

Hands-On Lab

Understanding Class Coupling with Visual Studio 2010 Ultimate

Lab version: 1.0.0

Last updated: 12/14/2010



Contents

[Overview 3](#_Toc261342346)

[Exercise 1: Introduction to Class Dependency Graphs 4](#_Toc261342347)

[Exercise 2: Discovering Circular References 14](#_Toc261342348)

Overview

* 1. Understanding the relationship between classes on a new code project can be difficult. In this lab, you'll use the new DGML diagrams in Visual Studio 2010 Ultimate to drill down into an existing code base and figure out how types are related.

# System Requirements

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

# Prerequisites

It is recommended that you complete the “Code Discovery using Visual Studio 2010 Ultimate Architecture Tools” lab before exploring this lab.

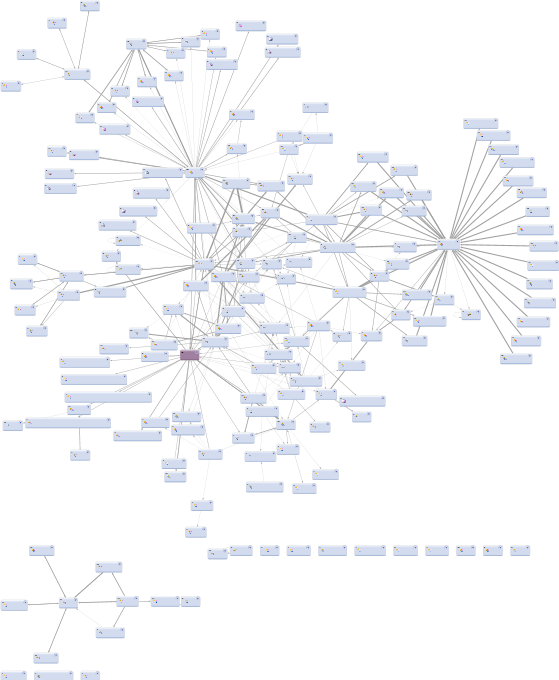
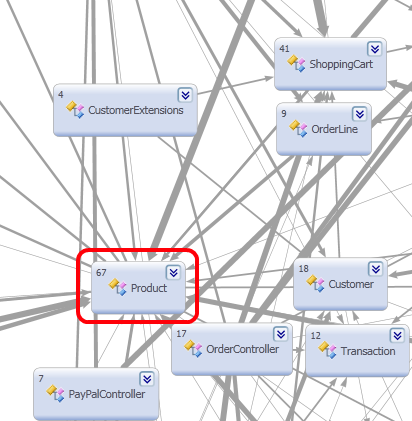
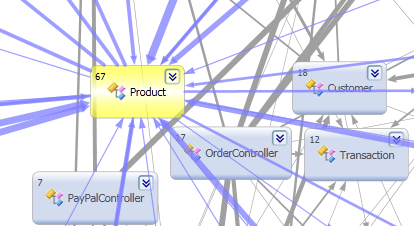
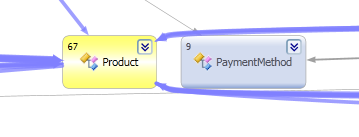
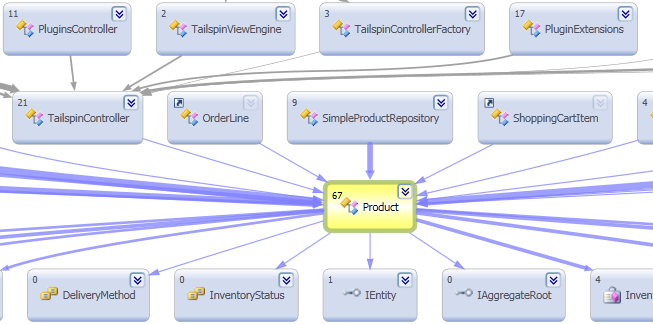
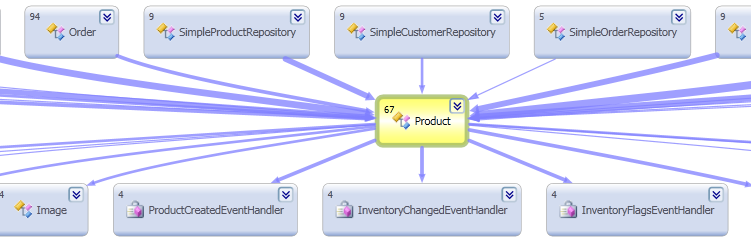
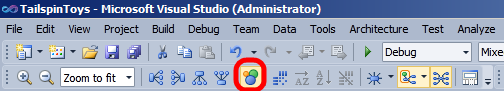
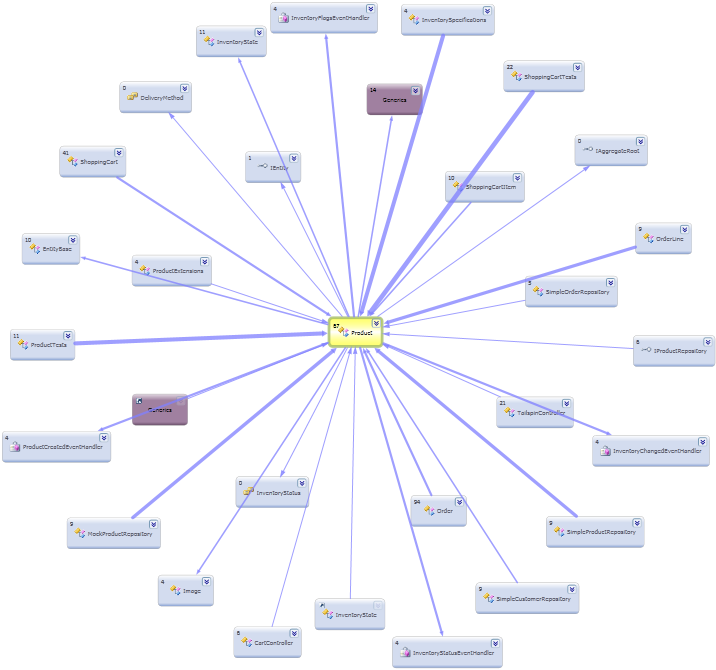
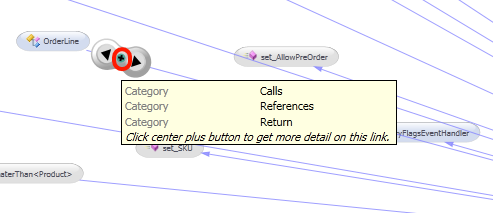
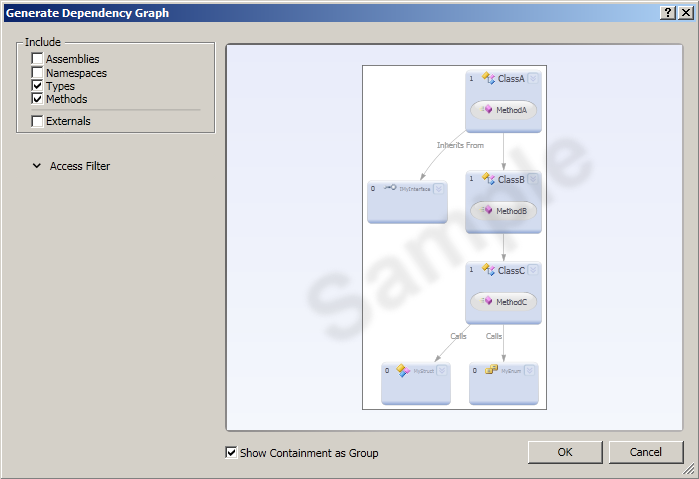
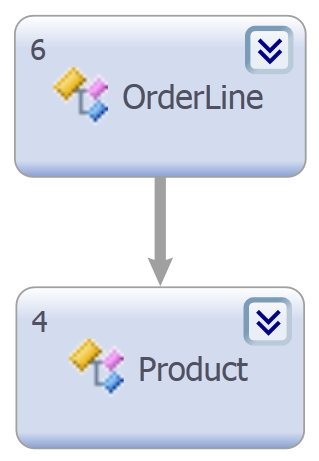
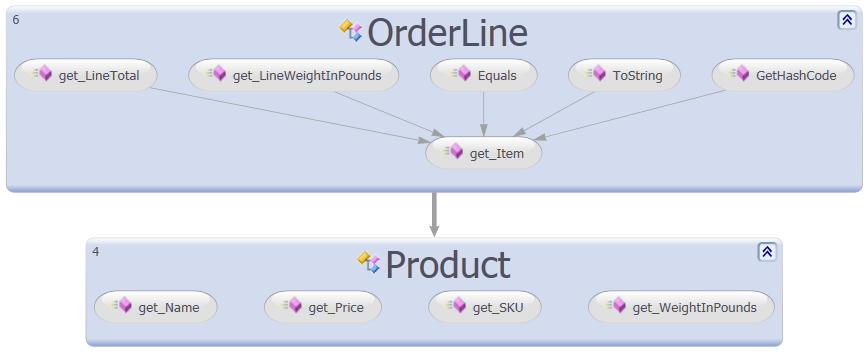
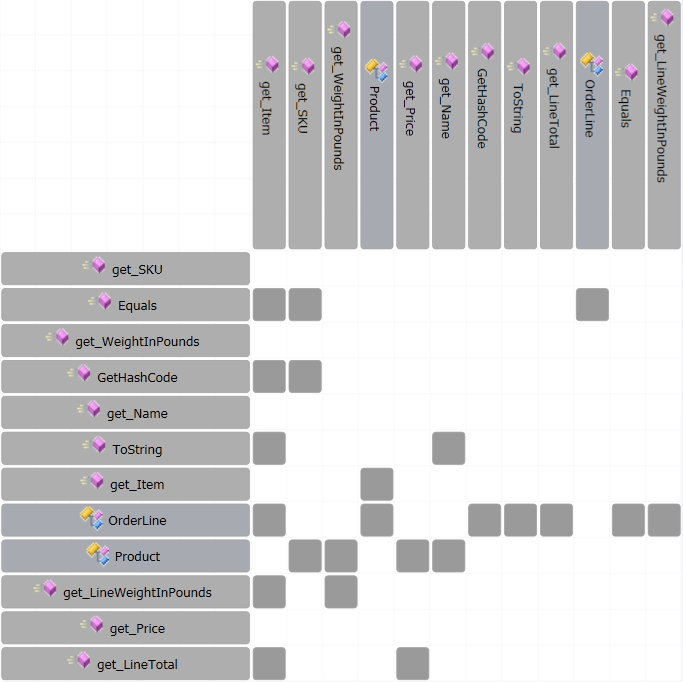
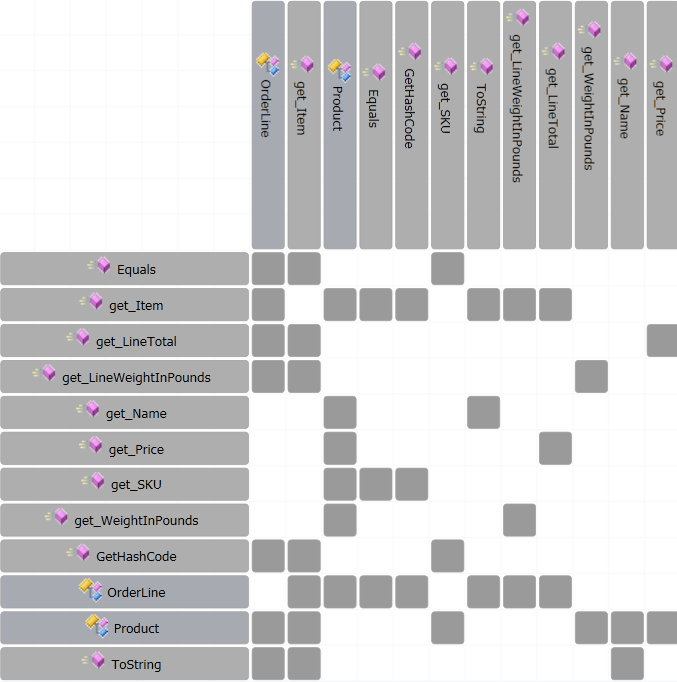
# Exercises

* 1. This Hands-On Lab comprises the following exercises:
  2. Introduction to Class Dependency Graphs
  3. Discovering Circular References
  4. Estimated time to complete this lab: 3**0 minutes**.

# Next Step

Exercise 1: Introduction to Class Dependency Graphs

Exercise 1: Introduction to Class Dependency Graphs

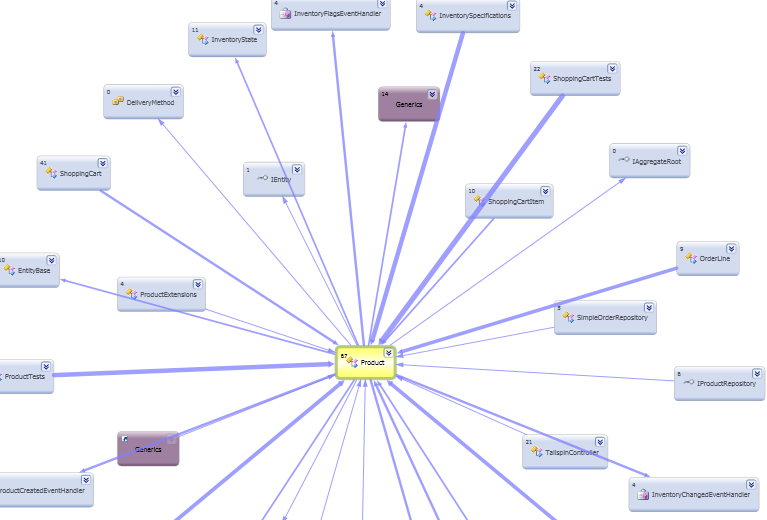
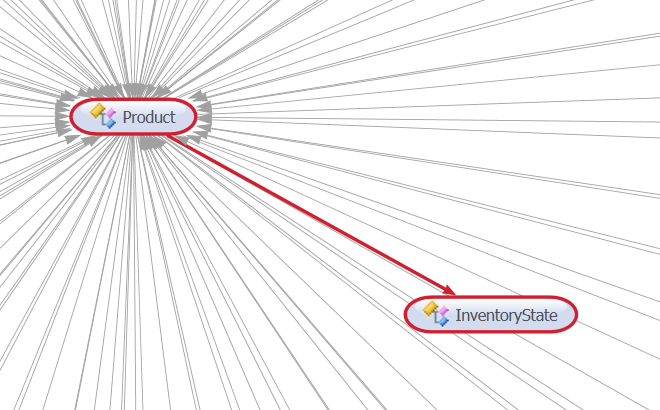
1. In this exercise, you will learn how to generate and navigate a class dependency graph.
   1. Log in as **Abu Obeida Bakhach (Dev)** if you have not already done so. The password is P2ssw0rd (capital letter P, the number two, the letter s, the letter s, the letter w, the number zero, the letter r, and the letter d). Please see “Working with the Visual Studio 2010 RTM Virtual Machine” for instructions on how to log into the VM.
   2. Open Microsoft Visual Studio from **Start | All Programs | Microsoft Visual Studio 2010 | Microsoft Visual Studio 2010**.
   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. Create a new class dependency graph (**Architecture | Generate Dependency Graph | By Class**). The class dependency graph is stored in a **Directed Graph Markup Language** format (hereafter referred to as **DGML**), which allows you to work with it using Visual Studio 2010 as well as other standard tools.
      1. **Note:** Generating the class dependency graph for all projects in the solution may take a minute to complete.
   7. Select the **Quick Clusters** button from the Directed Graph toolbar to view all classes and their relationships at a glance.
      1. 
      2. Figure 1
      3. Quick Clusters layout button location
      4. 
      5. Figure 2
      6. Quick Clusters view of generated class dependency graph
      7. **Note:** The generated dependency graph views that you see may be different from the screenshots shown in this lab manual. You may need to perform additional zooming, scrolling, and visually searching for objects specified in the lab steps.
   8. The Quick Clusters layout view of this class dependency graph only gives us an idea of the number of classes in all TailspinToys projects and their degree of dependency between each other. This is of limited usefulness to us, so **zoom** in towards the center of the graph by holding down the **Ctrl** key and using the mouse scroll wheel until you can clearly see the **Product** class.
      1. **Note:** Zooming can also be done using the Zoom drop down box in the Directed Graph toolbar, or alternatively by using the **+** and **–** magnifying glass icons next to the Zoom drop down.
      2. 
      3. Figure 3
      4. Close up view of class dependency graph showing the Product class
      5. **Note:** Gray lines of varying thicknesses represent the magnitude of relationship interdependencies between classes, with thicker lines equating to more relationships.
   9. Select the **Product** class by left-clicking on it to highlight only the direct relationships (purple color) between the Product class and the other classes represented in the graph.
      1. 
      2. Figure 4
      3. Product class selected showing direct relationship lines
   10. Selecting the Product class node from the graph helps highlight its direct relationship to other classes, but it is still difficult to distinguish between incoming and outgoing dependency relationships. To help visually make this distinction, change the graph layout by selecting the **Top to Bottom** button from the Directed Graph toolbar.
       1. 
       2. Figure 5
       3. Top to Bottom layout button location
       4. 
       5. Figure 6
       6. Result of Top to Bottom layout
   11. The top to bottom layout shows incoming dependency relationship lines on the top and outgoing lines on the bottom. This is an improvement, but there is still a lot of visual noise from dependency relationships between the other classes (shown in gray). Select the **Butterfly Mode** button to hide the relationship lines that do not live on a path through the selected Product node.
       1. 
       2. Figure 7
       3. Butterfly Mode button location
       4. 
       5. Figure 8
       6. Butterfly Mode view
   12. We can simplify the visualization of dependency relationships and classes even further by choosing the depth to branch out. Select the **Neighborhood Browse Mode** button from the Directed Graph toolbar and select the **1 Link** option.
       1. 
       2. Figure 9
       3. Neighborhood Browse Mode button location
       4. 
       5. Figure 10
       6. Result of Neighborhood Browse Mode with 1 Link
   13. Select the **Quick Clusters** layout button to return to the clusters layout view.
       1. 
       2. Figure 11
       3. Quick Clusters layout button location
   14. Zoom out as necessary to fit the entire graph into view.
       1. 
       2. Figure 12
       3. Quick Clusters view showing direct dependencies between Product and other classes
   15. Locate the **OrderLine** class node and hover the mouse cursor over the purple relationship line that connects to the **Product** class node. When the navigation control appears, click on the **+** button to see the details of this relationship. This will start the process of generating a new dependency graph.
       1. 
       2. Figure 13
       3. Location of + button to create a new graph
       4. **Note:** The navigation control that appears when hovering over a purple relationship line exposes three actions. The two arrows navigate to either the source or the target node of the relationship (depending upon context). The + button will help you generate a new graph with just the source and target nodes.
   16. In the **Generate Dependency Graph** window, only include **Types** and **Methods**. We will leave Assemblies, Namespaces, and Externals out of the new graph. Select the option to **Show Containment as group** and select the **OK** button to generate the new graph.
       1. 
       2. Figure 14
       3. Generate Dependency Graph options
       4. 
       5. Figure 15
       6. Initial view of generated dependency graph
   17. Let’s expand on this initial view by right-clicking and selecting **Group | Expand All**.
   18. Select the **Top to Bottom** layout button.
       1. 
       2. Figure 16
       3. Expanded dependency graph
   19. So far, the only additional thing that we have learned about the relationship between these two classes is the methods that are involved. To see more details about the relationship between the methods, select the **Dependency Matrix View** button.
       1. 
       2. Figure 17
       3. Dependency Matrix View button location
       4. 
       5. Figure 18
       6. Dependency matrix view
   20. The dependency matrix view allows sorting rows or columns by display name. Select the **Sort Rows by Display Name** button just to the right of the Dependency Matrix View button.
   21. The dependency matrix view currently shows the relationship going from the OrderLine class to the Product class, as you can see from the arrow direction in Figure 15.
   22. Another option that you have is to use the Reflexive view in order to see the relationships regardless of the direction. Select the **Show Reflexive View** button.
       1. 
       2. Figure 19
       3. Show Reflexive View button location
       4. 
       5. Figure 20
       6. Reflexive view of dependency matrix

# Next Step

Exercise 2: Discovering Circular References

Exercise 2: Discovering Circular References

In this exercise, you will learn how to use the circular reference analyzer in order to discover classes that are strongly coupled to each other.

* 1. Return to the original class dependencies graph (**ClassDependencies1.dgml**) and make sure that the **Product** class node is selected.
     1. 
     2. Figure 21
     3. Class dependency graph with Product selected
  2. Right-click somewhere within the dependency graph and select **Analyzers | Circular References** to identify types that are strongly coupled.
  3. De-select the Product node by clicking on a blank spot within the graph. Note that there are red outlines around types that are strongly coupled to the Product class.
     1. 
     2. Figure 22
     3. Class dependency graph showing circular references between Product and other types
     4. **Note:** Red relationship lines indicate direct circular references whereas the normal gray relationship lines indicate indirect circular references (there is at least one other type in-between).

To give feedback please write to [VSKitFdbk@Microsoft.com](mailto:VSKitFdbk@Microsoft.com)

Copyright © 2010 by Microsoft Corporation. All rights reserved.