

Developing a List Definition and Event Receiver in Visual Studio 2010

Hands-On Lab

Lab Manual – SPCHOL300

SPCHOL300 – Building a List Definition and Event Receiver – C#

This document is provided “as-is”. Information and views expressed in this document, including URL and other Internet Web site references, may change without notice. You bear the risk of using it.

This document does not provide you with any legal rights to any intellectual property in any Microsoft product. You may copy and use this document for your internal, reference purposes.

© 2010 Microsoft. All rights reserved.

**Contents**

[Developing a List Definition and Event Receiver in Visual Studio 2010 2](#_Toc259797226)

[Lab Objective 2](#_Toc259797227)

[Additional Resources 2](#_Toc259797228)

[Getting Started 2](#_Toc259797229)

[Logging in to the Virtual Machine 2](#_Toc259797230)

[Locations 2](#_Toc259797231)

[Lab Pre-requisites 3](#_Toc259797232)

[Copying code samples from Word document 3](#_Toc259797233)

[Code Snippets 4](#_Toc259797234)

[Exercise 1 – Create a SharePoint List Definition Project with List Instance 5](#_Toc259797235)

[Task 1 – Create a SharePoint List Definition Project 5](#_Toc259797236)

[Task 2 – Customize the SharePoint List Definition and List Instance 7](#_Toc259797237)

[Task 3 – Add an Event Receiver 21](#_Toc259797238)

[Exercise 2 – Using the Developer Dashboard to review performance of the event receiver 27](#_Toc259797239)

[Task 1 – Enable the Developer Dashboard 27](#_Toc259797240)

[Task 2 – Using the Developer Dashboard 28](#_Toc259797241)

[Lab Summary 30](#_Toc259797242)

# Developing a List Definition and Event Receiver in Visual Studio 2010

Estimated time to complete this lab: **30 minutes**

## Lab Objective

This lab demonstrates how to create a List Definition with a corresponding List Instance, and then attach an Event Receiver to the list to perform some tasks.

The objectives of this lab are to learn how to build List Definitions, List Instances, and Event Receivers using Visual Studio 2010 and how to use the Developer Dashboard in SharePoint 2010.

* Create a SharePoint List Definition Project with List Instance
* Customize the SharePoint List Definition and List Instance
* Add an Event Receiver
* Enable the Developer Dashboard
* Use the Developer Dashboard

## Additional Resources

This lab includes the following additional resources:

All resources are found in the directory \Supporting Files\SPCHOL300\Resources\CS

|  |  |  |
| --- | --- | --- |
| This Lab Manual | SPCHOL300\_Manual\_CS.docx | This document |
| Source Code | \Completed\CS\Ex1\End.sln | Completed lab source code. |
| Resources | \Resources\CS\Ex1 | Various resources used throughout this lab. |

## Getting Started

### Logging in to the Virtual Machine

Please log into the virtual machine as the following user:

**Username:** Administrator

**Password:** pass@word1

### Locations

This Hands-On Lab contains a number of additional resources in fixed locations. By default, it is assumed that the base HOL directory is ***C:\Content Packs\Packs\SharePoint 2010 Developer Labs 1.0\SUPPORTING FILES\SPCHOL300\Resources****.*

The default working folder for this lab is **C:\SPHOLS\SPCHOL300**.

### Lab Pre-requisites

Browse to base HOL directory ***Supporting Files\SPCHOL300\Resources***and execute the **optimize.ps1** PowerShell script:

1. Right click on **optimize.ps1** and select **Run with PowerShell**:

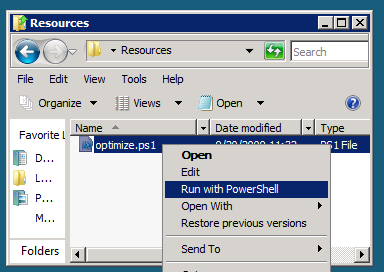


Figure 1 - Execute the PowerShell script

1. This will open a PowerShell window to execute the script. Please wait until the PowerShell script completes executing the script and closes the PowerShell window:

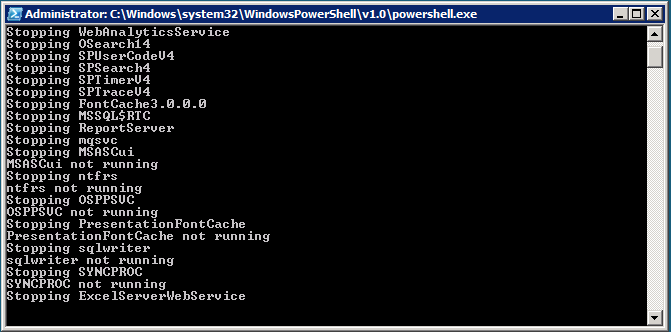


Figure 2 - PowerShell Window executing the script

### Copying code samples from Word document

Copying and pasting code from this Word document to Visual Studio is only safe for the sections of formatted code, e.g.:

Console.WriteLine("This is safe code!");

Code not in these sections may contain Unicode or other invisible characters that are not valid XML or C#/VB code, e.g.:

**Console.WriteLine(“This is NOT safe code !!”);**

### Code Snippets

You can also use Code Snippets to insert the appropriate code in the lab.

To use the required code snippet for this lab:

* Right click on the code file where you want to insert the code snippet.
* Select **Insert Snippet:**

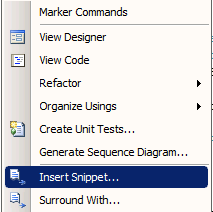


Figure 3 - Visual Studio Code Context Menu.

* Select code snippets from **My Code Snippets** gallery.

## Exercise 1 – Create a SharePoint List Definition Project with List Instance

### Task 1 – Create a SharePoint List Definition Project

This task will guide you through creating the Visual Studio 2010 Solution using the SharePoint 2010 Project template for a List Definition.

1. Open Visual Studio 2010 by going to **Start Menu | All Programs | Microsoft Visual Studio 2010 | Microsoft Visual Studio 2010**.
2. From the Visual Studio 2010 menu, select **File | New | Project...**.
3. Select the **Visual C# | SharePoint | 2010 |** **List Definition** project template.
4. Enter **Bugs** in the Name textbox.
5. Enter **C:\SPHOLS\SPCHOL300\CS\Ex1\** in the Location textbox.

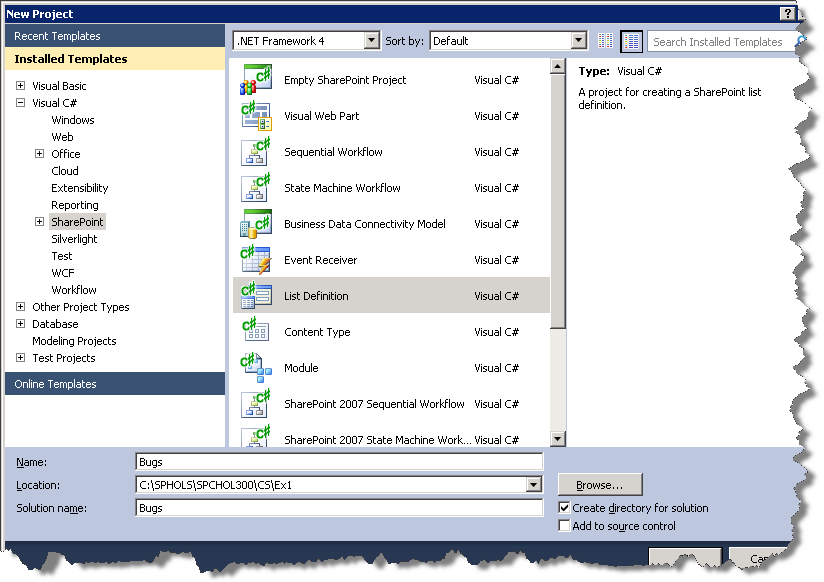


Figure 4 - New Project Dialog

1. Click **Ok**.
2. In the SharePoint Customization Wizard:
3. Enter **http://intranet.contoso.com/** for the local site.

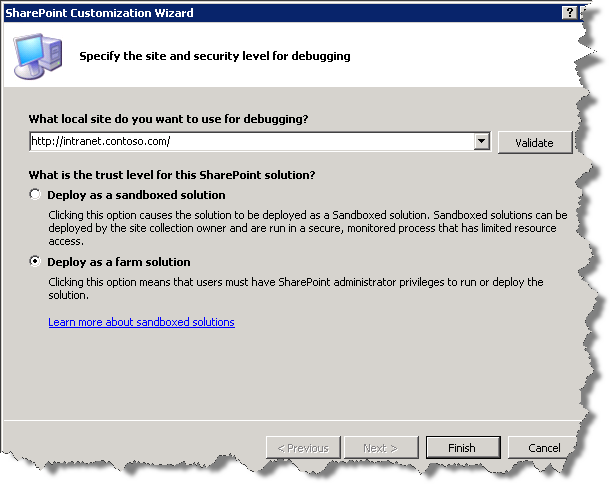


Figure 5 - SharePoint Customization Wizard Step1.

1. Set the trust level to **Deploy as a farm solution**.
2. Click **Next**.
3. Enter **Bugs** in the **What is the display name of the list definition?** textbox.
4. Select **Custom List** in the **What is the type of the list definition?** drop-down box.
5. Tick **Add a list instance for this list definition**.

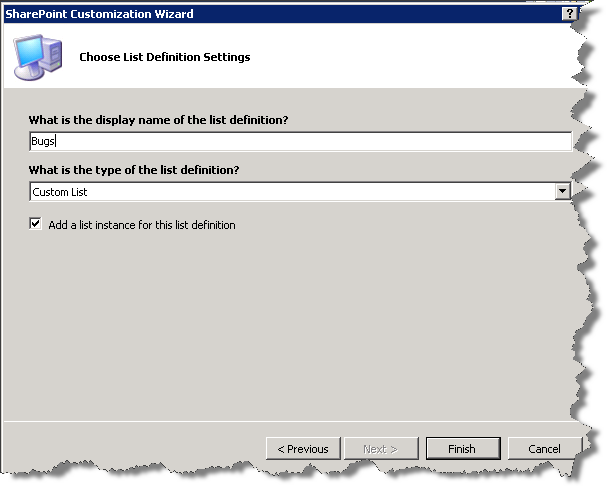


Figure 6 - SharePoint Customization Wizard Step2.

1. Click **Finish**.

### Task 2 – Customize the SharePoint List Definition and List Instance

This task will guide you through customizing the List Definition project template by changing the list name, the content type it stores, and the fields that are displayed in the new, edit, and display forms.

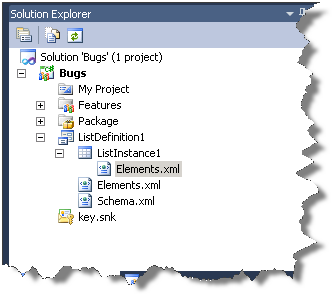
1. In the Solution Explorer, expand **ListInstance1** and open the **Elements.xml file**.  
     
   

Figure 7 - Solution Explorer highlighting Elements.xml

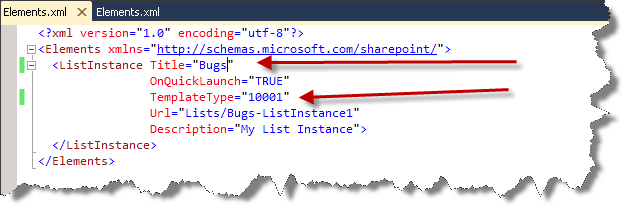
1. Within the **ListInstance** element change the Title attribute to **Bugs** and change theTemplateType attribute to **10001**.  
     
    

Figure 8 - Elements.xml file after editing Title and TemplateType

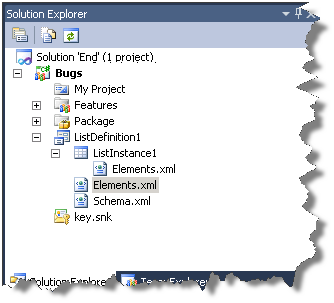
1. In the Solution Explorer, open **Elements.xml**.  
     
   

Figure 9 - Solution Explorer highlighting Elements.xml

1. Within the **ListTemplate** element, change the Type attribute to **10001**and add the **DisallowContentTypes=”FALSE”** attribute, as shown below.

DisallowContentTypes="FALSE"



Figure 10 - Elements.xml file after editing Type and adding the DisallowContentTypes attribute

1. Insert the following xml into the top of the **Elements** element in the Elements.xml file. This XML describes the Bug Item content type the list will store.   
   **Code Snippet:** *My Xml Snippets | spchol300\_ex1\_contenttype*

<!-- Start Add as part of SPCHOL300 Ex1 -->

<ContentType

ID="0x010089E3E6DB8C9B4B3FBB980447E313CE94"

Name="Bug Item"

Group="Custom Content Types"

Description="Bug item content type."

Version="0">

<FieldRefs>

<FieldRef ID="{fa564e0f-0c70-4ab9-b863-0177e6ddd247}" />

<FieldRef ID="{cb55bba1-81a9-47b6-8e6c-6a7da1d25602}" />

<FieldRef ID="{0248c82f-9136-4b3a-b802-d0b77280b3bc}" />

<FieldRef ID="{aa4a82dd-5b32-4507-9874-4e1c7bca3279}" />

</FieldRefs>

</ContentType>

<!-- End Add as part of SPCHOL300 Ex1 -->

1. Insert the following xml into the top of the **Elements** element in the Elements.xml file. This XML describes the Fields the Bug Item content type uses.   
   **Code Snippet:** *My Xml Snippets | spchol300\_ex1\_fields*

<!-- Start Add as part of SPCHOL300 Ex1 -->

<Field Type="Note" DisplayName="Description" Required="FALSE" NumLines="6" RichText="FALSE" Sortable="FALSE" ID="{cb55bba1-81a9-47b6-8e6c-6a7da1d25602}" StaticName="BugDescription" Name="BugDescription" Group="Custom Columns" />

<Field Type="Text" DisplayName="Project" Required="FALSE" MaxLength="255" ID="{0248c82f-9136-4b3a-b802-d0b77280b3bc}" StaticName="BugProject" Name="BugProject" Group="Custom Columns" />

<Field Type="Text" DisplayName="Assigned To" Required="FALSE" MaxLength="255" ID="{aa4a82dd-5b32-4507-9874-4e1c7bca3279}" StaticName="BugAssignedTo" Name="BugAssignedTo" Group="Custom Columns" />

<!-- End Add as part of SPCHOL300 Ex1 -->

1. The Elements.xml file should now look like the following.  
     
   

Figure 11 - Elements.xml file after the insertion of the two code snippets.

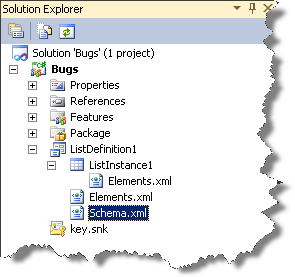
1. In the Solution Explorer, open **Schema.xml**  
   

Figure 12 - Solution Explorer highlighting Schema.xml

1. Add the **EnableContentTypes=”TRUE”**attribute to the List element inside of the Schema.xml file.

EnableContentTypes="TRUE"



Figure 13 - Schema.xml after the adding of the EnableContentTypes attribute

1. Insert the following XML into the **ContentTypes** element in the schema.xml file. This XML describes the Bug Item content type this list will store.   
   **Code Snippet:** *My Xml Snippets | spchol300\_ex1\_contenttyprefs*

<!-- Start Add as part of SPCHOL300 Ex1 -->

<ContentTypeRef ID="0x010089E3E6DB8C9B4B3FBB980447E313CE94" />

<!-- End Add as part of SPCHOL300 Ex1 -->

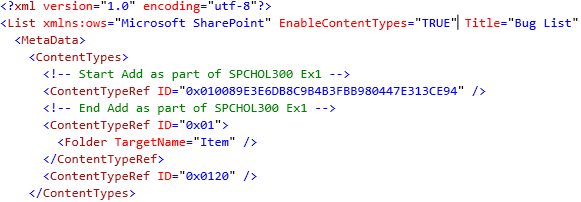


Figure 14 - Schema.xml after inserting the ContentTypeRefs snippet

1. Insert the following XML into the **Fields** element.This XML describes the fields the list will store. These are directly related to the fields in the Content Type we added in the previous step. **Code Snippet:** *My Xml Snippets | spchol300\_ex1\_fields*

<!-- Start Add as part of SPCHOL300 Ex1 -->

<Field Type="Note" DisplayName="Description" Required="FALSE" NumLines="6" RichText="FALSE" Sortable="FALSE" ID="{cb55bba1-81a9-47b6-8e6c-6a7da1d25602}" StaticName="BugDescription" Name="BugDescription" Group="Custom Columns" />

<Field Type="Text" DisplayName="Project" Required="FALSE" MaxLength="255" ID="{0248c82f-9136-4b3a-b802-d0b77280b3bc}" StaticName="BugProject" Name="BugProject" Group="Custom Columns" />

<Field Type="Text" DisplayName="Assigned To" Required="FALSE" MaxLength="255" ID="{aa4a82dd-5b32-4507-9874-4e1c7bca3279}" StaticName="BugAssignedTo" Name="BugAssignedTo" Group="Custom Columns" />

<!-- End Add as part of SPCHOL300 Ex1 -->

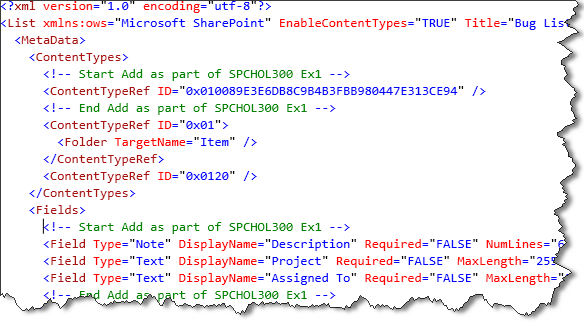


Figure 15 - Schema.xml after Fields xml insert

1. Insert the following XML into the **ViewFields** element in the 2nd view, **BaseViewID="1"**. This XML describes which fields should be visible in this particular view. BaseView 1 is set as the default view and this is the view the newly created fields should display in.  
   **Code Snippet:** *My Xml Snippets | spchol300\_ex1\_fieldrefs*

<!-- Start Add as part of SPCHOL300 Ex1 -->

<FieldRef Name="BugDescription"></FieldRef>

<FieldRef Name="BugProject"></FieldRef>

<FieldRef Name="BugAssignedTo"></FieldRef>

<!-- End Add as part of SPCHOL300 Ex1 -->

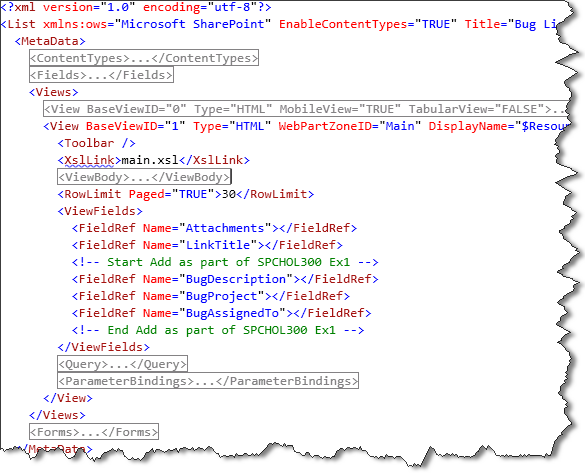


Figure 16 - Schema.xml after Fields XML insert

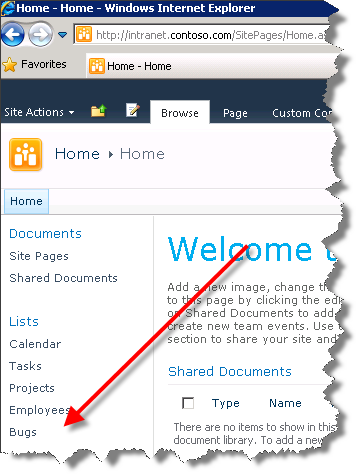
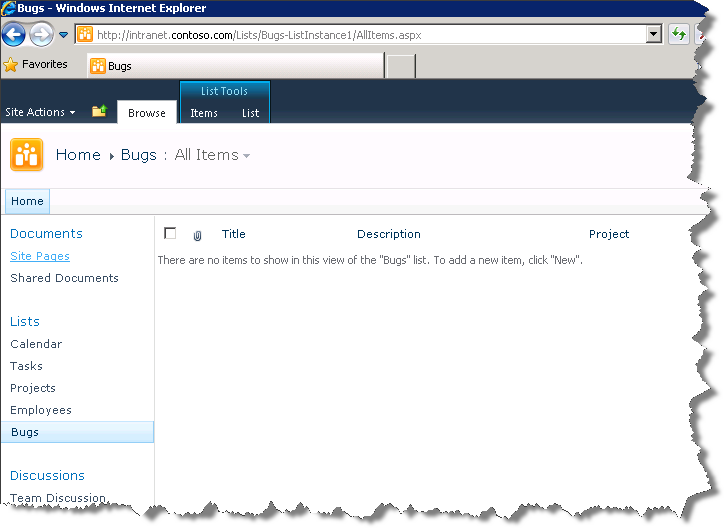
1. From the menu, select **Build | Deploy Bugs**.
2. Open <http://intranet.contoso.com> in a browser session. Under **Lists** in the left-side navigation, click on **Bugs**.  
     
   

Figure 17 - Bugs list link in left navigation

1. This is the **Bugs** list based on the List Definition.

  
Figure 18 - List Panel

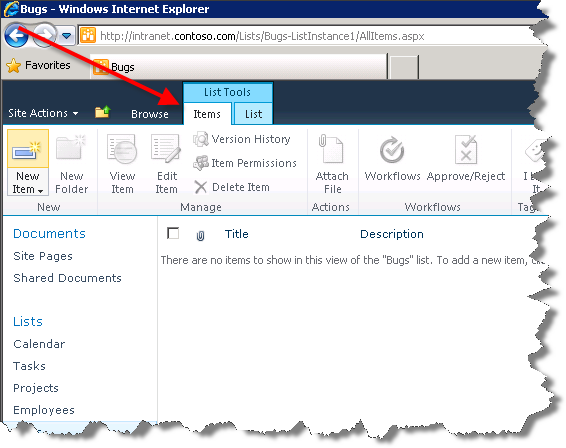
1. To add an item Click **Items**.  
     
   

Figure 19 - List Tools panel tabs

1. Click **New Item** with the arrow down next to it and select **Bug Item**.

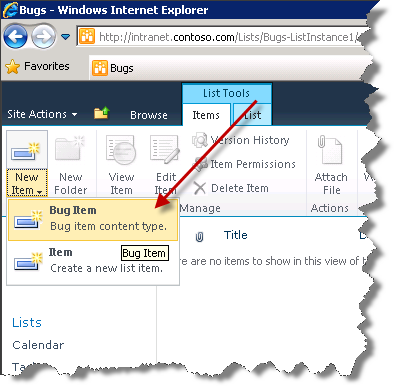


Figure 20 - New Item button

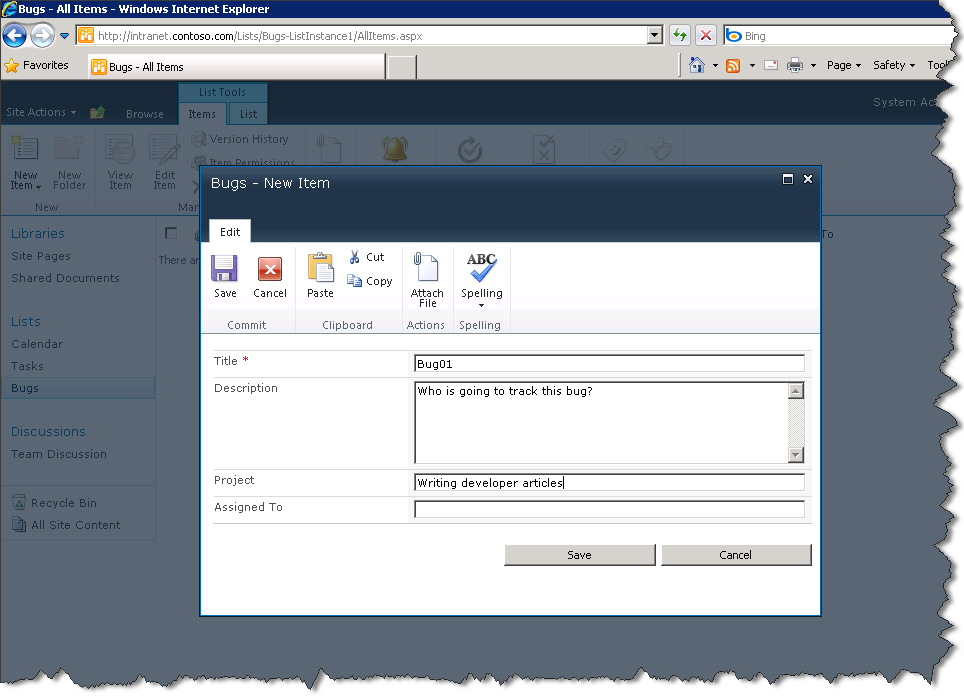
1. Enter **Bug01** in the Title textbox.
2. Enter ***Who is going to track this bug?*** *In the* Description textbox
3. Enter ***Writing developer articles*** in the Project textbox.
4. Click **Save**.  
   

Figure 21 - New Item dialog

1. You have successfully customized the List Definition and List Instance.

### Task 3 – Add an Event Receiver

This task will guide you through creating and attaching an Event Receiver to the List Definition we created in the previous step. This event receiver will prevent the deletion of bug items and automate the filling out of the **Assigned To** field.

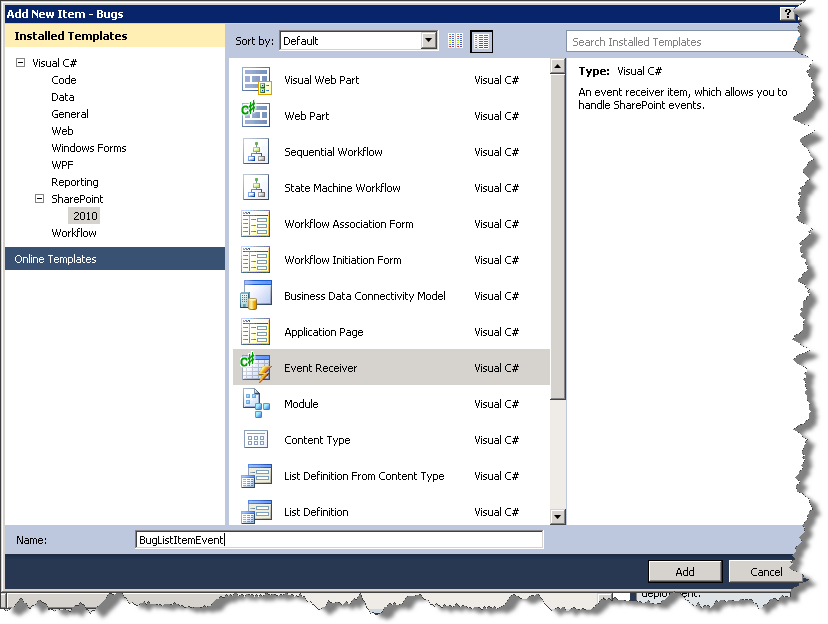
1. In **Visual Studio 2010**, right-click on the **Bugs** project and from the drop-down cascade select **Add | New Item…**.
2. Select the **Visual C# | SharePoint | 2010** | **Event Receiver** project type.
3. Enter **BugListItemEvent** in the Name textbox.  
   

Figure 22 - Add New Item dialog

1. Click **Add**.
2. In the SharePoint Customization Wizard, tick the following 3 events:
   * An item is being added
   * An item is being updated
   * An item is being deleted

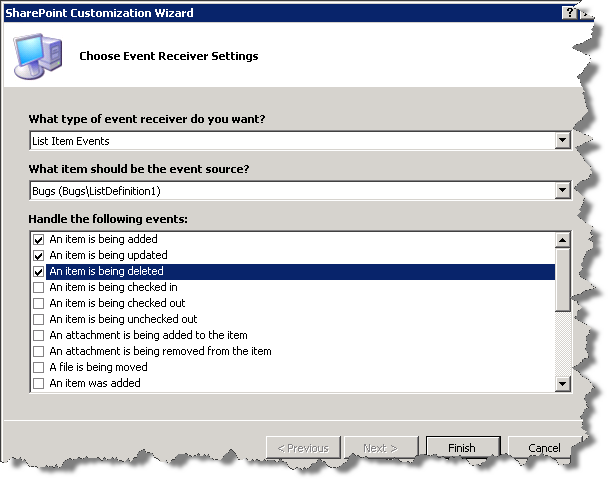


Figure 23 - Insert Event Reciever Settings dialog

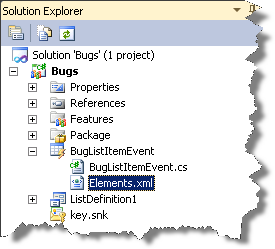
1. Click **Finish**.
2. In the **Solution Explorer,** expand **BugListItemEvent**and open the **Elements.xml** file.  
     
   

Figure 24 - Solution Explorer highlighting Elements.xml

1. To make sure the event receivers only bind to our custom list, we need to ensure the **ListTemplateId** attribute in the **Receivers** element matches the type of our list. If not, change the **ListTemplateId** attribute to *10001*.  
     
   

Figure 25 - Elements.xml file with updated ListTemplateId

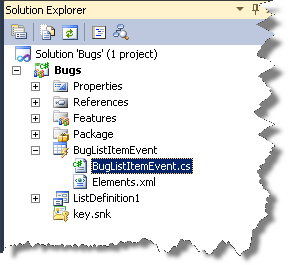
1. In the **Solution Explorer,** expand *BugListItemEvent* and open the **BugListItemEvent.cs**. There you should see the three override methods we ticked above.  
     
   

Figure 26 - Solution Explorer highlighting BugListItemEvent.cs

1. Insert the following code into the body of **public override void ItemDeleting(SPItemEventProperties properties)** after **base.ItemDeleting(properties);**. This code will prevent the deletion of an item and present the user with an error message when they attempt to delete.   
   **Code Snippet:** *My Code Snippets | spchol300\_ex1\_deleting*

try

{

properties.Cancel = true;

properties.ErrorMessage = "Bugs can only be resolved not deleted!";

}

catch (Exception ex)

{

//ToLog ex

return;

}

finally

{

this.EventFiringEnabled = true;

}

1. Insert the following code into body of **public override void ItemAdding(SPItemEventProperties properties)** after **base.ItemAdding(properties);.** This code will try and resolve the **Assigned To** field by looking up the project **Title** in the project list and mapping the **Contact Person** from the project list to the **Assigned To** field.   
   **Code Snippet:** *My Code Snippets | spchol300\_ex1\_adding*

try

{

this.EventFiringEnabled = false;

AutoAssign(properties);

}

catch (Exception ex)

{

//ToLog ex

return;

}

finally

{

this.EventFiringEnabled = true;

}

1. Insert the following code into body **of public override void ItemUpdating(SPItemEventProperties properties)** **after base.ItemUpdating(properties);**. This code will try and resolve the **Assigned To** field by looking up the project **Title** in the project list and mapping the **Contact Person** from the project list to the **Assigned To** field. a  
   **Code Snippet:** *My Code Snippets | spchol300\_ex1\_updating*

try

{

this.EventFiringEnabled = false;

AutoAssign(properties);

}

catch (Exception ex)

{

//ToLog ex

return;

}

finally

{

this.EventFiringEnabled = true;

}

1. Insert the following method into the body of the **BugListItemEvent** class. This is the code used by ItemAdding and ItemUpdating.   
   **Code Snippet:** *My Code Snippets | spchol300\_ex1\_autoassign*

private static void AutoAssign(SPItemEventProperties properties)

{

string projectName = properties.AfterProperties["BugProject"].ToString();

SPList projectList = properties.Web.GetList("Lists/Projects");

foreach (SPListItem item in projectList.Items)

{

if (item.Title.ToLower().Contains(projectName.ToLower()))

{

string contact = item["Primary Contact"].ToString();

if (contact.Contains(";#"))

{

contact = contact.Split('#')[1];

}

properties.AfterProperties["BugAssignedTo"] = contact;

break;

}

}

}

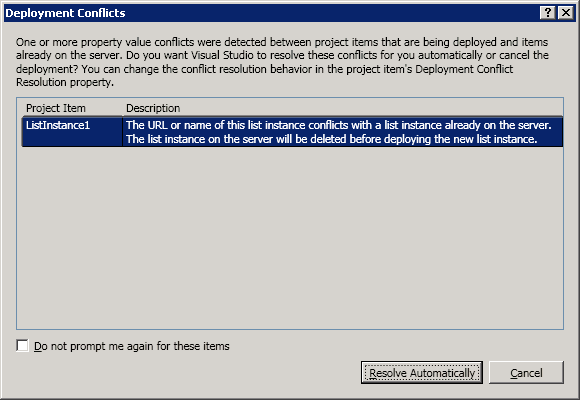
1. From the menu, select **Build | Deploy Bugs**.
2. This will be the third time we are deploying this solution and because we are working with a List Instance we will get a Deployment Conflict.  
     
   

Figure 27 - Deployment Conflict dialog

1. Visual Studio can automatically resolve this conflict. In this case it will delete and recreate the list. Click **Resolve Automatically**.
2. Open <http://intranet.contoso.com> in a browser session. Click on the **Bugs** link under **Lists** in the left hand navigation.  
     
   

Figure 28 - Bugs list link

1. You can now test the three event receivers we just added. Try adding, updating, and deleting items. (Note that if you try to delete an item, you will get an exception dialog, and that is expected behavior.)

## Exercise 2 – Using the Developer Dashboard to review performance of the event receiver

### Task 1 – Enable the Developer Dashboard

This task will show how to enable the developer tool dashboard via the command prompt using the stsadm.exe.

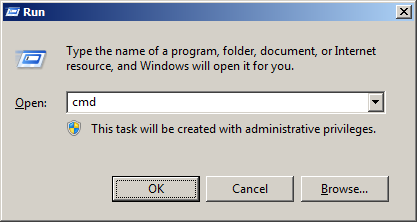
1. To enable the developer tool bar, go to **Start | Run.** Type **cmd** and click **OK**.
2. 

Figure 29 - Run dialog

1. Type in *"C:\Program Files\Common Files\Microsoft Shared\Web Server Extensions\14\BIN\stsadm.exe" -o setproperty -pn developer-dashboard -pv OnDemand*

"C:\Program Files\Common Files\Microsoft Shared\Web Server Extensions\14\BIN\stsadm.exe" -o setproperty -pn developer-dashboard -pv OnDemand

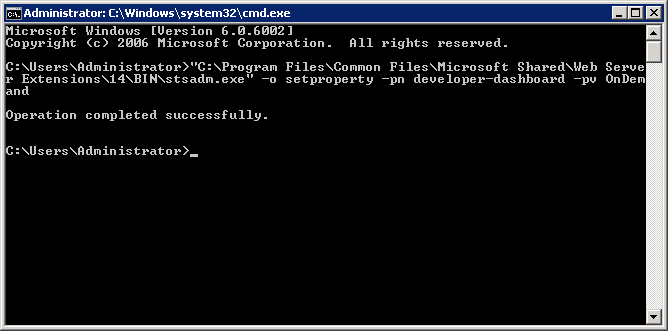
1. Press **Enter**.
2. This has enabled the Developer Dashboard.

Figure 30 - Terminal window after running the stsadm.exe

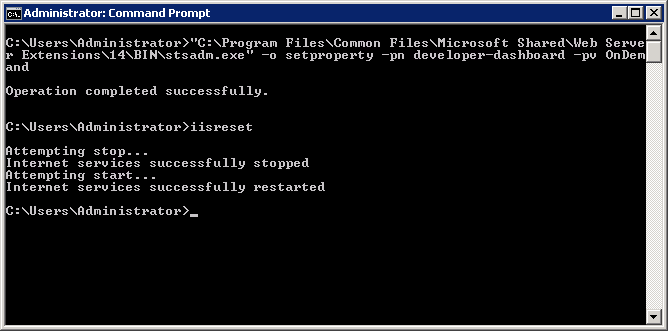
1. Type in *iisreset* and press **Enter**.  
     
   

Figure 31 - Terminal window after running iisreset

### Task 2 – Using the Developer Dashboard

This task demonstrates how to toggle the Developer Dashboard view on and off and points out some of its interesting features.

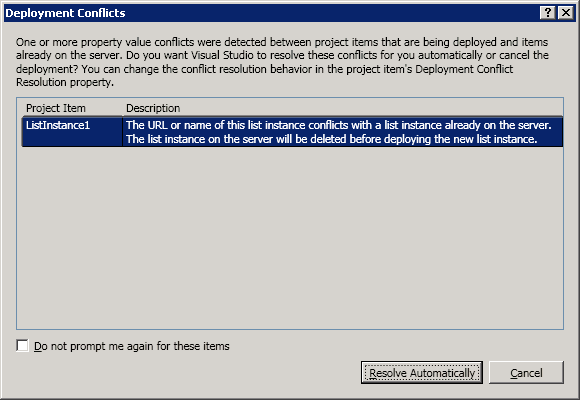
1. If you have just completed **Exercise 1** go to **Step 4**, otherwise open the **Exercise 1 | End** solution as described in Steps 2 & 3.
2. From the menu, select **Build | Deploy Bugs**.  
     
   

Figure 32 - Deployment Conflict dialog

1. Visual Studio can automatically resolve this conflict. In this case it will delete and recreate the list. Click **Resolve Automatically**.
2. Open <http://intranet.contoso.com> in a browser session. Click on the **Bugs** link under **Lists** in the left hand navigation.  
   

Figure 33 - Bugs list link

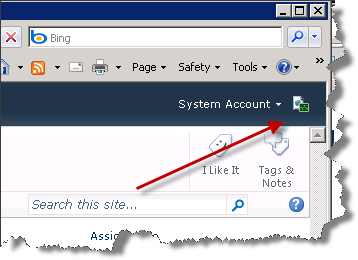
1. In the browser, activate the Developer Dashboard by clicking on the icon in the top right corner of the browser window.  
     
   

Figure 34 - Developer Dashboard icon

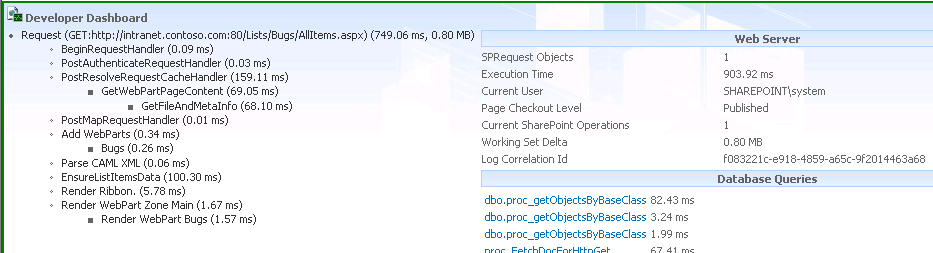
1. This will refresh the browser and display the tool bar at the bottom of the page.   
   

Figure 35 - Developer Dashboard panel

1. Some key information displayed in the Developer Dashboard are:
   * The SQL queries they have and their timings.
   * WebPart Events Offsets for testing performance against custom web parts.

## Lab Summary

In this lab you performed the following exercises.

* Created a SharePoint List Definition Project with List Instance.
* Customized the SharePoint List Definition and List Instance.
* Added an Event Receiver.
* Enabled the Developer Dashboard.
* Used the Developer Dashboard.