****

**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
  1. RequiredFieldValidator
  2. CompareValidator
  3. RegularExpressionValidator
  4. 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.

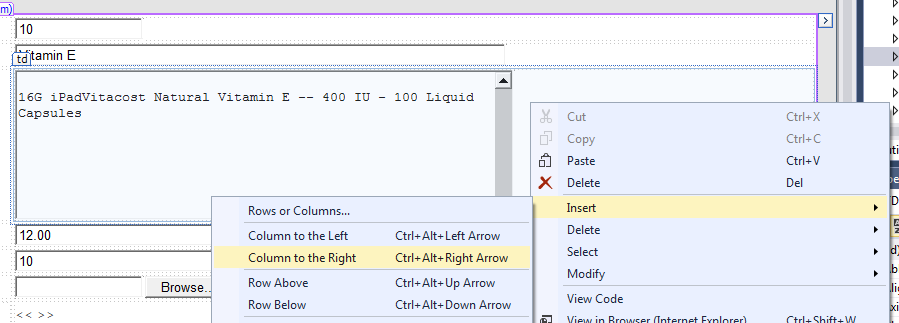
In all the web form created previously, we should encounter errors when user submits a Web Form with empty data in the Textboxes or a Form Controls.

Exercise 1

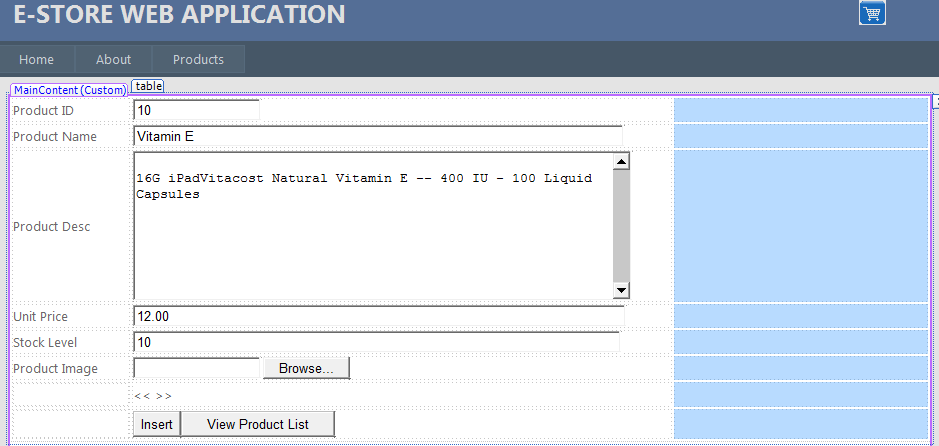
1. Open last week’s practical (Practical 8) solution.

* Start the Visual Studio 2019. Select Open a Project, locate **Lab06.sln**

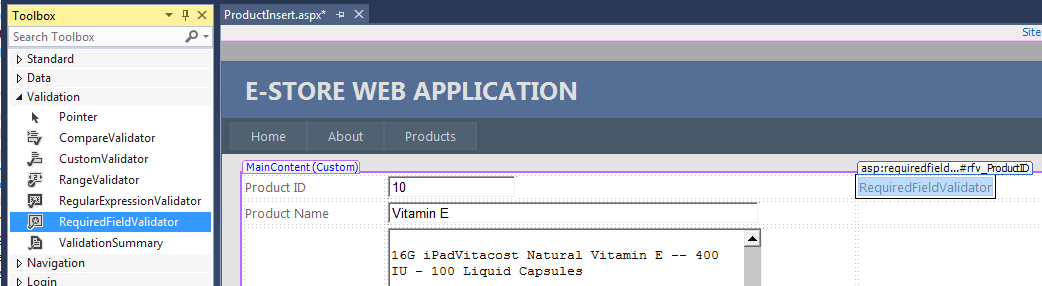
1. In ProductInsert.aspx, add a new column to the table containing the UI.
2. Right-Click on any empty space after the product description textbox -> choose **Insert** -> **Column to the Right**.



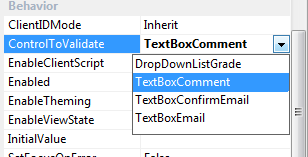
1. Adjust the new column by expanding its width.



1. Add the following Form Validation:
   1. Add a RequiredFieldValidator for validating Product ID textbox and set the following property. (DO NOT TYPE “tb\_ProductID”, use the dropdown recommended list)



|  |  |
| --- | --- |
| **Property** | **Value** |
| ControlToValidate | tb\_ProductID |
| ErrorMessage | Please enter Product ID |
| Display | Static |
| ForeColor | Red |
| ID | rfv\_ProductID |



* 1. 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 |

* 1. 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 |

* 1. Add 2 Validators to this controls “Unit Price” – One for RequiredFieldValidator and the other for CompareValidator
     1. 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 |
| ForeColor | Red |
| ID | rfv\_UnitPrice |

* + 1. Add a CompareValidator for validating “Unit Price” textbox value such that it must be Double

|  |  |
| --- | --- |
| **Property** | **Value** |
| ControlToValidate | tb\_UnitPrice |
| ErrorMessage | Only Numeric value is allowed |
| Display | Static |
| ForeColor | Red |
| Operator | DataTypeCheck |
| Type | Double |
| ID | cv\_UnitPrice |

* 1. Add 2 Validators to this controls “Stock Level” – One for RequiredFieldValidator and the other for CompareValidator.
     1. 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 |
| ForeColor | Red |
| ID | rfv\_StockLevel |

* + 1. Add a CompareValidator for validating “Stock Level” textbox value must be integer.

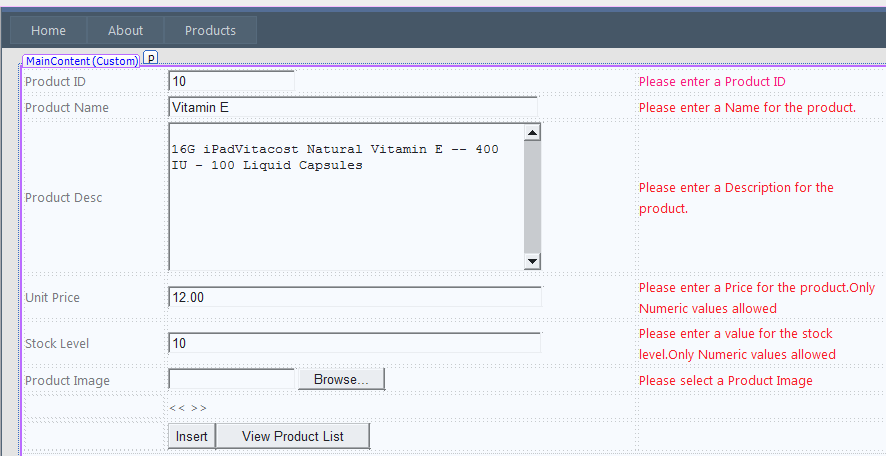
|  |  |
| --- | --- |
| **Property** | **Value** |
| ControlToValidate | tb\_StockLevel |
| ErrorMessage | Only Numeric Integer is allowed |
| Display | Static |
| ForeColor | Red |
| Operator | DataTypeCheck |
| Type | Integer |
| ID | cv\_StockLevel |

* 1. Add a RequiredFieldValidator for validating “Product Image” fileUpload\_Image and set the following property

|  |  |
| --- | --- |
| **Property** | **Value** |
| ControlToValidate | FileUpload1~~\_Image~~ |
| ErrorMessage | Please select a Product Image |
| Display | Static |
| ForeColor | Red |
| ID | rfv\_ProductImage |

1. Final output should be similar to Figure 1.

Figure 1 ProductInsert.aspx



1. Add the following to the web.config file (to disable the use of unobtrusive JavaScript for client-side validation):

<configuration>

<appSettings>

<add key="ValidationSettings:UnobtrusiveValidationMode" value="None" />

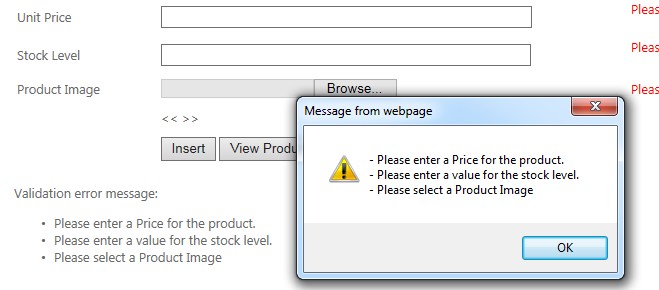
</appSettings>

</configuration>

1. 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.
2. What happens if you click on “View Product List” when some textboxes were left blank?

Answer:

1. 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.
2. 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*

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

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

1. 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

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

<connectionStrings>

<add name="HealthDBContext"

connectionString="Data Source=(LocalDB)\MSSQLLocalDB; AttachDbFilename=|DataDirectory|\MyDB.mdf;

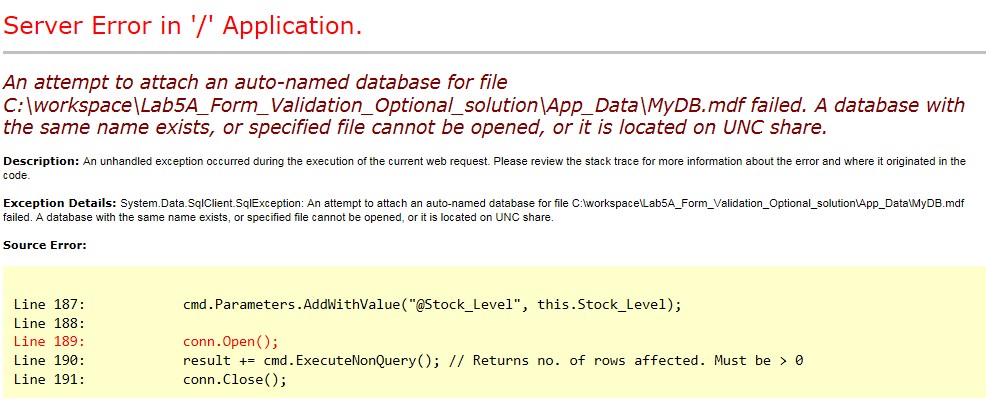
Integrated Security=True"

providerName="System.Data.SqlClient"

/>

</connectionStrings>

1. Run ProductInsert.aspx and write down your observation.
2. Most probably you have the following fatal error! You web application had just crashed!



1. To prevent failure to your web application, you will need to handle any unexpected or fatal errors that might happened.
2. 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;

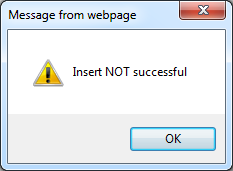
}

}//end Insert

Note :

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

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



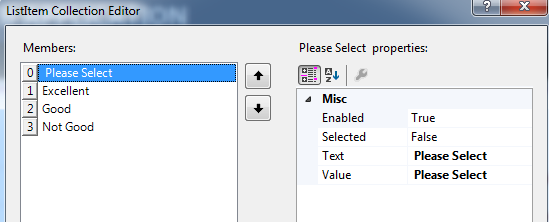
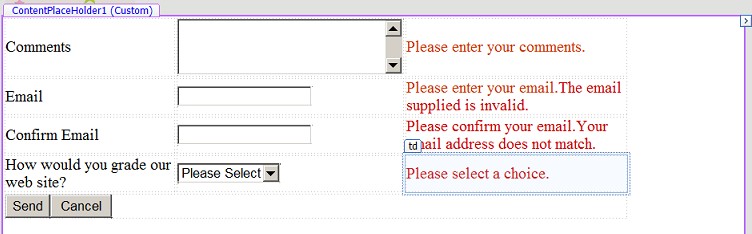
1. Try to do change from Exception to SqlException and also implement Try..catch for the other DB operations. (e.g Delete, Update)
2. Remember to revert your web.config file to the original working state.

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



Website grading list

Follow the instructions below :

1. Add a table with controls as shown in Figure 3.
   1. Set Textbox control for comments property as follows

|  |  |
| --- | --- |
| **Property** | **Value** |
| TextMode | MultiLine |
| ID | tb\_Comment |

* 1. Add a RequiredFieldValidator for Comments and set the following property

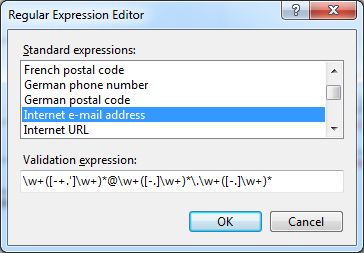
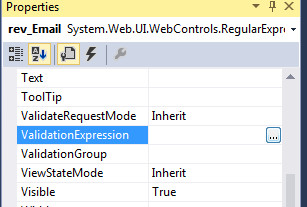
|  |  |
| --- | --- |
| **Property** | **Value** |
| ControlToValidate | tb\_Comment |
| ErrorMessage | Please enter your comments. |
| Display | Static |
| ID | Rfv\_Comments |

* 1. Add a TextBox control for email with ID = tb\_Email
  2. Add a RequiredFieldValidator for email and set the following property

|  |  |
| --- | --- |
| **Property** | **Value** |
| ControlToValidate | tb\_Email |
| ErrorMessage | Please enter your email. |
| Display | Static |
| ID | Rfv\_Email |

* 1. 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 |



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

* 1. Add a TextBox control to confirm the email address with ID = tb\_ConfirmEmail
  2. 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 |

* 1. Add a DropDownList and add the following items into it with ID = ddl\_Grade
     + Please Select
     + Excellent
     + Good
     + Not Good
  2. 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**.

* 1. Add two buttons ‘Send’ and ‘Cancel’. Their IDs are btn\_Send and btn\_Cancel respectively.

1. Run the web form using ‘View in Browser’
   1. Enter a badly formed email address, for example: john\_peh&hotmail.com (type '&' instead of '@') and click Send button. Which Error Messages showed up?
   2. Enter a proper email address, but leave "Confirm Email" empty and click Send button. Which Error Messages showed up?
   3. Enter a different email address in "Confirm Email" and click Send button. Which Error Messages showed up?
   4. Enter the same email address in "Confirm Email" and click Send button. Did the email validator Error Messages show up?
2. 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



1. 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--