School of Information Technology

Practical 09: Form Validation and Exception Handling

OBJECTIVES:

By the end of this Practical students should be able to:

- Make use of the Validation Control supplied by Visual Studio namely
    a. RequiredFieldValidator
    b. CompareValidator
    c. RegularExpressionValidator
    d. ValidationSummary
- Proper Use of Try … Catch statement to catch unexpected errors.

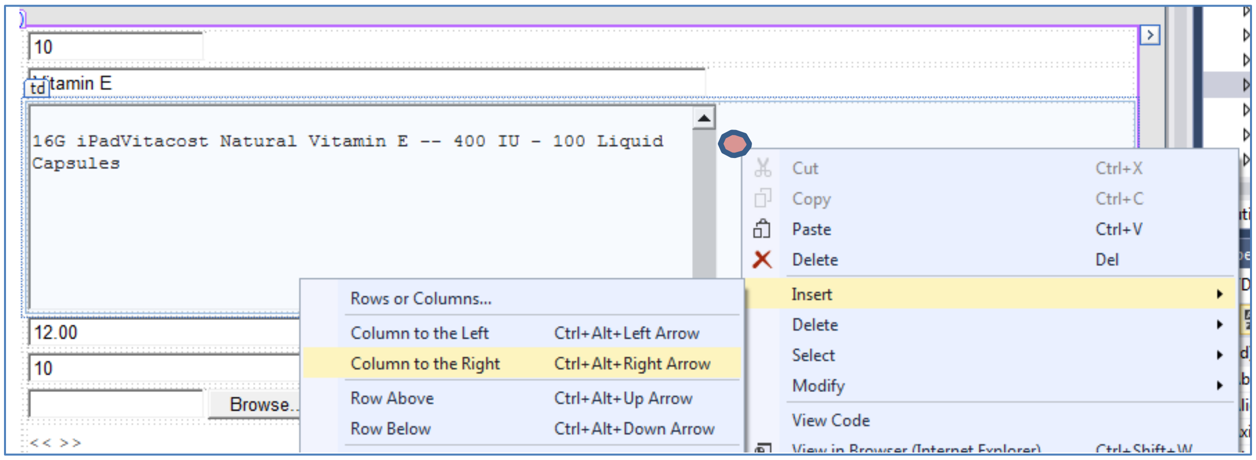
Introduction

In today's lab, we will continue with our previous lab web application. We will improve ProductInsert.aspx with validation checks so as to ensure the page is filled with valid data before user submit the form to the server.

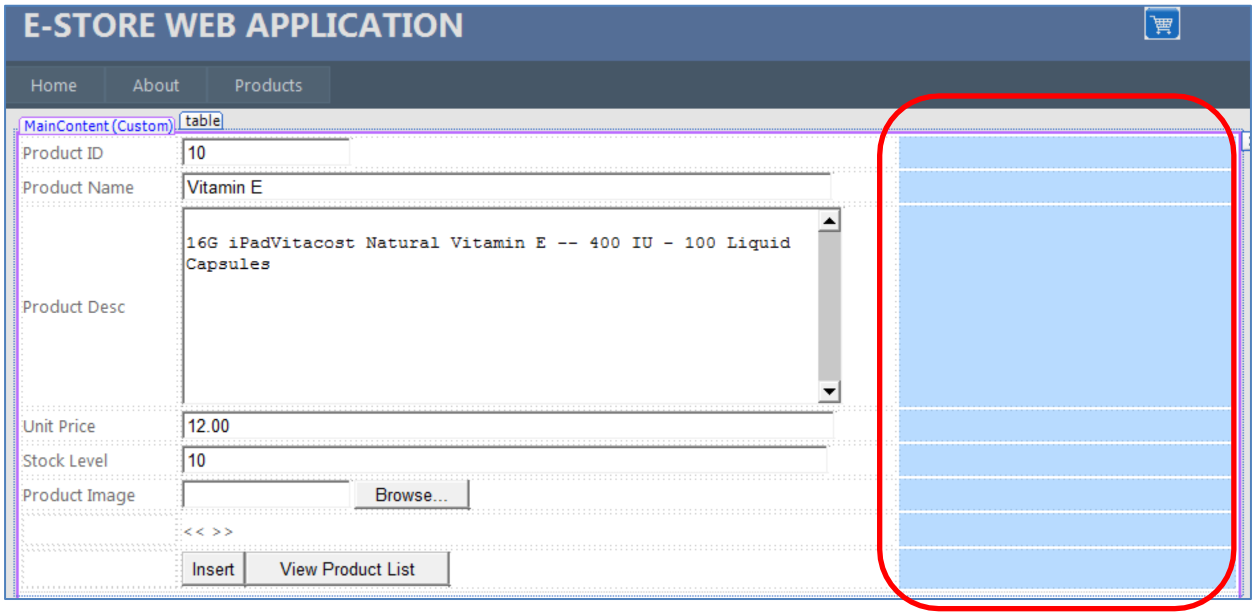
In all the web form created previously, we should encountered errors when user submits a Web Form with empty data in the Textboxes or a Form Controls.

Exercise 1

1. Download the template "Lab5A_Form_Validation template.zip" form blackboard and un- compress the file.
2. Open the website.
3. In ProductInsert.aspx, add a new column to the table holding the form UI.
4. Right-Click on any empty space after the product description textbox -> choose "Insert" -> Column to the Right.


10
Vitamin E
16G iPadVitacost Natural Vitamin E -- 400 IU - 100 Liquid Capsules
12.00
10
Browse...
<< >>
Rows or Columns...
Column to the Left Ctrl+Alt+Left Arrow
Column to the Right Ctrl+Alt+Right Arrow
Row Above Ctrl+Alt+Up Arrow
Row Below Ctrl+Alt+Down Arrow
Cut Ctrl+X
Copy Ctrl+C
Paste Ctrl+V
Delete Del
Insert
Delete
Select
Modify
View Code
View in Browser (Internet Explorer) Ctrl+Shift+W

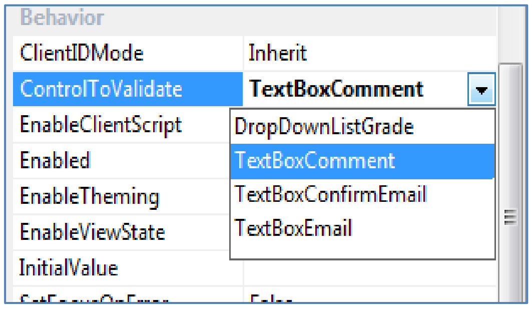
5. Adjust the new column by expanding its width.



Product ID	10
Product Name	Vitamin E
Product Desc	16G iPadVitacost Natural Vitamin E -- 400 IU - 100 Liquid Capsules
Unit Price	12.00
Stock Level	10
Product Image	
	<< >>

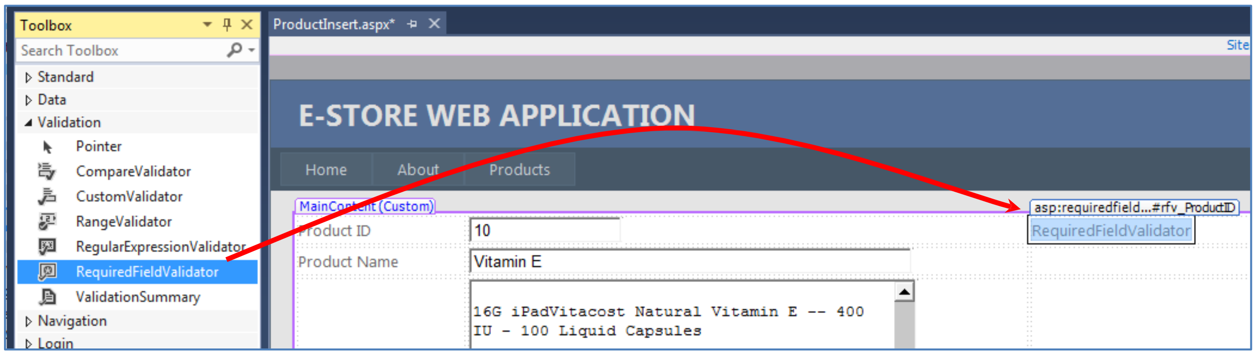
6. Add the following Form Validation:

a. Add a RequiredFieldValidator for validating Product ID textbox and set the following property. (<u>DO NOT TYPE </u>"tb_ProductID", use the dropdown recommended list)



ClientIDMode	Inherit
ControlToValidate	TextBoxComment
EnableClientScript	
Enabled	
EnableTheming	
EnableViewState	
InitialValue	

Property	**Value**
ControlToValidate	tb_ProductID
ErrorMessage	Please enter Product ID
Display	Static
ForeColor	Red
ID	rfv_ProductID



b. Add a RequiredFieldValidator for validating "Product Name" textbox and set the following property

Property	**Value**
ControlToValidate	tb_ProductName
ErrorMessage	Please enter a name for the product.
Display	Static
ForeColor	Red
ID	rfv_ProductName

c. Add a RequiredFieldValidator for validating "Product Description" textbox and set the following property

Property	**Value**
ControlToValidate	tb_ProductDesc
ErrorMessage	Please enter a description for the product.
Display	Static
ForeColor	Red
ID	rfv_ProductDesc

d. Add 2 Validators to this controls "Unit Price" – One for RequiredFieldValidator and the other for CompareValidator
    i. Add a RequiredFieldValidator for validating "Unit Price" textbox and set the following property

Property	**Value**
ControlToValidate	tb_UnitPrice
ErrorMessage	Please enter a Unit Price for the product.
Display	Static
ID	rfv_UnitPrice

    ii. Add a CompareValidator for validating "Unit Price" textbox value such that it must be Double
    iii.

Property	**Value**
ControlToValidate	tb_UnitPrice
ErrorMessage	Only Numeric value is allowed
Display	Static
ForeColor	Red
Operator	DataTypeCheck
Type	Double
ID	cv_UnitPrice

e. Add 2 Validators to this controls "Stock Level" – One for RequiredFieldValidator and the other for CompareValidator.
    i. Add a RequiredFieldValidator for validating "Stock Level" textbox and set the following property

Property	**Value**
ControlToValidate	tb_StockLevel
ErrorMessage	Please enter a value for the Stock Level
Display	Static
ID	rfv_StockLevel

    ii.

iii. Add a CompareValidator for validating “Stock Level” textbox value must be integer.

iv.

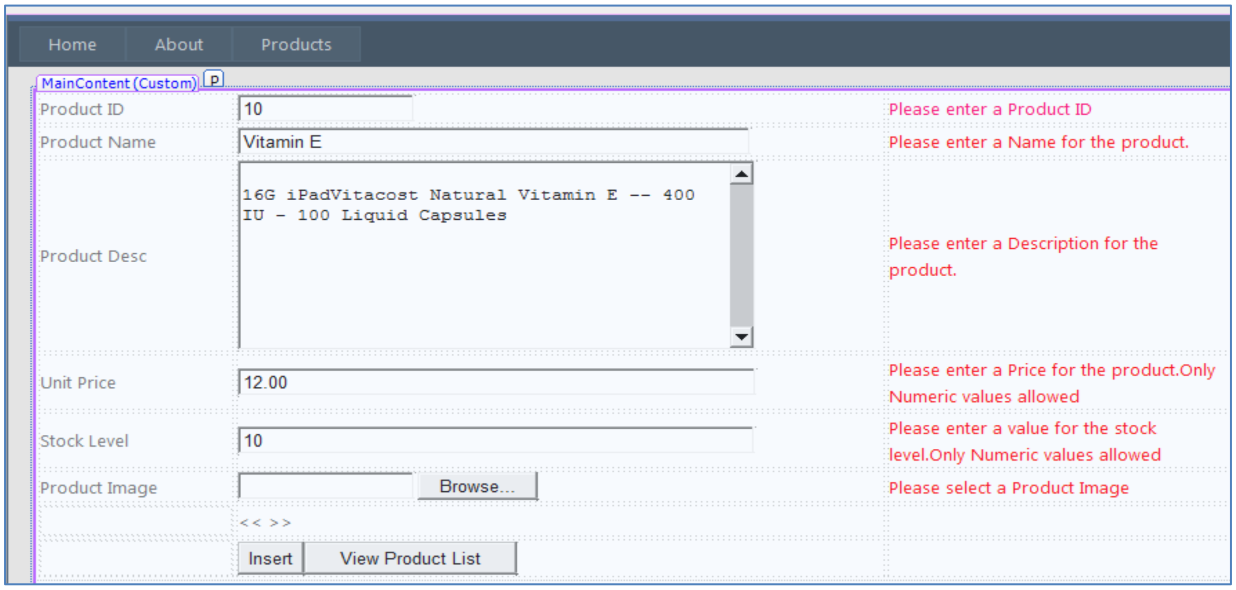
Property	Value
ControlToValidate	tb_StockLevel
ErrorMessage	Only Numeric Integer is allowed
Display	Static
ForeColor	Red
Operator	DataTypeCheck
Type	Integer
ID	cv_ StockLevel

f. Add a RequiredFieldValidator for validating “Product Image” fileUpload_Image and set the following property

Property	Value
ControlToValidate	FileUpload_Image
ErrorMessage	Please select a Product Image
Display	Static
ForeColor	Red
ID	rfv_ProductImage

7. Final output should be similar to Figure 1.

Figure 1 ProductInsert.aspx



Product ID	10	Please enter a Product ID
Product Name	Vitamin E	Please enter a Name for the product.
Product Desc	16G iPadVitacost Natural Vitamin E -- 400 IU - 100 Liquid Capsules	Please enter a Description for the product.
Unit Price	12.00	Please enter a Price for the product.Only Numeric values allowed
Stock Level	10	Please enter a value for the stock level.Only Numeric values allowed
Product Image		Please select a Product Image

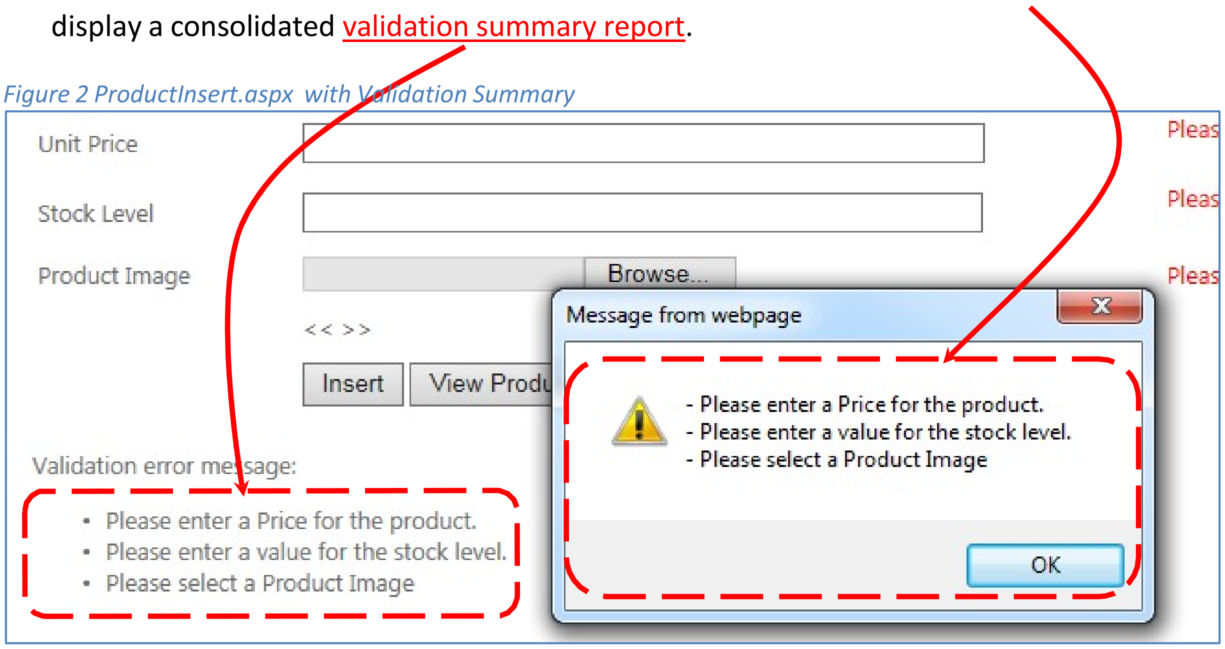
8. Test run the web form. Set ProductInsert.aspx to "start page" and hit F5 or use View in Browser'. Before you hit "Insert" button, remember to leave some of the textboxes blank so that you can test if the validation is working or not.
9. What happens if you click on "View Product List" when some textboxes were left blank?

Answer:

10. To overcome the above issue, you will need to do a little tweaking to the properties of the "View Product List" button. Set property CausesValidation to False.
11. ValidationSummary – This feature allow developer to either display a pop-up box and/or

display a consolidated validation summary report.

Figure 2 ProductInsert.aspx with Validation Summary



- Please enter a Price for the product.
- Please enter a value for the stock level.
- Please select a Product Image

12. Add a ValidationSummary as shown below and set the following properties:

Property	Value	Remarks
ShowMessage	True	If this is set to true, webpage will show the message on message box as shown in Figure 2.
ShowSummary	True	

13. Test your web application now.

Exercise 2 – Perform Exception Handling in Database codes

1. Continue from Exercise 1. Open ProductInsert.cs and Product.cs file.
2. Observe the codes and understand how a Product is being inserted into the database.
3. In Product.cs, go to the insertProduct method :

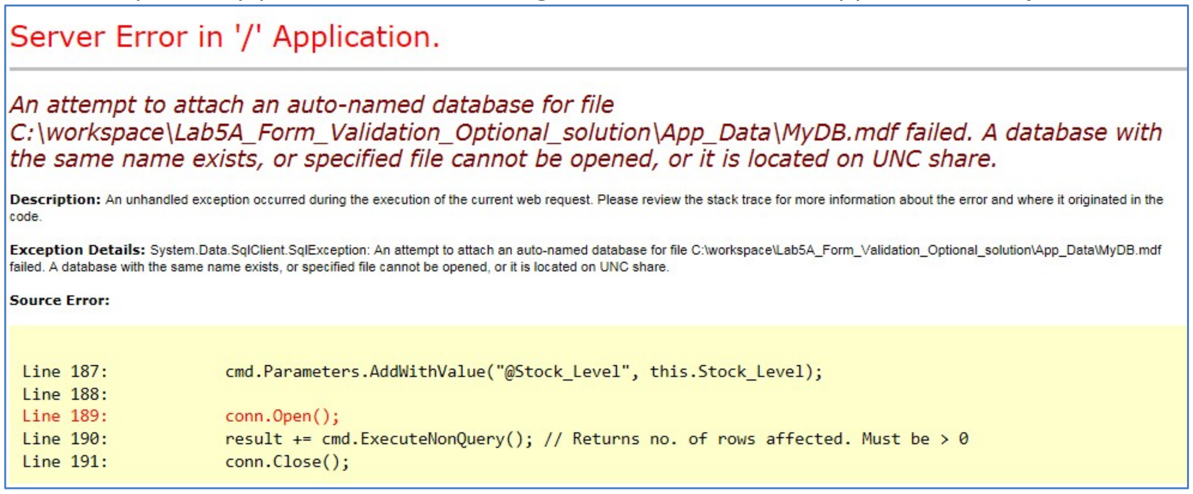
```
        public int ProductInsert()
        {
            string msg = null;
            int result = 0;
            string queryStr = "INSERT INTO Products(Product_ID,Product_Name, Product_Desc … … …“
            SqlConnection conn = new SqlConnection(_connStr);
            SqlCommand cmd = new SqlCommand(queryStr, conn);
            cmd.Parameters.AddWithValue("@Product_ID", this.Product_ID);
            cmd.Parameters.AddWithValue("@Product_Name", this.Product_Name);
            cmd.Parameters.AddWithValue("@Product_Desc", this.Product_Desc);
            cmd.Parameters.AddWithValue("@Unit_Price", this.Unit_Price);
            cmd.Parameters.AddWithValue("@Product_Image", this.Product_Image);
            cmd.Parameters.AddWithValue("@Stock_Level", this.Stock_Level);
            conn.Open();
            result += cmd.ExecuteNonQuery(); // Returns no. of rows affected. Must be > 0
            conn.Close();
            return result;
        }//end Insert
```

4. What will happen if we change the database file HealthDB.mdf TO MyDB.mdf which does not exist in the web.config file ?
5. Update the web.config with MyDB.mdf as the main DB and run your web app.

```
<connectionStrings>
<add name="HealthDBContext"
connectionString="Data Source=(LocalDB)\v11.0;
AttachDbFilename=|DataDirectory|\MyDB.mdf; Integrated
Security=True"  providerName="System.Data.SqlClient"/>
</connectionStrings>
```

6. Run ProductInsert.aspx and write down your observation.

7. Most probably you have the following fatal error! You web application had just crashed!



Server Error in '/' Application.

An attempt to attach an auto-named database for file C:\workspace\Lab5A_Form_Validation_Optional_solution\App_Data\MyDB.mdf failed. A database with the same name exists, or specified file cannot be opened, or it is located on UNC share.

Description: An unhandled exception occurred during the execution of the current web request. Please review the stack trace for more information about the error and where it originated in the code.

Exception Details: System.Data.SqlClient.SqlException: An attempt to attach an auto-named database for file C:\workspace\Lab5A_Form_Validation_Optional_solution\App_Data\MyDB.mdf failed. A database with the same name exists, or specified file cannot be opened, or it is located on UNC share.

Source Error:

```
Line 187:           cmd.Parameters.AddWithValue("@Stock_Level", this.Stock_Level);
Line 188:
Line 189:           conn.Open();
Line 190:           result += cmd.ExecuteNonQuery(); // Returns no. of rows affected. Must be > 0
Line 191:           conn.Close();
```

8. To prevent failure to your web application, you will need to handle any unexpected or fatal errors that might happened.
9. Make a slight modification to Product.cs

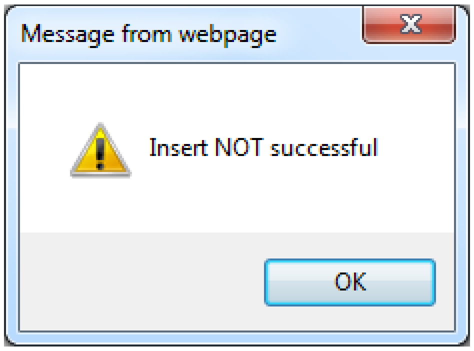
```
public int ProductInsert()
        {
            int result = 0;
            string queryStr = "INSERT INTO Products(Product_ID,Product_Name, Product_Desc … … …“
            try {
              SqlConnection conn = new SqlConnection(_connStr);
              SqlCommand cmd = new SqlCommand(queryStr, conn);
              cmd.Parameters.AddWithValue("@Product_ID", this.Product_ID);
              cmd.Parameters.AddWithValue("@Product_Name", this.Product_Name);
              cmd.Parameters.AddWithValue("@Product_Desc", this.Product_Desc);
              cmd.Parameters.AddWithValue("@Unit_Price", this.Unit_Price);
              cmd.Parameters.AddWithValue("@Product_Image", this.Product_Image);
              cmd.Parameters.AddWithValue("@Stock_Level", this.Stock_Level);
              conn.Open();
              result += cmd.ExecuteNonQuery(); // Returns no. of rows affected. Must be > 0
              conn.Close();
              return result;
            }
            catch (Exception ex)
              {  return 0;
            }

        }//end Insert
```

Note :

- Exception –for normal/general types of exception.
- SqlException –for SQL specific errors. In Product.cs, you may replace Exception with

SqlException.

10. Test run your web application again. This time, you won't get the messy error messages.



Message from webpage

Insert NOT successful

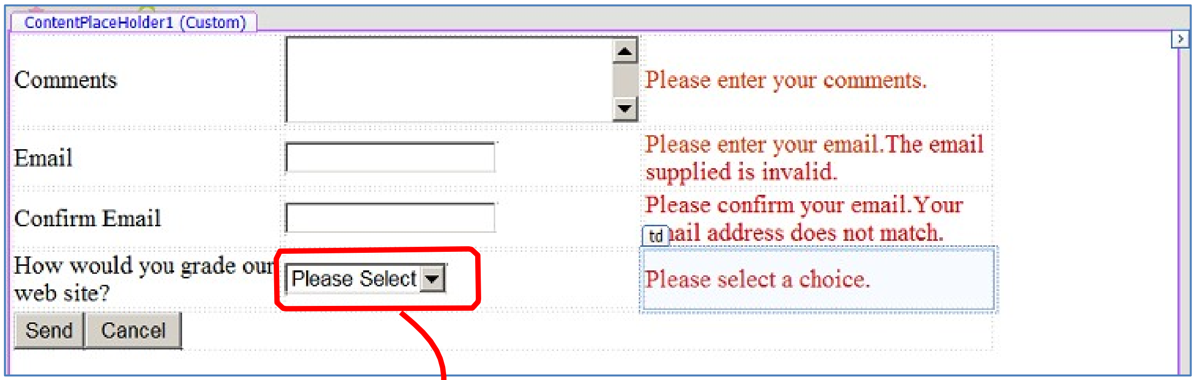
11. Try to do change from Exception to SqlException and also implement Try..catch for the other DB operations. (e.g Delete, Update)
12. Remember to revert your web.config file to the original working state.

OPTIONAL (FeedbackForm.aspx) – SDL/E-Learning

This is an optional practical. Creating a Customer Feedback Form.

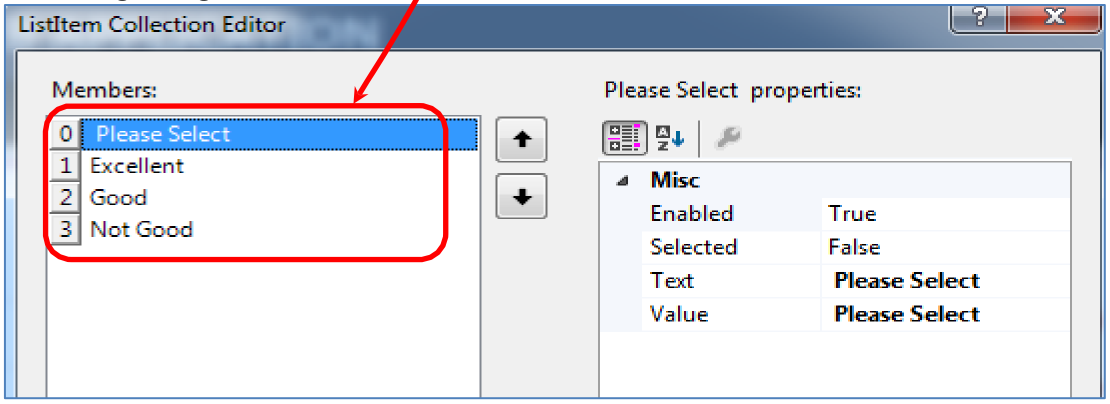
1. Create a web form feedback.aspx selecting master page as shown in Figure 3.

Figure 3 Feedback.aspx



Comments		Please enter your comments.
Email		Please enter your email.The email supplied is invalid.
Confirm Email		Please confirm your email.Your email address does not match.
How would you grade our web site?	Please Select	Please select a choice.

Website grading list



0	Please Select
1	Excellent
2	Good
3	Not Good

Enabled	True
Selected	False
Text	**Please Select**
Value	**Please Select**

Follow the instructions below :

2. Add a table with controls as shown in Figure 3.

a. Set Textbox control for comments property as follows

Property	**Value**
TextMode	MultiLine
ID	tb_Comment

b. Add a RequiredFieldValidator for Comments and set the following property

Property	**Value**
ControlToValidate	TextBoxComment
ErrorMessage	Please enter your comments.
Display	Static
ID	Rfv_Comments

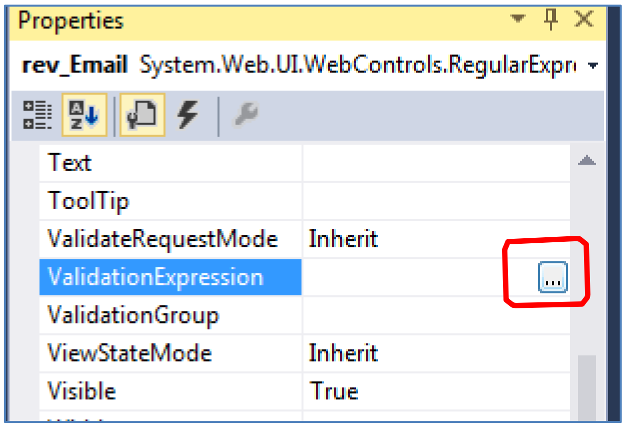
c. Add a TextBox control for email with ID = tb_Email

d. Add a RequiredFieldValidator for email and set the following property

Property	**Value**
ControlToValidate	TextBoxEmail
ErrorMessage	Please enter your email.
Display	Static
ID	Rfv_Email

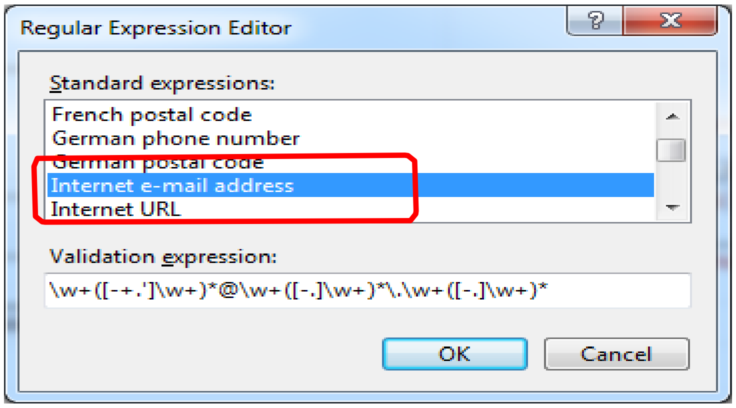
e. Add a RegularExpressValidator for email and set the following property

Property	**Value**
ControlToValidate	tb_Email
ErrorMessage	The email supplied is invalid.
Display	Static
ValidationExpression	Internet e-mail address
ID	rev_Email



rev_Email System.Web.UI.WebControls.RegularExpr

Text	
ToolTip	
ValidateRequestMode	Inherit
ValidationExpression	
ValidationGroup	
ViewStateMode	Inherit
Visible	True



Standard expressions:

French postal code
German phone number
German postal code
Internet e-mail address
Internet URL

Validation expression:

```
\w+([-+.']\w+)*@\w+([-.]\w+)*\.\w+([-.]\w+)*
```

If you select "Internet e-mail address" option, the validation expression will be automatically fill up for you.

f. Add a TextBox control to confirm the email address with ID = tb_ConfirmEmail

g. Add a CompareValidator for tb_ConfirmEmail and set the following property

Property	**Value**
ControlToCompare	tb_Email
ControlToValidate	tb_ConfirmEmail
ErrorMessage	Your email does not match.
Display	Static
Operator	Equal
Type	String
ID	cpv_ConfirmEmail

h. Add a DropDownList and add the following items into it with ID = ddl_Grade

- Please Select
- Excellent
- Good
- Not Good

i. Add a CompareValidator for DropDownList and set the following property

Property	Value	Remarks
ControlToValidate	ddl_Grade	
ErrorMessage	Please select a choice.	
Display	Static	
Operator	NotEqual	
Type	String	
ValueToCompare	Please Select	This MUST be exactly match what you typed into the DropDownListGrade control.
ID	cpv_Grade	

- The above setting will enforce user to select a value from the drop down list.
- DropDownListGrade is VALID if the selected item is **NotEqual** to the value of **ValueToCompare**.

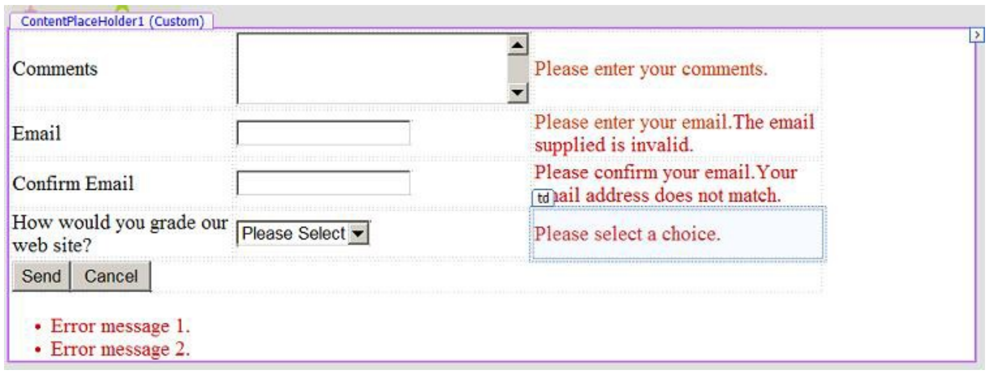
j. Add two buttons 'Send' and 'Cancel'. Their IDs are btn_Send and btn_Cancel respectively.

3. Run the web form using 'View in Browser'
   a. Enter a badly formed email address, for example: john_peh&hotmail.com (type '&' instead of '@') and click Send button. Which Error Messages showed up?
   b. Enter a proper email address, but leave "Confirm Email" empty and click Send button. Which Error Messages showed up?
   c. Enter a different email address in "Confirm Email" and click Send button. Which Error Messages showed up?
   d. Enter the same email address in "Confirm Email" and click Send button. Did the email validator Error Messages show up?

4. Next add a ValidationSummary as shown in Figure 2 and set the following properties

Property	**Value**	**Remarks**
ShowMessage	False	
ShowSummary	True	
ForeColor	#CC0000	

Figure 4 Feedback form with Validation Summary



Comments		Please enter your comments.
Email		Please enter your email.The email supplied is invalid.
Confirm Email		Please confirm your email.Your email address does not match.
How would you grade our web site?	Please Select	Please select a choice.

- Error message 1.
- Error message 2.

5. Test run your web application.

RegularExpressionValidator

The RegularExpressionValidator allows validating the input text by matching against a pattern of a regular expression. The regular expression is set in the ValidationExpression property.

The following table summarizes the commonly used syntax constructs for regular expressions:

Character Escapes	Description
\b	Matches a backspace.
\t	Matches a tab.
\r	Matches a carriage return.
\v	Matches a vertical tab.
\f	Matches a form feed.
\n	Matches a new line.
\	Escape character.

Apart from single character match, a class of characters could be specified that can be matched, called the metacharacters.

Metacharacters	Description
.	Matches any character except \n.
[abcd]	Matches any character in the set.
[^abcd]	Excludes any character in the set.
[2-7a-mA-M]	Matches any character specified in the range.
\w	Matches any alphanumeric character and underscore.
\W	Matches any non-word character.
\s	Matches whitespace characters like, space, tab, new line etc.
\S	Matches any non-whitespace character.
\d	Matches any decimal character.
\D	Matches any non-decimal character.

Quantifiers could be added to specify number of times a character could appear.

Quantifier	Description
*	Zero or more matches.
+	One or more matches.
?	Zero or one matches.
{N}	N matches.
{N,}	N or more matches.
{N,M}	Between N and M matches.

--- End of Practical--