



ENTERPRISE ARCHITECT

用户指南系列

混合脚本

Author: Sparx Systems

Date: 13/11/2024

Version: 17.0

创建于  **ENTERPRISE
ARCHITECT**

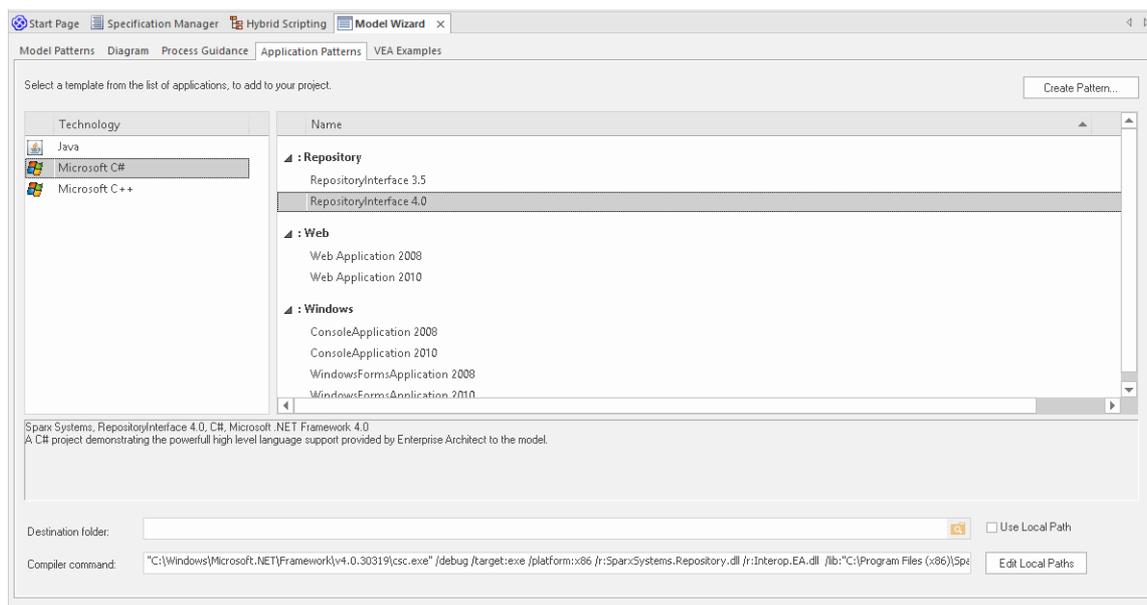
目录

混合脚本	3
C#示例	5
Java示例	7

混合脚本

混合脚本将标准脚本环境的功能扩展到Java和C#等高级语言。混合脚本提供了优于传统脚本的速度优势，还允许脚本作者利用流行编程语言中的现有技能。

访问

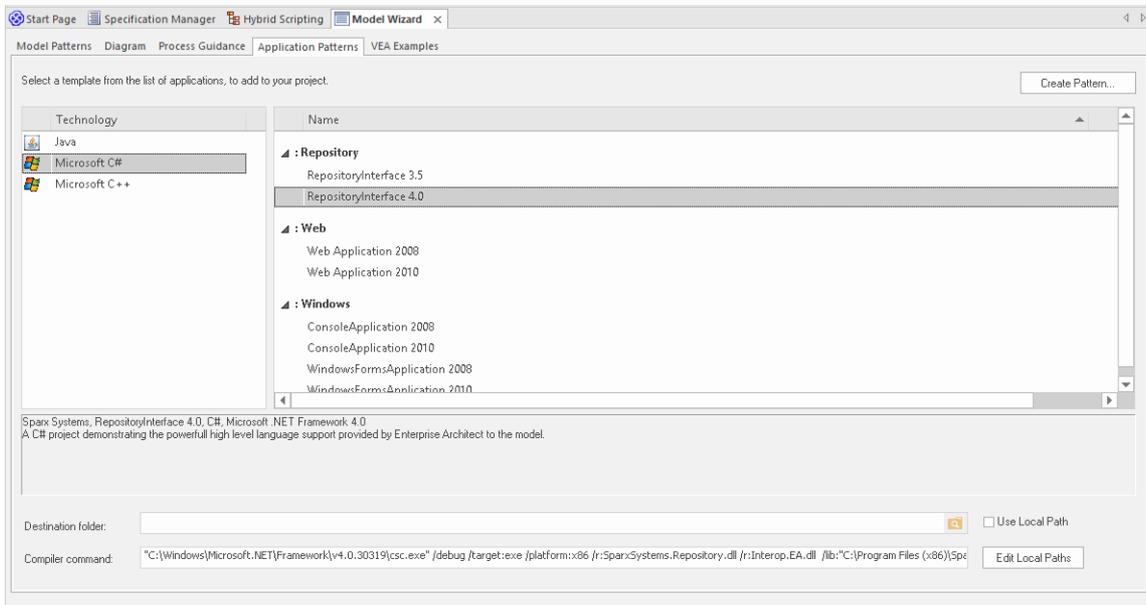


功能区

开发 > 源代码 > Create From模式 > Application模式

特征

- 卓越的执行速度
- 增强的互操作性
- 全面可视化执行分析器支持



C#示例

该示例程序演示了使用任何 Microsoft .NET语言导航、查询和报告当前模型是多么容易。此示例是用 C# 编写的。

运行时，会打印出您当前使用的模型中每个包的名称。

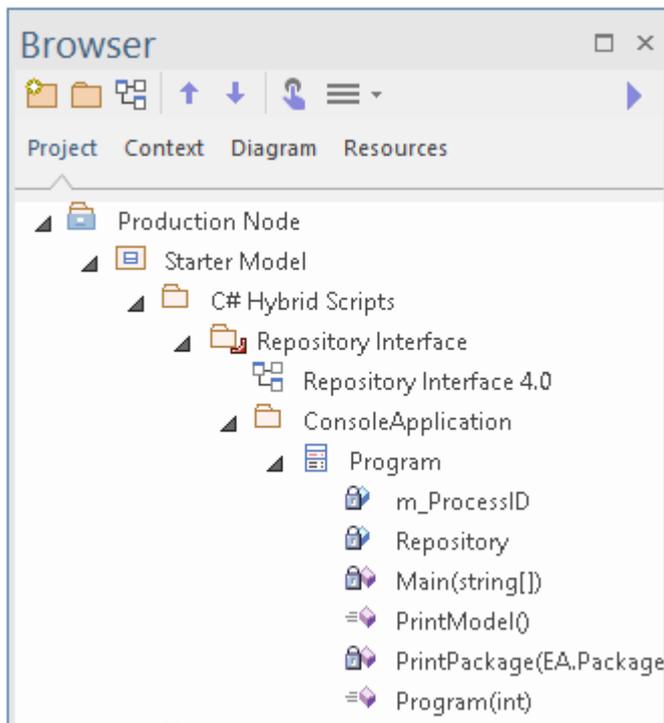
创建项目

在浏览器窗口中，选择要在其中创建模板的包，然后使用 开发>源代码>从模式创建”功能区选项显示模式窗口；单击 应用程序模式”选项。

从 应用程序模式”页面中，选择Microsoft C#> RepositoryInterface模板。（您可以从 3.5 或 4.0 框架版本中进行选择。）在文件系统上指定将创建项目模板的目标文件夹，然后单击确定按钮。

打开项目

将为您创建A此类似的包结构。



展开结构，直到找到存储库接口nn图并打开它。

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)

编译脚本

此图上的命令将在活动构建配置上运行。在执行它们之前，双击分析器脚本链接并选中“存储库接口”构建配置旁边的复选框。

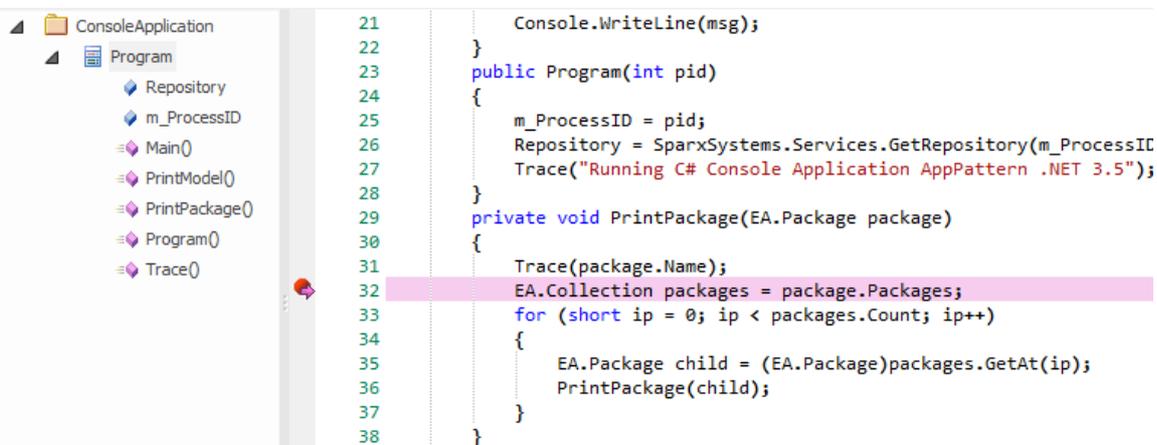
运行脚本

双击运行链接打开控制台。控制台将在完成后暂停，以便您读取程序的输出；此输出也将发送到系统输出窗口的“脚本”选项卡。您可以通过更改代码来更改它。

调试脚本

从浏览器窗口中选择“程序”类，然后按 Ctrl+E 打开源代码。

在其中一个函数中放置一个断点，然后双击 *DebugRun* 链接。当遇到断点时，该行代码将在编辑器中高亮显示，如图所示：



```
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 }
```

Java示例

该示例程序演示了使用Java等高级语言导航、查询和报告当前模型是多么容易。
运行时打印当前加载的模型中每个包的名称。

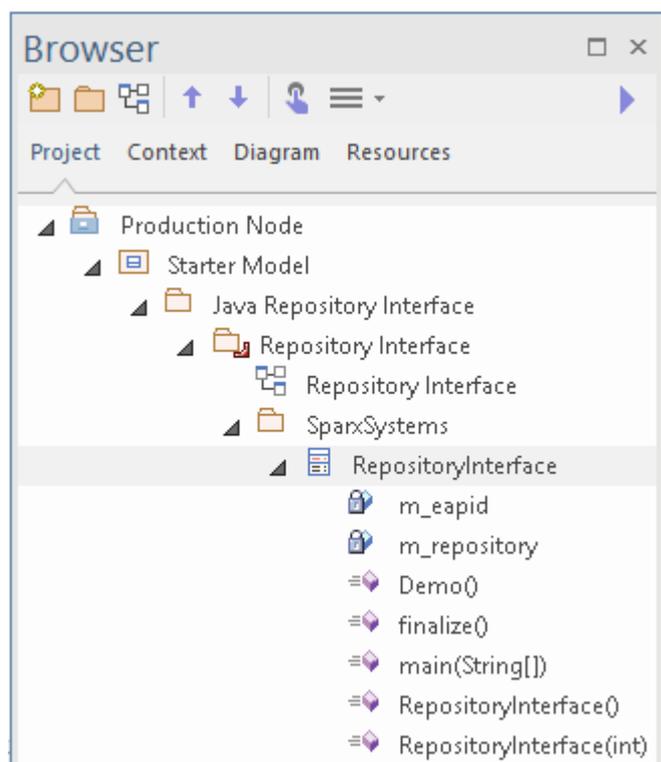
创建项目

在浏览器窗口中，选择要在其中创建模板的包，然后使用 开发>源代码>从模式创建”功能区选项显示模式窗口；单击 应用程序模式”选项。

从 应用程序模式”页面中，选择Java > RepositoryInterface模板。在文件系统上指定将创建项目模板的目标文件夹，然后单击确定按钮。

打开项目

将为您创建A此类似的包结构。



展开结构，直到找到 存储库接口”图并打开它。

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.

Build the project

Run your program

Debug the program

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

编译脚本

图表上的命令将在活动构建配置上运行。在执行它们之前，双击分析器脚本链接并选中“存储库接口”构建配置旁边的复选框。

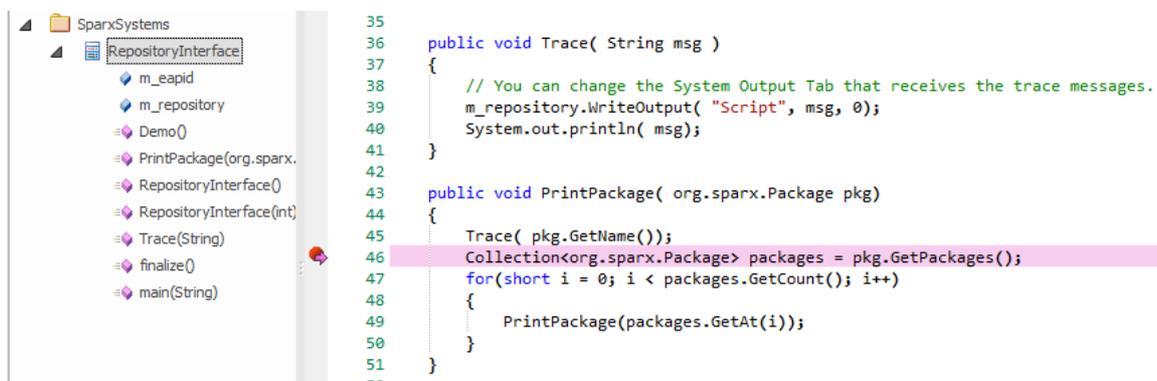
运行脚本

双击运行链接；控制台将打开。控制台将在完成后暂停，以便您阅读输出。程序的输出也会输出到系统输出窗口的“脚本”选项卡中。您可以通过更改代码来更改它。

调试脚本

从浏览器窗口中选择“程序”类，然后按 Ctrl+E 打开源代码。

在其中一个函数中放置一个断点，然后双击 *DebugRun* 链接。当遇到断点时，代码行将在编辑器中突出显示，如图所示。



```
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 }
52
```

