Lab 2: Working with Content Types

**Lab Time: 60 Minutes**

**Lab Directory: ECM401.ContentTypes**

**Lab Overview:**

In this lab, you will write code that manipulates SharePoint Content Types via the SharePoint object model and API. The exercises in this lab take an imperative (code-based) rather than a declarative (xml-based) approach in order to familiarize you with the objects and methods provided by the SharePoint API for building solutions. Your code will focus both on discovering existing type definitions and creating new ones.

# Exercise 1: Exploring the SharePoint API

1. In this exercise, you will write a console application the displays information about content types for a given site. Your utility will accept three parameters; the URL of the target site to scan, the name of a content type to be displayed and whether to recursively display all content types which inherit from the one specified.
2. Start by creating a new Console Application project in Visual Studio. Give it the name **ECM401.ContentTypeBrowser** .
3. In order to use the SharePoint API, you must add a reference to the Microsoft.SharePoint.dll file. Right-click the **References** node and select **Add Reference…** . Click on the **.NET** tab and scroll to the bottom of the list. Select **Windows® SharePoint® Services** and click OK.
4. Add the following line of code to the top of the **Program.cs** file.

using Microsoft.SharePoint;

1. Add three data members to the **Program** class as follows:

bool Recursive=false;  
string Url = "http://litwareinc.com";  
string TargetType = null;

1. Add the following code to the static **Main** method.

Code Snippet: 'Main Method'

static void Main(string[] args)  
 {  
 bool isType = false;  
 bool recursive = false;  
 string baseType = null;  
 string url = "http://localhost:401/docman";  
  
  
 foreach (string option in args)  
 {  
 switch (option.ToLower())  
 {  
 case "-type": isType = true; break;  
 case "-recursive": recursive = true; break;  
 default:  
 if (isType)  
 {  
 baseType = option.Replace("\"", "");  
 isType = false;  
 }  
 else  
 {  
 url = option.Replace("\"", "");  
 }  
 break;  
 }  
 }  
  
 new Program(url, baseType, recursive).Run();  
  
 Console.WriteLine("Press any key...");  
 Console.ReadKey();  
 }

1. This handles the program arguments. Now you will implement the Program constructor and Run method. Add the following code to the Program class.

/// <summary>  
/// Program constructor  
/// </summary>  
public Program(string url, string baseType, bool recursive)  
{  
this.Url = url;  
this.TargetType = baseType;  
this.Recursive = recursive;  
}

1. The Run method simply checks the data members to determine what to do. Enter the following code just after the Program constructor.

Code Snippet: 'Run Method'

/// <summary>  
 /// Executes the program.  
 /// </summary>  
 public void Run()  
 {  
 // Open the site.  
 using (SPSite site = new SPSite(this.Url))  
 {  
 // Get the root web.  
 using (SPWeb web = site.OpenWeb())  
 {  
 // Prevent endless recursion.  
 if (this.TargetType == null)  
 this.Recursive = false;  
  
 // Get the available content types.  
 foreach (SPContentType contentType in web.AvailableContentTypes)  
 {  
 if (this.TargetType == null)  
 DisplayContentType(0,contentType);  
 else if (contentType.Name.Equals(this.TargetType))  
 DisplayContentType(0,contentType);  
 }  
 }  
 }  
 }

1. This method starts at the root web of the designated URL and obtains the catalog of available content types for the website. The AvailableContentTypes collection includes all content types that are visible from the target website up to the root web of the site collection. It does not include content types for website beneath the target web.
2. Now you will implement the **DisplayContentType** method, which simply displays the content type name at the specified indent level. If the recursive flag is turned on, then you will see a tree of content types showing the inheritance level for the selected type. Enter the following code beneath the Run method.

Code Snippet: 'DisplayContentType Method'

/// <summary>  
 /// Writes content type information to the console.  
 /// </summary>  
 /// <param name="indentLevel"></param>  
 /// <param name="contentType"></param>  
 public void DisplayContentType(int indentLevel, SPContentType contentType)  
 {  
 for (int i = 0; i < indentLevel; i++)  
 Console.Write(" ");  
 Console.WriteLine("{0}",contentType.Name);  
 //Console.WriteLine(contentType.SchemaXml);  
 if (this.Recursive)  
 {  
 foreach (SPContentType subType in contentType.ParentWeb.AvailableContentTypes)  
 if (DerivesFrom(contentType, subType))  
 DisplayContentType(indentLevel + 1, subType);  
 }  
 }

1. To determine if one content type derives from another, you can use the SharePoint API to walk up the inheritance tree. Enter the following method beneath the DisplayContentType method.

Code Snippet: 'DerivesFrom Method'

/// <summary>  
 /// Returns true if the child is derived from the parent.  
 /// </summary>  
 /// <param name="parentType"></param>  
 /// <param name="childType"></param>  
 /// <returns></returns>  
 private bool DerivesFrom(SPContentType parentType, SPContentType childType)  
 {  
 while (childType.Name != "System")  
 {  
 childType = childType.Parent;  
 if (childType == parentType)  
 return true;  
 }  
 return false;  
 }

1. Build the project and open a command prompt in the bin\debug folder. Enter the following command line:

ECM401.ContentTypeBrowser.exe –type Item –recursive

1. The resulting window should resemble the following diagram.

# Exercise 2: Creating Content Types using XML

1. In the next exercise, you will create a SharePoint Feature that installs a custom content type called Promissory Note. The Promissory Note content type will include the following metadata fields:

* Amount
* Due Date
* Interest Rate

1. Start by creating a new **SharePoint Feature** project in Visual Studio. Give it the name **ECM401.PromissoryNote** and delete the auto-generated **FeatureReceiver.cs** and **FeatureReceiverBase.cs** files. You won’t be adding any code to the project, just XML.
2. Open the generated **feature.xml** file for editing. Remove the two highlighted lines shown below.

<?xml version="1.0" encoding="utf-8" ?>  
<Feature xmlns="http://schemas.microsoft.com/sharepoint/"  
Id="db85ec29-731a-4ace-810d-2675b3ef7b79"  
Hidden="FALSE"  
Title="ECM401.PromissoryNote"  
Description="Creates a simple promissory note content type."  
ImageUrl="Ted Pattison Group\bug\_brwn\_pith.gif"  
ReceiverAssembly="ECM401.PromissoryNote, Version=1.0.0.0, Culture=neutral, PublicKeyToken=a9ca7666007df3c3"  
ReceiverClass="ECM401.PromissoryNote.FeatureReceiver"  
Scope="Site"  
Version="1.0.0.0">  
  
<ElementManifests>  
<ElementManifest Location="elements.xml"/>  
</ElementManifests>  
  
</Feature>

Note: The actual feature **Id** value in your code will be different.

1. Now open the **elements.xml** file for editing. Select the entire contents of this file and replace the contents with the following XML code.

XML Snippet: 'Promissory Note Content Type'

<?xml version="1.0" encoding="utf-8" ?>  
<Elements xmlns="http://schemas.microsoft.com/sharepoint/">  
 <ContentType ID="0x01010070ECF29474514071B9E7F9D991D3DC05"  
 Name="Promissory Note"  
 Group="ECM401"  
 Description="Defines fields for a simple promissory note."  
 Sealed="FALSE"  
 Version="0"  
 ReadOnly="FALSE"  
 BaseType="0x0101">  
  
 <FieldRefs>  
 </FieldRefs>  
  
 <DocumentTemplate TargetName="/\_layouts/ecm401/promissory note.docx" />  
 </ContentType>  
</Elements>

1. Working with content types in XML can be challenging, because in order to declare field references, you have to know the unique identifier of each field you want to use. Because of this, you will use a trick to obtain the field identifiers for the title and comments fields.
2. Right-click on the project node and select **Add -> Existing Item…** from the context menu. Browse to **C:\Student\Resources\** and select the files **fieldswss.xml** and **fieldswss.xslt** . The fieldswss.xml file is a copy of the default field declaration file used by WSS. The fieldswss.xslt is a custom XSLT stylesheet you will use to transform the raw field declaration file into a table with field declarations you can simply copy into your content type definition.
3. The XSLT code you will reference looks like this:

XML Snippet: 'SharePoint Fields XSL Stylesheet'

<?xml version="1.0" encoding="utf-8" ?>  
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform" xmlns:wss="http://schemas.microsoft.com/sharepoint/">  
 <xsl:output method="html" version="1.0" encoding="utf-8" indent="yes" />  
 <xsl:template match="wss:Elements">  
 <html>  
 <body>  
 <h2>SharePoint 3.0 Built-In Fields</h2>  
 <table border="0" width="100%" style="font-size:9pt;">  
 <tr bgcolor="#9acd32">  
 <th align="left">Field</th>  
 <th align="left">Group</th>  
 <th align="left">Type</th>  
 <th align="left">Declaration</th>  
 </tr>  
 <xsl:apply-templates>  
 <xsl:sort select="@Name" />  
 </xsl:apply-templates>  
 </table>  
 </body>  
 </html>  
 </xsl:template>  
  
 <xsl:template match="wss:Field">  
 <tr>  
 <td>  
 <xsl:value-of select="@Name" />  
 </td>  
 <td>  
 <xsl:value-of select="@Group" />  
 </td>  
 <td>  
 <xsl:value-of select="@Type" />  
 </td>  
 <td>  
 <!-- Emit The Full Declaration -->  
 <xsl:element name="FieldRef">  
 <xsl:attribute name="ID">  
 <xsl:value-of select="@ID"/>  
 </xsl:attribute>  
 <xsl:attribute name="Name">  
 <xsl:value-of select="@Name"/>  
 </xsl:attribute>  
 <xsl:attribute name="DisplayName">  
 <xsl:value-of select="@DisplayName"/>  
 </xsl:attribute>  
 </xsl:element>  
 </td>  
 </tr>  
 </xsl:template>  
</xsl:stylesheet>

1. Double-click the **fieldswss.xml** file to open it for editing. Open the property pane and enter fieldswss.xslt into the Stylesheet property.
2. With the fieldswss.xml file still open, select **Show XSLT Output** from the **XML** menu in Visual Studio. When you are done, you should see a table like the one in the following diagram.
3. Browse the list of fields until you find the **Title** and **Comments** fields. Copy the contents of the **Declaration** column and paste the full declaration into the elements.xml file just inside the **<FieldRefs>** tag of the content type declaration.
4. In addition to the two built-in fields, you will also add three custom fields for the **Due Date** , **Interest Rate** and **Total Amount** of the promissory note. Add the following lines to the elements.xml file just under the opening <Elements> tag.

XML Snippet: 'Promissory Note Field References'

<!-- Declare some fields for use in the content type -->  
  
<Field ID="{E718EA82-8127-481b-A704-41EA74F14E87}"  
 Name="DueDate"  
 SourceID="http://schemas.microsoft.com/sharepoint/v3"  
 StaticName="DueDate"  
 Group="ECM401"  
 Type="DateTime"  
 Sealed="FALSE"  
 ReadOnly="FALSE"  
 Hidden="FALSE"  
 DisplayName="Due Date"  
 StorageTZ="TRUE">  
</Field>  
  
<Field ID="{4BA9D3E0-ABA5-415d-89CD-CC34B1169071}"  
 Name="InterestRate"  
 SourceID="http://schemas.microsoft.com/sharepoint/v3"  
 StaticName="InterestRate"  
 Group="ECM401"  
 Type="Number"  
 DisplayName="Interest Rate"  
 ReadOnly="FALSE"  
 Sealed="FALSE">  
</Field>  
  
<Field ID="{4C736512-4E2A-4a7b-9C96-D4624B51995E}"  
 Name="TotalAmount"  
 SourceID="http://schemas.microsoft.com/sharepoint/v3"  
 StaticName="TotalAmount"  
 Group="ECM401"  
 Type="Number"  
 DisplayName="Total Amount"  
 ReadOnly="FALSE"  
 Sealed="FALSE">  
</Field>

Note: When creating new field declarations, you need a unique GUID for each field identifier.

1. To complete your content type declaration, you must add field references to the three new fields you have just declared. When you are finished, the content definition should look like this.

<!-- Declare the content type -->  
  
<ContentType ID="0x01010070ECF29474514071B9E7F9D991D3DC05"  
Name="Promissory Note"  
Group="ECM401"  
Description="Defines fields for a simple promissory note."  
Sealed="FALSE"  
Version="0"  
ReadOnly="FALSE"  
BaseType="0x0101">  
  
<FieldRefs>  
<FieldRef ID="{fa564e0f-0c70-4ab9-b863-0177e6ddd247}" Name="Title" DisplayName="$Resources:core,Title;" Required="FALSE"/>  
<FieldRef ID="{9da97a8a-1da5-4a77-98d3-4bc10456e700}" Name="Comments" DisplayName="$Resources:core,Description;" Required="FALSE"/>  
<FieldRef ID="{E718EA82-8127-481b-A704-41EA74F14E87}" Name="DueDate" DisplayName="Due Date" Required="TRUE"/>  
<FieldRef ID="{4BA9D3E0-ABA5-415d-89CD-CC34B1169071}" Name="InterestRate" DisplayName="Interest Rate" Required="TRUE"/>  
<FieldRef ID="{4C736512-4E2A-4a7b-9C96-D4624B51995E}" Name="TotalAmount" DisplayName="Total Amount" Required="TRUE"/>  
</FieldRefs>  
  
<DocumentTemplate TargetName="/\_layouts/ecm401/promissory note.docx" />  
</ContentType>

Note: The **DocumentTemplate** tag is used to specify the template that should be used for the content type.

1. Now you are ready to test your work. Save all files and build the project.
2. Although you are not actually writing any compiled code, the post-build event batch will copy the feature definition into the appropriate locations and then call **STSADM** to install the feature into the site collection.

Note: You will still need to manually activate the feature for any sub-site in which you wish to use the promissory note content type.

1. Open the browser and navigate to thehttp://localhost:401/docmansite. Click on Site Settings from the Site Actions menu and then choose Site Collection features. You should see **ECM401 - Promissory Note** next to an Indiana Jones hat. Click the **Activate** button and then return to the home page of the site.
2. Click on the **Shared Documents** link and then select **Document Library Settings** from the **Settings** menu. Click the **Advanced Settings** link under **General Settings** and select **Yes** under **Allow management of content types?** and then click **OK** .
3. On the **Customize Shared Documents** page, under the **Content Types** section, click the **Add from existing site content types** link. You should see **Promissory Note** in the list. Double-click it and then click the **OK** button at the bottom of the page.
4. Return to the **Shared Documents** page and click the **New** button. Select **Promissory Note** from the dropdown menu. **Microsoft Word** should open with a new document based on the promissory note document template.
5. Click the **Save** button to save the document back to the document library. When Word prompts you to select a document type, select **Promissory Note** . You will now see an error message saying that required fields are missing. This is because you specified that the **Due Date** , **Interest Rate** and **Total Amount** fields are required.

Note: If you see an error message when you try to save the document, it means that the **Event Notification** service is running. This service can sometimes interfere with network connectivity from within Word 2007. Disable the service by running **net stop sens** from the command line.

1. Open the **Document Information Panel** as directed, and enter some values for these fields. When you are done, continue saving the file to the **Shared Documents** library.

**This concludes the lab exercises.**