5.1.0 – Message User Control

Introduction

In lessons 4.2.0 and 5.1.0, you created a working web form. In lesson 4.3.0, you were shown how to create and configure a `ListView` control. In this lesson, you will be introduced to a way to display some of the errors on the web page, or to display information messages to the user. The code you will use you do not have to memorize, or code for yourself; it is provided free.

User Controls

User Controls are reusable web controls; once created, they can be used on many different web forms. They differ from web forms in that they need to be added to a web form to work.

On Moodle, in the **Supporting Files (5.1.0)** folder, you will find a file called **MessageUserControl.zip**. This file contains the free code for the User Control we need for this course. You do not need to know how this code works, only how to add it to a web form, and to use it in your code.

There are many methods in this control (some are overloaded to make the control easier to use):

- **`ShowInfo()`**: Displays a message in a message panel
- **`TryTun()`**: Process a request through a callback delegate within a `try/catch` block
- **`HandleDataBoundException()`**: Checks for an exception from an `ObjectDataSource` event and handles it by showing the details of the exception

Adding the MessageUserControl

1. Download the ZIP file from Moodle and extract the files to a location on your computer where you can easily find them. You should see the following files:

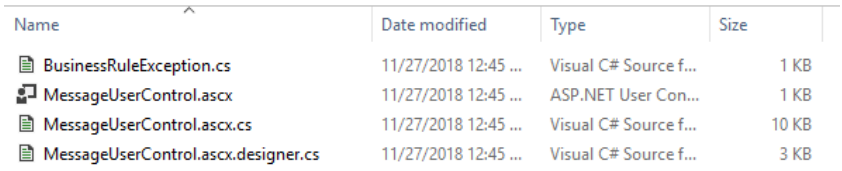


Figure 1: Message User Control Files

2. Open your Visual Studio solution.
3. On your web site project, add a new folder called **`UserControls`**.
4. Right-click this new folder and add an existing item.

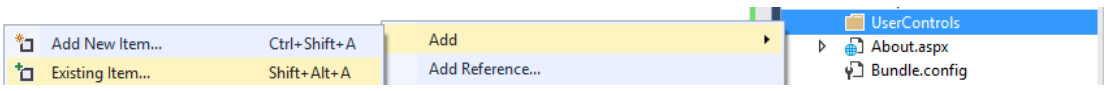


Figure 2: Add Existing Item

5. Browse to the location where you extracted the ZIP file to and select **ONLY** the **`MessageUserControl.ascx`** file.
6. Once added, expand the **`MessageUserControl.ascx`** to show the **`MessageUserControl.ascx.cs`** and **`MessageUserControl.ascx.designer.cs`** files:

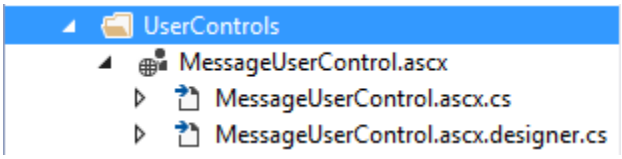


Figure 3: MessageUserControl Files Added

7. Open the **`MessageUserControl.ascx.cs`** file and expand the Additional Namespaces. Notice the code on line 89, and that there is an error (there are other similar errors throughout this file):

```
79          public void HandleDataBoundException(ObjectDataSourceStatusEventArgs e)
80          {
81              var ex = e.Exception;
82              if (ex is TargetInvocationException)
83                  ex = ex.InnerException;
84              if (ex is DbEntityValidationException)
85              {
86                  HandleException(e.Exception as DbEntityValidationException);
87                  e.ExceptionHandled = true;
88              }
89              else if (ex is BusinessRuleException)
90              {
91                  HandleException(ex as BusinessRuleException);
92                  e.ExceptionHandled = true;
93              }
94              else if (ex is Exception)
95              {
96                  HandleException(ex as Exception);
97                  e.ExceptionHandled = true;
98              }
99          }
```

8. To fix all these errors we need to add the **`BusinessRuleException.cs`** file to the **BLL** folder of the **eStoreSystem** project. Right-click the folder and select **Add → New Item…**, browse to where you extracted the MessageUserControl.ZIP files to, then select the **`BusinessRuleException.cs`** code file and press Add. This will give you:

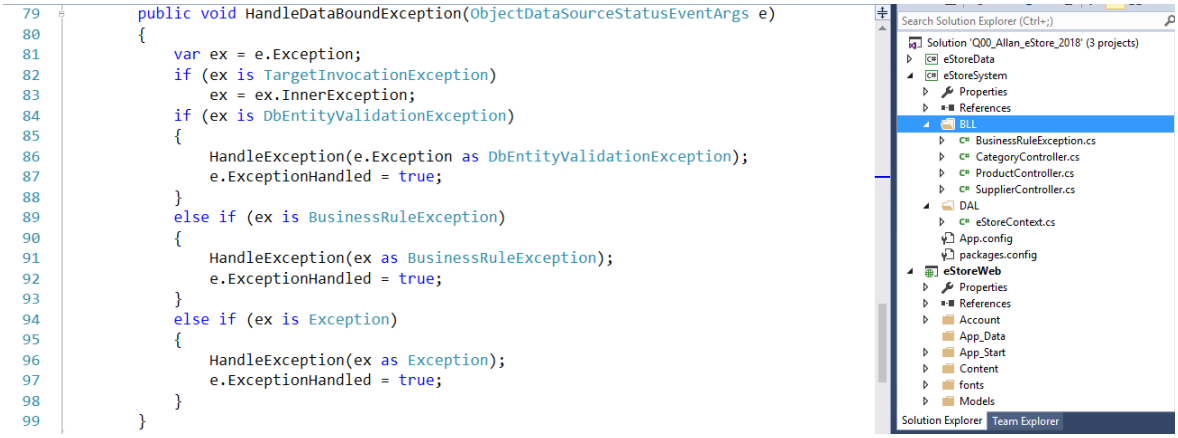


Figure 4: BusinessRuleException Class Added

The User Control is now ready to be added to any web form we need it on (take time to look at the **MessageUserControl.asox.cs** file to see the error on line 89 has been fixed). We will add it to the ListView web form we created in the last 2 lessons (**ShoppingCart.aspx** and **PurchaseOrders.aspx**).

ShoppingCart.aspx

To add a User Control to a web form is easy, just drag the control file from its location in the Solution Explorer and drop it near the top of the web form (Source View). For us, we will create a new `div row`:

Listing 1: New `div row`

```
<div class="jumbotron">
    <h1>Products For Sale</h1>
    <p>
        This web form will load a DropDownList of Categories which the Customer can
select from.
        Once a Category is selected, a list of Products for the Category will be
displayed.
    </p>
</div>
<div class="row">

</div>
```

New <div row>

Click and drag the **MessageUserControl.ascx** file from the Solution Explorer to this new `div row`:

Listing 2: MessageUserControl Added to ShoppingCart.aspx

```
<%@ Register Src="~/UserControls/MessageUserControl.ascx" TagPrefix="uc1"
TagName="MessageUserControl" %>


<asp:Content ID="Content1" ContentPlaceHolderID="MainContent" runat="server">
    <div class="jumbotron">
        <h1>Products For Sale</h1>
        <p>
            This web form will load a DropDownList of Categories which the Customer can
select from.
```

```
            Once a Category is selected, a list of Products for the Category will be
displayed.
        </p>
    </div>
    <div class="row">
        <uc1:MessageUserControl runat="server" id="MessageUserControl" />
    </div>
```

You will see the <%@ Register line (should be line 3 or 4). This line registers the User Control for use on the web form. Next, you will see the line of code in the new `div row`:

```
<uc1:MessageUserControl runat="server" ID="MessageUserControl" />
```

This is the User Control. The `uc1` is defined by the `TagPrefix="uc1"` when the control was registered.

You may see a green underline underneath the code. This should disappear once the solution runs.

Using the MessageUserControl

For the `MessageUserControl` to be useful, we need to make some modifications to our web form by replacing the `GridView` control (**`ProductListGV`**), with a `ListView` called **`ProductListLV`**. Follow the steps from Lesson 4.3.0 to add and configure the `ListView` so that is will display all the data; do not forget to delete the unneeded templates. Also, change the Label that displays the cost to a `TextBox`, with decimal formatting for both the **`ItemTemplate`** and the **`AlternatingItemTemplate`**. When completed, before using the `MessageUserControl`, your output should be like:

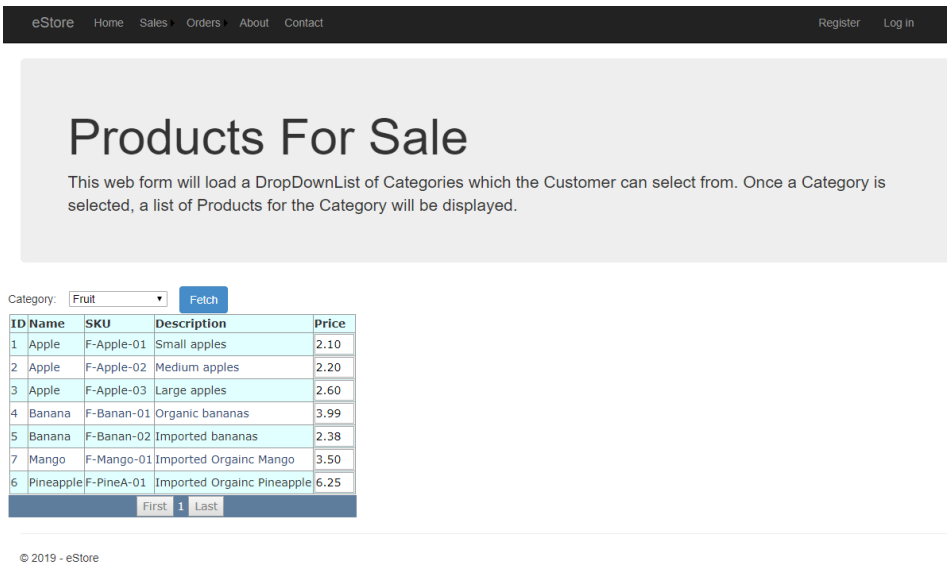


Figure 5: ShoppingCart with ListView

First, we need to modify the `LayoutTemplate` by adding another column. The column we will add will have a `LinkButton` control. When we pressed, the data from that row on the `MessageUserControl`:

Listing 3: Modified LayoutTemplate

```
<LayoutTemplate>
    <table runat="server">
        <tr runat="server">
            <td runat="server">
                <table runat="server" id="itemPlaceholderContainer" style="background-
color: #FFFFFF; border-collapse: collapse; border-color: #999999; border-style: none;
border-width: 1px; font-family: Verdana, Arial, Helvetica, sans-serif;" border="1">
                    <tr runat="server" style="background-color: #E0FFFF; color:
#333333;">
                        <th runat="server">ID</th>
                        <th runat="server">Name</th>
                        <th runat="server">SKU</th>
                        <th runat="server">Description</th>
                        <th runat="server">Price</th>
                        <th runat="server"></th>
                    </tr>
                    <tr runat="server" id="itemPlaceholder"></tr>
                </table>
            </td>
        </tr>
        <tr runat="server">
            <td runat="server" style="text-align: center; background-color: #5D7B9D;
font-family: Verdana, Arial, Helvetica, sans-serif; color: #FFFFFF">
                <asp:DataPager runat="server" ID="DataPager1">
                    <Fields>
                        <asp:NextPreviousPagerField ButtonType="Button"
ShowFirstPageButton="True" ShowNextPageButton="False"
ShowPreviousPageButton="False"></asp:NextPreviousPagerField>
                         <asp:NumericPagerField></asp:NumericPagerField>
                         <asp:NextPreviousPagerField ButtonType="Button"
ShowLastPageButton="True" ShowNextPageButton="False"
ShowPreviousPageButton="False"></asp:NextPreviousPagerField>
                    </Fields>
                </asp:DataPager>
            </td>
        </tr>
    </table>
</LayoutTemplate>
```

New column; no header text

Next, modify the `ItemTemplate` to be:

Listing 4: Modified ItemTemplate

```
<ItemTemplate>
    <tr style="background-color: #E0FFFF; color: #333333;">
        <td>
            <asp:Label Text='<%# Eval("ID") %>' runat="server" ID="IDLabel" /></td>
        <td>
             <asp:Label Text='<%# Eval("Name") %>' runat="server" ID="NameLabel" /></td>
        <td>
            <asp:Label Text='<%# Eval("SKU") %>' runat="server" ID="SKULabel" /></td>
        <td>
            <asp:TextBox Text='<%# string.Format("{0:0.00}", Eval("Cost")) %>'
runat="server" ID="CostTextBox" Width="50" /></td>
```

```
        <td>
            <asp:TextBox Text='<%# Eval("ROL") %>' runat="server" ID="ROLTextBox"
Width="50" /></td>
        <td>
            <asp:TextBox Text='<%# Eval("QOH") %>' runat="server" ID="QOHTextBox"
Width="50" /></td>
        <td>
            <asp:TextBox Text='<%# Eval("Ordered") %>' runat="server" ID="OrderedTextBox"
Width="50" /></td>
        <td>
            <asp:LinkButton runat="server" ID="SelectButton" CommandName="Select"
Text="Select" /></td>
    </tr>
</ItemTemplate>
```

New column

Make a similar change to the `AlternatingItemTmplate`:

Listing 5: AlternatingItemTemplate Modified

```
<AlternatingItemTemplate>
    <tr style="background-color: #FFFFFF; color: #284775;">
        <td>
            <asp:Label Text='<%# Eval("ID") %>' runat="server" ID="IDLabel" /></td>
        <td>
             <asp:Label Text='<%# Eval("Name") %>' runat="server" ID="NameLabel" /></td>
        <td>
             <asp:Label Text='<%# Eval("SKU") %>' runat="server" ID="SKULabel" /></td>
        <td>
             <asp:TextBox Text='<%# string.Format("{0:0.00}", Eval("Cost")) %>'
runat="server" ID="CostTextBox" Width="50" /></td>
        <td>
             <asp:TextBox Text='<%# Eval("ROL") %>' runat="server" ID="ROLTextBox"
Width="50" /></td>
        <td>
             <asp:TextBox Text='<%# Eval("QOH") %>' runat="server" ID="QOHTextBox"
Width="50" /></td>
        <td>
             <asp:TextBox Text='<%# Eval("Ordered") %>' runat="server"
ID="OrderedTextBox" Width="50" /></td>
        <td>
             <asp:LinkButton runat="server" ID="SelectButton" CommandName="Select"
Text="Select" /></td>
    </tr>
</AlternatingItemTemplate>
```

New column

Finally, we need to make the Button work. To do that we need to add a command parameter to the `ListView`. Add the `OnItemCommand` parameter just before the closing tag. Once you have the `=""`, Visual Studio will prompt you for an event method:

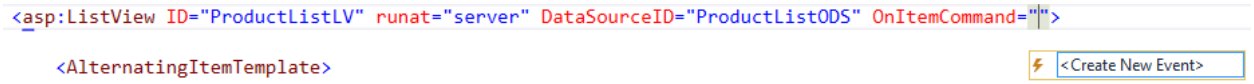


Figure 6: Add OnItemCommand

We need a new event, so press `<Create New Event>`. What this will do is complete the parameter for the `ListView` and create an event method in the code behind page

(**ProductsForSupplier.aspx.cs**). When the `ListView` line looks like the code below, open the code behind page:

Listing 6: OnItemCommand Paramnter Added

```
<asp:ListView ID="ProductListLV" runat="server" DataSourceID="ProductListODS"
OnItemCommand="ProductListLV_ItemCommand">
```

Listing 7: Event Method Stub Created

```
protected void ProductListLV_ItemCommand(object sender, ListViewCommandEventArgs e)
{

}
```

In this event method, we want to get all the data from a row in the `ListView` control. In the selected row, we have asp web controls (`Label` and `TextBox` controls). We can get the data from these controls through their control IDs.

Listing 8: ProductListLV_ItemCommand Event Code

```
protected void ProductListLV_ItemCommand(object sender, ListViewCommandEventArgs e)
{
    string msg = null;
    //Get the data from the selected displayed row of the ListView
    ListViewDataItem lvItem = e.Item as ListViewDataItem;
    //First column (0) Label control
    //The "IDLabel" MUST be he same as the ID from the ItemTemplate row
    int id = int.Parse((lvItem.FindControl("IDLabel") as Label).Text);
    //Second column (1) Label control
    string name = (lvItem.FindControl("NameLabel") as Label).Text;
    //Third column (2) Label control
    string sku = (lvItem.FindControl("SKULabel") as Label).Text;
    //Fourth column (3) Label control
    string description = (lvItem.FindControl("DescriptionLabel") as Label).Text;
    //Fifth column (4) TextBox
    decimal price = decimal.Parse((lvItem.FindControl("PriceTextBox") as TextBox).Text);
    //Write the msg to the MessageUserControl
    msg = "ID: " + id.ToString()
        + ", Name: " + name
        + ", SKU: " + sku
        + ", Description: " + description
        + ", Price: " + price.ToString();
    MessageUserControl.ShowInfo(msg);
}//eom
```

The data displayed on a `ListView` is text (i.e. string values).

When you run this web form, select a category to get a listing of products, then select a product, you should get something like:

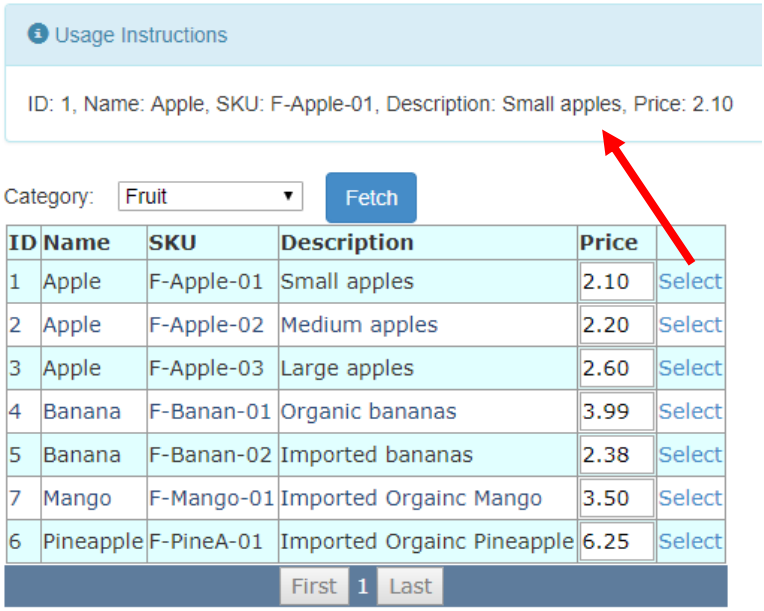


Figure 7: Web Form Running Showing MessageUserControl.ShowInfo()

MessageUserControl.ShowInfo()

There are 2 overloaded methods for the `ShowInfo()`:

Listing 9: ShowInfo(message)

```
public void ShowInfo(string message)
{
    ShowInfo(STR_TITLE_UsageInstructions, message);
}
```

Listing 10: ShowInfo(title, message)

```
public void ShowInfo(string title, string message)
{
    ShowInfo(message, title, STR_TITLE_ICON_info, STR_PANEL_info);
}
```

If you do not like the default title, `Usage Instructions`, you could call the method in Listing 11. You will need to pass in a string value for your title. For example, change the line

```
MessageUserControl.ShowInfo(msg);
```

To be:

```
MessageUserControl.ShowInfo("Selected Product Information", msg);
```

Making this change give us the following:

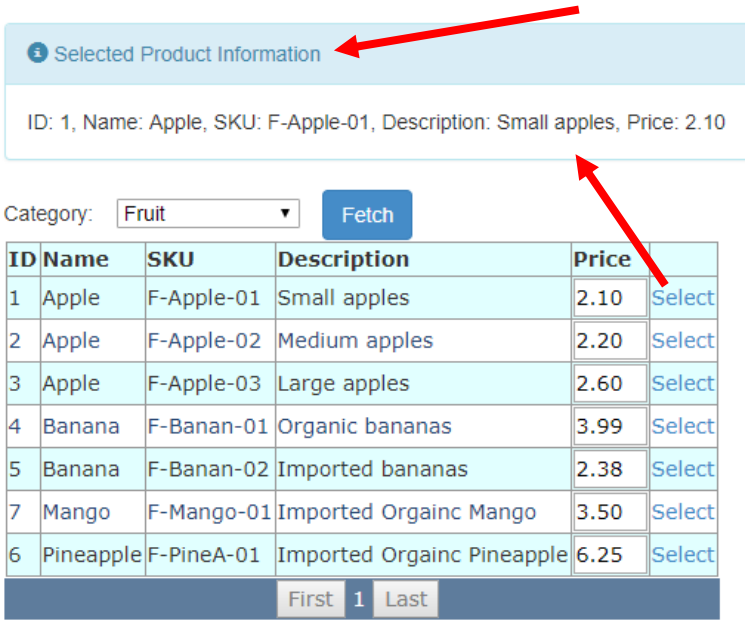
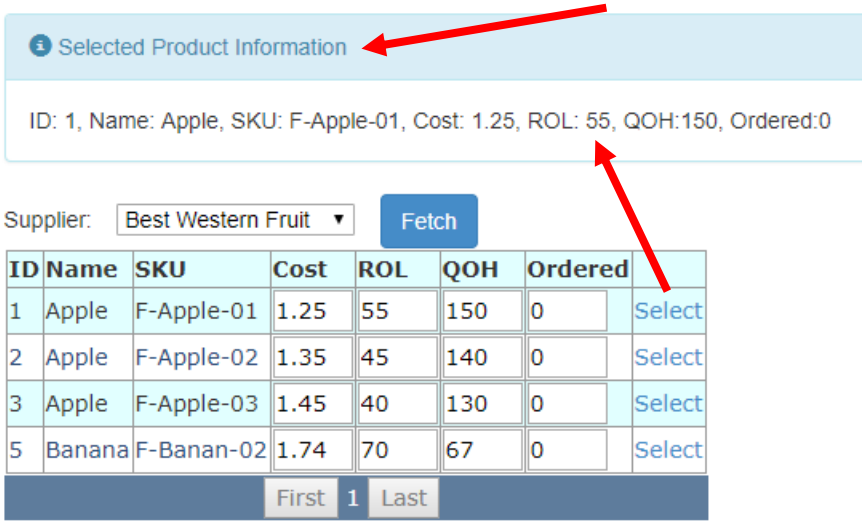


Figure 8: Custom MessageUserControl.ShowInfo()

PurchaseOrders.aspx

Repeat the steps you did with **`ShoppingCart.aspx`**, skipping the change from the `GridView` to the `ListView` (**`PurchaseOrders.aspx`** already has a `ListView`). Once done, your output should look like the figure below:



Exercise

Complete Exercise 5.1.1