**WebPage general error handling**

There are several ways we can handle and display error messages. This will enable your webpage to nicely handle various types of errors whether they are generated from your code, entity annotations for EntityFramework or system errors. The key to this is each error is treated as a single string. Each error is loaded to a global List<string>. The list is then passed to a display method along with the appropriate bootstrap css alert. If you **do not** wish to use the alert class, pass the value ***null***.

**Coding: WebPage: Form Control**

<asp:DataList ID="Message" runat="server">

<ItemTemplate>

<%# Container.DataItem %>

</ItemTemplate>

</asp:DataList>

**Coding: WebPage: Code-Behind Setup and clear old messages**

Add: List<string> and clearing of Message control

List<string> errormsgs = new List<string>();

protected void Page\_Load(object sender, EventArgs e)

{

Message.DataSource = null;

Message.DataBind();

if (!Page.IsPostBack)

{

* + - Your code goes here

}

}

**Coding: WebPage: Code-Behind General user Messages**

Add your general user message to the List<T>. Pass to the display method.

errormsgs.Add("Please select a product for the search");

LoadMessageDisplay(errormsgs, "alert alert-warning");

**Coding: WebPage: Code-Behind Additonal Namespaces**

using System.Data.Entity.Validation;

using System.Data.Entity.Infrastructure;

using System.Data.Entity.Core;

**Coding: WebPage: Code-Behind various Exception Catches**

The Exception catch you use is dependent on what you are trying to catch. The Db exceptions are related to EntityFramework. The General catch should be the last coded catch in your Try/Catch logic.

catch (DbUpdateException ex)

{

UpdateException updateException = (UpdateException)ex.InnerException;

if (updateException.InnerException != null)

{

errormsgs.Add(updateException.InnerException.Message.ToString());

}

else

{

errormsgs.Add(updateException.Message);

}

LoadMessageDisplay(errormsgs, "alert alert-danger");

}

catch (DbEntityValidationException ex)

{

foreach (var entityValidationErrors in ex.EntityValidationErrors)

{

foreach (var validationError in entityValidationErrors.ValidationErrors)

{

errormsgs.Add(validationError.ErrorMessage);

}

}

LoadMessageDisplay(errormsgs, "alert alert-danger");

}

catch (Exception ex)

{

errormsgs.Add(GetInnerException(ex).ToString());

LoadMessageDisplay(errormsgs, "alert alert-danger");

}

**Coding: ADD WebPage general method for obtaining inner errors**:

//use this method to discover the inner most error message.

//this routine has been created by the user

protected Exception GetInnerException(Exception ex)

{

//drill down to the inner most exception

while (ex.InnerException != null)

{

ex = ex.InnerException;

}

return ex;

}

**Coding: ADD WebPage general method for displaying errors**:

//use this method to load a DataList with a variable

//number of message lines.

//each line is a string

//the strings (lines) are passed to this routine in

// a List<string>

//second parameter is the bootstrap cssclass

protected void LoadMessageDisplay(List<string> errormsglist, string cssclass)

{

Message.CssClass = cssclass;

Message.DataSource = errormsglist;

Message.DataBind();

}