Enterprise Architect

User Guide Series

Hybrid Scripting

Hybrid Scripting in Sparx Systems Enterprise Architect extends standard scripting environments to high-level languages like Java or C#. It is faster than conventional scripting and helps scriptors apply existing skills in popular programming languages.

Author: Sparx Systems

Date: 7/08/2019

Version: 1.0

CREATED WITH ENTERPRISE ARCHITECT

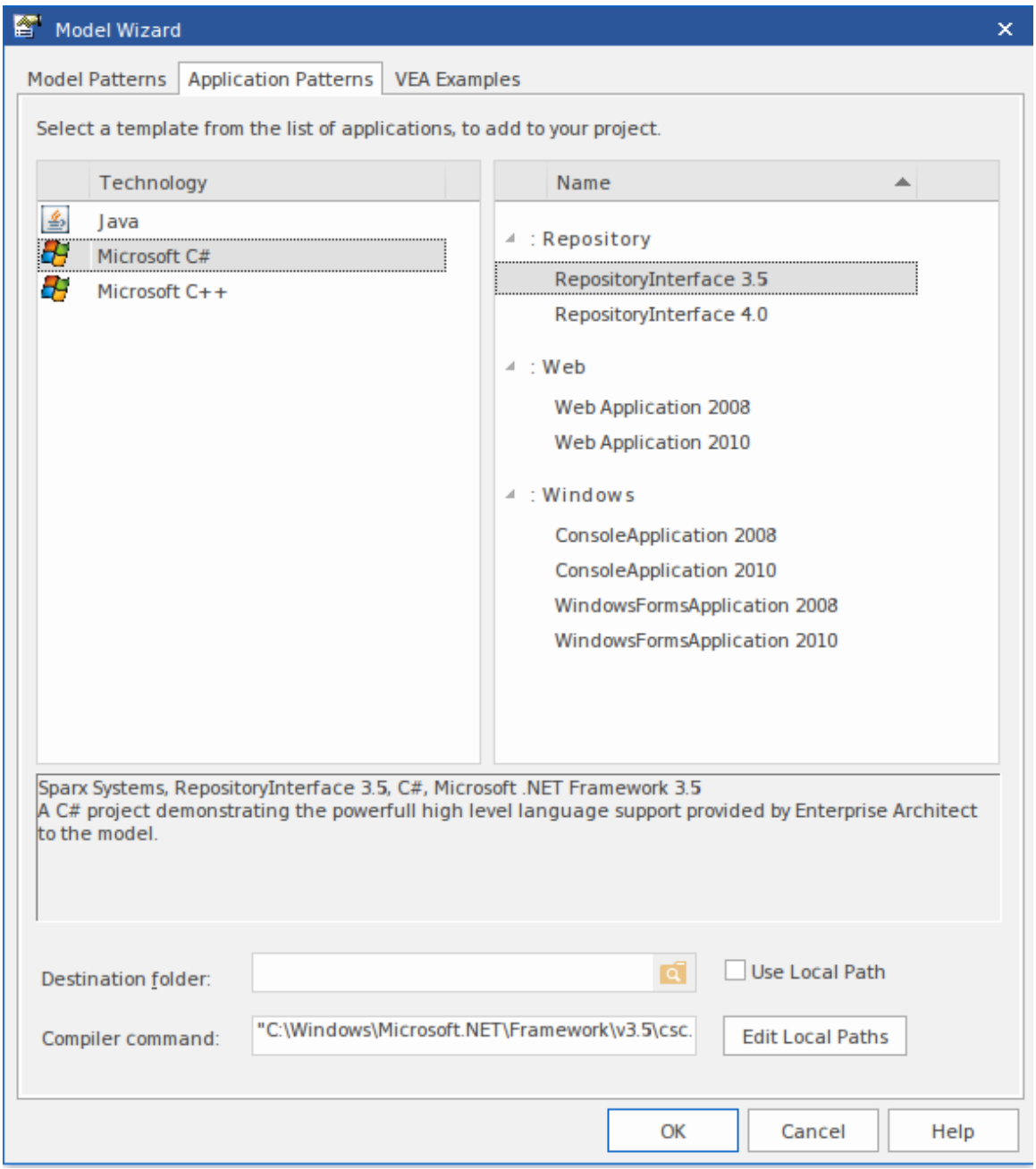
Table of Contents

Hybrid Scripting 3
C# Example 5
Java Example 7

Hybrid Scripting

Hybrid scripting extends the capabilities of the standard scripting environment to high level languages such as Java and C#. Hybrid scripting provides a speed advantage over conventional scripting, and also allows script authors to leverage existing skills in popular programming languages.

Access



Ribbon	Design > Model > Add > Insert > Model Wizard > Application Patterns
Context Menu	Right-click on Package | Add a Model using Wizard | Application Patterns

Keyboard Shortcuts	Ctrl+Shift+M | Application Patterns
Other	Browser window header bar menu | New Model from Pattern | Application Patterns

Features

- Superior execution speed
- Enhanced interoperability
- Full Visual Execution Analyzer support

C# Example

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.

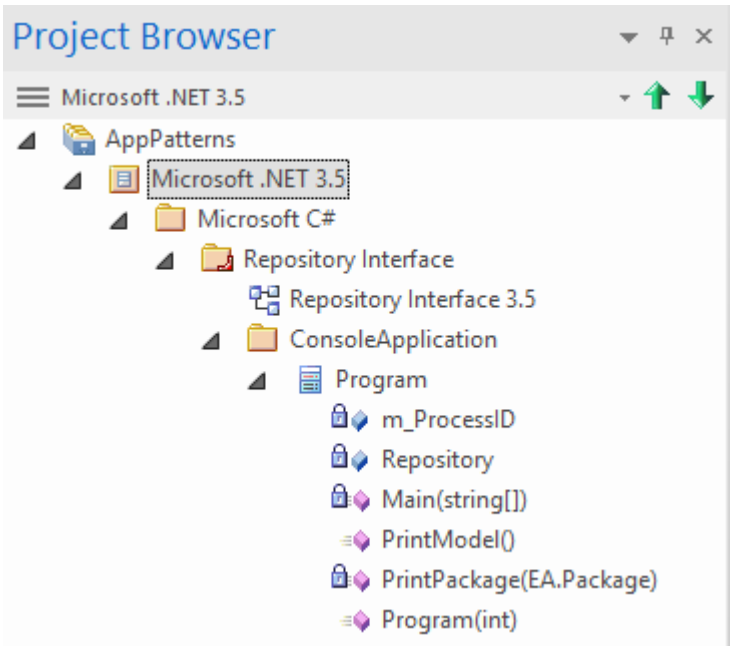
Create the project

In the Browser window, select the Package in which to create the template and then use the ribbon or context menu to display the Model Wizard; select the 'Application Patterns' tab.

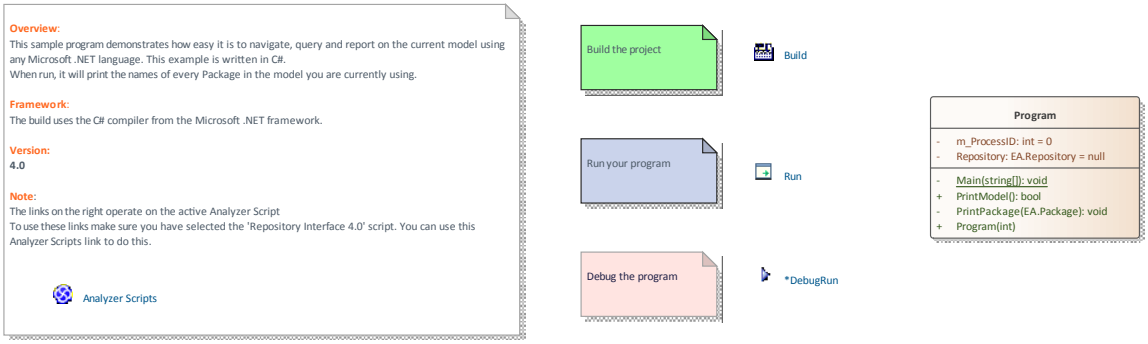
On the 'Application Patterns' tab, select the *Microsoft C# > RepositoryInterface* template. (You can choose from either the 3.5 or the 4.0 framework versions.) Specify the destination folder on the file system where the project template will be created, and click on the OK button.

Open the project

A Package structure similar to this will be created for you.



Expand the structure until you locate the *Repository Interface n.n* diagram and open it.



Build the script

The commands on this diagram will operate on the active build configuration. Before executing them, double-click on the *Analyzer Scripts* link and select the checkbox next to the 'Repository Interface' build configuration.

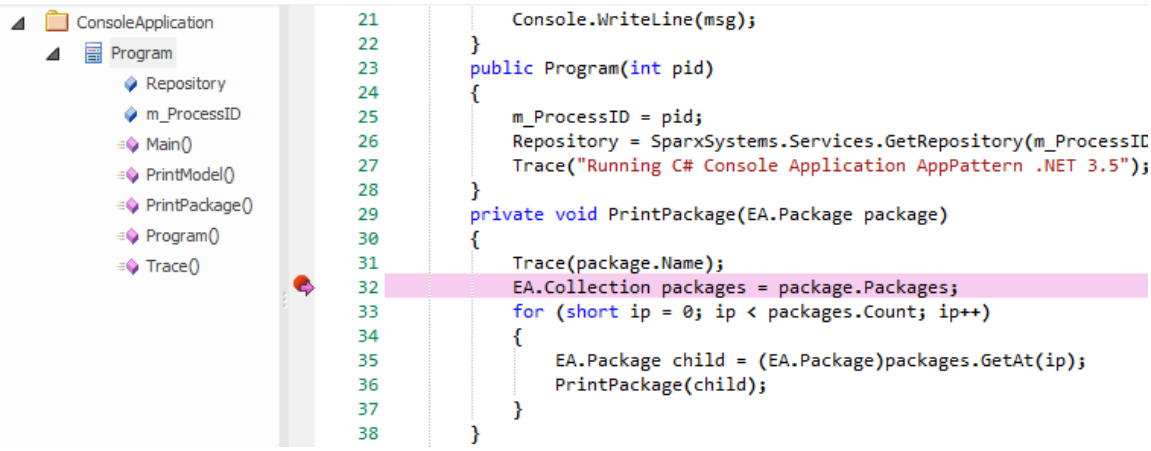
Run the script

Double-click on the *Run* link to open the Console. The Console will pause after completion so you can read the output from the program; this output will also be sent to the 'Script' tab of the System Output window. You can alter this by changing the code.

Debug the script

Select the 'Program' Class from the Browser window and press Ctrl+E to open the source code.

Place a breakpoint in one of the functions and then double-click on the *DebugRun* link. When the breakpoint is encountered, the line of code will become highlighted in the editor, as shown:



Java Example

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.

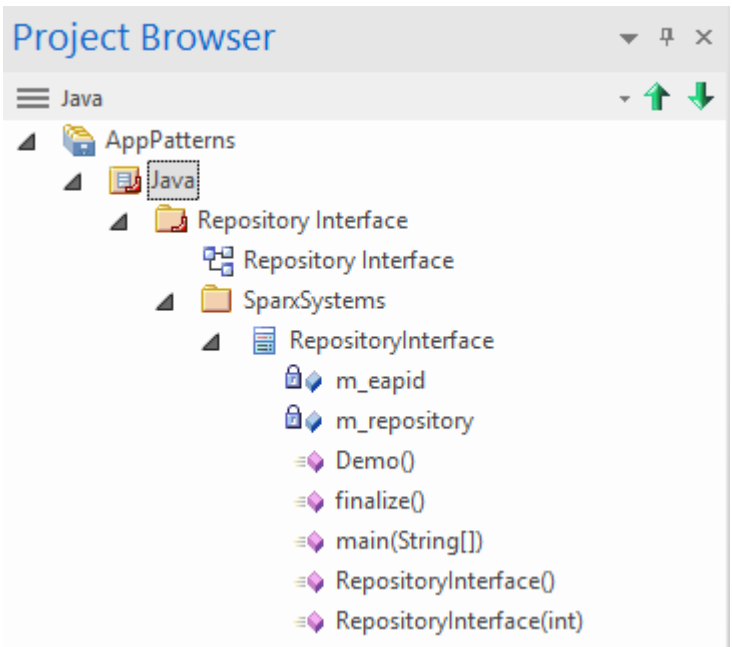
Create the project

In the Browser window, select the Package in which to create the template, then use the ribbon or context menu to display the Model Wizard; click on the 'Application Patterns' tab.

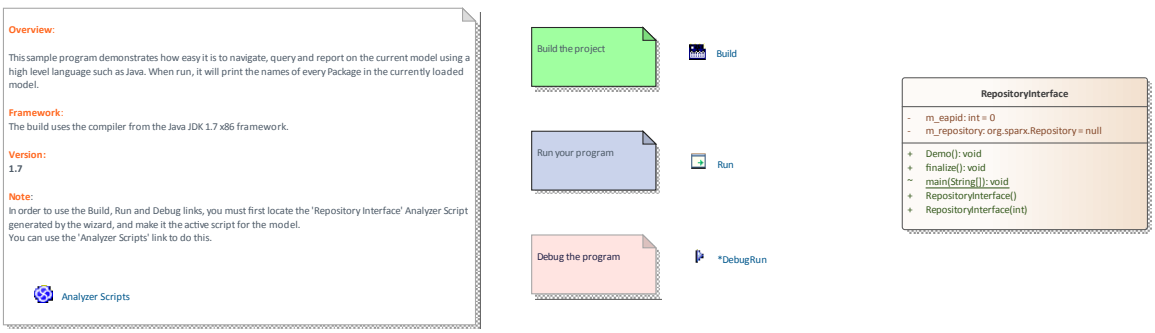
From this tab, select the *Java* > *RepositoryInterface* template. Specify the destination folder on the file system in which the project template will be created, and click on the OK button.

Open the project

A Package structure similar to this will be created for you.



Expand the structure until you locate the 'Repository Interface' diagram and open it.



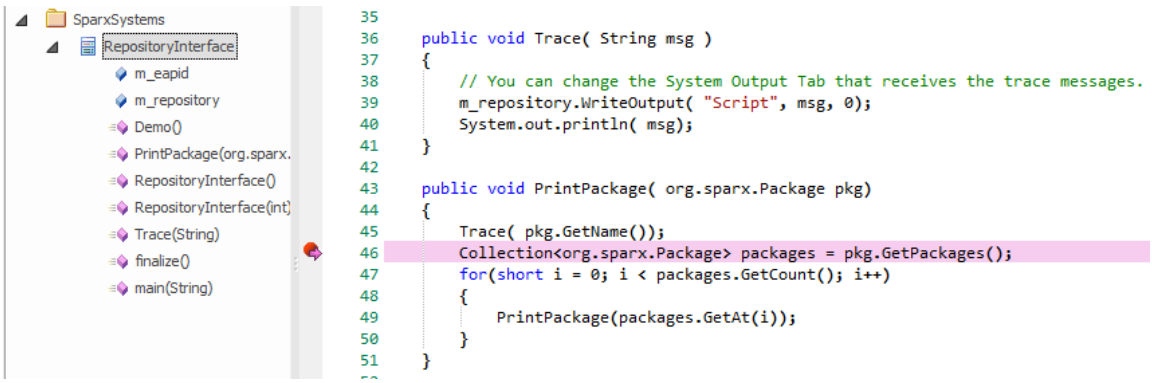
Build the script

The commands on the diagram will operate on the active build configuration. Before executing them, double-click on the *Analyzer Scripts* link and select the checkbox next to the 'Repository Interface' build configuration.

Run the script

Double-click on the *Run* link; a Console will open. The Console will pause after completion so you can read the output. The output from the program will also be output to the 'Script' tab of the System Output window. You can alter this by changing the code.

Debug the script

Select the 'Program' Class from the Browser window and press Ctrl+E to open the source code.

Place a breakpoint in one of the functions and then double-click on the *DebugRun* link. When the breakpoint is encountered the line of code will become highlighted in the editor, as shown.