Delaware Health and Social Services
Web Application Template Documentation v1.0
Michael Singleton – Web Manager of Base Technology

Overview

The purpose of this document is to guide Information Resource Management (IRM) developers and vendors/contractors how to develop ASP.NET web based applications to adhere with standards and direction of the organization.

The .NET solution supplied with this document consists of various projects that deal different aspects of web application development. In this document each project and their relationships to each other will be defined. The solution will also utilize Integrated Authorization System (IAS) to handle user authentication for system access.

Prerequisites

- Internet Information Services (IIS) 5.0 or later.
- Microsoft Visual Studio 2005 w/Service Pack 1
    - C# Development must be selected.
- Microsoft SQL Server 2005 Developer Edition.

Installation

In this section it is assumed you have the prerequisites installed in their default locations. If this is not the case please adjust the following steps accordingly.

1. Copy .NET solution directory DhssWebApplicationTemplate supplied with this document to:

   My Documents\Visual Studio 2005\Projects

2. Open IIS manager and create the following virtual directories:

   Virtual Directory Name

   Dhss.Division.AppName.Web

Target Location

My Documents\Visual Studio 2005\Projects\DhssWebApplicationTemplate\Dhss.Division.AppName.Web

Virtual Directory Name

Dhss.Division.AppName.Facade

Target Location

My Documents\Visual Studio 2005\Projects\DhssWebApplicationTemplate\Dhss.Division.AppName.Facade

3. Open Microsoft SQL Server 2005 Management Studio and complete the following steps:

   - Create a local database called DMS_DvdLibrary_TEST.
   - Create a SQL Server user called DvdUser with DBO rights to the DMS_DvdLibrary_TEST database.
   - Located in the My Documents\Visual Studio 2005\Projects\Dhss.Division.AppName.Database directory please run the following scripts in the order listed below:
     - Catalog_T1.sql
     - TranslateTypes_T1.sql
     - TranslateCodes_T1.sql
     - Catalog_Procedures.sql
     - Translation_Procedures.sql
     - DataImport.sql

   * Note the database steps mentioned above assumes the individual or organization contains some sort of SQL Server database administrator expertise. The exact details are beyond the scope of this document.

Testing

Now that the application template has been installed it is time to make sure the installation is working properly. Please complete the following steps:

- Open the DhssWebApplicationTemplate.sln file located in the My Documents\Visual Studio 2005\Projects\DhssWebApplicationTemplate directory with the Visual Studio 2005 IDE.
- Simply run the application by pressing the F5 key.

If everything runs properly you will be greeted with an application login page. Sections of the layout will be addressed later in this document. At this point you should be able to view the current DVD library under the “My Library” section. You can also add a DVD to the library under the “Add DVD” section.

II. Application Template/Framework Details

Overview

In this area of the document sections of the template will be described in-depth to show what can be changed and how. Then the various projects within the solution will be discussed and how the interoperate with one another in detail.

Layout

There are three general image placeholders that can be modified to reflect the theme of your web application. These are known as the Site Banner Graphic, Site Logo Graphic, and Subsystem Graphic. It is recommended that you utilize Adobe Photoshop to modify the template graphics.

Site Banner Graphic is located in the following directories:

Source File: My Documents\Visual Studio 2005\Projects\DhssWebApplicationTemplate\Dhss.Division.AppName.Web\Templates\Main\Images\PSD\sitebanner.psd

Runtime File: My Documents\Visual Studio 2005\Projects\DhssWebApplicationTemplate\Dhss.Division.AppName.Web\Templates\Main\Images\sitebanner.gif

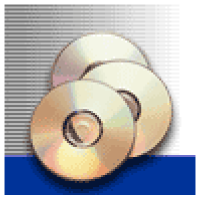
DHSS Web Application Template
Provided by Delaware Health & Social Services

* Note you should only change the text "DHSS Web Application Template" to something pertaining to the title of your web application. Other pieces of the of the graphic must be left the same.

Site Logo Graphic is located in the following directories:

Source File: My Documents\Visual Studio 2005\Projects\DhssWebApplicationTemplate\Dhss.Division.AppName.Web\Templates\Main\Images\PSD\sitelogo.psd

Runtime File: My Documents\Visual Studio 2005\Projects\DhssWebApplicationTemplate\Dhss.Division.AppName.Web\Templates\Main\Images\sitelogo.gif

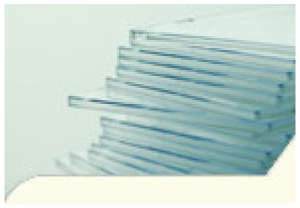


* Note the top layer in this case DVD's should only be replaced with something that reflects the theme of your web application. In some cases if you so desire you can remove the top layer and use that as your site logo graphic. This would look like the following.

Subsystem Graphic is located in the following directories:

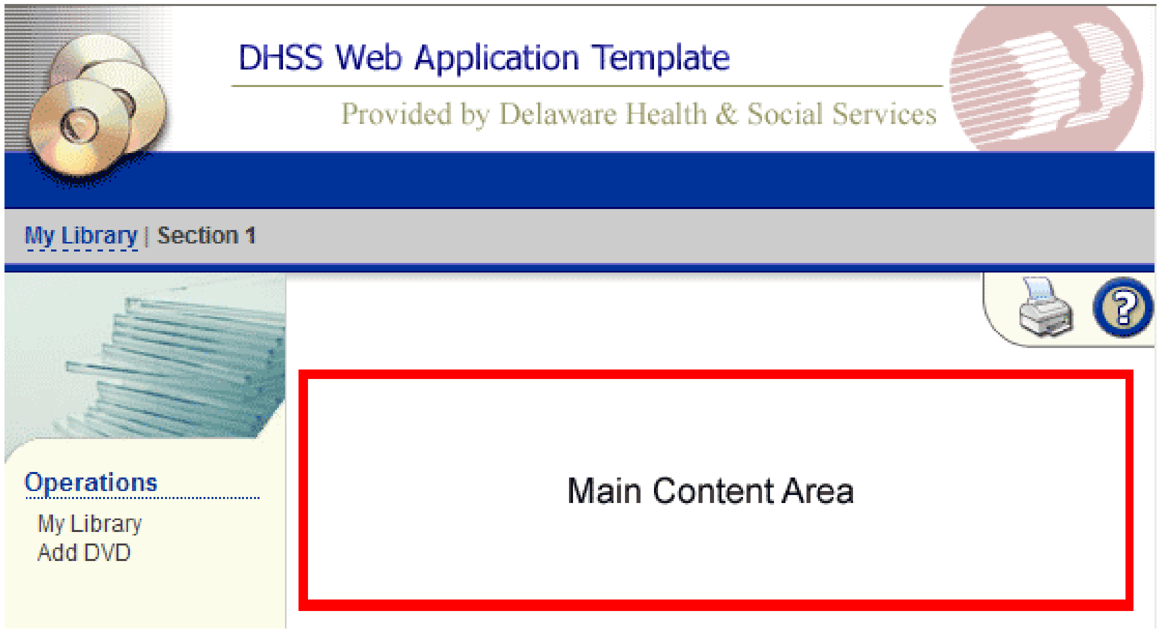
Source File: My Documents\Visual Studio 2005\Projects\DhssWebApplicationTemplate\Dhss.Division.AppName.Web\Templates\Main\Images\PSD\subsys_main.psd

Runtime File: My Documents\Visual Studio 2005\Projects\DhssWebApplicationTemplate\Dhss.Division.AppName.Web\Templates\Main\Images\subsys_main.gif



* Note each subsystem can have a different graphic pertaining to each section of the web application. This in turn will result in various runtime file generations such as: subsys_main.gif or subsys_addDvd.gif, etc… In order to specify which graphic will appear in what section this is handled in the application code. Later in this document we will discuss how this is accomplished.

The main content area of each web page is fair game to design with, but still must adhere to standards set and should utilize DHSS specific web controls whenever possible.


DHSS Web Application Template
Provided by Delaware Health & Social Services
My Library | Section 1
Operations
My Library
Add DVD
Main Content Area

* Note if you do not have appropriate software or expertise to modify layout graphics the Base Technology group will gladly assist in these efforts.

DHSS Specific Web Controls

The following is a list of DHSS specific web controls that were developed for you to utilize in your web application. The following list below will specify each current control and their purpose.

- CsvBuilder is only utilized within the code-behind of the web application. The purpose of this control is to generate comma separated value file of data from the web application to the end users web browser.

  Usage is as follows:

```
using Dhss.Framework.Web;

CsvBuilder csvBuilder = new CsvBuilder();

WriteCsvHeader(csvBuilder);
csvBuilder.NewRow();

foreach (CatalogData dvd in catalogList)
{
      WriteCsvRow(csvBuilder, dvd);
      csvBuilder.NewRow();
}

csvBuilder.ExportToBrowser("AllCatalogData.csv");

private void WriteCsvHeader(CsvBuilder csv)
{
      // Create Columns
      csv.AddColumnData("ID");
      csv.AddColumnData("foobar");
}
```

```
private void WriteCsvRow(CsvBuilder csv, CatalogData dvd)
{
        if (dvd == null)
        {
                return;
        }

        csv.AddColumnData(dvd.Id.ToString());
        csv.AddColumnData(dvd.foobar);
}
```

- DataGrid is utilized within a web form and code-behind of the web application. The purpose of this control is to standardize on how a datagrid is displayed in order to go with the web applications theme.

Web Form

```
<%@ Register TagPrefix="dhss" namespace="Dhss.Framework.Web" Assembly="Dhss.Framework.Web" %>

<div class="genericVisualContainer">
        <div class="datagridHeadline" style="float: left;">Filter by title:
<asp:HyperLink ID="lnkAf" Runat="server" NavigateUrl="?startLetter=a&stopLetter=j">A-J</asp:Hyperlink>
&nbsp;|&nbsp;
<asp:HyperLink ID="lnkKr" Runat="server" NavigateUrl="?startLetter=k&stopLetter=r">K-R</asp:HyperLink>
&nbsp;|&nbsp;
<asp:HyperLink ID="lnkSz" Runat="server" NavigateUrl="?startLetter=s&stopLetter=z">S-Z</asp:HyperLink>
&nbsp;|&nbsp;
<asp:HyperLink ID="lnk09" Runat="server" NavigateUrl="?startLetter=0&stopLetter=9">0-9</asp:HyperLink>
&nbsp;|&nbsp;
<asp:HyperLink ID="lnkAll" Runat="server" NavigateUrl="?listAll=1">All</asp:HyperLink></div><br /><br />
<dhss:DataGrid ID="dtgDvdLibrary" runat="server" AllowSorting="false" EmptyMessage="(No titles found)">
      <Columns>
            <asp:BoundColumn DataField="Id" HeaderText="Catalog Id"></asp:BoundColumn>
            <asp:BoundColumn DataField="foobar" HeaderText="Description"></asp:BoundColumn>
      </Columns>
      </dhss:DataGrid>
      </div>
</div>
```

* Note if datagrid filtering is not needed you can remove the filtering markup.

Code-Behind

The only thing left to do is bind an IList generated normally through the framework itself to the datagrid.

```
using Dhss.Framework.Web;
```

```
dtgDvdLibrary.DataSource = dvdList;
DataBind();
```

- DateLabel is utilized within a web form and code-behind of the web application. The purpose of this control is to display a date to the end user in the ShortDateTimeString format. This control also returns a DateTime value of the control of the Text property if successfully parsed as an actual DateTime value.

Web Form

```
<%@ Register TagPrefix="dhss" namespace="Dhss.Framework.Web" Assembly="Dhss.Framework.Web" %>
```

```
<dhss:DateLabel id="lblReleaseDate" runat="server"></dhss:DateLabel>
```

Code-Behind

```
using Dhss.Framework.Web;
```

```
lblReleaseDate.Value = foo;
```

- DateTextBox is utilized within a web form and code-behind of the web application. The purpose of this control is to display a date to the end user in the ShortDateTimeString format. This control also returns a DateTime value of the control of the Text property if successfully parsed as an actual DateTime value.

Web Form

```
<%@ Register TagPrefix="dhss" namespace="Dhss.Framework.Web" Assembly="Dhss.Framework.Web" %>
```

```
<dhss:DateTextBox id="txtReleaseDate" runat="server"></dhss:DateTextBox>
```

Code-Behind

```
using Dhss.Framework.Web;
```

```
txtReleaseDate.Value = foo;
```

- IntLabel is utilized within a web form and code-behind of the web application. The purpose of this control is to display an integer to the end user in the Int32 format. This control also returns a integer value of the control of the Text property if successfully parsed as an actual integer value.

Web Form

```
<%@ Register TagPrefix="dhss" namespace="Dhss.Framework.Web" Assembly="Dhss.Framework.Web" %>
```

```
<dhss:IntLabel id="lblRating" runat="server"></dhss:IntLabel>
```

Code-Behind

```
using Dhss.Framework.Web;
```

```
lblRating.Value = foo;
```

- IntTextBox is utilized within a web form and code-behind of the web application. The purpose of this control is to display an integer to the end user in the Int32 format. This control also returns a integer value of the control of the Text property if successfully parsed as an actual integer value.

Web Form

```
<%@ Register TagPrefix="dhss" namespace="Dhss.Framework.Web" Assembly="Dhss.Framework.Web" %>
```

```
<dhss:IntTextBox id="txtRating" runat="server"></dhss:IntTextBox>
```

Code-Behind

```
using Dhss.Framework.Web;
```

```
txtRating.Value = foo;
```

- SecureInputProcessor is only utilized within the code-behind of the web application. The purpose of this control is to validate whether the string parameter is a valid GUID value. If so the return value is True and False if the string is not an actual GUID.

Code-Behind

```
using Dhss.Framework.Web;
```

```
bool returnValue IsValidGuid(“foo”);
```

- TranslatedDropDownList is utilized within a web form of the web application. The purpose of this control is to display all items associated with the specific CodeGroup named in the control declaration from the database.

Web Form

```
<%@ Register TagPrefix="dhss" namespace="Dhss.Framework.Web" Assembly="Dhss.Framework.Web" %>
```

```
<dhss:TranslatedDropDownList ID="ddlRating" runat="server" CodeGroup="RTNTYP" InitialText="-- Select Rating
--" ValidatorId="validRating">
```

- TranslatedLabel is utilized within a web form and code-behind of the web application. The purpose of this control is to display the description of the assigned DataValue property by looking up the value in the specific CodeGroup named in the control declaration from the database.

Web Form

```
<%@ Register TagPrefix="dhss" namespace="Dhss.Framework.Web" Assembly="Dhss.Framework.Web" %>
```

```
<dhss:TranslatedLabel id="lblRating" Runat="server" CodeGroup="RTNTYP"></dhss:TranslatedLabel>
```

Code-Behind

```
using Dhss.Framework.Web;
```

```
lblRating.DataValue = “XXXXPG”;
```

- TranslationCache is only utilized within the code-behind of the web application. The purpose of this control is to display the description of the string parameter passed to the GetTranslation method by looking up the value in the specific CodeGroup named in the control declaration from the database.

Code-Behind

```
using Dhss.Framework.Web;
```

```
TranslationData translationData = TranslationCache.GetTranslation("RTNTYP", sRating);
```

```
if (translationData == null)
{
      return "";
}
else
{
      return translationData.CodeDescription;
}
```

- UserInterfaceUtilities is only utilized within the code-behind of the web application. The purpose of this control is to provide shortcuts for repetitive coding tasks within the web application. Currently there are three available user interface utilities: PrepareDdl, SetDropDownValue, and Redirect.

Code-Behind

- **PrepareDdl** is used initialize a DropDownListBox with its primary purpose of specifying the first ListBox item as description describing the action required. The secondary purpose is the ability to specify the validation control associated with the DropDownListBox.
    - Parameter 1: DropDownListBox control id.
    - Parameter 2: Description string.
    - Parameter 3: Validation control id.

```
using Dhss.Framework.Web;

UserInterfaceUtilities.PrepareDdl(ddlRating, "-- Select Rating --", validRating);
```

- **SetDropDownValue** is used to select a value within a DropDownListBox. If the value specified is not found the control will select the first item in the DropDownListBox as known as the DdlNoneSelected item.
    - Parameter 1: DropDownListBox control id.
    - Parameter 2: ListBox item to select string.

```
using Dhss.Framework.Web;

UserInterfaceUtilities.SetDropDownValue(ddlRating, value);
```

- **Redirect** is used to redirect the end user to a specified web page while stopping the execution of the current web page.
    - Parameter 1: Redirect path string.

```
using Dhss.Framework.Web;

UserInterfaceUtilities.Redirect(value);
```

III. Application Framework Logical Layers Programming

Overview

In this section we will use the DHSS Web Application Template as an example how the source code operates from User Interface all the way to the database layer. Each layer will be discussed and followed by a visual flowchart.

- **Dhss.Division.AppName.Web** project consists of all the user interface components and components to interact with the Dhss.Division.AppName.Facade project layer. Below various crucial pieces of this project will be discussed and how you must utilize them in your web application.
    - Subsystem.cs located in the root of the project has the task of keeping track of each Subsystem within the web application. All Subsystems should be specified before doing any of the tasks below.

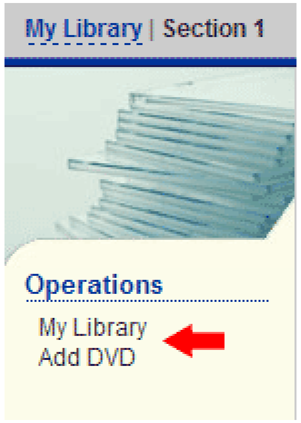
```
public enum Subsystem
{
      MyLibrary,
      Section1
}
```

    - Menu.xml located in the root of the project has a similar task as Subsystem.cs, but each Subsystem is stored in a XML file for easy reference at runtime. All Subsystems should be specified before doing any of the tasks below.

```
<?xml version="1.0" encoding="utf-8" ?>
<menu>
      <subsystem name="My Library" defaultUri="Default.aspx"></subsystem>
      <subsystem name="Section 1" defaultUri="AnotherSection.aspx"></subsystem>
</menu>
```

- DhssTemplate.Master located in the root of the project must be inherited by every web form created. This will keep the theme and functionality intact throughout the web application. When developing a new web application there are a few things you need to be aware of and modify within the DhssTemplate.Master file.
    - In the code-behind of the DhssTemplate.Master file modify each entry that has a path such as "/Dhss.Division.AppName.Web/…" to coincide with your web applications naming convention.
    - In the GetOperations method you must specify every operation(s) that associate with each Subsystem for example:

```
case Subsystem.MyLibrary:
operations = new OperationInfo[]
{
      new OperationInfo("My Library", ResolveUrl("~/Default.aspx")),
      new OperationInfo("Add DVD", ResolveUrl("~/DvdAdd.aspx"))
};

break;
```



    - In the HTML markup of the DhssTemplate.Master file areas need to be modified to reflect your web application.

```
<title>DHSS - AppName --  ← AppName needs to be changed to describe the web application.

var virtualDirectoryPath = '/Dhss.Division.AppName.Web/'; ← Path needs to be modified.
```

- Web.config located in the root of the project stores various web application settings. Some of these settings can and should be modified. Below these settings are explored.

```
<appSettings>
      <add key="VirtualDirectory" value="/Dhss.Division.AppName.Web/"/>
      <add key="AppName" value="AppName"/>
      <add key="AppVersion" value="10/22/2007"/>
      <add key="AppDatabase" value="TEST"/>
</appSettings>
```

  - VirtualDirectory needs to be modified to reflect the web applications naming convention file structure.
  - AppName needs to be modified to reflect the web applications name.
  - AppVersion needs to be modified to reflect the version date in which the web application was deployed.
  - AppDatabase needs to be modified to reflect what type of application was deployed to the end user. Whether it is a test or production (PROD) deployment.

  The following application settings are required and should only be modified to the appropriate path.

```
<add key="ActiveDirectoryFacadeUrl"
value="http://localhost/Dhss.Division.AppName.Facade/ActiveDirectoryFacade.rem"/>

<add key="TranslationFacadeUrl"
value="http://localhost/Dhss.Division.AppName.Facade/TranslationFacade.rem"/>

<add key="Evolve.TemplateConfiguration"
value="~/Templates/Email/Email.config"/>
```

  You will notice another application setting that is supplied with the template. This setting is for the purpose of telling the Dhss.Division.AppName.Web project where and how to communicate through .NET Remoting to the Dhss.Division.AppName.Facade project objects. Throughout the development of your web application you will develop various facade objects and those would be specified here in the same manner. More detail will also be provided in the Dhss.Division.AppName.Facade project detail later in this document.

```
<add key="DvdFacadeUrl" value="http://localhost/Dhss.Division.AppName.Facade/DvdFacade.rem"/>
```

- Global.asax located in the root of the project has code within the Application_Start method that loads Translation Code information from the database on the initial load of the web application. This is cached in order to give better performance for code lookups without going back and forth to the database on every web form.
- Templates Directory located in the root of the project consists of several directories and sub-directories. The files that reside in the directories should not be modified. They consist of Javascript, Stylesheets, and Images. Please note all images should be placed within the Templates\Main\Images directory that you will use in the web application. Javascript and Stylesheets that may be added during development should not change the overal layout of the web application. They also must adhere to state standards and accessibility standards. Those standards are beyond the scope of this document, but all final decisions will be made by the Web Manager of Base Technology. You will notice a directory Templates\Email this directory stores e-mail templates to utilize within the web application to submit e-mails to individuals or e-mail lists. This e-mail functionality will be discussed later in this document in detail.
- FacadeFactory located in the HelperComponents directory is used to communicate between the Dhss.Division.AppName.Web project and the Dhss.Division.AppName.Facade project. Each .NET Remoting object is referenced within this class.

  ```
  private static string m_dvdFacadeUrl =
  System.Configuration.ConfigurationManager.AppSettings["DvdFacadeUrl"];

  private static string m_activeDirectoryFacadeUrl =
  System.Configuration.ConfigurationManager.AppSettings["ActiveDirectoryFacadeUrl"];

  private static string m_translationFacadeUrl =
  System.Configuration.ConfigurationManager.AppSettings["TranslationFacadeUrl"];
  ```

  For each object there must be an Interface to interact with. Note the Interface must be created in the Dhss.Division.AppName.Common project under the Interfaces section. Actual detail on creating an Interface will be discussed in the Dhss.Division.AppName.Common section of this document.

```
public static IDvdFacade CreateDvdFacade()
{
        if (m_dvdFacadeUrl != null)
        {
                return (IDvdFacade)Activator.GetObject(typeof(IDvdFacade), m_dvdFacadeUrl);
        }
        else
        {
                return null;
        }
}
```

- **Dhss.Division.AppName.Common** project consists of two important sections DataModel and Interfaces. These classes are referenced throughout the framework to obtain or set data related to the task at hand. Below various crucial pieces of this project will be discussed and how you must utilize them in your web application.
    - DataModel is a class which represents the data elements to its associated database table. Your web application will have many DataModel classes throughout development. Generally this information automatically generated through a code generation tool called CodeSmith. This tool will be discussed in-depth later in this document. Each data element is declared and has a public get/set method:

```
private string m_title = "";

public string Title
{
        get
        {
                return m_title;
        }
        set
        {
                m_title = value;
                MarkDirty();
        }
}
```

Another important element specified within the class is required elements of data. This is automatically checked through the framework in the ValidatedExtendedData method.

```
ValidateLength("Title", m_title, 1, 100); ← Note 1 - 100 character value is required.
ValidateLength("Description", m_description, 0, 200); ← Note 0 denotes a nullable field.
```

- Interfaces is a class which represents new methods created during development. Each facade class has an interface located in Dhss.Division.AppName.Common\Interfaces. Every interface must have a prefix of the letter I for identification purposes. Below is an example of the IDvdFacade interface method declarations. Note it utilizes the DataModel for passing data as parameter.

  ```
  public interface IDvdFacade
  {
        bool SaveCatalogData(CatalogData catalogData);
        IList ListDvds();
        CatalogData RetrieveCatalogDataById(int id);
  }
  ```

- ApplicationConstants is a class that is available to store any type of application constants. Below is an example of declaring a constant to be used the web application and how to utilize it.

  ```
  public static readonly string string1 = "VALUE"; ← Declaration

  if (foo == ApplicationConstants.string1)
  {
        bSuccess = true;
  }
  ```

- **Dhss.Division.AppName.Facade** project ties together various projects and can reside on a separate web server in order to have an n-tier web application environment. Once a request reaches the facade layer each method makes a call to the appropriate class within the Dhss.Division.AppName.BunsinessLogic project. The responsibility of the BusinessLogic layer will be discussed later in this document. Each facade class and a corresponding interface class as mentioned in the Dhss.Division.AppName.Common section. Below is an example from the template that calls BusinessLogic class called Dvd.

  ```
  public bool SaveCatalogData(CatalogData catalogData)
  {
        Dvd catalog = new Dvd(catalogData);

        return catalog.Add();
  }
  ```

  Also note that each facade must inherit its associated interface. An example of this is:

  ```
  public class DvdFacade : MarshalByRefObject, IDvdFacade
  ```

- **Dhss.Division.AppName.BusinessLogic** project has the main purpose making the appropriate calls to the Dhss.Division.AppName.DataAccess project in order to retrieve information from the database. The secondary purpose is to make any business logic decisions or validations in the web application. Below is an example from the template that calls the DataAccess class called CatalogDvdDal.

```
public IList ListDvds()
{
        CatalogDal dvdDal = (CatalogDal)GetDalObject();

        return dvdDal.ListDvds();
}
```

- **Dhss.Division.AppName.DataAccess** project has the sole purpose of interacting with the database. Most of the database call functionality is built into the framework so the developer has little need to worry about database related code. Below is an example the DataAccess class calling a database stored procedure.

```
public IList ListDvds()
{
        IDataParameter[] parameters = new IDataParameter[]
        {
        };

        return ListByCriteria("Catalog_List_S1", parameters);
}
```

- **Dhss.Division.AppName.Database** project a nothing but a repository for all database related SQL scripts and files used for the web application.

IV. Example Programming Exercise

Overview

In this section we walk through an example exercise. We will add a new data element to capture on the DvdAdd.aspx web form called ratio. This DropDownListBox control will house two initial options Widescreen and Standard ratio's. The following is a step-by-step explanation of what needs to be accomplished to add this new element.

Exercise

1. First thing we need to do is add the two new ratio options to the Translate tables in the database. This can be done in many ways, but I recommend opening SQL Server Management Studio and modifying the tables directly. This will give a better idea of what data is actually housed in the tables.
    a. Open the **TranslateTypes_T1** table and you will see one record that represents the DVD rating type. Now you must enter a new record for the ratio type. Most of the required fields are standard InsertedBy and LastSavedBy Id's are generally 1 on initial application creation. One could think of it as an "Administrator" and could easily develop an interface within the application to update the translation tables and store the current users Id. In regards to the date fields they are initially the same value and are the date and time when the record was created. One very important thing to keep in mind is that the framework expects the *Type_CODE field to always have a six character value in length*. For this exercise we will make the Type_CODE and Description_TEXT values RTOTYP and Ratio Types. Below is valid example SQL Insert script to perform this task.

```
INSERT INTO [dbo].[TranslateTypes_T1]
([FirstInsertedBy_IDNO],
[FirstInserted_DTTM],
[LastSavedBy_IDNO],
[LastSaved_DTTM],
[Concurrency_DTTM],
[Type_CODE],
[Description_TEXT],
[Editable_INDC])
VALUES (1, '20080101 11:00:00.000', 1, '20080101 11:00:00.000', '20080101 11:00:00.000', 'RTOTYP',
'Ratio Types', 1)
```

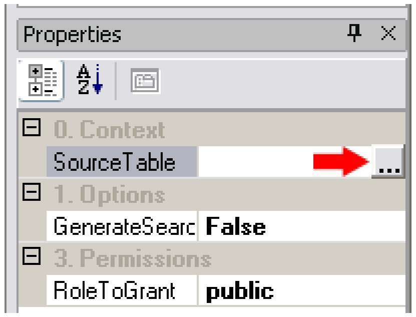
b. Open the **TranslateCodes_T1** table and you will see four records containing various rating codes. You will notice a lot of the same fields as in the TranslateTypes_T1 table. There are four fields you must familiarize yourself with. The TranslateType_IDNO field value references one record stored in the TranslateTypes_T1 table. Thus signifying what type of translate type it is associated with. The Key_CODE field value stores a unique code for each translate type group. Note this *field must always have a six character value in length*. The Sequence_NUMB field value represents a numeric order for each code of a particular translate group. This sequence will display translation codes to the end user via the numeric range from least to greatest. The Active_INDC field value is either true of false. True to set the code as active and False to inactivate the code. We want to add two new codes for Widescreen and Standard ratios for this exercise. You can add them through SQL Server Management Studio or via the following scripts below.

```
INSERT INTO [dbo].[TranslateCodes_T1] ([FirstInsertedBy_IDNO], [FirstInserted_DTTM],
[LastSavedBy_IDNO], [LastSaved_DTTM], [Concurrency_DTTM], [TranslateType_IDNO], [Key_CODE],
[LongDescription_TEXT], [Sequence_NUMB], [Active_INDC], [Editable_INDC])
VALUES (1, '20080101 11:00:00.000', 1, '20080101 11:00:00.000', '20080101 11:00:00.000', 2, 'XXSTND',
'Standard', 10, 1, 0)

INSERT INTO [dbo].[TranslateCodes_T1] ([FirstInsertedBy_IDNO], [FirstInserted_DTTM],
[LastSavedBy_IDNO], [LastSaved_DTTM], [Concurrency_DTTM], [TranslateType_IDNO], [Key_CODE],
[LongDescription_TEXT], [Sequence_NUMB], [Active_INDC], [Editable_INDC])
VALUES (1, '20080101 11:00:00.000', 1, '20080101 11:00:00.000', '20080101 11:00:00.000', 2, 'XXXXWS',
'Widescreen', 20, 1, 0)
```

2. Now that we have the new ratios added to the translation tables. We need to have a way to store the ratio for each DVD. In this example all of the DVD data is stored in the Catalog_T1 table. Currently it does not have a field to store ratios. We need to create it in the table and call it Ratio_CODE.
   a. This can be done through design view in SQL Server Management Studio or through the following SQL script below. Note the field must be a six character field in length in order work correctly the framework and translate tables within the database. For this exercise we are going to make the field nullable, but in a real world scenario you may want to make the field required. We will update the few records to reflect a standard ratio for simplicity sake.

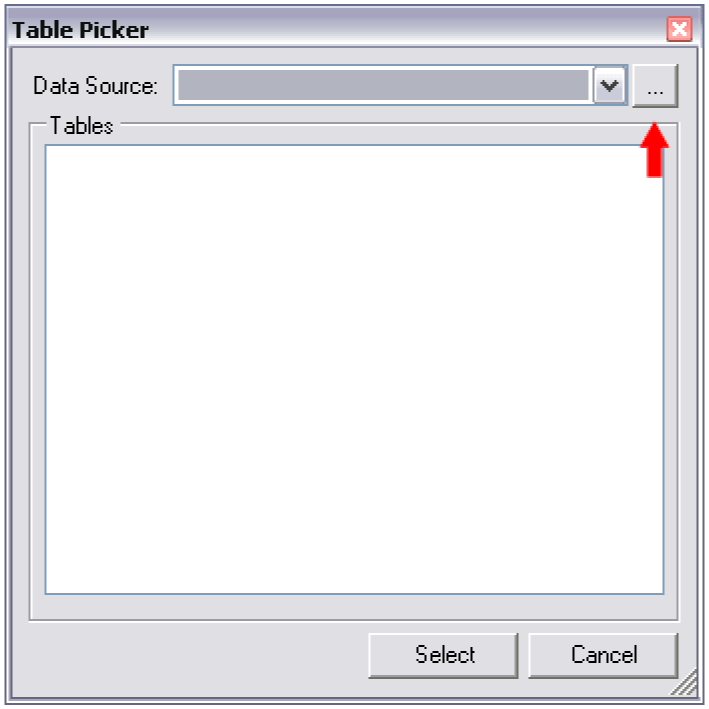
```
Alter Table dbo.Catalog_T1
Add Ratio_CODE char(6)

UPDATE Catalog_T1 SET Ratio_CODE = 'XXSTND'
```

3. The database table modifications are now complete. This leaves us with a few out of sync areas within our web application source code and database stored procedures. There are a few ways to update these areas one being manually and the other more convenient faster by utilizing CodeSmith tools. CodeSmith is a software application that generates source code related output to incorporate into an application. DHSS has pre-built templates that deal with web application framework. They are called: BusinessEntity.cst, DataAccessLayer.cst, and StoredProcedures.cst. These templates are included with this document. In the following steps we will go over how to update the database stored procedures and the web application source code. Note DHSS has CodeSmith license(s) please verify with your manager if you are licensed for this software. In any case we will go over what needs to be added and also show you how to make the modifications with CodeSmith.
    a. Stored Procedures are out of sync that relate to the Catalog_T1 table. In order update these procedures you must do the following:
        i. Open the CodeSmithStudio.exe located in the root of your CodeSmith installation directory.
        ii. Open the file StoredProcedures.cst file supplied with this document.
        iii. You will notice a properties section located on the right side of the screen. The appropriate Source Table needs to be selected. Simply click on the ellipses and follow the instructions below.

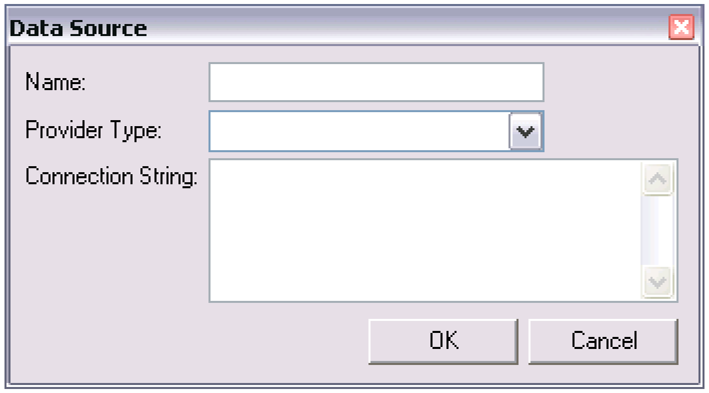


0. Context
SourceTable
1. Options
GenerateSearc False
3. Permissions
RoleToGrant public

Choose the ellipses to create a new data source to the database.

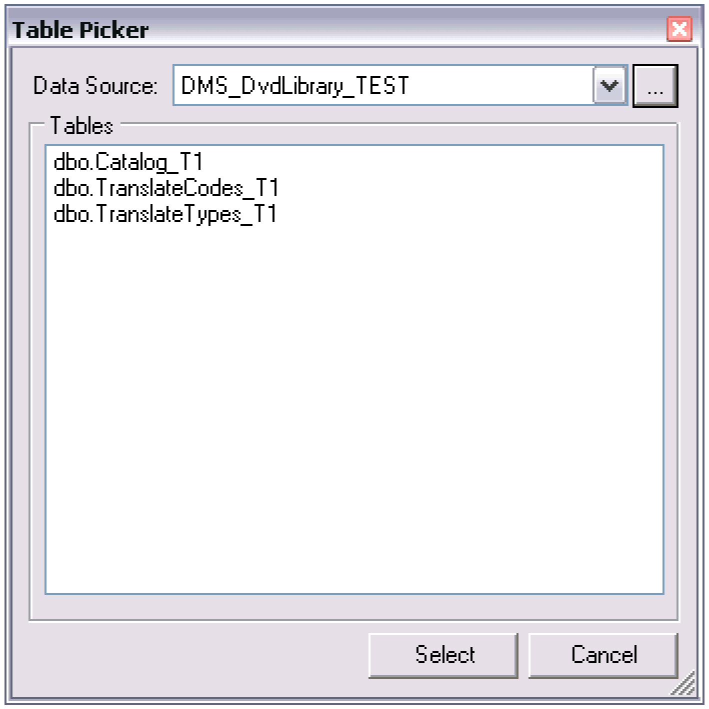


Next click the Add button and you will see the following screen. Enter the values specified below.



Name: DMS_DvdLibrary_TEST
Provider Type: SqlSchemaProvider
Connection String: server=localhost;database=DMS_DvdLibrary_TEST;uid=DvdUser;pwd=dvduser;

Next click the OK button and navigate back to the table picker screen and select the data source DMS_DvdLibrary_TEST. You should see a screen as the one below. If you do not try closing the table picker screen and re-opening it. This will refresh any new connections to the database.



DMS_DvdLibrary_TEST

dbo.Catalog_T1
dbo.TranslateCodes_T1
dbo.TranslateTypes_T1

Since the Catalog_T1 was the one modified you need to choose the entry dbo.Catalog_T1 and click the select button.

The CodeSmith template is now ready to be run. Press the F5 Key and this will generate the output of the appropriate SQL statements that need to be executed. The final step is to copy these SQL statements into SQL Server Management Studio and execute them against the database.

b. Next we must update business entity data model class this can be done through CodeSmith or manually. Follow the following steps in order to do so.
   i. Open the CodeSmithStudio.exe located in the root of your CodeSmith installation directory.
   ii. Open the file BusinessEntity.cst file supplied with this document.
   iii. You will notice a properties section located on the right side of the screen. The appropriate Source Table needs to be selected in this case the Catalog_T1. Simply click on the ellipses and follow the instructions above.
   iv. Then you will be asked for a namespace enter “Dhss.Division.AppName” and press the F5 Key. This will create the output class that needs to replace CatalogData.cs file contents located in the Dhss.Division.AppName.Common.DataModel project directory. Upon replacing the contents you will

notice the new data element Ratio has been automatically supplied to be utilized within the business entity data model class. Note that this refreshes the ValidateExtendedData method. You should verify that each validation appropriated matches the database field it relates to. In this case the Description property is a nullable field so make sure to change the min value of 1 to a 0 length.

```
ValidateLength("Title", m_title, 1, 100);
ValidateLength("Description", m_description, 1, 200);

ValidateCode("Rating", m_rating);
ValidateCode("Ratio", m_ratio);
```

c. The final step that utilizes CodeSmith is the modifications required to change the Data Access Layer(DAL) CatalogDal located in the Dhss.Division.AppName.DataAccess project. Follow the steps below or you must enter the changes manually.
   i. Open the CodeSmithStudio.exe located in the root of your CodeSmith installation directory.
   ii. Open the file DataAccessLayer.cst file supplied with this document.
   iii. You will notice a properties section located on the right side of the screen. The appropriate Source Table needs to be selected in this case the Catalog_T1. Simply click on the ellipses and follow the instructions above. Then you will be asked for a namespace enter “Dhss.Division.AppName” and press the F5 Key. This will create the output class that needs to replace CatalogDal.cs file contents located in the Dhss.Division.AppName.DataAccess project directory. Note that if you have custom DAL methods not to overwrite them with the newly generated CatalogDal code.
   iv. After finishing updating the web application source code. Do a few rebuild of the web application. If an error occurs verify each step was performed correctly in this section.

d. Now that ratio data element has been incorporated into the web application code. The final step is to update the Dhss.Division.AppName.Web project. Giving the ability for the end user to select a ratio within the web form.
   i. The first step is you will need to update the DvdAdd.aspx source to reflect the DropDownListBox that displays the different types of ratios. Below is the web form source code that needs to be added preferably underneath the rating DropDownListBox.

```
<div class="formRow">
      <div class="formColumnLabel">
            <label for="ddlRatio">Ratio:</label>
      </div>
      <div>
            <dhss:TranslatedDropDownList ID="ddlRatio" runat="server" CodeGroup="RTOTYP"
            InitialText="-- Select Ratio --">
            </dhss:TranslatedDropDownList>
      </div>
```

```
</div>
```

ii. Next in the code-behind file DvdAdd.aspx.cs you will need to add one line of code to capture the ratio value selected by the user.

```
catalogData.Ratio = ddlRatio.SelectedValue;
```

iii. Now we will want to show the ratio within the DVD library datagrid. Open the Default.aspx web form source and add the following source beneath rating section.

```
<asp:TemplateColumn HeaderText="Ratio" ItemStyle-Wrap="False">
      <ItemTemplate>
            <dhss:TranslatedLabel runat="server" CodeGroup="RTOTYP" DataValue='<%#
            DataBinder.Eval(Container.DataItem, "Ratio") %>'
            ID="lblRatioTypeTranslation"></dhss:TranslatedLabel>
      </ItemTemplate>
</asp:TemplateColumn
```

iv. Another are that needs to be updated is the DvdDetail.aspx web form to display the ratio when a DVD is selected.

1. Open the web form and add the following piece of source code beneath the ratings section.

```
<div class="staticFormRow">
      <div class="formColumnLabel">
            <label for="lblRatio" class="staticLabel">Ratio:</label>
      </div>
<div>
  <dhss:TranslatedLabel ID="lblRatio" CodeGroup="RTOTYP" runat="server" /></div>
</div>
```

2. Secondly we must assign the ratio value to the label control in the DvdDetail.aspx.cs code-behind file. Enter the following line of code below ratings section.

```
lblRatio.DataValue = catalogData.Ratio;
```

v. Finally the DvdEdit.aspx web form needs to be able to handle the addition of the ratio data element.

1. Open the web form and add the following piece of source code beneath the ratings section.

```
<div class="formRow">
      <div class="formColumnLabel">
            <label for="ddlRatio">Ratio:</label>
      </div>
      <div>
            <dhss:TranslatedDropDownList ID="ddlRatio" runat="server"
            CodeGroup="RTOTYP" InitialText="-- Select Ratio --">
            </dhss:TranslatedDropDownList>
      </div>
</div>
```

2. Secondly we must assign the ratio value to the DropDownListBox control in the DvdEdit.aspx.cs code-behind file. Enter the following line of code below ratings section.

```
ddlRatio.SelectedValue = catalogData.Ratio;
```

V. Miscellaneous Tips

Various informational howto tips will be addressed in this section at a later date. Topics will include security integration with the DHSS IAS application, E-Mail generation templates, and other topics.

1. In regards to database table design in accordance with the DHSS Framework each table for auditing reasons must include the following fields in this exact order:

```
[PrimaryKey_IDNO] [int] IDENTITY(1,1) NOT NULL
[FirstInsertedBy_IDNO] [int] NOT NULL
[FirstInserted_DTTM] [datetime] NOT NULL
[LastSavedBy_IDNO] [int] NOT NULL
[LastSaved_DTTM] [datetime] NOT NULL
[Concurrency_DTTM] [datetime] NOT NULL
```

* Note the PrimaryKey_IDNO field name can be changed accordingly for each table such as <name>_IDNO.

2. There are a few methods within the DHSS Framework that you should be aware of in order to interact with a database. These methods come from the PersistantBussinessObject class. Upon each new instance of a BusinessLogic class there is a set of methods available to be utilized which will be discussed below. For examples on usage please refer to the DHSS Web Application Template. A good place to start is the DvdFacade class.
    a. **Add() –** Add takes the current information supplied by the business entity data class and performs an insert into the table specified in the corresponding DataAccess class. The return is a boolean value of true or false as to whether the action executed successfully.
    b. **Save() –** Save takes the current information supplied by the business entity data class and performs an update to the table specified in the corresponding DataAccess class. The return is a boolean value of true or false as to whether the action executed successfully.
    c. **Load(int) –** Load takes the integer value passed and selects a single record by primary seed (The first field in the table) from the table specified in the corresponding DataAccess class. The return is the business entity class loaded with all the data elements for the record selected.
    d. **List() –** List brings back all the records within the corresponding table. The return is in IList collection.

If the need presents itself and you need to write a custom stored procedure. There are two methods available to you through the DataAccessObject class utilized in each DataAccessLayer class. You can view a commented out example in the CatalogDal class.

a. **ListByCriteria(string, IDataParameter[ ]) –** ListByCriteria consists of two paramenters the first represents the name of the custom stored procedure. The second parameter is a collection of parameters the stored procedure expects. The return is a IList collection of this information retrieved.
b. **RunProcedure(string, IDataParameter[ ]) –** RunProcedure consists of two paramenters the first represents the name of the custom stored procedure. The second parameter is a collection of parameters the stored procedure expects. The return is void.

VI. General Usage Agreement

Delaware Health and Social Services
1901 North DuPont Highway
New Castle, DE 19720
Biggs Building
RE: Web Manager of Base Technology
February 21, 2008

By utilizing the DHSS Framework, Web Application Template, and this document you must adhere to the following guidelines:

- The software and information conveyed in this document cannot be distributed to 3rd party vendors/contractors without written consent from the Web Manager of Base Technology.
- Software and related documentation cannot be used for private use outside the State of Delaware.
- Software and related documentation must not be modified or changed in anyway without written consent of the Web Manager of Base Technology. If suggestions or comments are required they must be in writing and submitted for review to the Web Manager of Base Technology.

Sincerely,

ACCEPTED AND AGREED:

Please Print:

First Name: ______________________ Last Name: ______________________

Signature: ______________________ Date: _____________________________