil de mémoire	18
Fuites de Mémoire	20
Options de Réglage	23
Démarrer et Arrêter le Profileur	25
Rapports de Ligne de Fonction	27
Générer, Enregistrer et Charger des Rapports de Profil	30
Enregistrer Rapport dans Bibliothèque d'Équipe	35

Profilage

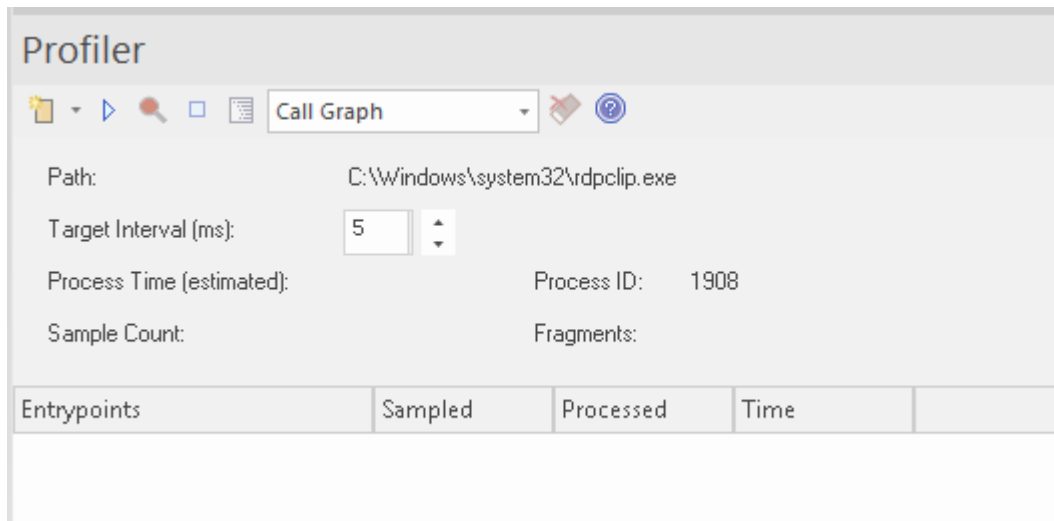


Au cours de la durée de vie des applications logicielles, il n'est pas rare d'enquêter sur des tâches d'application dont l'exécution s'avère plus lente que prévu. Vous pourriez aussi simplement vouloir savoir ce qui se passe lorsque vous « appuyez sur ce bouton » ! Vous pouvez résoudre ce problème assez rapidement dans Enterprise Architect en utilisant son profileur. Les résultats peuvent généralement être produits en quelques secondes et vous pourrez rapidement voir les actions qui consomment l'application et les fonctions impliquées. Dans l'Analyseur d'Exécution, la fonctionnalité emploie deux stratégies distinctes ; *Échantillonnage de processus* et *accrochage de processus*. Dans l'un, des échantillons sont prélevés à intervalles réguliers pour identifier motifs gourmands en CPU, tandis que dans l'autre, le processus est lié à l'enregistrement des demandes faites en mémoire. Les données sont analysées pour produire un Graphique d'Appel pondéré. Les comportements sont généralement identifiables comme des nœuds racines (points d'entrée) dans le graphique ou des branches à proximité de ces points. Tous les rapports peuvent être consultés sur demande. Ils peuvent être enregistrés dans un fichier au sein du modèle, à la fois en tant qu'éléments d'artefact et en tant que publications Bibliothèque d'Équipe .

Accéder

Ruban	Exécuter > Outils > Profileur
Autre	Barre d'outils Analyseur d'Exécution : Analyseur Windows Profileur

Échantillonnage d'appels



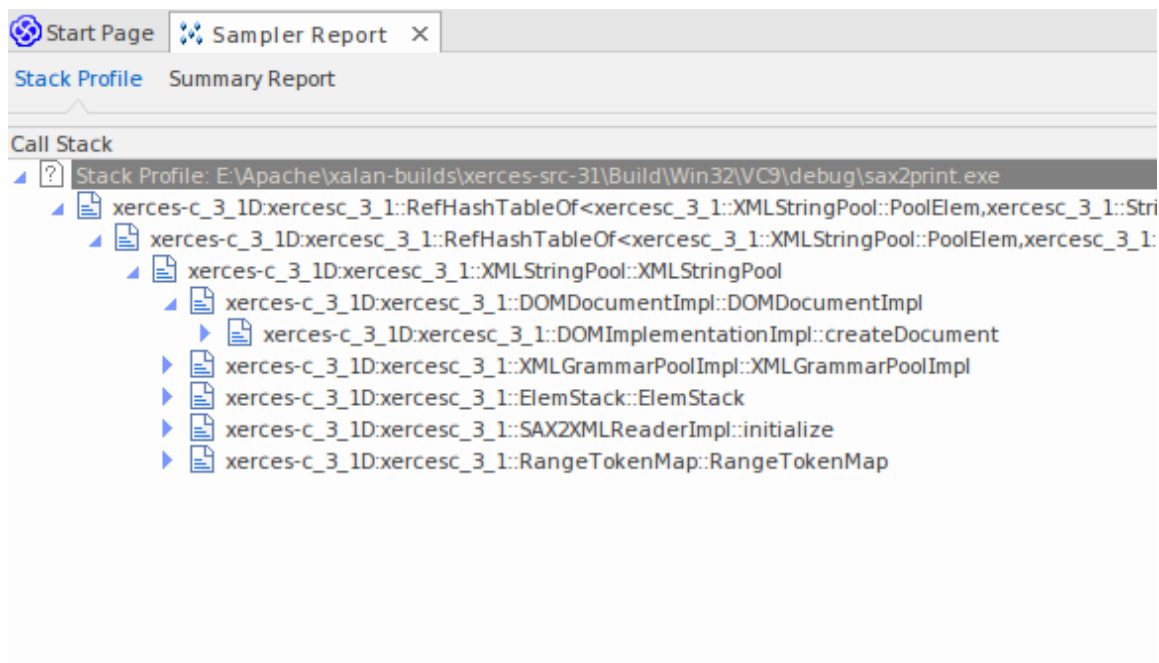
Le Profiler est contrôlé à l'aide des boutons de sa barre d'outils. Ici, vous pouvez attacher le profileur à un processus existant (ou JVM) ou lancer l'application pour le script Analyzer actif. La fenêtre Profiler affiche les détails du processus cible tel qu'il est profilé. Ces détails fournissent des commentaires, vous permettant de voir le nombre d'échantillons prélevés. Vous disposez également d'options pour suspendre et reprendre la capture, effacer les données capturées et générer des rapports. Vous pouvez accéder à la fonctionnalité de reporting en suspendant la capture - la fonctionnalité de reporting est désactivée pendant que la capture des données est en cours.

Graphique d'Appel pondéré

Call Stack	Inclusive Hits	Hits
xercesc_3_1::SAX2XMLReaderImpl::parse	16051	
xercesc_3_1::XMLScanner::scanDocument	16051	
xercesc_3_1::IGXMLScanner::scanDocument	16051	
xercesc_3_1::IGXMLScanner::scanContent	16051	
xercesc_3_1::IGXMLScanner::scanStartTagNS	16051	
xercesc_3_1::IGXMLScanner::resolveSchemaGrammar	16051	
xercesc_3_1::SchemaValidator::preContentValidation	16049	
xercesc_3_1::ComplexTypeInfo::checkUniqueParticleAttribution	16049	
xercesc_3_1::ComplexTypeInfo::makeContentModel	16049	
xercesc_3_1::DFAContentModel::DFAContentModel	16047	
xercesc_3_1::DFAContentModel::buildDFA	15998	515
xercesc_3_1::CMStateSet::operator =	8174	8093
memcpy	32	32
xercesc_3_1::CMStateSet::allocateChunk	27	1
_security_check_cookie	21	21
TrailUpVec	1	1
xercesc_3_1::CMStateSet::~CMStateSet	3573	4
xercesc_3_1::XMemory::operator delete	841	2
xerces-c_3_1D	4416	2
xercesc_3_1::CMStateSet::getBit	1036	1036
xercesc_3_1::DFAContentModel::buildSyntaxTree	528	3
xercesc_3_1::CMStateSet::CMStateSet	373	3
xercesc_3_1::CMStateSet::getBitCountInRange	285	285
xercesc_3_1::XMemory::operator new	211	2
xercesc_3_1::CMStateSet::zeroBits	154	
xercesc_3_1::CMStateSetEnumerator::nextElement	153	136
xercesc_3_1::RefHashTableOf<xercesc_3_1::XMLInteger,>	59	2
xercesc_3_1::RefHashTableOf<xercesc_3_1::XMLInteger,>	28	2
xercesc_3_1::RefHashTableOf<xercesc_3_1::XMLInteger,>	25	
xercesc_3_1::DFAContentModel::makeDefStateList	25	2

Ce rapport détaillé montre l'ensemble unique de piles d'appels/comportements sous forme de Graphique d'Appel pondéré. Le poids de chaque branche est représenté par un nombre de hits, qui correspond au total des hits de cette branche plus toutes les branches à partir de ce point. En suivant la piste des hits, vous pouvez identifier rapidement les zones de code qui ont le plus occupé le programme pendant la période de capture.

Profil de Pile



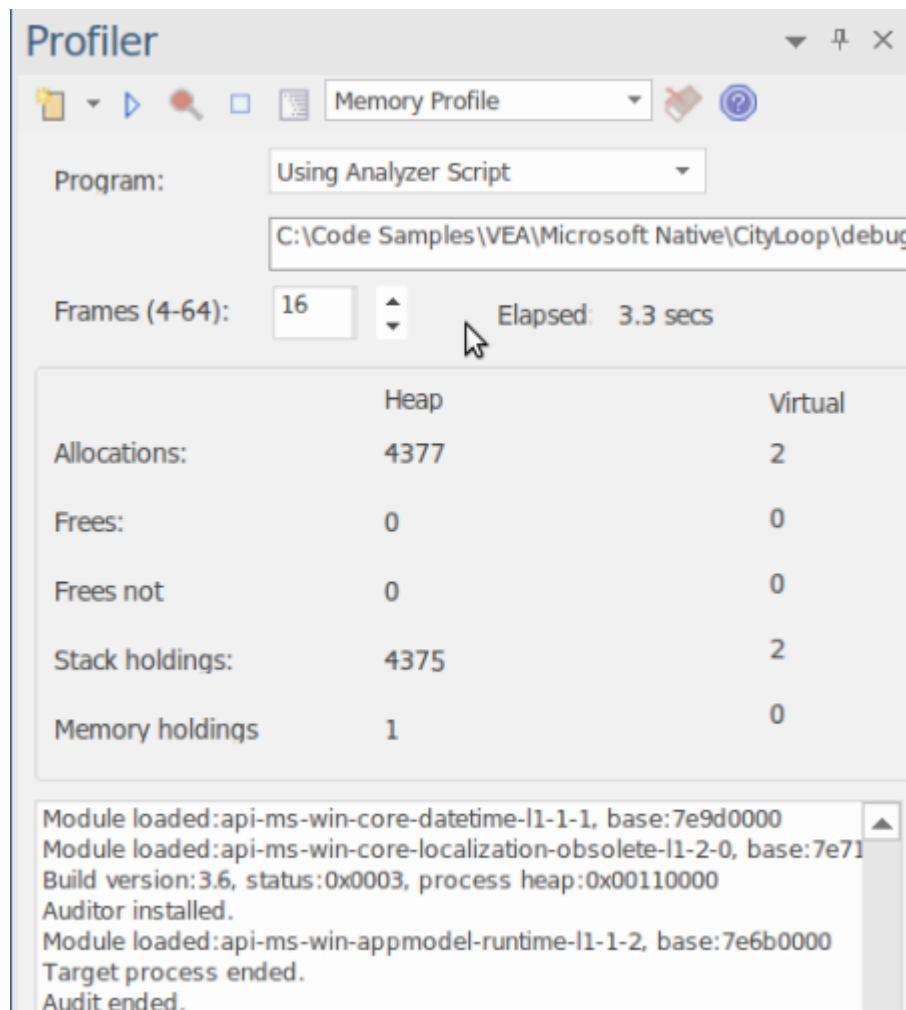
Les profils de pile sont utilisés pour découvrir les différentes manières (piles) et le nombre de façons dont une fonction particulière est invoquée pendant l'exécution du programme. Contrairement aux autres modes du profileur, ce profil est activé via l'utilisation d'un point de profil, qui est un type spécial de marqueur de point d'arrêt. Le marqueur est défini dans le code source comme n'importe quel autre point d'arrêt. Lorsque le programme rencontre le point d'arrêt, la pile est capturée. Lorsque vous produisez ultérieurement le rapport, les piles sont analysées et un graphique d'appel pondéré est produit. Le graphique montre les piles uniques impliquées dans cette fonction pendant l'exécution du profileur. La colonne « Hit Count » indique le nombre de fois où la même pile s'est produite.

```

106
107 template <class TVal, class THasher>
108 void RefHashTableOf<TVal, THasher>::initialize(const XMLSize_t modulus)
109 {
110     if (modulus == 0)
111         ThrowXMLwithMemMgr(IllegalArgumentException, XMLExcepts::HshTbl_ZeroMo
112
113     // Allocate the bucket list and zero them
114     fBucketList = (RefHashTableBucketElem<TVal>**) fMemoryManager->allocate
115     (
116         fHashModulus * sizeof(RefHashTableBucketElem<TVal>*)
117     );
118     for (XMLSize_t index = 0; index < fHashModulus; index++)
119         fBucketList[index] = 0;
120 }
121

```

Profils de Mémoire



Le profil de mémoire suit les allocations, ignorant le moment où la mémoire est libérée. Il utilise ces informations pour évaluer les demandes de mémoire du code en cours d'exécution, non pas en termes de quantité de mémoire mais de fréquence des demandes. Le chiffre *des allocations* correspond au nombre total d'allocations de mémoire demandées. Le *Stack Holdings* est le nombre de traces de pile prises à ces moments-là, et le chiffre *Heap Holding* est la quantité totale de mémoire obtenue par ces appels. Note que le profilage peut être activé et désactivé à la demande. Il n'est pas non plus nécessaire de reconstruire votre programme pour le faire fonctionner car aucun lien n'est impliqué.

Graphique de Mémoire

profile		
Start Page *leaks *profile x		
Call Stack	Instances	Bytes
E:\Apache\alan-builds\xerces-src-31\Build\Win32\VC9\debug\DOMPrint.exe C:\test\materials\portrait.xml	0	0
ntdll:RtlAllocateHeap	7,068	4,830,947
ntdll:RtlpNtSetValueKey	7,068	4,830,947
ntdll:RtlDestroyMemoryBlockLookaside	7,068	4,830,947
ntdll:RtlAllocateHeap	7,068	4,830,947
ntdll	7,068	4,830,947
msvcr90d:malloc_base	4,813	3,985,885
msvcr90d:malloc_dbg	4,813	3,985,885
msvcr90d:malloc_dbg	4,813	3,985,885
msvcr90d:malloc_dbg	4,813	3,985,885
msvcr90d:malloc	4,812	3,985,734
msvcr90d:operator new	4,812	3,985,734
xerces-c_3_1d:xercesc_3_1::MemoryManagerImpl::allocate	4,807	3,985,526
xerces-c_3_1d:xercesc_3_1::RangeToken::expand	803	396,984
xerces-c_3_1d:xercesc_3_1::ValueHashTableOf<bool,xercesc_3_1::StringHasher>::put	791	37,968
xerces-c_3_1d:xercesc_3_1::XMemory::operator new	753	239,048
xerces-c_3_1d:xercesc_3_1::XMemory::operator new	333	21,652
xerces-c_3_1d:xercesc_3_1::RangeToken::doCreateMap	291	19,788
xerces-c_3_1d:xercesc_3_1::RangeToken::addRange	287	28,700
xerces-c_3_1d:xercesc_3_1::XMLString::replicate	218	12,914
xerces-c_3_1d:xercesc_3_1::RefHashTableOf<xercesc_3_1::RangeTokenElemMap,xercesc_3_1	146	7,008
xerces-c_3_1d:xercesc_3_1::RefHashTableOf<xercesc_3_1::CPMapEntry,xercesc_3_1::StringHi	144	6,912
xerces-c_3_1d:xercesc_3_1::Win32TransService::Win32TransService	112	6,612
xerces-c_3_1d:xercesc_3_1::Win32TransService::Win32TransService	106	6,258

Cet exemple concerne un rapport produit à partir du profilage d'un programme de démonstration dans le projet Xerces d'Apache. Le programme parcourt le Document Object Modèle (DOM) pour un fichier XML fourni.

Rapport Sommaire de Fonction

Name	Inclusive Hits
profiler/Example.Run	156
profiler/Example.main	156
java/io/FileOutputStream.write	154
java/io/PrintStream.println	154
profiler/Example.Print	154
profiler/Example.MakeItalianCars	2
profiler/Example.NewCar	2

Ce rapport récapitulatif répertorie les fonctions et uniquement les fonctions exécutées pendant la période d'échantillonnage. Les fonctions sont répertoriées par nombre total d'appels, avec une fonction qui se présente deux fois dans des piles d'appels distinctes apparaissant avant une fonction qui n'apparaît qu'une seule fois.

Rapport de Ligne de Fonction

LineNo	Hits	Code
54	1	for(int n = 0; n < 10000; n++)
55		{
56	1408	m_Cars = new Collection<Car>();
57	1408	if((n % 3)>0)
58		{
59	938	for(int i = 0; i < 1000; i++)
60		{
61	938000	MakeItalianCars();
62		}

Ce rapport détaillé montre le code source d'une fonction ligne par ligne, affichant à côté le nombre total de fois où chacune a été exécutée. Nous avons découvert du code à l'aide de ce rapport, qui exposait des instructions case dans du code qui ne semblaient jamais être exécutées.

Support

Le Profiler est pris en charge pour les programmes écrits en C, C++, Visual Basic, Java et les langages Microsoft .NET . Le profilage de la mémoire est actuellement disponible pour les programmes natifs C et C++.

Notes

- Le profileur est disponible dans Enterprise Architect Professional Edition et versions ultérieures.
- Le Profiler peut également être utilisé sous WINE (Linux et Mac) pour profiler les applications Windows standards déployées dans un environnement WINE .

Exigences du système

À l'aide du Profiler, vous pouvez analyser les applications conçues pour ces plates-formes :

- Microsoft TM Natif (C++, C, Visual basic)
- Microsoft .NET (prenant en charge un mélange de code managé et non managé)
- Java

Applications natives Microsoft

Pour les applications C, C++ ou Visual Basic, le Profiler nécessite que les applications soient compilées avec le compilateur Microsoft TM Native et que pour chaque application ou module d'intérêt, un fichier PDB soit disponible. Le profileur peut échantillonner les configurations de débogage et de publication d'une application, à condition que le fichier PDB de chaque exécutable existe et soit à jour.

Applications Microsoft .NET

Pour les applications Microsoft .NET , le Profiler nécessite que le framework Microsoft .NET approprié soit installé et que pour chaque application ou module à analyser, un fichier PDB soit disponible.

Java

Pour Java, le profileur nécessite que le JDK approprié d'Oracle soit installé.

Les classes d'intérêt doivent également avoir été compilées avec des informations de débogage. Par exemple : "java -g *.java"

- Une nouvelle instance de VM d'application est lancée à partir Enterprise Architect - aucune autre action n'est requise
- La machine virtuelle d'application existante est attachée à partir d' Enterprise Architect : la Virtual Machine Java cible doit avoir été lancée avec l'agent de profilage Enterprise Architect .

Voici des exemples de lignes de commande pour créer une VM Java avec un agent JVMTI spécifique :

1. `java.exe -cp "%classpath%;\" -agentpath:"C:\Program Files (x86)\ Sparx Systems \EA\vea\x86\ssamplerlib32" monapplication`
2. `java.exe -cp "%classpath%;\" -agentpath:"C:\Program Files (x86)\ Sparx Systems \EA\vea\x64\ssamplerlib64" monapplication`

(Reportez-vous à la documentation du JDK pour plus de détails sur l'option de démarrage de la VM -agentpath.)

Démarrage

Le Profiler peut être utilisé pour enquêter sur les problèmes de performances, en vous proposant quatre outils distincts parmi lesquels choisir, à savoir :

- Graphique d'Appel
- Profil de Pile
- Profil de mémoire
- Fuites de Mémoire

Vous sélectionnez ces outils dans la barre d'outils du profileur.








Accéder

Ruban	Exécuter > Outils > Profileur
-------	-------------------------------

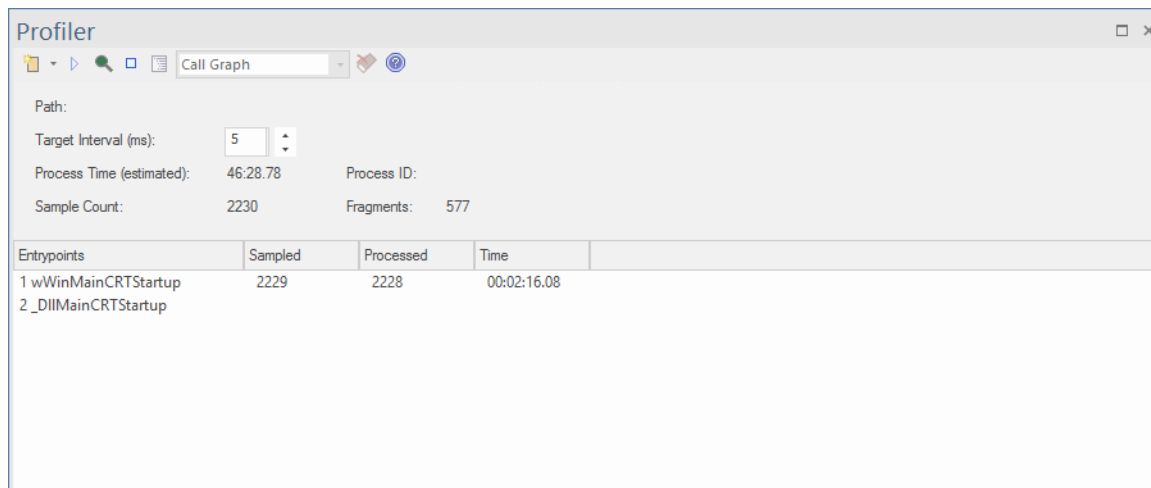
Outils

Outil	Description
Graphique d'Appel	Analyse les performances en prélevant des échantillons lors d'une activité dans un programme. Chaque échantillon représente une pile. Les échantillons sont prélevés à des intervalles contrôlés à l'aide de la barre d'outils. Dans ce scénario, les mauvaises performances sont évaluées en fonction des motifs de comportement qui se répètent le plus au cours de la période d'échantillonnage. Ce chiffre sert à pondérer le Graphique d'Appel produit.
Profil de mémoire	Analyse les performances en accrochant les allocations de mémoire effectuées par un programme. Dans ce scénario, les mauvaises performances sont évaluées en fonction des activités qui demandent le plus de mémoire. Ce chiffre sert à pondérer le Graphique d'Appel produit.
Profil de Pile	<p>Le Stack Profiler vous permet de définir un marqueur dans votre code source afin que chaque fois que l'exécution atteint ce marqueur, une trace complète de la pile soit capturée. Au fur et à mesure que l'application continue de s'exécuter et que la position marquée est accessible à partir de divers endroits dans l'exécutable en cours d'exécution, une image très détaillée et utile est créée montrant les points chauds et les scénarios d'utilisation pour un point particulier du code.</p> <p>Le rapport Profil de Pile, comme le rapport Profil de mémoire, est affiché dans l'ordre « pile inversée ». Cela signifie que la racine du rapport est toujours un nœud unique (dans ce cas, le marqueur) et que l'arborescence se déploie ensuite pour montrer tous les différents endroits à partir desquels la position marquée a été accédée.</p>
Fuites de Mémoire	Analyse les fuites de mémoire en accrochant les opérations de mémoire effectuées par un programme. Ce qui est produit est un Graphique d'Appel présentant les Call Stacks qui alloué de la mémoire pour lesquelles une opération libre n'a pas été détectée.

Boutons de la barre d'outils

Bouton	Action
	Affiche un menu d'options pour gérer votre session de profilage.
	Lance l'application configurée à profiler. Par défaut, il s'agit de l'application configurée dans le script Analyzer actif.
	Indique l'état de l'échantillonneur. Lorsqu'il est vert, l'échantillonnage est activé ; lorsqu'il est rouge, l'échantillonnage est désactivé.
	Arrête le processus du profileur ; si des échantillons ont été collectés, les boutons Rapport et Supprimer les données sont actifs.
	Génère un rapport à partir de la collecte de données actuelle.
	Affiche l'outil de profilage utilisé, qui détermine les champs affichés dans la fenêtre Profiler. Cliquez sur la flèche déroulante et sélectionnez un autre outil, qui modifie les champs de la fenêtre.
	Supprime les données collectées. Vous êtes invité à confirmer la suppression.
	Affiche la rubrique d'aide pour cette fenêtre.

Graphique d'Appel



- Découvrez rapidement ce que fait un programme à tout moment
- Identifiez facilement les problèmes de performances
- Soyez surpris de la rapidité avec laquelle vous pouvez réaliser des améliorations
- Voyez vos améliorations au travail et ayez la preuve
- Support en charge des plateformes C/C++, .NET et Java

Usage

L'option « Graphique d'Appel » est généralement utilisée dans les situations où une activité s'exécute plus lentement que prévu, mais elle peut également être utilisée simplement pour mieux comprendre les motifs de comportement en jeu lors d'une activité.

Opération

Le Profiler fonctionne en prélevant des échantillons - ou piles d'appels - à intervalles réguliers sur une période donnée ; l'intervalle est défini à l'aide de la barre d'outils du profileur. Vous utilisez le Profiler pour exécuter un programme particulier ou vous pouvez vous attacher à un processus existant. La capture du Profiler est contrôlée et vous pouvez suspendre et reprendre la capture à tout moment. Vous pouvez également choisir de lancer la capture immédiatement au démarrage du profileur. Si nécessaire, vous pouvez supprimer tous les échantillons capturés et recommencer au cours de la même session. Si vous ne pouvez pas continuer la même session, redémarrer le Profiler est simple et rapide.

Note que le champ « Durée du processus (estimé) » affiche une estimation de la durée d'exécution du processus en cours de profilage, en tenant compte des interruptions du processus par le profileur lors de la collecte d'échantillons.

Résultats

Les résultats peuvent être produits à tout moment pendant la session ; cependant, la capture doit être désactivée pour que le bouton Rapport devienne actif. C'est à vous de décider combien de temps vous laissez le Profiler exécuter . Vous savez peut-être quand une activité est terminée ou cela peut être apparent pour d'autres raisons. La raison pour laquelle vous êtes ici peut être qu'une activité ne se termine pas du tout.

Le bouton Rapport sera activé soit en suspendant la capture, soit en arrêtant complètement le profileur.

Les résultats sont affichés dans une vue Rapport . Le rapport s'ouvre avec trois onglets initialement visibles : le Graphique d'Appel , le Rapport Sommaire et les onglets Analyse Hit . Les rapports peuvent être enregistrés dans un fichier, stockés dans le modèle en tant qu'artefacts ou affichés dans la Bibliothèque d'Équipe .

L'onglet Graphique d'Appel

	Inclusive Hits	UFP	Hits	Inclusive Hits%	Hits%
EA:CNIEMSchemaImporterDlg::OnBnClickedImport	1,283	1		54%	
EA:CNIEMSchemaImporter::ImportSchemas	862	1		36%	
EA:CNIEMNamespaceCreator::CreateNIEMNamespace	398	1		17%	
EA:CNIEMNamespaceCreator::CreateSchemaTypeProperties	259	1		11%	
EA:CNIEMNamespaceCreator::CreateComplexTypeProperties	147	1		6%	
EA:CNIEMNamespaceCreator::CreateNIEMAttribute	109	35		5%	
EA:CDAODataMan::UpdateEx	109	43		5%	
EA:CDAODataMan::UpdateAutoCounter	76	32		3%	
EA:CSSRecordset::Update	76	46		3%	
EA:CSSRecordset::Update	15	7		1%	
EA:SACCommand::SACCommand	15	13		1%	
EA:SACCommand::setCommandText	15	14		1%	
EA:SACCommand::ParseCmd	15	14		1%	
EA:SACCommand::ParseInputMarkers	11	10			
EA:SACCommand::CreateParam	8	7			
EA:saParams::find	8	7			
EA:SACCommand::CompareIdentifier	8	7			
EA:SACCommand::CompareIdentifier	4	4			
EA:SACCommand::CompareIdentifier	4	4			
EA:SACCommand::CompareIdentifier	3	2			
EA:SACCommand::CompareIdentifier	1	1	1		
EA:SACCommand::CreateParam	1	1			
EA:SACCommand::CreateParam	1	1			

L'onglet Rapport récapitulatif

Functions	Inclusive Hits	Depth	Modules	Occurrences
invoke_main	2392	4	EA	1
wWinMain	2392	5	EA	1
__scrt_common_main	2392	2	EA	1
__scrt_common_main_seh	2392	3	EA	1
CBCGPDIALOG::DoModal	1771	80	EA	2
CMainFrame::WindowProc	1671	103	EA	9
CBCGPFrameWnd::OnCommand	1625	21	EA	1
CMainFrame::OnCmdMsg	1625	24	EA	1
CMainFrame::OnImportNIEMXSD	1625	27	EA	1
CSSDialog::DoModal	1622	28	EA	1
CBCGPDIALOG::PreTranslateMessage	1538	92	EA	3
CSSDialog::PreTranslateMessage	1535	40	EA	2
CBCGPButton::OnLButtonUp	1533	105	EA	2
CBCGPDIALOG::OnCommand	1533	123	EA	2
CNIEMSchemaImporterDlg::OnBnClickedImport	1283	77	EA	1
CNIEMSchemaImporter::ImportSchemas	862	78	EA	1
CNIEMNamespaceCreator::CreateNIEMNamespace	398	79	EA	1
CDAODataMan::UpdateEx	398	86	EA	20
CSSRecordset::Update	322	89	EA	23
CNIEMSchemaImporter::ImportSchemas	304	78	EA	1

L'onglet Analyse Hit

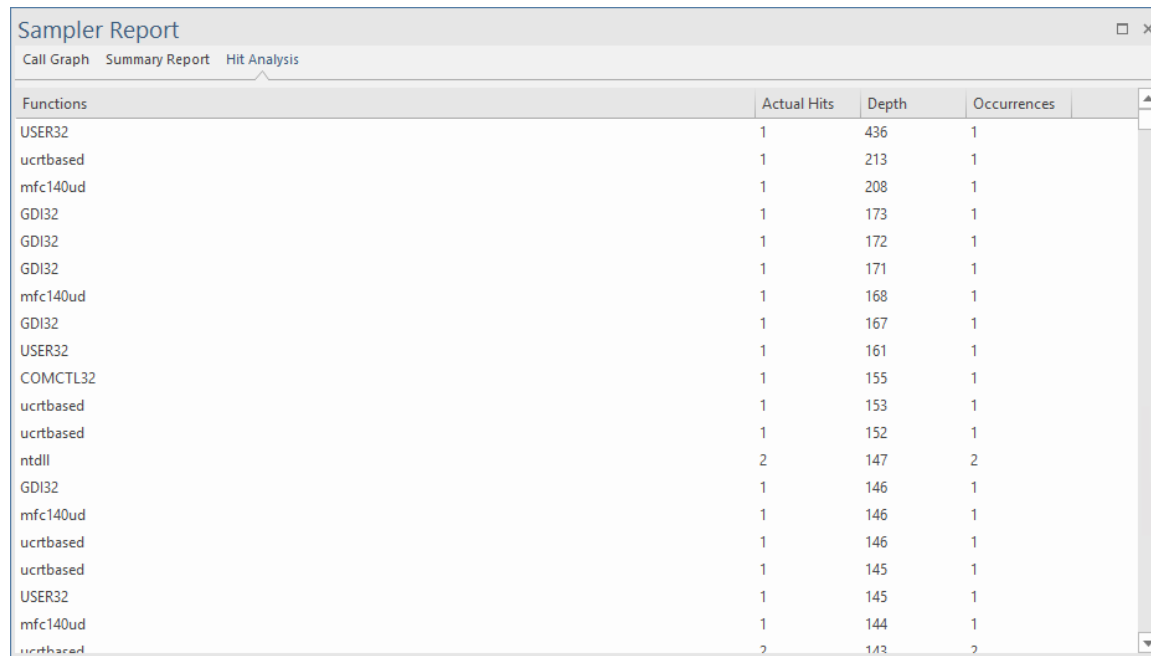
L'onglet « Analyse des Hit » affiche un certain nombre de colonnes :

- Fonction : le nom de la fonction (ou du module si aucun symbole pour le module)
- Hits : le nombre d'échantillons prélevés au cours desquels la fonction s'exécutait.
- Profondeur : le numéro d'image ou la profondeur de pile à laquelle l'accès a eu lieu.

- Occurrences : le nombre de fois où la fonction a été touchée à cette profondeur de pile particulière

Le nombre d'accès à une fonction particulière est agrégé en fonction de la profondeur du cadre de pile lors de l'échantillonnage.

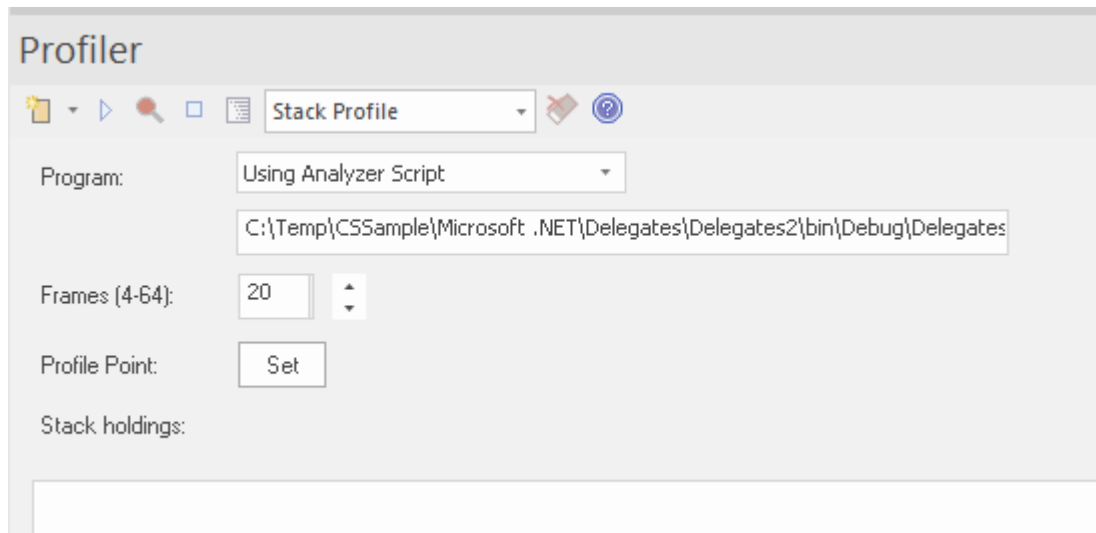
Si le nom de la fonction n'est pas disponible, par exemple les DLL du système Windows telles que User32 ou les DLL sans informations de débogage, le nom du module est affiché à la place.



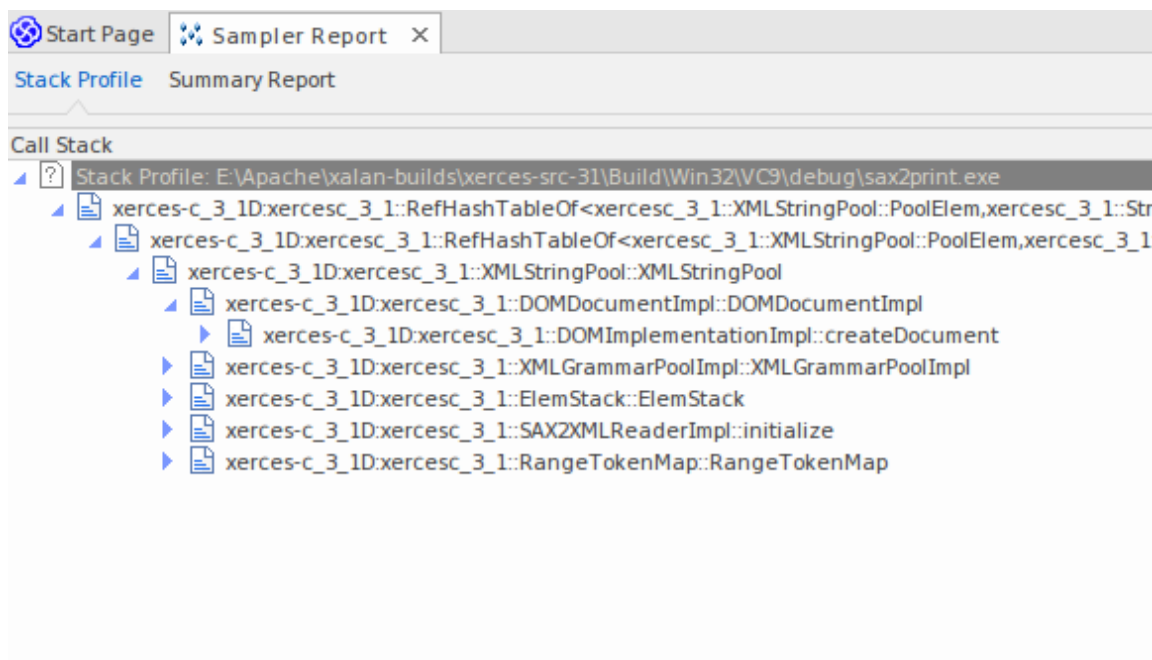
Functions	Actual Hits	Depth	Occurrences
USER32	1	436	1
ucrtbased	1	213	1
mfc140ud	1	208	1
GDI32	1	173	1
GDI32	1	172	1
GDI32	1	171	1
mfc140ud	1	168	1
GDI32	1	167	1
USER32	1	161	1
COMCTL32	1	155	1
ucrtbased	1	153	1
ucrtbased	1	152	1
ntdll	2	147	2
GDI32	1	146	1
mfc140ud	1	146	1
ucrtbased	1	146	1
ucrtbased	1	145	1
USER32	1	145	1
mfc140ud	1	144	1
ucrtbased	2	143	2

Profil de Pile

Le Stack Profiler vous permet de définir un marqueur dans votre code source afin que chaque fois que l'exécution atteint ce marqueur, une trace complète de la pile soit capturée. Au fur et à mesure que l'application continue de s'exécuter et que la position marquée est accessible à partir de divers endroits dans l'exécutable en cours d'exécution, une image très détaillée et utile est créée montrant les points chauds et les scénarios d'utilisation pour un point particulier du code.



Le rapport Profil de Pile, comme le rapport Profil de mémoire, est affiché dans l'ordre « pile inversée ». Cela signifie que la racine du rapport est toujours un nœud unique (dans ce cas, le marqueur) et que l'arborescence se déploie ensuite pour montrer tous les différents endroits à partir desquels la position marquée a été accédée.



Usage

Utilisez le mode Profil de Pile pour produire un rapport qui montre les manières uniques dont une fonction peut être invoquée pendant l'exécution d'un programme. Déterminez les parties du modèle qui dépendent de cette fonction et leur fréquence.

Opération

```
106
107 template <class TVal, class THasher>
108 void RefHashTableOf<TVal, THasher>::initialize(const XMLSize_t modulus)
109 {
110     if (modulus == 0)
111         ThrowXMLwithMemMgr(IllegalArgumentException, XMLExcepts::HshTbl_ZeroMo
112
113     // Allocate the bucket list and zero them
114     fBucketList = (RefHashTableBucketElem<TVal>**) fMemoryManager->allocate
115     (
116         fHashModulus * sizeof(RefHashTableBucketElem<TVal>*)
117     );
118     for (XMLSize_t index = 0; index < fHashModulus; index++)
119         fBucketList[index] = 0;
120 }
121
```

Les modes du profileur sont sélectionnés à l'aide de la barre d'outils de contrôle du profileur. Si un point Profiler est déjà créé, il est affiché. Le point Profiler est le point auquel les traces de pile sont capturées. Vous pouvez définir le point de profileur à l'aide du bouton Définir sur la commande elle-même, une fois le mode sélectionné. Après avoir choisi le point de profil, créez le projet pour vous assurer que tout est à jour, puis démarrez le profileur. Le nombre de titres de pile uniques détectés est visible lors de l'exécution.

Résultats

Un résultat peut être produit en cliquant sur le bouton de rapport dans la barre d'outils du contrôle Profiler. Ce bouton est activé lorsque :

- La capture est désactivée (à l'aide du bouton Pause) ou
- Le Profiler est arrêté (à l'aide du bouton Stop)

Les résultats produits sont affichés sous forme de graphique d'appel pondéré, où les lignes du graphique représentent une pile unique, et pondérées pour afficher en premier les piles de fréquences les plus élevées. Le rapport peut ensuite être enregistré, soit dans un fichier, soit dans le modèle, à l'aide du menu contextuel du rapport lui-même.

Profil de mémoire

Profiler

Memory Profile

Program: Using Analyzer Script

C:\Code Samples\VEA\Microsoft Native\CityLoop\debug

Frames (4-64): 16 Elapsed: 3.3 secs

	Heap	Virtual
Allocations:	4377	2
Frees:	0	0
Frees not	0	0
Stack holdings:	4375	2
Memory holdings	1	0

Module loaded: api-ms-win-core-datetime-l1-1-1, base: 7e9d0000
 Module loaded: api-ms-win-core-localization-obsolete-l1-2-0, base: 7e71
 Build version: 3.6, status: 0x0003, process heap: 0x00110000
 Auditor installed.
 Module loaded: api-ms-win-appmodel-runtime-l1-1-2, base: 7e6b0000
 Target process ended.
 Audit ended.

Start Page CityLoop *leaks *profile x				
Memory Profile Summary Report				
Call Stack	Instances	Bytes	Line	
ntdll:RtlAllocateHeap	4,375	1,237,419	-1	
user32:PostMessageA ? (+0x00BD, +189)	913	233,728	-1	
user32:ShutdownBlockReasonDestroy ? (+0x0A6A, +2666)	632	40,448	-1	
ucrtbased:toupper	541	100,374	-1	
user32:GetClientRect ? (+0x0103, +259)	480	245,760	-1	
gdi32:SetAbortProc ? (+0x032F, +815)	193	17,756	-1	
gdi32:CreateFontIndirectExW ? (+0x0051, +81)	145	13,340	-1	
winex11:SetFocus ? (+0x0A79, +2681)	121	4,992	-1	
ntdll:LdrGetDllHandle ? (+0x0072, +114)	88	54,628	-1	
ntdll:RtlDosPathNameToNtPathName_U_WithStatus ? (+0x0300, +768)	74	14,864	-1	
gdi32>SelectObject ? (+0x00E2, +226)	58	33,456	-1	
user32:GetTitleBarInfo ? (+0x09AF, +2479)	56	16,296	-1	
user32:ShutdownBlockReasonDestroy ? (+0x0A6A, +2666)	52	29,212	-1	
gdi32:GetCharWidthInfo ? (+0x0461, +1121)	52	832	-1	

- Évaluez rapidement les performances des activités qui vous intéressent
- Rien n'influence plus une discussion que les preuves
- Récompensez vos efforts en travaillant dans les domaines qui feront la différence

- Surprenez-vous en proposant des optimisations dont vous ignoriez peut-être l'existence

Usage

Le profil de mémoire peut être utilisé pour révéler comment les activités se déroulent en ce qui concerne la consommation de mémoire. En utilisant ce mode, un utilisateur serait intéressé à s'interroger sur la fréquence des demandes de mémoire au cours d'une tâche. Ils seraient moins intéressés par la quantité réelle consommée. Une activité bien gérée peut effectuer relativement peu d'appels pour allouer des ressources mais allouer suffisamment de mémoire pour faire son travail efficacement. D'autres activités peuvent générer plusieurs milliers de requêtes, ce qui les rend généralement moins efficaces. Ce mode est utile pour détecter ces scénarios.

Opération

Le profil de mémoire fonctionne en accrochant le processus en question, de sorte que ce programme doit être lancé à l'aide de l'outil d'Enterprise Architect. Contrairement à l'option Graphique d'Appel, vous ne pouvez pas vous rattacher à une démarche existante. Lorsque le programme est démarré, les mécanismes de hooking suivent l'allocation de mémoire ; ces informations sont collectées et rassemblées dans Enterprise Architect. Vous pouvez facilement surveiller le nombre d'allocations effectuées. De plus, le processus est contrôlé ; c'est-à-dire que les hooks de mémoire peuvent être activés et désactivés à la demande. Si vous avez mal programmé une action, vous pouvez suspendre la capture, supprimer les données et reprendre facilement la capture.

Résultats

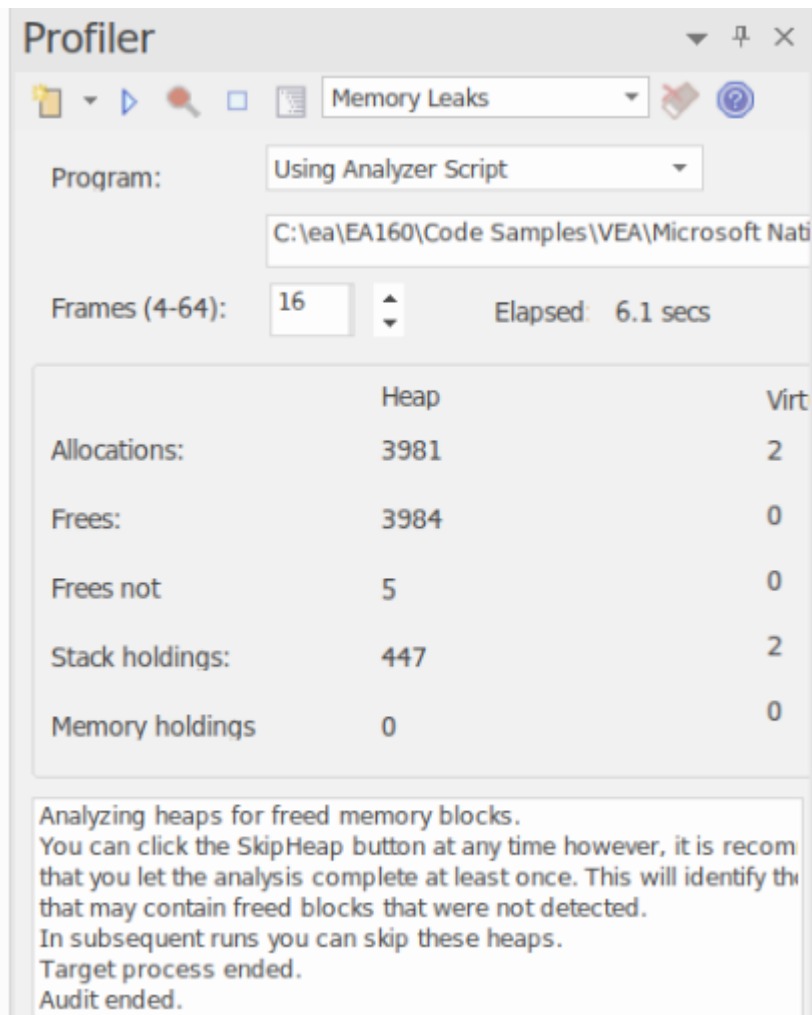
Les résultats peuvent être produits à tout moment pendant la session ; cependant, la capture doit être désactivée pour que le bouton Rapport devienne actif. C'est à vous de décider combien de temps vous laissez le Profileur exécuter. Vous activez le bouton Rapport en suspendant la capture ou en arrêtant complètement le profileur.

Les résultats sont affichés dans une vue Rapport. Le rapport s'ouvre initialement avec deux onglets visibles ; un seul Graphique d'Appel pondéré et un Résumé des Fonctions. Le Graphique d'Appel représente toutes les Call Stacks ayant conduit à des allocations de mémoire, qui sont agrégées et pondérées en fonction de la fréquence du motif.

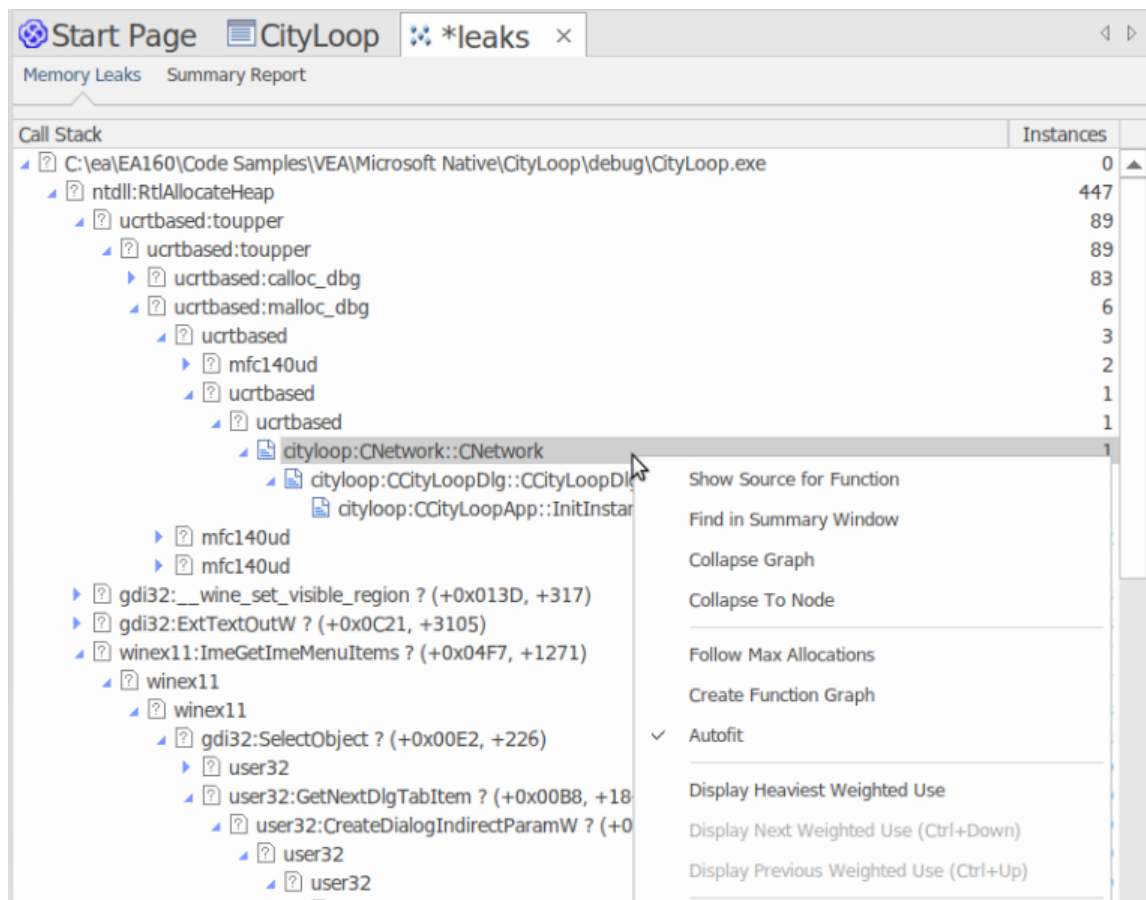
Exigences

Pour de meilleurs résultats, l'image et ses modules doivent être créés avec les informations de débogage incluses et sans optimisations. Tout module doté de l'optimisation FPO (Frame Pointer Omission) est susceptible de produire des résultats trompeurs.

Fuites de Mémoire



Le contrôle Profiler, affichant le nombre d'allocations de mémoire et le nombre d'opérations qui sont libres de mémoire.



Un programme bien mené.

La détection des fuites de mémoire est un chemin bien parcouru. Bien que de nombreuses autres bonnes options soient disponibles, nous pensons que notre approche présente des avantages majeurs, tels que :

- Aucun changement dans la construction du projet existant
- Aucun fichier d'en-tête requis par le code du projet
- Aucune dépendance d'exécution à craindre
- Aucune configuration système à penser

Usage

Une personne utiliserait ce mode pour suivre les fuites de mémoire dans une application ou dans une activité au sein de l'application. Du point de vue du profileur, une fuite de mémoire est un appel réussi à une fonction d'allocation de mémoire qui renvoie une adresse mémoire pour laquelle aucun appel correspondant n'est effectué pour libérer cette adresse.

Opération

La détection des fuites de mémoire fonctionne par accrochage. Les routines de mémoire du processus sont conçues pour suivre le moment où la mémoire est à la fois allouée et libérée. Les piles d'appels sont capturées au moment de l'allocation et ces informations sont rassemblées dans Enterprise Architect pour produire un rapport sous la forme d'un Graphique d'Appel. La capture est contrôlée ; c'est-à-dire que les mécanismes d'accrochage peuvent être activés ou désactivés à la demande.

En fonction du type de programme et de sa consommation de mémoire, vous pouvez employer une stratégie appropriée. Pour les petits programmes, vous pouvez suivre le programme du début à la fin. Pour les programmes à fenêtres plus

grandes, vous feriez probablement mieux d'activer la capture avant et après une tâche spécifique pour éviter de suivre trop de données.

Résultats

Les résultats peuvent être produits à tout moment pendant la session ; cependant, la capture doit être désactivée pour que le bouton Rapport devienne actif. C'est à vous de décider combien de temps vous laissez le Profiler exécuter . Vous activez le bouton Rapport en suspendant la capture ou en arrêtant complètement le profileur.

Les résultats sont affichés dans une vue Rapport . Le rapport s'ouvre initialement avec deux onglets visibles ; un seul Graphique d'Appel pondéré et un Résumé des Fonctions. Le Graphique d'Appel représente toutes les piles d'appels qui ont conduit à des allocations de mémoire, et sont agrégées et pondérées en fonction de la fréquence du motif .

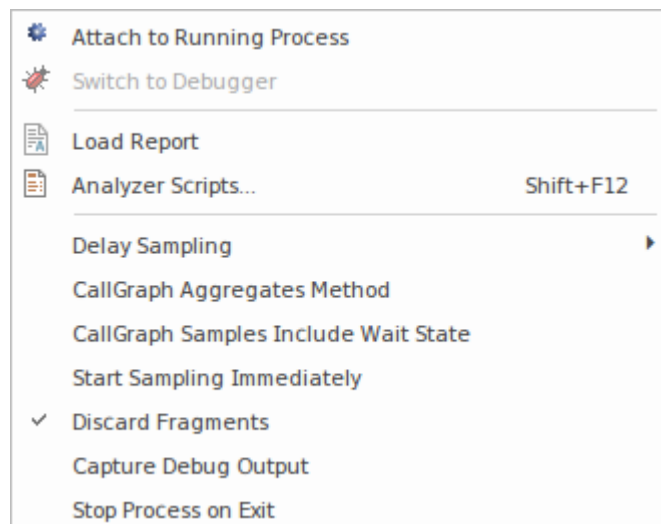
Rapports peuvent contenir une quantité variable de « bruit ». Pour vous concentrer sur un domaine qui vous préoccupe spécifiquement, localisez une fonction que vous connaissez dans le rapport de synthèse et utilisez-la pour naviguer directement dans la ligne du graphique où elle apparaît.

Exigences

Pour de meilleurs résultats, l'image et ses modules doivent être créés avec les informations de débogage incluses et sans optimisations. Tout module doté de l'optimisation FPO (Frame Pointer Omission) est susceptible de produire des résultats trompeurs.

Options de Réglage

La première icône de la barre d'outils de la fenêtre Profiler affiche une liste d'options que vous pouvez définir pour personnaliser votre session de profilage.



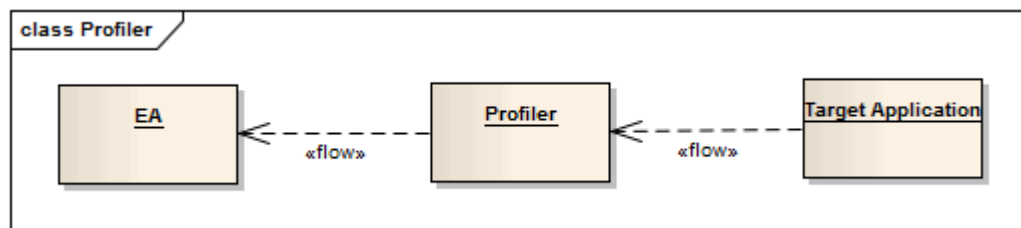
Possibilités

Option	Description
Attacher au processus en cours d'exécution	Sélectionnez cette option pour afficher la dialogue « Attacher au processus », à partir de laquelle vous choisissez un processus actif à profiler.
Passer au Débogueur	Sélectionnez cette option pour modifier les opérations du profilage au débogage. Le Débogueur dispose d'une option de menu déroulant équivalente que vous pouvez utiliser pour passer du débogage au profilage.
Rapport de charge	Sélectionnez cette option pour charger un rapport précédemment enregistré à partir du système de fichiers.
Scripts d'Analyseur	Sélectionnez cette option pour ouvrir la fenêtre Analyzer Script, qui est le référentiel de modèles pour la configuration des builds, le débogage et toutes les autres options Analyseur d'Exécution Visuelle .
Échantillonnage différé	Sélectionnez cette option pour définir un délai entre le clic sur une option « Démarrer le profilage » et le début réel du profilage. Le délai peut être de 3, 5 ou 10 secondes. Sélectionnez « Aucun » pour annuler tout délai défini.
Méthode d'agrégations CallGraph	Lorsque cette option est sélectionnée, les instances des séquences de pile identiques sont agrégées par méthode. C'est-à-dire que les numéros de ligne/instructions au sein d'une méthode sont ignorés, donc deux piles seront comptées comme une seule lorsqu'elles ne diffèrent que par le numéro de ligne dans leur trame finale.
Les exemples CallGraph incluent State d'attente	Lorsque cette option est sélectionnée, le profileur échantillonne tous les threads, y compris ceux en état d'attente. Lorsqu'il n'est pas sélectionné, le profileur échantillonne uniquement les threads qui ont accumulé du temps CPU depuis

	l'expiration du dernier intervalle.
Démarrer l'échantillonnage immédiatement	Sélectionnez cette option pour déclencher la collecte de données immédiatement au lancement. Vous utiliserez généralement cette option pour profiler un processus lors du démarrage.
Jeter les fragments	<p>Lorsque les piles ne peuvent pas être réconciliées avec le point d'entrée d'un thread, elles sont appelées fragments. Le nombre de fragments rencontrés lors de l'échantillonnage est affiché dans la fenêtre Résumé de l'échantillonneur. Vous pouvez définir cette option pour collecter ou supprimer des fragments ; lorsque l'option Supprimer les fragments est :</p> <ul style="list-style-type: none">• Sélectionnés, les fragments n'apparaissent pas dans les rapports, bien que le nombre rencontré soit toujours mis à jour• Désélectionnée, une collection spéciale nommée « fragments » est créée dans le graphe d'appel pour les héberger et pour garantir que leurs données ne sont pas mélangées aux échantillons complets.
Capturer la sortie Déboguer	(S'applique à l'échantillonnage de processus). Lorsqu'elle est sélectionnée, la sortie normalement visible pendant le débogage est capturée et affichée dans la fenêtre Déboguer . Note que seules les versions de débogage émettent généralement une sortie de débogage.
Arrêter le processus à la sortie	Cette option détermine le comportement de terminaison du profileur. Lorsque l'option est sélectionnée, le processus cible se terminera lorsque le profileur sera arrêté.

Démarrer et Arrêter le Profileur

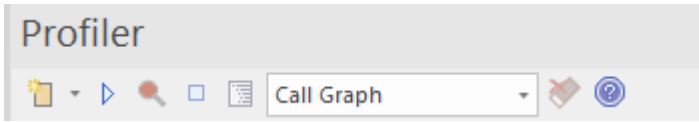
Le profilage est un processus en deux étapes de collecte de données et de reporting. Dans Enterprise Architect la collecte de données a l'avantage d'être une tâche en arrière-plan : vous êtes donc libre de faire autre chose pendant son exécution. Les informations renvoyées à Enterprise Architect sont stockées jusqu'à ce que vous génériez un rapport. Pour visualiser un rapport, la capture doit être désactivée. Une fois le rapport produit, vous pouvez reprendre la capture en cliquant sur un bouton. Si, pour une raison quelconque, vous décidez de supprimer vos données et de recommencer, vous pouvez le faire facilement et sans avoir à arrêter et redémarrer le programme.



Accéder

Ruban	Exécuter > Outils > Profiler > Ouvrir le profileur
Autre	Barre d'outils Analyseur d'Exécution : Analyseur Windows Profileur

Actions

Action	Détail
Barre d'outils	
Sélection de stratégie	Sélectionnez la stratégie de profilage parmi les options disponibles dans la barre d'outils.
Démarrer le Profileur	Cliquez sur le bouton Exécuter dans la fenêtre Profiler
Arrêter le Profileur	<p>Le processus se termine si :</p> <ul style="list-style-type: none">• Vous cliquez sur le bouton Stop• L'application cible se termine, ou• Vous fermez le modèle actuel <p>Si vous arrêtez le profileur et que le processus est toujours en cours, vous pouvez rapidement vous y connecter à nouveau.</p>
Pause et Reprise de Capture	<p>Vous pouvez suspendre et reprendre la capture à tout moment pendant une session. Lorsque la capture est activée, des échantillons sont collectés sur la cible. Lorsqu'il est en pause, le profileur entre et reste dans un état d'attente jusqu'à ce que la</p>

	capture soit activée, que le profileur soit arrêté ou que l'application se termine.
Générer Rapports	Le bouton Rapport est désactivé pendant la capture mais est disponible lorsque la capture est désactivée.
Mode déroulant	Cliquez sur le menu déroulant et sélectionnez le mode de Profilage - Graphique d'Appel , Profil de Pile , Memory Profile ou Fuites de Mémoire .
Collecte de données Effacer	Vous pouvez effacer tous les échantillons de données collectés et reprendre à tout moment. Suspendez d'abord la capture en cliquant sur le bouton Pause. Le bouton Supprimer, comme pour le bouton Rapport , est activé chaque fois que la capture est désactivée. En cliquant sur le bouton Supprimer, il vous sera demandé de confirmer l'opération. Cette action ne peut pas être annulée.

Rapports de Ligne de Fonction

Après avoir exécuter le Profiler sur une application en cours d'exécution et généré un rapport Sampler, vous pouvez analyser plus en détail l'activité d'une fonction spécifique répertoriée dans le rapport en générant un rapport de ligne de fonction à partir de cet élément. Un rapport de ligne de fonction indique le nombre de fois où chaque ligne de la fonction a été exécutée. Vous produisez un rapport de ligne de fonction à la fois, sur n'importe quelle méthode du rapport Sampler qui possède un fichier source valide. Le rapport Ligne de fonction est particulièrement utile pour les fonctions qui exécutent des boucles contenant un branchement conditionnel ; la couverture peut fournir une image des parties de code les plus fréquemment et les moins fréquemment exécutées au sein d'une seule méthode.

Le rapport de ligne que vous générez est enregistré lorsque vous enregistrez le rapport Sampler. Le corps de la fonction est également enregistré avec le rapport Ligne de fonction afin de préserver l'état de la fonction à ce moment-là.

Cette facilité ne s'applique pas aux rapports de profil de mémoire.

Plateformes prises en charge

Java, Microsoft .NET et code natif Microsoft

Créer un Rapport de Ligne

Dans le rapport Sampler, cliquez-droit sur le nom de la fonction à analyser, et sélectionnez l'option 'Créer Rapport ligne pour la fonction'.

Une fois que le profileur lie la méthode, le rapport de ligne de fonction s'ouvre dans la fenêtre Sampler Rapport . Le rapport affiche le corps de la fonction, y compris les numéros de ligne et le texte. Au fur et à mesure que chaque ligne est exécutée, une valeur de frappe s'accumulera par rapport à cette ligne. Une minuterie mettra à jour le rapport environ une fois par seconde.

Call Tree Report Summary Report ConsoleApplication::CQuickSort::Quicksort		
Module: ConsoleApplication Function: CQuickSort::Quicksort Date: 20/09/2013 2:53:21 PM Author: smeagher Iterations: 28679		
LineNo	Hits	Code
21	28645	{
22	28644	if (r <= l)
23	14460	return;
24	14184	int i = l-1, j = r, p = l-1, q = r;
25		for (;;)
26		{
27	439580	while (a[++i] < a[r]) ;
28	14185	while (a[--j] > a[r])
29		if (j == i)
30		break;
31		if (j >= j)
32	14185	break;
33		
34		Exchange(a, i, j);
35		if (a[i] == a[r])
36		Exchange(a, ++p, i);
37		
38		if (a[j] == a[r])
39		Exchange(a, j, --q);
40		
41		}
42	14185	Exchange(a, i, r);
43	14185	j = i-1; i = i+1;
44	14185	for (int k = l; k < p; k++, j--)

Terminer Capture du Rapport de Ligne

Une fois que suffisamment d'informations sont capturées ou que la fonction est terminée, cliquez sur le bouton Arrêter de la barre d'outils du profileur pour arrêter l'enregistrement de la capture.

Enregistrer Rapports

Utilisez le bouton Enregistrer de la barre d'outils Pile d'Appel pour enregistrer le rapport Sampler et tous les rapports Function Line dans un fichier.

Supprimer Rapports de ligne


La fermeture de l'onglet 'Line Rapport ' fermera ce rapport mais les données du rapport ne seront supprimées que lorsque le rapport sera enregistré.

Générer, Enregistrer et Charger des Rapports de Profil

Rapports peuvent être produits à tout moment au cours d'une session, ou naturellement à la fin d'un programme. Cependant, pour activer le bouton Rapport pendant l'exécution du programme, vous devez suspendre le profilage en basculant le bouton Pause/Reprendre ou en mettant fin au Profiler avec le bouton Arrêter. Vous disposez de plusieurs options pour réviser et partager les résultats :

- Vue le rapport
- Enregistrez le rapport dans un fichier
- Distribuer le rapport comme ressource Bibliothèque d'Équipe
- Joindre le rapport en tant que document à un élément Artefact
- Synchronisez le modèle par rétro-ingénierie du code source qui a participé au profil


Accéder

Ruban	Exécuter > Outils > Profileur > Créer Rapport à partir des données actuelles
Profileur	Depuis la fenêtre Profiler, cliquez sur l'icône  dans la barre d'outils.

Charger Rapport à partir d'un fichier

L'option est disponible dans le menu déroulant de la fenêtre Profiler

Générer Rapport

Depuis la fenêtre Profiler, cliquez sur l'icône  dans la barre d'outils.

Rapport de fréquence d'appel

Call Stack	Inclusive Hits	Hits
xercesc_3_1::SAX2XMLReaderImpl::parse	16051	
xercesc_3_1::XMLScanner::scanDocument	16051	
xercesc_3_1::IGXMLScanner::scanDocument	16051	
xercesc_3_1::IGXMLScanner::scanContent	16051	
xercesc_3_1::IGXMLScanner::scanStartTagNS	16051	
xercesc_3_1::IGXMLScanner::resolveSchemaGrammar	16051	
xercesc_3_1::SchemaValidator::preContentValidation	16049	
xercesc_3_1::ComplexTypeInfo::checkUniqueParticleAttribution	16049	
xercesc_3_1::ComplexTypeInfo::makeContentModel	16049	
xercesc_3_1::DFAContentModel::DFAContentModel	16047	
xercesc_3_1::DFAContentModel::buildDFA	15998	515
xercesc_3_1::CMStateSet::operator =	8174	8093
memcpy	32	32
xercesc_3_1::CMStateSet::allocateChunk	27	1
__security_check_cookie	21	21
TrailUpVec	1	1
xercesc_3_1::CMStateSet::~CMStateSet	3573	4
xercesc_3_1::XMemory::operator delete	841	2
xerces-c_3_1D	4416	2
xercesc_3_1::CMStateSet::getBit	1036	1036
xercesc_3_1::DFAContentModel::buildSyntaxTree	528	3
xercesc_3_1::CMStateSet::CMStateSet	373	3
xercesc_3_1::CMStateSet::getBitCountInRange	285	285
xercesc_3_1::XMemory::operator new	211	2
xercesc_3_1::CMStateSet::zeroBits	154	
xercesc_3_1::CMStateSetEnumerator::nextElement	153	136
xercesc_3_1::RefHashTableOf<xercesc_3_1::XMLInteger>	59	2
xercesc_3_1::RefHashTableOf<xercesc_3_1::XMLInteger>	28	2
xercesc_3_1::RefHashTableOf<xercesc_3_1::XMLInteger>	25	
xercesc_3_1::DFAContentModel::makeDefStateList	25	2

Résumé des fonctions

Name	Inclusive Hits	Occurrences
mainCRTStartup	7408	1
__tmainCRTStartup	7407	1
xercesc_3_1::XMLFormatter::handleUnEscapedChars	7351	10
xercesc_3_1::XMLFormatter::formatBuf	7351	10
xercesc_3_1::XMLFormatter::specialFormat	7351	10
SAX2PrintHandlers::writeChars	7350	10
xercesc_3_1::XMLScanner::scanDocument	7350	1
main	7350	1
xercesc_3_1::SAX2XMLReaderImpl::parse	7350	1
xercesc_3_1::XMLScanner::scanDocument	7349	1
xercesc_3_1::IGXMLScanner::scanDocument	7348	1
xercesc_3_1::XMLFormatter::formatBuf	4042	8

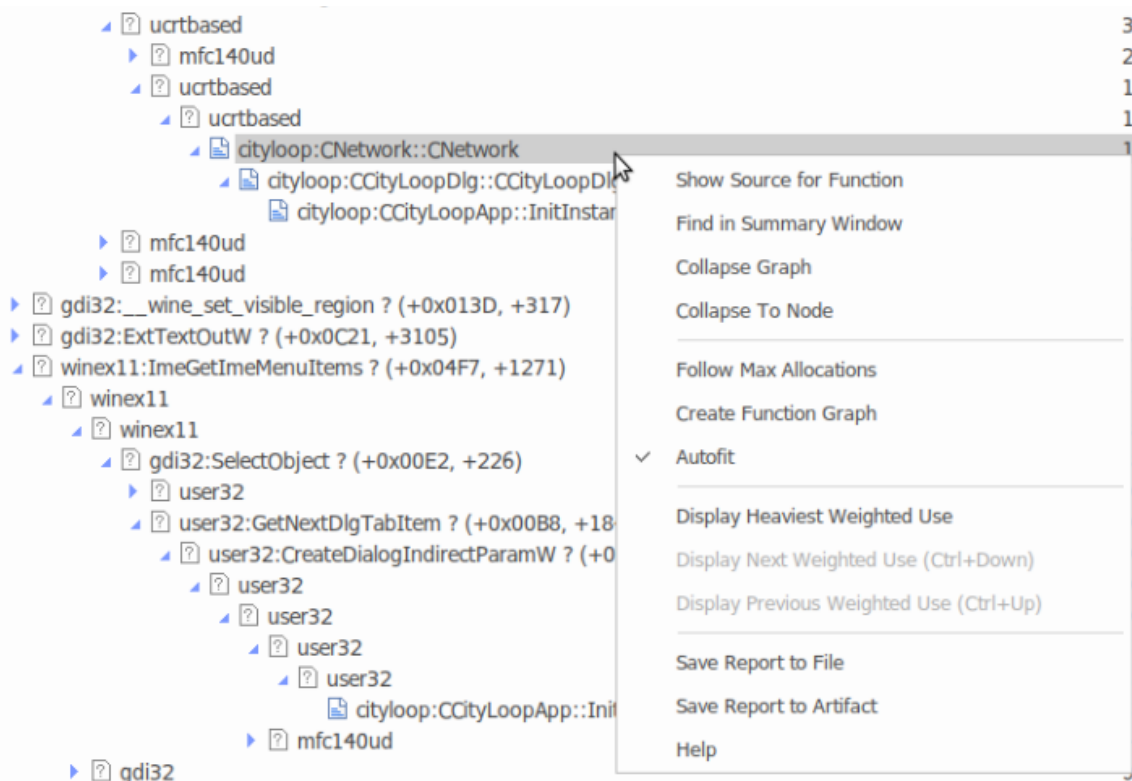
Rapport récapitulatif non filtré répertoriant toutes les fonctions participantes par ordre d'accès inclus.

Name	Inclusive Hits	Occurrences
SAX		
SAX2PrintHandlers::write Chars	7350	10
xercesc_3_1::SAX2XMLReaderImpl::parse	7350	1
xercesc_3_1::SAX2XMLReaderImpl::docCharacters	3309	2
SAX2PrintHandlers::characters	3309	2
xercesc_3_1::SAX2XMLReaderImpl::endElement	2114	1
xercesc_3_1::SAX2XMLReaderImpl::startElement	1925	1
SAX2PrintHandlers::endElement	1523	1

Vous pouvez filtrer et réorganiser les informations dans le rapport, de la même manière que vous le faites pour les résultats d'une Recherche Modèle .

Options Rapport

Cliquez-droit sur l'état pour afficher le menu contextuel.



Note que les options répertoriées dépendent du type de rapport affiché ; le rapport illustré ici est un rapport de profil de mémoire.

Action	Détail
Afficher la source de la fonction	Pour le frame sélectionné, sélectionnez cette option pour afficher la ligne de code correspondante dans un éditeur de code. Les images dont la source est disponible sont identifiables par leur icône.
Rechercher dans la fenêtre Résumé	Sélectionnez cette option pour localiser la fonction dans la fenêtre Résumé.
Réduire le graphique	Sélectionnez cette option pour réduire l'intégralité du graphique, y compris les

	nœuds enfants, visibles ou non.
Réduire au nœud	Sélectionnez cette option pour réduire l'intégralité du graphique, puis développez et définissez le focus sur le nœud sélectionné.
Suivre les allocations maximales	Sélectionnez cette option pour développer une ligne entière dans le graphique.
Créer Rapport de ligne pour la fonction	<p>Sélectionnez cette option pour lancer le Profiler (s'il n'est pas déjà en cours d'exécution), lier immédiatement la fonction sélectionnée et la préparer pour l'enregistrement. Une fois lié, un onglet supplémentaire est ouvert dans le Rapport Vue actuel. Ce rapport sera mis à jour instantanément, indiquant le nombre de fois où chaque ligne a été exécutée. Bien entendu, le rapport continuera à enregistrer l'activité dans la fonction même s'il n'est pas visible.</p> <p>Notes :</p> <ul style="list-style-type: none"> • Dans les programmes fenêtrés, il est généralement nécessaire d'effectuer une action dans l'application pour que la fonction soit invoquée. • Cette option ne s'applique pas aux rapports de profil de mémoire
Créer un graphique de fonctions	Sélectionnez cette option pour créer un onglet supplémentaire, qui affiche la fonction sélectionnée de manière isolée. Pour un profil de fréquence d'appel, cela produit un graphique montrant toutes les lignes qui ont conduit à l'appel de cette fonction (c'est-à-dire les appelants). Pour un profil de mémoire, cela produit un graphique montrant toutes les lignes qui émanent de cette fonction (c'est-à-dire les appelés).
Marquer le cadre initial pour Diagramme Pile d'Appel	<p>À utiliser avant de créer un diagramme Séquence Pile d'Appel pour limiter la longueur de la pile. Lorsque cette option est sélectionnée, le cadre est marqué et son texte est mis en surbrillance. Les images au-dessus de celle-ci seront alors exclues de tout diagramme Séquence produit.</p> <p>Cette option ne s'applique pas aux rapports de profil de mémoire.</p>
Supprimer la marque	<p>Supprime la marque d'un cadre précédemment marqué comme « Initial ».</p> <p>Cette option ne s'applique pas aux rapports de profil de mémoire.</p>
Créer Diagramme Pile d'Appel	<p>Génère un diagramme Séquence pour une seule pile dans le graphique. La trame sélectionnée est représentée comme la trame terminale dans la pile. La trame initiale de la pile est par défaut le nœud racine si aucune trame « Initiale » n'a été marquée.</p> <p>Cette option ne s'applique pas aux rapports de profil de mémoire.</p>
Créer Graphique d'Appel Diagramme	<p>Génère un diagramme Séquence qui présente une séquence pour chaque branchement de pile visible à partir du cadre sélectionné. En développant et en réduisant les nœuds qui vous intéressent, vous pouvez adapter le contenu du diagramme Séquence à votre guise.</p> <p>Cette option ne s'applique pas aux rapports de profil de mémoire.</p>
Afficher l'utilisation la plus lourde	Sélectionnez cette option pour afficher la ligne du graphique ayant le poids le plus élevé dans lequel cette fonction apparaît.
Afficher la prochaine utilisation pondérée	<p>Sélectionnez cette option pour accéder à la ligne suivante du graphique où la fonction apparaît.</p> <p>Vous pouvez utiliser la combinaison de touches de raccourci Ctrl+Flèche vers le bas.</p>

Afficher l'utilisation pondérée précédente	Sélectionnez cette option pour accéder à la ligne précédente du graphique où cette fonction apparaît. Vous pouvez également utiliser la combinaison de touches de raccourci Ctrl+Flèche vers le haut.
Importer le code source	Sélectionnez cette option pour importer le code source sélectionné dans le rapport. Cette option ne s'applique pas aux rapports de profil de mémoire.
Ajustement automatique	Lorsqu'il est activé, adapte automatiquement les colonnes à la zone d'affichage disponible.
Enregistrer Rapport dans un fichier	Sélectionnez cette option pour afficher la dialogue « Enregistrer sous », vous permettant de choisir où stocker le rapport.
Enregistrer Rapport dans l'artefact	Note : Avant de sélectionner cette option, allez dans la fenêtre Navigateur et sélectionnez le Paquetage ou l'élément sous lequel créer l'élément Artefact. Vous êtes invité à fournir un nom pour le rapport (et l'élément) ; tapez ceci et cliquez sur le bouton OK . L'élément Artefact est créé dans la fenêtre Navigateur , sous le Paquetage ou l'élément sélectionné. Si vous ajoutez l'artefact à un diagramme sous la forme d'un simple lien, lorsque vous double-cliquez sur l'élément, le rapport est rouvert.

Notes

- Si vous ajoutez le rapport Profiler à un élément Artefact et que vous joignez également un document lié, le rapport Profiler est prioritaire et s'affiche lorsque vous double-cliquez sur l'élément ; vous pouvez afficher le document lié à l'aide de l'option de menu contextuel « Modifier le document lié »

Enregistrer Rapport dans Bibliothèque d'Équipe

Vous pouvez enregistrer n'importe quel rapport actuel en tant que ressource pour une catégorie, un sujet ou un document dans la Bibliothèque d'Équipe . Le rapport peut ensuite être partagé et révisé à tout moment car il est enregistré avec le modèle. Cela vous aide à :

- Conserver un rapport Profiler pour le comparer aux exécutions futures
- Autoriser d'autres personnes à enquêter sur le profil

Accéder

Menu Contexte	Cliquez-droit dans la fenêtre Bibliothèque Partager la ressource Rapport Actif Profiler
---------------	---------------------------------------------------------------------------------------------

