

Chapter 5

Exception Handling and Validation Controls

No one is so brave that he is not perturbed by the unexpected.

Julius Caesar,
De Bello Gallico, 6.39.

PRENTICE HALL

core
INTERNET
APPLICATION DEVELOPMENT
WITH ASP.NET 2.0



▼ Get up to speed quickly with numerous real-world walkthrough exercises, code listings, in-depth examples, and snippets

▼ Learn valuable design principles and best practices of ASP.NET Web application development

▼ Gain in-depth knowledge of many key ASP.NET 2.0 topics, including controls, data binding, security, Web services, and a sneak peek at ASP.NET AJAX

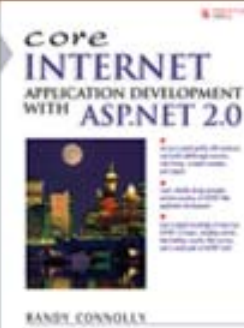
RANDY CONNOLLY

Overview

- Exception handling in C#
- Exception handling in ASP.NET
- Validation controls

2 Exceptions and Validation

COPYRIGHT © 2007 RANDY CONNOLLY



CORE INTERNET
APPLICATION DEVELOPMENT
WITH
ASP.NET 2.0

Prentice Hall, 2007

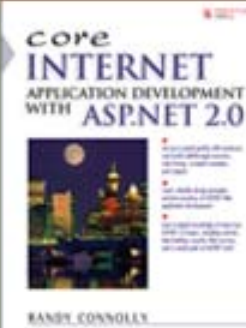
www.randyconnolly.com/core

Error Handling

- Even the best written Web application can suffer from runtime errors.
 - Most complex Web applications must interact with external systems such as databases, Web services, RSS feeds, email servers, file system, and other externalities that are beyond your control.
 - A failure in any one of these systems means that your application can also no longer run successfully.
 - It is vitally important that your applications can gracefully handle such problems.

3 Exceptions and Validation

COPYRIGHT © 2007 RANDY CONNOLLY



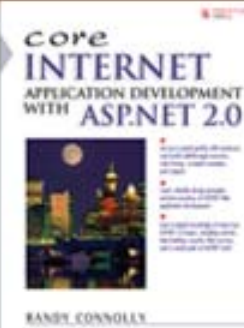
CORE INTERNET
APPLICATION DEVELOPMENT
WITH
ASP.NET 2.0

Prentice Hall, 2007

www.randyconnolly.com/core

.NET Exception Handling

- When an error occurs, something called an **exception** is raised, or **thrown** in the nomenclature of .NET.
 - When an error occurs, either the system or the currently executing application reports it by throwing an exception containing information about the error.
 - When thrown, an exception can be handled by the application or by ASP.NET itself.



CORE INTERNET
APPLICATION DEVELOPMENT
WITH
ASP.NET 2.0

Prentice Hall, 2007

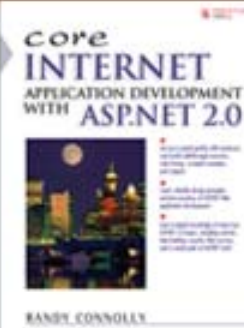
www.randyconnolly.com/core

Exception Handling Model

- In the .NET exception handling model, exceptions are represented as objects.
 - The ancestor class for all exceptions is `Exception`.
 - This class has many subclasses.
 - Every `Exception` object contains information about the error.

5 Exceptions and Validation

COPYRIGHT © 2007 RANDY CONNOLLY

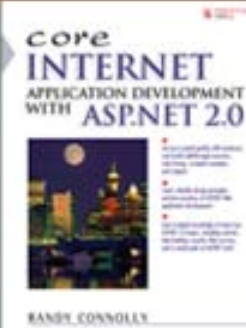


CORE INTERNET
APPLICATION DEVELOPMENT
WITH
ASP.NET 2.0

Prentice Hall, 2007
www.randyconnolly.com/core

Default Error Page

- When an exception is raised but not handled by the application, ASP.NET displays the **default error page**.
- This page displays:
 - the exception message
 - the exception type
 - the line that it occurred on
 - stack trace



CORE INTERNET
APPLICATION DEVELOPMENT
WITH
ASP.NET 2.0

Prentice Hall, 2007

www.randyconnolly.com/core

Default Error Page

Input string was not in a correct format. - Windows Internet Explorer

http://localhost:3141/Chapter5/TryCatch.aspx

Input string was not in a correct format.

Server Error in '/Chapter5' Application.

Input string was not in a correct format. ← Exception message

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.FormatException: Input string was not in a correct format. ← Exception type

Source Error:

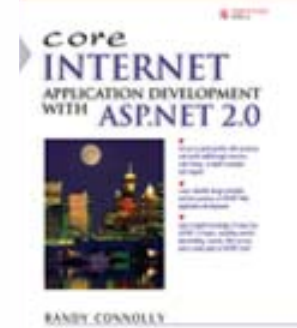
```
Line 21: //try
Line 22: //{
Line 23:     double dVal = Convert.ToDouble(txtValue.Text); ← Line that generated the exception
Line 24:     double result = dVal * 100.0;
Line 25:
```

Source File: c:\Documents and Settings\Owner\My Documents\teachmy books\CoreASPNET\CurrentVersion\Exercises\Done\Chapter5\TryCatch.aspx.cs Line: 23

Stack Trace:

```
[FormatException: Input string was not in a correct format.]
System.Number.StringToNumber(String str, NumberStyles options, NumberBuffers number, NumberFormatInfo info, Boolean parseDecimal) +2752723
System.Number.ParseDouble(String value, NumberStyles options, NumberFormatInfo numfmt) +113 ← Method that raised exception ...
System.Double.Parse(String s, NumberStyles style, NumberFormatInfo info) +188 ← ... which was called by this method ...
System.Convert.ToDouble(String value) +68 ← ... which was called by this method ...
TryCatch.btnDivide_Click(Object sender, EventArgs e) in c:\Documents and Settings\Owner\My Documents\teachmy books\CoreASPNET\CurrentVersion\
System.Web.UI.WebControls.Button.OnClick(EventArgs e) +75
System.Web.UI.WebControls.Button.RaisePostBackEvent(String eventArgument) +97
System.Web.UI.WebControls.Button.System.Web.UI.IPostBackEventHandler.RaisePostBackEvent(String eventArgument) +7
System.Web.UI.Page.RaisePostBackEvent(IPostBackEventHandler sourceControl, String eventArgument) +11
System.Web.UI.Page.RaisePostBackEvent(NameValueCollection postData) +33
System.Web.UI.Page.ProcessRequestMain(Boolean includeStagesBeforeAsyncPoint, Boolean includeStagesAfterAsyncPoint) +4919
```

Version Information: Microsoft .NET Framework Version 2.0.50727.42; ASP.NET Version 2.0.50727.42



Stack trace

Method that raised exception ...

... which was called by this method ...

... which was called by this method ...

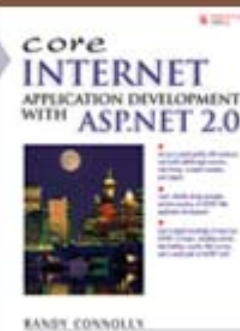
... and so on ...

Handling Exceptions

- Although this ASP.NET default error page is quite useful when developing and debugging an application, you might not always want to display this page when an exception occurs.
- Instead, you might want to **handle** the exception.
- There are three different ways or levels where you can do so:
 - At the class level
 - At the page level
 - At the application level.

8 Exceptions and Validation

COPYRIGHT © 2007 RANDY CONNOLLY



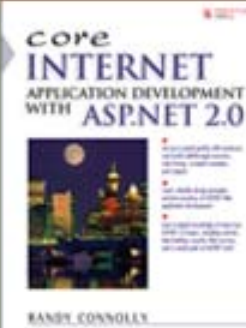
CORE INTERNET
APPLICATION DEVELOPMENT
WITH
ASP.NET 2.0

Prentice Hall, 2007

www.randyconnolly.com/core

Class level exception handling

- All .NET languages provide a mechanism for separating regular program code from exception handling code.
- In C#, this is accomplished via the `try...catch` block.
- If a runtime error occurs during the execution of any code placed within a try block, the program does not crash ...
 - ... but instead tries to execute the code contained in one of the catch blocks.



CORE INTERNET
APPLICATION DEVELOPMENT
WITH
ASP.NET 2.0

Prentice Hall, 2007

www.randyconnolly.com/core

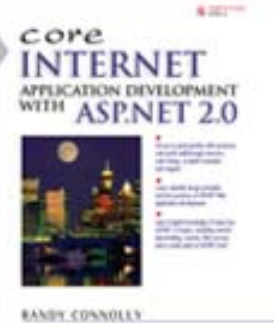
try...catch block

10 Chapter title here

COPYRIGHT © 2007 RANDY CONNOLLY

CORE INTERNET
APPLICATION DEVELOPMENT
WITH
ASP.NET 2.0

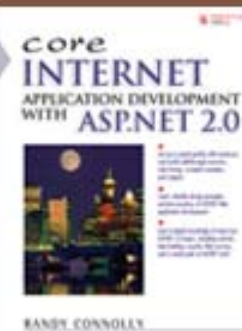
Prentice Hall, 2007
www.randyconnolly.com/core



```
try
{
    double dVal1 = Convert.ToDouble(txtValue1.Text);
    double dVal2 = Convert.ToDouble(txtValue2.Text);
    double result = dVal1 / dVal2;
    labMessage.Text = txtValue1.Text + "/" + txtValue2.Text;
    labMessage.Text += "=" + result;
}
catch (FormatException ex1)
{
    labMessage.Text = "Please enter a valid number";
}
catch (Exception ex2)
{
    labMessage.Text = "Unable to compute a value with these values";
}
```

finally block

- There may be times when you want to execute some block of code regardless of whether an exception occurred.
- The classic example is closing a database connection no matter whether the SQL operation was successful or generated an exception.
- In such a case, you can use the optional `finally` block



CORE INTERNET
APPLICATION DEVELOPMENT
WITH
ASP.NET 2.0

Prentice Hall, 2007

www.randyconnolly.com/core

finally block

12 Chapter title here

COPYRIGHT © 2007 RANDY CONNOLLY

CORE INTERNET
APPLICATION DEVELOPMENT
WITH
ASP.NET 2.0

Prentice Hall, 2007
www.randyconnolly.com/core

Core
INTERNET
APPLICATION DEVELOPMENT
WITH ASP.NET 2.0



RANDY CONNOLLY

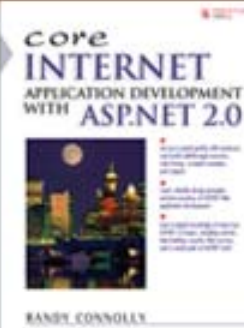
```
try
{
    // Open a database connection

    // Execute SQL statement

}
catch (DbException ex)
{
    // Handle database exception
}
finally
{
    // Close database connection if it exists
}
```

Cost of Exceptions

- Throwing exceptions is relatively expensive in terms of CPU cycles and resource usage.
- As a result, one should try to use exceptions to handle only exceptional situations.
- If your code relies on throwing an exception as part of its normal flow, you should refactor the code to avoid exceptions, perhaps by using a return code or some other similar mechanism instead.



CORE INTERNET
APPLICATION DEVELOPMENT
WITH
ASP.NET 2.0

Prentice Hall, 2007

www.randyconnolly.com/core

Using Exceptions

14 Chapter title here

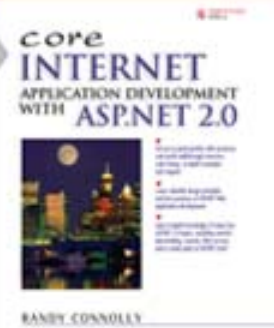
COPYRIGHT © 2007 RANDY CONNOLLY

```
try
{
    SomeBusinessObject.Login(email);

    // Other code dependent upon a successful login
}
catch (Exception ex)
{
    // Display message that email was not found
}
```

CORE INTERNET
APPLICATION DEVELOPMENT
WITH
ASP.NET 2.0

Prentice Hall, 2007
www.randyconnolly.com/core



bad

```
bool okay = SomeBusinessObject.Login(email);
if (! okay)
{
    // Display error message on page
}
else
{
    // Other code dependent upon a successful login
}
```

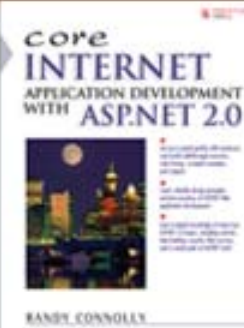
good

Exception Handling Strategies

- If you design your code so that exceptions are thrown only in truly exceptional situations, what do you do when one of these exceptional exceptions occurs?

15 Exceptions and Validation

COPYRIGHT © 2007 RANDY CONNOLLY



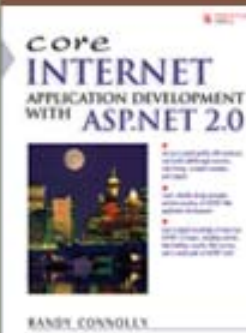
CORE INTERNET
APPLICATION DEVELOPMENT
WITH
ASP.NET 2.0

Prentice Hall, 2007

www.randyconnolly.com/core

Exception Handling Strategies

- Possible strategies:
 - “Swallow” the exception by catching and ignoring the exception by continuing normal execution.
 - Almost never appropriate.
 - Completely handle the exception within the catch block.
 - Ignore the exception by not catching it (and thus let some other class handle it).
 - Catch the exception and rethrow it for some other class to handle it.

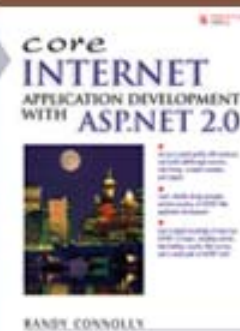


CORE INTERNET
APPLICATION DEVELOPMENT
WITH
ASP.NET 2.0

Prentice Hall, 2007
www.randyconnolly.com/core

Exception Handling Strategies

- You may want to know when an exception occurs in a production application so that you can change the code to prevent it from occurring in the future.
 - In this case, you might not want to catch the exception but instead let some other class “higher” in the calling stack handle it, perhaps by recording the exception to some type of exception log.
 - Even if you are not recording an exception log, you should remember that in general, you should not catch exceptions in a method unless it can handle them, such as by:
 - logging exception details,
 - performing some type of page redirection,
 - retrying the operation,
 - performing some other sensible action.

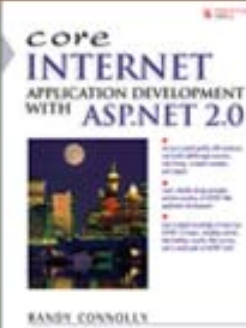


CORE INTERNET
APPLICATION DEVELOPMENT
WITH
ASP.NET 2.0

Prentice Hall, 2007
www.randyconnolly.com/core

Page Level Exception Handling

- ASP.NET allows the developer to handle errors on a page basis via the page's `Page_Error` event handler.
- The `Page_Error` event handler is called whenever an uncaught exception occurs during the execution of the page.



CORE INTERNET
APPLICATION DEVELOPMENT
WITH
ASP.NET 2.0

Prentice Hall, 2007
www.randyconnolly.com/core

Page_Error event handler

19 Chapter title here

COPYRIGHT © 2007 RANDY CONNOLLY

CORE INTERNET
APPLICATION DEVELOPMENT
WITH
ASP.NET 2.0

Prentice Hall, 2007
www.randyconnolly.com/core

Core
INTERNET
APPLICATION DEVELOPMENT
WITH ASP.NET 2.0



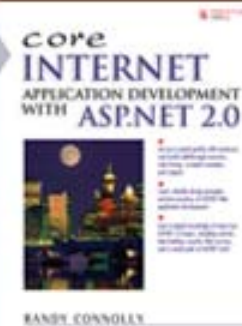
RANDY CONNOLLY

```
public partial class PageExceptionTest : System.Web.UI.Page
{
    ...

    private void Page_Error(object sender, EventArgs e)
    {
        Exception ex = Server.GetLastError();
        Response.Write("<h1>An error has occurred</h1>");
        Response.Write("<h2>" + ex.Message + "</h2>");
        Response.Write("<pre>" + ex.StackTrace + "</pre>");
        Context.ClearError();
    }
}
```

Application level exception handling

- There are two different ways that you can handle an exception at the application level:
 - using a `Application_Error` event handler
 - using the ASP.NET error page redirection mechanism.

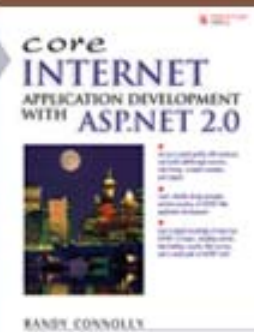


CORE INTERNET
APPLICATION DEVELOPMENT
WITH
ASP.NET 2.0

Prentice Hall, 2007
www.randyconnolly.com/core

Using the Application_Error Handler

- ASP.NET allows the developer to handle errors on an application-wide basis via the `Application_Error` event handler.
- This handler resides in the application's `Global.asax` file and is often the preferred location to handle uncaught exceptions in an application.
 - Because you often want to do the same thing for all unhandled exceptions in your application.
 - Rather than have the same type of error-logging code on every single page, it makes sense to centralize this code into a single spot.



CORE INTERNET
APPLICATION DEVELOPMENT
WITH
ASP.NET 2.0

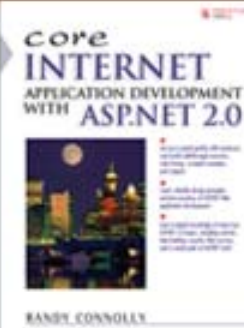
Prentice Hall, 2007

www.randyconnolly.com/core

Custom Error Page

- To use a custom error page, you can change the settings of the `<customErrors>` element in the `Web.config` file.
- In this element, you can specify the custom page that is to be displayed.

```
<system.web>  
  <customErrors mode="On" defaultRedirect="FriendlyErrorPage.aspx" />  
  ...  
</system.web>
```



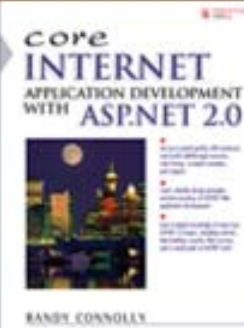
CORE INTERNET
APPLICATION DEVELOPMENT
WITH
ASP.NET 2.0

Prentice Hall, 2007
www.randyconnolly.com/core

Custom Error Pages

- You can create custom error pages for different HTTP error codes.
 - For example, a common feature of many Web sites is to provide custom HTTP 404 (requested page not found) and HTTP 500 (server error) error pages.
- You can specify custom pages for HTTP error codes within the `<customErrors>` element.

```
<customErrors mode="On" defaultRedirect="FriendlyErrorPage.aspx" >  
  <error statusCode="404" redirect="custom404.aspx" />  
  <error statusCode="500" redirect="custom500.aspx" />  
</customErrors>
```



CORE INTERNET
APPLICATION DEVELOPMENT
WITH
ASP.NET 2.0

Prentice Hall, 2007
www.randyconnolly.com/core

Validation Server Controls

- These are a special type of Web server control.
- They significantly reduce some of the work involved in validating user data.
- They are used to validate or verify that certain input server controls (such as `TextBox`, `RadioButtonList`, or `DropDownList`) contain correct data.



CORE INTERNET
APPLICATION DEVELOPMENT
WITH
ASP.NET 2.0

Prentice Hall, 2007

www.randyconnolly.com/core

Validation Server Controls

RequiredFieldValidator

- Ensures that the input control is not empty.

CompareValidator

- Compares a user entry against another value or control.

RangeValidator

- Checks if a user entry is between a lower and upper boundary.

RegularExpressionValidator

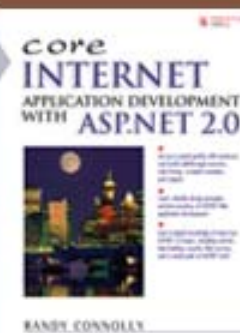
- Checks if a user entry matches a pattern defined by a regular expression

CustomValidator

- Checks a user entry using custom validation logic.

ValidationSummary

- Displays the error messages from all validation controls in a single location.



CORE INTERNET
APPLICATION DEVELOPMENT
WITH
ASP.NET 2.0

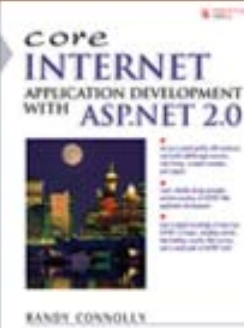
Prentice Hall, 2007

www.randyconnolly.com/core

Validation Server Controls

- You use validation server controls as you do other server controls.
 - That is, you add the markup to your .aspx file where you would like an error indicator to be displayed (typically adjacent to the field it is validating).
 - Each validation control references another input server control elsewhere on the page.

```
<asp:TextBox ID="txtUserName" runat="server" />  
  
<asp:RequiredFieldValidator Id="reqUser" runat="server"  
  ControlToValidate="txtUserName"  
  Text="Please enter a User Name" />
```



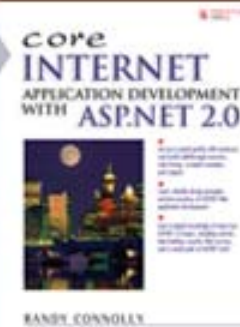
CORE INTERNET
APPLICATION DEVELOPMENT
WITH
ASP.NET 2.0

Prentice Hall, 2007

www.randyconnolly.com/core

Form Validation Process

- When a form that uses these validators is submitted, the user's input is validated first by using Javascript on the client side if enabled and if supported by the browser.
 - If there is an error, an error message is displayed without a round-trip to the server.
 - If no error (or no Javascript or if client validation is disabled), the data is passed to the server and the data is checked once again on the server side.
- If the data is not valid, an error message is generated and ultimately sent back to the browser (along with all the other form data).



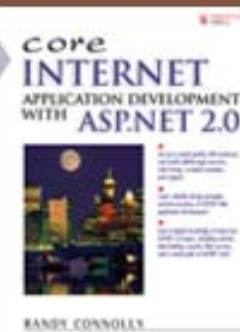
CORE INTERNET
APPLICATION DEVELOPMENT
WITH
ASP.NET 2.0

Prentice Hall, 2007

www.randyconnolly.com/core

Form Validation Process

- Why is both client-side and server-side data validation necessary?
 - Client-side validation is useful because it reduces round-trips to the server.
 - This provides immediate feedback to the user as well as improves server performance.
 - Client-side validation by itself is not sufficient. The user could be using a browser that does not support scripting.
 - that is, using an ancient browser or, more commonly, has scripting turned off via the browser preferences.
 - Client-side scripting is also potentially vulnerable to “script exploits.”



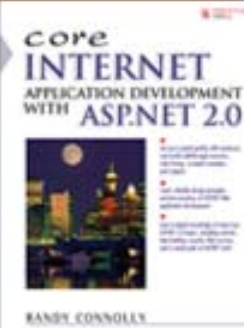
CORE INTERNET
APPLICATION DEVELOPMENT
WITH
ASP.NET 2.0

Prentice Hall, 2007

www.randyconnolly.com/core

Form Validation Process

- User data must thus be validated on both the client and the server side.
- Validation controls automatically generate the Javascript necessary for client-side validation as well as perform, behind the scenes, the server-side validation.



CORE INTERNET
APPLICATION DEVELOPMENT
WITH
ASP.NET 2.0

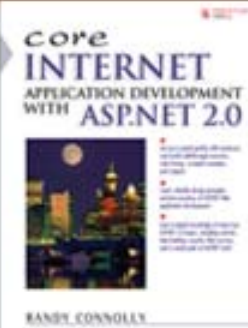
Prentice Hall, 2007
www.randyconnolly.com/core

Validation Controls

- See pages 280-308

30 Exceptions and Validation

COPYRIGHT © 2007 RANDY CONNOLLY



CORE INTERNET
APPLICATION DEVELOPMENT
WITH
ASP.NET 2.0

Prentice Hall, 2007

www.randyconnolly.com/core