

Hands-On Lab

Using Code Analysis with Visual Studio 2010 to Improve Code Quality

Lab version: 1.0.0

Last updated: 12/15/2010



Contents

[Overview 3](#_Toc265504314)

[Exercise 1: Introduction to Code Analysis 4](#_Toc265504315)

[Exercise 2: Suppressing Code Analysis Warnings 8](#_Toc265504316)

Overview

* 1. The Code Analysis feature of Visual Studio performs static code analysis on code to help developers identify potential design, globalization, interoperability, performance, security, and a host of other categories of potential problems. In Visual Studio 2010, improvements have been made to the code analysis engine itself, as well as ability to configure rule sets rather than picking and choosing from one flat list of rules. Code Analysis can be run manually at any time from within the Visual Studio IDE, or even setup to automatically run as part of a Team Build or check-in policy for TFS.
  2. In this lab, you will be introduced to Code Analysis, how to configure rules sets to use, and finally how to suppress specific rules at a project and source code level.
  3. **Note:** Code Analysis can be found in the Premium and Ultimate editions of Visual Studio 2010.

# 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”.

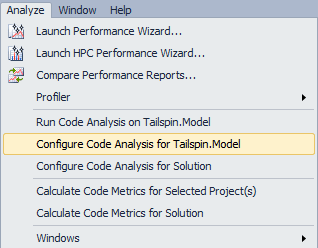
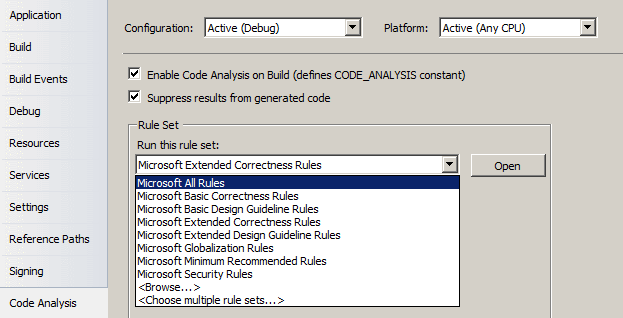
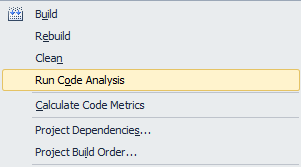
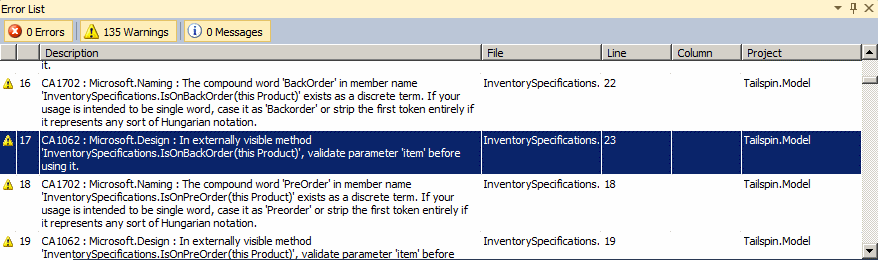
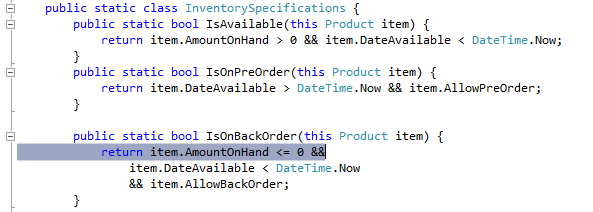
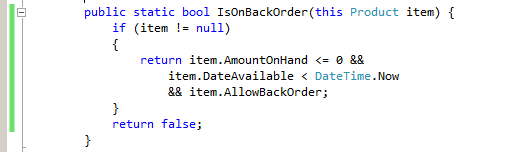
# Exercises

* 1. This Hands-On Lab comprises the following exercises:
  2. Introduction to Code Analysis
  3. Suppressing Code Analysis Warnings
  4. Estimated time to complete this lab: 3**0 minutes**.

# Next Step

Exercise 1: Introduction to Code Analysis

Exercise 1: Introduction to Code Analysis

1. In this exercise, you will learn about new Code Analysis features in Visual Studio 2010 by configuring the rule set used, performing code analysis on a sample project, and addressing some of the warnings that are raised.
   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. In Solution Explorer, left-click on the **Tailspin.Model** project node and select **Analyze | Configure Code Analysis for Tailspin.Model** from the main menu, which will load the project properties and go to the **Code Analysis** tab.
      1. 
      2. Figure
      3. Configuring Code Analysis for a project
      4. **Note:** The Code Analysis tab is new to Visual Studio 2010, and it allows you to choose from sets of rules rather than picking and choosing from one flat list of rules, as was the case in previous versions.
   6. Select the “**Microsoft All Rules**” option for the **Rule Set** to select the comprehensive set of rules.
      1. 
      2. Figure
      3. Configuring Code Analysis rule set to use
   7. In Solution Explorer, right-click on the **Tailspin.Model** project node and select **Run Code Analysis** from the context menu that appears.
      1. 
      2. Figure
      3. Location of contextual Run Code Analysis command
   8. The Code Analysis feature runs through static code analysis rules as defined by Microsoft and displays the results as warnings. Scroll through the list of warnings and read a few of them.
      1. 
      2. Figure
      3. Code Analysis warnings
      4. **Note:** Depending on the version of Tailspin Toys that you are running, you may see more or less warnings than depicted in screenshots.
      5. **Note:** Code Analysis rules can also be configured to show up as errors if desired.
   9. The warnings produced by Code Analysis provide a wealth of information including a unique category ID (such as CA1062 in screenshot above), a group name (Microsoft.Design in screenshot above), and a brief description of the problem or suggested fix, and the file location of the offending code.
      1. **Note:** Pressing F1 while selecting one of the warnings will take you more descriptive help online. This may not be available to you within the virtual environment.
   10. Find a warning that looks like it will be easy to fix quickly, such as the warning for **CA1062** in the screenshot above, and double-click on it to load the location in code.
       1. 
       2. Figure
       3. Code Analysis errors are linked to source code
   11. Perform the code fix necessary to resolve the warning. For CA1062, we are told that we should validate parameter ‘item’ before using it. One potential fix can be seen in the following screenshot.
       1. 
       2. Figure

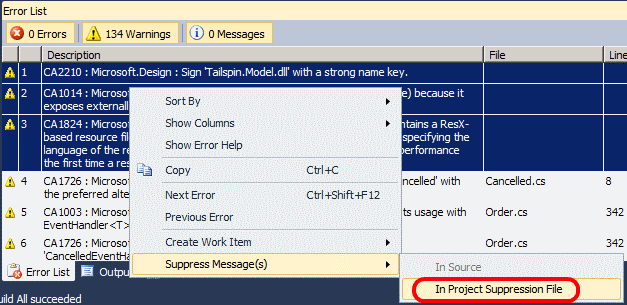
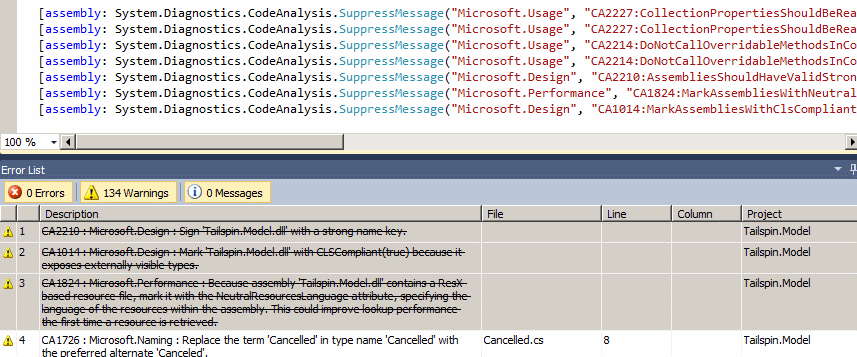
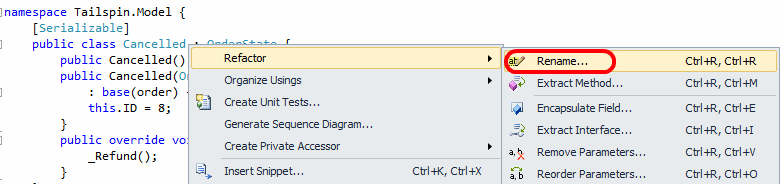
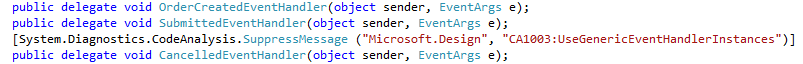
Validating input parameter before use resolves CA1062

* 1. Select **Analyze | Run Code Analysis on Tailspin.Model** from the main menu and verify that the warning disappears. There should be one less warning than before.

# Next Step

Exercise 2: Suppressing Code Analysis Warnings

Exercise 2: Suppressing Code Analysis Warnings

* 1. In this exercise, you will learn how to suppress Code Analysis warnings at the project and source level.
  2. In the **Error List** window, select the first three warnings that appear which are associated with the project. Note that they are not associated with a file by looking in the File column. Imagine that we do not want to address the selected issues and no longer want them to appear when Code Analysis executes.
  3. Right-click on the selected warnings and select **Supress Message(s) | In Project Suppression File** from the context menu that appears. This will add assembly level metadata to a project level GlobalSuppressions.cs file.
     1. 
     2. Figure
     3. Suppressing specific code analysis rules at the project level
  4. Open the **GlobalSuppressions.cs** file to view the added code.
     1. 
     2. Figure
     3. Suppressed rules are crossed out and GlobalSuppressions.cs changes
  5. Take a look at the next Code Analysis warning that is listed, **CA1726**, which suggests changing the type name ‘Cancelled’ with the preferred alternate ‘Canceled’. **Double-click** on it to go to the correct source location for the fix.
  6. Right-click on the class name ‘Cancelled’ and select **Refactor | Rename…** from the context menu that appears.
     1. 
     2. Figure
     3. Renaming class to fix spelling
  7. In the Rename window, change ‘Cancelled’ to ‘Canceled’ and select the **OK** button to continue.
  8. In the Preview Changes – Rename window, review the proposed changes and select the **Apply** button to complete the refactoring process.
  9. Move on to the next Code Analysis warning by selecting it. Imagine that we want to suppress this specific rule, but this time we only want applied to this particular source file. Right-click on the warning and select **Suppress Message(s) | In Source** from the context menu that appears. This applies a **SuppressMessage** attribute to the delegate for which the Code Analysis rule was identifying.
     1. 
     2. Figure
     3. Suppressing a Code Analysis rule at the source code level
  10. Select **Analyze | Run Code Analysis on Tailspin.Model** from the main menu and verify that more of the warnings have been successfully addressed.
  11. At this point, there are additional Code Analysis warnings that we could address, but imagine that we simply want to ignore the remaining items for now. Return to the **Code Analysis** section of the project properties and select the ‘**Microsoft Minimum Recommended Rules**’ option.
  12. Select **Analyze | Run Code Analysis on Tailspin.Model** from the main menu and verify that the number of warnings has dramatically been reduced. The warnings that appear as the result of using this rule set are more likely to be problematic during runtime.

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

Copyright © 2010 by Microsoft Corporation. All rights reserved.