## Lists & Event Handlers

**Lab Time:** 45 minutes

**Lab Folder**: C:\Student\Labs\Lists

**Lab Overview:** In this lab you will work with some of the new capabilities Microsoft added to events in Windows SharePoint Foundation 2010 as well as the new Visual Studio 2010 SharePoint Tools. You will first create a custom list definition, template and instance using Visual Studio 2010. Next, you will implement referential integrity between two SharePoint lists so that items in one list cannot be deleted until referenced items in a child list are first removed so there are no orphans. Finally, you’ll create a synchronous event receiver that is triggered when new sites are created and, when certain criteria are met, you’ll cancel the error and redirect the user to a custom application page for more information on the error.

Lab Setup Requirements

* Before you begin this lab, you must run the batch file named **SetupLab.bat** in the folder for this lab. This batch file creates a new SharePoint site collection at the location **http://intranet.wingtip.com/sites/Lists**.

### Exercise 1: Creating Lists with Visual Studio 2010

In this exercise you will create a new SharePoint list using the SharePoint development tools included in Visual Studio 2010.

1. Launch the Internet Explorer and navigate to the top-level site at http://intranet.wingtip.com/sites/Lists. Take a moment to inspect the site and make sure it behaves as expected. Note that the setup script creates a new site collection with a team site as its top-level site.
2. Launch Visual Studio 2010 and create a new **Empty SharePoint Project** by selecting **File » New » Project** and give it a name of **SharePointLists**.
3. Complete the wizard that appears using the following information.

**Debugging site:** http://intranet.wingtip.com/sites/Lists

**Deploy as a farm solution**: selected

1. The first step is to create the site columns and content types that will be applied to the new list that you will create. This is done by adding a new project item to the project. Right-click the **SharePointLists** project in the **Solution Explorer** tool window and select **Add » New Item**. Pick **Content Type** from the list of **SharePoint » 2010** templates and give it a name of **Product**. The wizard dialog will then prompt you for the content type the new content type will inherit from. Select the **Item** content type from the dropdown list.
2. Before creating the content type, the first step is to create two new site columns. Add the following CAML just before the opening <ContentType> element. Note that you don’t need to use the same GUIDs as shown here. Just make sure you track the ones you use because you’ll need to reference them in future steps.

Take care to assure the ID="" attribute is in CAPS. IntelliSense tries to use the invalid Id="" format.

<?xml version="1.0" encoding="utf-8"?>

<Elements xmlns="http://schemas.microsoft.com/sharepoint/">

<Field SourceID="http://schemas.microsoft.com/sharepoint/v3"

ID="{36819A9B-E748-47D5-9949-A65DD195BF80}"

Name="ProductDescription"

DisplayName="Product Description"

Group="My Custom Columns"

Type="Text"

DisplaceOnUpgrade="TRUE" />

<Field SourceID="http://schemas.microsoft.com/sharepoint/v3"

ID="{5CD2C0C1-67AC-4F9E-BF21-463CFEE9B382}"

Name="ProductID"

DisplayName="Product ID"

Group="My Custom Columns"

Type="Number"

DisplaceOnUpgrade="TRUE" />

</Elements>

1. With the site columns created, you can add the columns to the content type. Add a <FieldRefs> element containing <FieldRef> elements for each column to the <ContentType> nodes. Add also an extra <FieldRef> element for the Title field. This adds three columns to the content type. The resulting content type should look like the following CAML. Ensure the correct GUIDs are used for the ProductDescription and the ProductID fields, referencing the columns created in the previous step.

<?xml version="1.0" encoding="utf-8"?>

<Elements xmlns="http://schemas.microsoft.com/sharepoint/">

<Field SourceID="http://schemas.microsoft.com/sharepoint/v3" />

<Field SourceID="http://schemas.microsoft.com/sharepoint/v3" />

<ContentType ID="0x0100fb1ad12faa9b4834ad4d590f0f030151"

Name="SharePointLists - Product"

Group="Custom Content Types"

Description=”My Content Type”

Version="0">

<FieldRefs>

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

Name="Title"

DisplayName="Product Name" />

<FieldRef ID="{36819A9B-E748-47D5-9949-A65DD195BF80}"

Name="ProductDescription" />

<FieldRef ID="{5CD2C0C1-67AC-4F9E-BF21-463CFEE9B382}"

Name="ProductID" />

</FieldRefs>

</ContentType>

</Elements>

1. Now the list template and instance can be created that will leverage this content type. Right-click the **SharePointLists** project in the Solution Explorer tool window, select **Add » New Item**, pick the **List Definition** template from the **SharePoint » 2010** list and specify a name of **ProductList**. In the following dialog, enter a display name of **Product List** for the question **What is the display name of the list definition?** Enter a value of **CustomList** for the question **What is the type of the list definition?** Leave the option for **Add a list instance for this list definition** checked and click **Finish**.
2. Now it’s time to modify the list definition (aka: template) and schema. The list definition, located in the SharePointLists\ProductList\Elements.xml file in the **SharePointLists** solution contains the list template. Change the Type attribute from **10000** to **10001** to give it a unique **ID**.

<?xml version="1.0" encoding="utf-8"?>

<Elements xmlns="http://schemas.microsoft.com/sharepoint/">

<ListTemplate

Name="ProductList"

Type="10001"

BaseType="0"

OnQuickLaunch="TRUE"

SecurityBits="11"

Sequence="410"

DisplayName="Product List"

Description="My List Definition"

Image="/\_layouts/images/itgen.gif"/>

</Elements>

1. Next, open the schema.xml file in the **SharePointLists\ProductList** section of the project. This file contains all the details about the list such as what fields will appear, any content types and the views in the list.
2. First, add the attribute EnableContentTypes="TRUE" to the <List> node.
3. Second, add the content type created previously to the <ContentTypes> section near the top of the schema.xml file like so:

<?xml version="1.0" encoding="utf-8"?>

<List xmlns:ows="Microsoft SharePoint" Title="ProductList" EnableContentTypes="TRUE" FolderCreation="FALSE" Direction="$Resources:Direction;" Url="Lists/SharePointLists-ProductList" BaseType="0" xmlns="http://schemas.microsoft.com/sharepoint/">

<MetaData>

<ContentTypes>

<ContentTypeRef ID="0x01">

<Folder TargetName="Item" />

</ContentTypeRef>

<ContentTypeRef ID="0x0120" />

<ContentTypeRef ID="0x0100fb1ad12faa9b4834ad4d590f0f030151" />

</ContentTypes>

Make sure you use the same content type ID that was generated when you created the content type previously.

1. Now it's time to add the fields that will be included in the list by adding them to the <Fields> section in the schema.xml file. Add the three fields that are part of the content type like so:

<?xml version="1.0" encoding="utf-8"?>

<List xmlns:ows="Microsoft SharePoint" Title="ProductList" EnableContentTypes="TRUE" FolderCreation="FALSE" Direction="$Resources:Direction;" Url="Lists/SharePointLists-ProductList" BaseType="0" xmlns="http://schemas.microsoft.com/sharepoint/">

<MetaData>

<ContentTypes>…</ContentTypes>

<Fields>

<Field ID="{fa564e0f-0c70-4ab9-b863-0177e6ddd247}"

Name="Title"

DisplayName="Product Name"

Type="Text" />

<Field ID="{36819A9B-E748-47D5-9949-A65DD195BF80}"

Name="ProductDescription"

DisplayName="Product Description"

Type="Text" />

<Field ID="{5CD2C0C1-67AC-4F9E-BF21-463CFEE9B382}"

Name="ProductID"

DisplayName="ProductID"

Type="Number" />

</Fields>

1. With the content type and fields added to the list, the last part is to add the fields to two default list views. Search for the <ViewFields> elements and add the fields. There are two default views in the schema.xml file so you’ll find two of them. The first one should look like this:

<ViewFields>

<FieldRef Name="LinkTitleNoMenu"></FieldRef>

<FieldRef ID="{36819A9B-E748-47D5-9949-A65DD195BF80}"

Name="ProductDescription"

DisplayName="Product Description" />

<FieldRef ID="{5CD2C0C1-67AC-4F9E-BF21-463CFEE9B382}"

Name="ProductID"

DisplayName="ProductID" />

</ViewFields>

1. And the second one should look like this:

<ViewFields>

<FieldRef Name="Attachments"></FieldRef>

<FieldRef Name="LinkTitle"></FieldRef>

<FieldRef ID="{36819A9B-E748-47D5-9949-A65DD195BF80}"

Name="ProductDescription"

DisplayName="Product Description" />

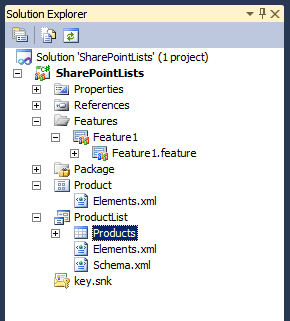
<FieldRef ID="{5CD2C0C1-67AC-4F9E-BF21-463CFEE9B382}"

Name="ProductID"

DisplayName="ProductID" />

</ViewFields>

1. With the list template and definition complete, now you need to modify the list instance that will be created based off this template. First, Rename the node **ProductList\ListInstance1** to **ProductList\Products**:



1. Next, open the ProductList\Products\Elements.xml file. Change the <ListInstance> element to match the following, changing the Title, TemplateType and Url attributes:

<?xml version="1.0" encoding="utf-8"?>

<Elements xmlns="http://schemas.microsoft.com/sharepoint/">

<ListInstance Title="Products"

OnQuickLaunch="TRUE"

TemplateType="10001"

Url="Lists/Products"

Description="My List Instance">

</ListInstance>

</Elements>

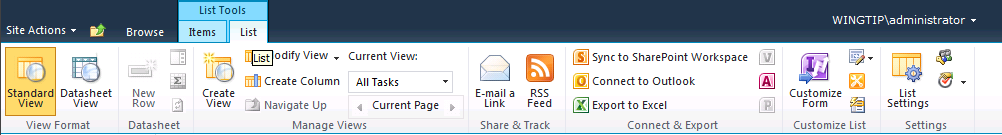
1. Test your work by saving all changes and pressing **[CTRL]+[F5]** to build and deploy the solution. Eventually Visual Studio will launch the site in the browser and you should see the **Products** list in the Quick Launch.
2. You can look in the **Site Column Gallery** and **Site Content Type Gallery** to see the other assets created by the Visual Studio 2010 project.

In this exercise you used the new SharePoint Tools in Visual Studio 2010 to create new site columns, a content type, list template and an instance of that list.

### Exercise 2: Implement Referential Integrity between SharePoint Lists

In this exercise you will add referential integrity between two lists that you will create in this exercise. Referential integrity is a new capability added in SharePoint Foundation 2010.

1. Open a browser and navigate to the site created in the previous exercise.
2. Create a new list named **Companies** by selecting **Site Actions » More Options…**, and then select the **Custom List** template.
3. Edit the list to add columns by clicking the **List** tab in the ribbon, selecting **List Settings**:



1. Now create two new columns for this list using the **Create column** link below the **Columns** section of the list settings page. Create the two columns using the following values leaving all other fields their default values:
2. Create a column named **Stock Ticker** based on column type of **Single line of text**.
3. Create a column named **Headquarters** based on column type of **Single line of text**.
4. Go back to the list by clicking **Companies** in the breadcrumb at the top of the page. Now add three items to the list. This is done using the ribbon again. Select the **Items** tab, then **New Item** and finally **New Item**.
5. Add an item to the list using the following information.

**Title**: Microsoft

**Stock Ticker**: MSFT

**Headquarters**: Redmond

1. Add another item to the list using the following information.

**Title**: Adventure Works Travel

**Stock Ticker**: AWT

**Headquarters**: London

1. Add another item to the list using the following information.

**Title**: Wingtip, Inc.

**Stock Ticker**: WNG

**Headquarters**: Seattle

1. The **Companies** list will serve as the parent list in the join. Now you need to create a new list named **SKUs** that will contain products and be treated as the child in the relationship. Refer to the previous instructions to create a new list **SKUs** (using the **Custom List** template).
2. First add a lookup column that will (1) act as the join column & (2) enforce referential integrity between **Companies** and **SKUs**.
3. Name the new column **Manufacturer** based on a column type of **Lookup**.
4. For the column setting of **Get information from** assign a value of **Companies**.
5. For the column setting of **In this column** assign a value of **Title**.
6. Add a column to show each of these additional fields: **Headquarters**. Note that this setting will cause SharePoint to project the **Companies.Headquarters** field into the views within the **SKUs** list containing the value corresponding to the list item selected in the **Manufacturer** field.
7. Make sure the column setting **Enforce relationship behavior** is checked
8. Make sure the column setting **Restrict Delete** is checked

**Note:** These last two selections will prohibit users from deleting items in the **Companies** list if that item is referenced by items in the **SKUs** list.

1. When prompted, **accept the dialog** prompting you to create an index on this column. Linked columns in joins must be indexed.

If you cancel the dialog, the column will not be created & added to the list.

1. Now, create two additional columns for this list.
2. Create a column named **Description** based on a column type of **Single line of text**.
3. Create a column named **Cost** based on a column type of **Currency**.
4. Add an item to the **SKUs** list using the following information.

**Title:** SharePoint Foundation 2010

**Manufacturer**: Microsoft

**Description**: SharePoint Foundation 2010

**Cost**: $0.00

1. Add another item to the **SKUs** list using the following information.

**Title:** Office

**Manufacturer**: Microsoft

**Description**: Office 2010 Client Applications

**Cost**: $100.00

1. Add another item to the **SKUs** list using the following information.

**Title:** Destination Guide Seattle

**Manufacturer**: Adventure Works Travel

**Description**: Travel guide for Seattle, Washington

**Cost**: $25.00

1. Add another item to the **SKUs** list using the following information.

**Title:** Destination Guide St Johns

**Manufacturer**: Adventure Works Travel

**Description**: Travel guide for St. Johns, Florida

**Cost**: $25.00

1. Add another item to the **SKUs** list using the following information.

**Title:** Do-it-Yourself Advanced Fireworks Set

**Manufacturer**: Wingtip

**Description**: Create your own fireworks indoors with this advanced fireworks set. Real gunpowder! Age 4-7

**Cost**: $15.00

1. Add another item to the **SKUs** list using the following information.

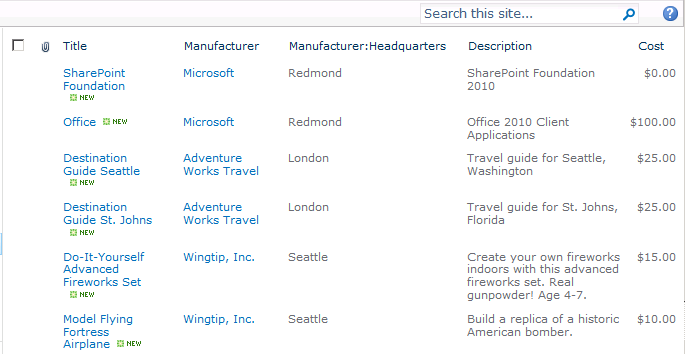
**Title:** Model Flying Fortress Airplane

**Manufacturer**: Wingtip

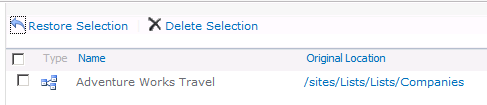
**Description**: Build a replica of a historic American bomber.

**Cost**: $10.00

1. When finished, you should see a list of products. However, you should notice that fields from the **Companies** list were pulled down into this list!



1. Now, test the referential integrity of the lists. Go try to delete the **Adventure Works Travel** item form the **Companies** list. You will be sent to an error page.
2. To test out the cascading delete capability, go change the **Manufacturer** column in the **SKUs** list to **Cascade Delete**. Now, delete the same **Adventure Works Travel** item from Companies. If you browse to the Products list, you’ll see two items are now gone.
3. When the parent and child items were deleted, they were all deleted in a single atomic unit. Look inside the **Recycle Bin** (available via the Quick Launch menu). You’ll notice a special icon indicating there are multiple items in this deletion set.



In this exercise you created two joined lists that utilized the referential integrity constraints provided in SharePoint.

### Exercise 3: Create URL Redirection upon Event Errors

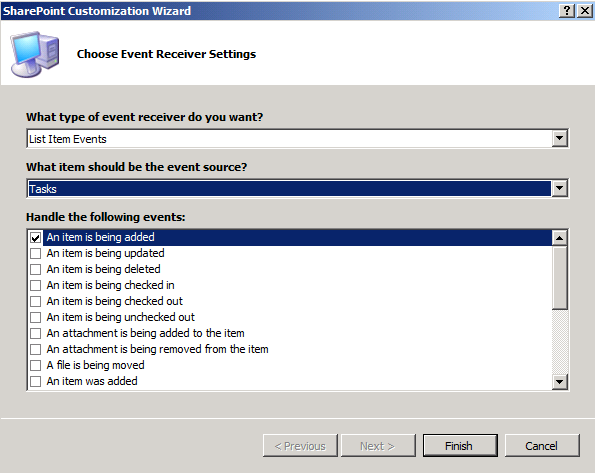
In this exercise you will create an error page the user will be redirected to when there is an error during an event. Cancelling events was introduced in WSS 3.0 but the capability to cancel and redirect the user to an error page is a new capability added in SharePoint Foundation 2010.

1. Launch **Visual Studio 2010** and create a new **Empty SharePoint Project** by selecting **File » New » Project** and give it a name of **CustomErrorPage**.
2. Complete the wizard that appears using the following information.

**Debugging site:** http://intranet.wingtip.com/sites/Lists

**Deploy as a farm solution**: selected

1. The first step is to create the event receiver that will test if a due date is filled out when a task is created. This is done by adding a new project item to the project. Right-click the **CustomErrorPage** project in the Solution Explorer tool window and select **Add » New Item**. Pick **Event Receiver** from the list of **SharePoint » 2010** templates and give it a name of **DueDateEventReceiver**. The wizard dialog will then prompt you for the event receiver properties. Use the following to complete the wizard dialog:
2. When prompted for type of event receiver, choose a value of **List Item Events**.
3. The second dropdown is populated with all list templates from the target site. Choose the **Tasks** list.
4. In the **Handle the following events list** check the value **An item is being added**.
5. Click the Finbish button.



1. Open the DueDateEventReceiver\DueDateEventReceiver.cs code file. Replace the contents of the ItemAdding event with the following code:

public override void ItemAdding(SPItemEventProperties properties) {

if ((properties.AfterProperties[“DueDate”] == null) ||

(properties.AfterProperties[“DueDate”] == string.Empty))

{

properties.Status = SPEventReceiverStatus.CancelWithRedirectUrl;

properties.RedirectUrl = "/\_layouts/CustomErrorPage/DueDateErrorPage.aspx";

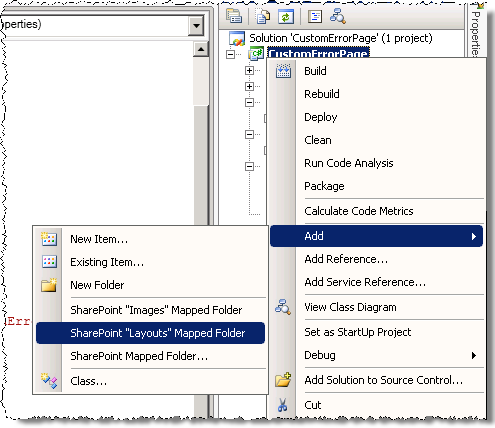
}

}

Each time a task is created within the current site, the event receiver will fire checking if a due date is filled out. If not, the event receiver will cancel the creation of the task and redirect the user to a custom ASPX page named DueDateErrorPage.aspx that you will create in the next few steps.

First, think about why you would want to create a custom error page. Using this technique you can provide the user with rich contextual information on the problem and guidance about how to resolve it. For now you’ll just create a simple custom error page to tell the user that it is not possible to add a task without a due date.

1. Now it's time to create a custom application page that will be used by your event handler to redirect the user when things go wrong. Right-click the **CustomErrorPage** project and select **Add » SharePoint “Layouts” Mapped Folder**:



1. Right-click the **Layouts\CustomErrorPage** folder, select **Add Item » New Item**, select the **Application Page** template from the **SharePoint » 2010** group and name the file DueDateErrorPage.aspx.
2. Open DueDateErrorPage.aspx in code view and update the Main, PageTitle and PageTitleInTitleArea content placeholder to the following:

<asp:Content ID="Main" contentplaceholderid="PlaceHolderMain" runat="server">

<p>

ERROR: You cannot create a task without a due date.

</p>

</asp:Content>

<asp:Content ID="PageTitle" contentplaceholderid="PlaceHolderPageTitle" runat="server">

Due Date Error Page

</asp:Content>

<asp:Content ID="PageTitleInTitleArea" contentplaceholderid="PlaceHolderPageTitleInTitleArea" runat="server" >

Due Date Error Page

</asp:Content>

1. Test your work by saving all changes and pressing **[CTRL]+[F5]** to build and deploy the solution. Eventually Visual Studio will launch the site in the browser. If no task list is available, create one using the **More Options** menu item of the **Site Actions** menu.
2. Create a task with a due date. Create a second task without a due date. When saving the task, the custom error page is displayed:



In this exercise you created a new event that used the error page event handler options.