Hands-On Lab

Using the Architecture Explorer in Visual Studio 2010 Ultimate to Analyze Your Code

Lab version: 1.0.0
Last updated: 12/10/2010
CONTENTS

OVERVIEW...........................................................................................................................................3

EXERCISE 1: INTRODUCTION TO ARCHITECTURE EXPLORER.........................................................4

EXERCISE 2: ANALYZING PROJECT STRUCTURE USING ARCHITECTURE EXPLORER AND DGML ........................................................................................................................................13

EXERCISE 3: ANALYZING EXTERNAL ASSEMBLIES USING ARCHITECTURE EXPLORER ..........24
Overview

The Visual Studio 2010 Ultimate Architecture Explorer provides an easy way to drill down into your code and analyze relationships between projects, types, their assemblies and references. In this lab, you'll see how you can use the Architecture Explorer to examine an existing solution and create dynamic DGML diagrams.

System Requirements

In order to complete this lab you will need the Visual Studio 2010 RC virtual machine provided by Microsoft. For more information on acquiring and using this virtual machine, please see “Working with the Visual Studio 2010 Virtual Machine”.

Prerequisites

It is recommended that you complete the following labs before exploring this lab:

- Code Discovery using Visual Studio 2010 Ultimate Architecture Tools
- Understand Class Coupling with Visual Studio 2010 Ultimate

Exercises

This Hands-On Lab comprises the following exercises:

1. Introduction to Architecture Explorer
2. Analyzing Project Structure using Architecture Explorer and DGML
3. Analyzing External Assemblies using Architecture Explorer

Estimated time to complete this lab: 60 minutes.

Next Step

Exercise 1: Introduction to Architecture Explorer
Exercise 1: Introduction to Architecture Explorer

In this exercise, you will learn how to use the Architecture Explorer tool to navigate project source structure and to generate DGML diagrams that help illuminate subsets of that chosen structure.

1. Log in as Abu Obeida Bakhach (Dev) if you have not already done so. The password is P2ssw0rd (capital letter P, the number two, the letter s, the letter s, the letter w, the number zero, the letter r, and the letter d). Please see “Working with the Visual Studio 2010 Virtual Machine” for instructions on how to log into the VM.


3. In Source Control Explorer (View | Other Windows | Source Control Explorer), navigate to Tailspin Toys | Development | Iteration 2 and double-click on the TailspinToys.sln file to open the Tailspin Toys solution.

4. Rebuild the solution (Build | Rebuild Solution from the main menu). This step may take a few minutes to complete.

5. Close the Output window after the build has successfully completed.

6. Load Architecture Explorer (View | Architecture Explorer). Architecture Explorer allows you to navigate the types and methods from code starting from either a class view or a solution view.

7. Select the Class View option from Architecture Explorer to view all of the namespaces that are part of the TailspinToys solution. From this point on, clicking on an item from the far right pane will allow you to drill down further into types and members.
8. Scroll down to find the **Tailspin.Admin.App** namespace and select it to load its types.

9. Select the **ProductRelationship** class type to load its members.
10. Select the **Product** property member to see that it contains two methods.

11. Select the **get_Product** method.

**Note:** This step illustrates that Architecture Explorer is operating on compiled code using reflection. As you will see later on in this lab, you can explore compiled .NET code directly from compiled assemblies.
12. Each level we drill down into starting with Types has a vertical bar that exposes filtering, grouping, and other commands. Select the vertical **Calls** bar to expose these options.

![Diagram of Node Navigation and Inbound Navigation](image.png)

**Figure 6**
*Selecting get_Product exposing vertical Calls bar*

13. Select the **Return Types** option found under Outbound Navigation in order to show the return type for get_Product.

![Diagram of Outbound Navigation and Analysis](image.png)

**Figure 7**
*Options available when working with type and member nodes*

**Note:** The figure above is not an exact representation of what you will see. You will have to scroll through to see all of these options.
14. Select the **Product** return type to display its members. Note that we are now navigating the Product type, but we still have the historical steps of our navigation to look through also.

15. Select the vertical **Members** bar immediately to the left of the Product members list and scroll down to the **Inbound Navigation** section.

16. Select the **All Inbound** navigation option to see all calls that reference the Product type.
17. Although this may seem redundant from a navigation perspective, select the **Product** class type from the list of inbound references. Once again, you will see `get_Product` and `set_Product` listed.

18. Select the **Create New Graph** button from the left side of Architecture Explorer. This will create a new DGML diagram showing the entire navigation path that you manually took up until this point.
Since this is a DGML diagram, we can use all of the standard tools to view it and add or remove nodes. Select the `TailspinToys` node and press the `Delete` key to remove it from the diagram.

Return to Architecture Explorer and find the `set_Product` method that is contained within the `Product` class. Drag and drop `set_Product` onto the DGML diagram.
Adding set_Product to the DGML diagram

**Note:** You can view the source for a type by right clicking on it and selecting View Content.

21. Clear the selected columns in Architecture Explorer by selecting the **Clear Columns** button.

---

**Figure 14**

*Adding set_Product to the DGML diagram*

**Figure 15**

*Clear Columns button location*
Next Step

Exercise 2: Analyzing Project Structure using Architecture Explorer and DGML
Exercise 2: Analyzing Project Structure using Architecture Explorer and DGML

In this exercise, you will learn more about the features of Architecture Explorer from the Solution View. This includes more exposure to node navigation, filtering support, and a sample scenario that demonstrates the combined power of Architecture Explorer and DGML diagrams to aid in architectural analysis.

1. Select the **Solution View** option from Architecture Explorer to view all projects that are part of the TailspinToys solution.

![Figure 16](image)
*Solution view showing projects from the TailspinToys solution*

2. Select the **Tailspin.Web** project to see a flat view of its files.

![Figure 17](image)
*Viewing all files for the Tailspin.Web project*
3. At the top of the source files list is a filter box. Enter ".cs" into the filter box and press Enter in order to restrict display to files that have the .cs extension.

![Figure 18](Image)

*Filtering source files*

4. Press Ctrl + A to select all of the .cs files and select the **Create New Graph** button.

![Figure 19](Image)

*DGML diagram showing source files from Tailspin.Web project*

Note: If Ctrl + A does not work, make sure that the list you are trying to select currently has focus by first clicking somewhere within that list.

5. **Close** the new graph and return to Architecture Explorer.
6. Select the Tailspin.Web project again to return to an unfiltered view of the files.

7. Select the vertical Files bar and select the Classes navigation option to view all of the classes from the Tailspin.Web project.

![Diagram showing tailspin.web classes](image)

**Figure 20**
Viewing Tailspin.Web classes

8. Press Ctrl + A to select all of the classes and select the Create New Graph button.

![DGML diagram showing all classes from Tailspin.Web project](image)

**Figure 21**
DGML diagram showing all classes from Tailspin.Web project

9. Close the new graph and return to Architecture Explorer.

10. Clear the selected columns in Architecture Explorer by selecting the Clear Columns button.

11. Select the Solution View, navigate to the Tailspin.Web project once again and select the References option from the vertical Files bar. This will show us all of the referenced assemblies used by this project.
12. Select the **Filter** button at the top of the referenced assemblies list and select the check box for **Project**. This will filter out all references that are external to the TailspinToys solution.

13. Select the **OK** button to accept the filter changes.

14. **Select all** of the filtered project references and select the **Create New Graph** button.

15. **Delete** the **TailspinToys** node from the diagram.
16. Now we can explore the referential relationship between Tailspin.SimpleSqlRepository and the other projects in the diagram. In Architecture Explorer, select just the Tailspin.SimpleSqlRepository item and change the Files viewing options to References.

**Figure 25**
Listing references for the Tailspin.SimpleSqlRepository project

17. Select the Tailspin.Infrastructure and Tailspin.Model items and then drag and drop them onto the existing DGML diagram.
Figure 26
Updated DGML diagram showing outgoing references from Tailspin.SimpleSqlRepository project

18. Now we can create a better view of how the Product related classes for these projects fit into this picture. Return to the Architecture Explorer and select the three project references from the Tailspin.Web project once again.

19. Select the vertical Files bar and select the Classes option to view all classes from the selected projects.

20. In the Filter box, enter “Product” and press Enter to see all classes that contain that term.

21. Select all the filtered classes, then drag and drop them onto the existing DGML diagram.
22. Select the **ProductList** class from the DGML diagram and select the **Butterfly Mode** button from the Directed Graph toolbar. This removes nodes that do not directly reference ProductList.

23. Select the **Butterfly Tree** button once again to turn it off.

24. Select the **Quick Clusters** button and zoom to fit the entire graph to the available screen space. Note that it is now easy to see that the Tailspin.SimpleSqlRepository project and its Product related classes are responsible for data access.
25. Now we can see how the Tailspin administrator application fits into the overall architecture that we have looked at so far. Clear the selected columns in Architecture Explorer by selecting the **Clear Columns** button.

26. Select the **Solution View** and select **Tailspin.Admin.App**.

27. Select the vertical **Files** bar and select the **References** option. Note that there are no references to any of the Tailspin projects. There are references to System.Data.dll and System.Data.Linq.dll, so perhaps the admin application is going directly to the database.

28. Now we will investigate the admin application further to see if there is truly cause for concern. Select the vertical **Files** bar and select the **Classes** option.

29. **Select all** classes that start with **Product**.
30. Select the **Create New Graph** button.

31. **Delete** the **TailspinToys** solution node from the diagram.

32. Note that there are no outbound calls to the Tailspin.SimpleSqlRepository data access code. **Right-click** somewhere within the current DGML diagram and select the **Save As XPS...** option. We are going to create a work item and assign it to someone to investigate further.
33. Name the XPS file “AdminAppReview” and select the Save button.

34. In Team Explorer, right-click on the Work Items node for the TailspinToys project and select the New Work Item | Task option.

35. Fill in the following fields as shown in the table below:

<table>
<thead>
<tr>
<th>Work Item Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title</td>
<td>Review why Admin app is not using the Data Access Layer project</td>
</tr>
<tr>
<td>Assigned To</td>
<td>Michael Affronti (PM)</td>
</tr>
<tr>
<td>Stack Rank</td>
<td>1</td>
</tr>
<tr>
<td>Details Description</td>
<td>Take a look at the attached diagram. Notice how we’re using Product data but not getting it via the Tailspin data access project.</td>
</tr>
</tbody>
</table>

36. Select the Attachments tab and click the Add button.
37. Select the **Browse** button and select the XPS file that you saved.

38. In the **Comment** field, enter “**Snapshot of current Admin app.**”

39. Select the **OK** button to add the attachment to the new work item.

40. Select the **Save Work Item** button to finish the process of creating the new work item.

![Figure 36](image)
*Save Work Item button location*

41. Close all open windows in Visual Studio including the new work item and existing graphs. You do not need to save the existing graphs.

42. Clear the selected columns in Architecture Explorer by selecting the **Clear Columns** button.

---

**Next Step**

*Exercise 3: Analyzing External Assemblies Using Architecture Explorer*
Exercise 3: Analyzing External Assemblies Using Architecture Explorer

In this exercise, you will learn how to load and navigate external .NET assemblies. This can help you gain insight into the architecture of frameworks or applications for which you do not have the original project or source files.

1. Select the **Select Files** option from Architecture Explorer. This allows you to load compiled .NET assemblies and analyze them as you already have done with the Class and Solution Views.

2. In the **File Name** box, enter “c:\windows\microsoft.net\framework\v4.0.30128” and press the **Enter** key.

   **Note:** If the v4.0.30128 directory is not available, select the most recent one available.

3. In the **File Name** box, enter “System.Data.dll” and select the **Open** button to continue.

4. Select the **System.Data.dll** file to view its types.
5. Select the **Filter** button at the top of the types list and select the check box for **Class** and the check box for **IsPublic**.

6. Press the **OK** button to see the list of public classes in the System.Data.dll assembly.

7. Enter “**Sql**” into the filter box at the top of the current type list and press Enter so that we can view types that contain that text.
8. **Select all** of the listed types and press the **Create New Graph** button.
To give feedback please write to VSKitFdbk@Microsoft.com

Copyright © 2010 by Microsoft Corporation. All rights reserved.