LESSON 3

98-363 Web Development Fundamentals

3.1 Debug a Web Application

3.2 Handle Web Application Errors

MTA Web Development Fundamentals 3 Test

LESSON 3.1

98-363 Web Development Fundamentals

Debug a Web Application

Lesson Overview

In this lesson, you will learn:

- Techniques for tracing a Web application
- How custom error pages can be created
- The appropriate use of error pages
    - Viewing error messages in a Web page

Tracing Web Applications

- ASP.NET tracing can be integrated with system-level tracing to provide multiple levels of tracing output in distributed and multi-tier applications.

- When you enable tracing for an application, you can display trace output in any page in the application by setting the *pageOutput* attribute of the trace element to True in the Web.config file.

- The Trace Viewer (Trace.axd) can be used to view trace information that is collected and cached by ASP.NET when tracing is enabled.

Example of enabling tracing in the Web.config file

```
<configuration>
   <system.web>
         <trace enabled="true" requestLimit="20" localOnly="false" />
   </system.web>
</configuration>
```

Note: The example shows an application trace configuration that collects trace information for up to 20 requests. It also enables browsers on computers other than the server to display the Trace Viewer.

How is tracing useful?

- ASP.NET tracing enables you to view diagnostic information about a single request for an ASP.NET page.
- ASP.NET tracing enables you to follow a page's execution path, display diagnostic information at run time, and debug your application.

Debugging

- It is difficult, if not impossible, to catch every possible error in code when you first develop a Web application.
- Errors can be either compile-time errors, logic errors, or run-time errors.
  - — The Microsoft Visual Studio 2008 compiler finds compile-time errors.
- To find run-time errors and logic errors, use the Visual Studio 2008 debugger, the Trace object, or the Debug object.

Assignment

- Complete student activity 3.1

LESSON 3.2

98-363 Web Development Fundamentals

Handle Web Application Errors

Lesson Overview

In this lesson, you will learn about:

- Hypertext Transfer Protocol (HTTP) error trapping
- Common HTTP errors

HTTP Error Trapping

Error handling in Web applications occurs on four different levels, each of which generally traps different types of errors.

1. Code-block level:
   - Error handling is done within a page in `try-catch-finally` blocks.
   - Both Java Server Pages (JSP) and Microsoft ASP.NET support this structure.
2. Page level:
   - Errors that occur on a JSP or an ASP.NET page (for example, compilation errors) are generally processed by specialized error pages.
   - Redirection to error pages is accomplished through page directives.
3. Application level:
   - These errors apply to entire Web applications and are generally handled and controlled by settings within configuration files, such as deployment descriptors in JSP applications or the Web.config file in ASP.NET.

HTTP Error Trapping (continued)

4. Server level:
   - This applies to all applications running on a server and is generally configurable in the settings for the particular server.
   - Error handling of this nature is vendor-specific.

When errors occur, they move up the levels.

- For example, if an error can’t be handled at the code block level, it “bubbles up” to the page level, and so on.
- It is important to catch the errors in one of these four levels or the error message could contain insecure information.

▪ See *http://msdn.microsoft.com/en-us/library/aa478986.aspx* for language-specific error handling (such as `try-catch`).

Code Block Example Using Microsoft Visual Basic.NET

```
Try
 ' Code that might throw an exception
Catch
 ' Jump to here when the exception is thrown
Finally
 ' This code will be executed after the try and catch
 ' code, regardless of whether an exception is
 ' thrown or there is a break or continue

End Try
```

Example of Page-Level Error Handling in Visual Basic.NET

```
Private Sub Page_Error(ByVal sender As Object, ByVal e As EventArgs)
  ' Get last error from the server
  Dim exc As Exception = Server.GetLastError

  ' Handle exceptions generated by Button 1
  If TypeOf exc Is InvalidOperationException Then
    ' Pass the error on to the Generic Error page
    Server.Transfer("GenericErrorPage.aspx", True)

  ' Handle exceptions generated by Button 2
  ElseIf TypeOf exc Is ArgumentOutOfRangeException Then
    ' Give the user some information, but stay on the default page
    Response.Write("<h2>Default Page Error</h2>" & vbLf)
    Response.Write(("<p>Provide as much information here as is " _
      & "appropriate to show to the client.</p>" & vbLf))
    Response.Write(("Return to the <a href='Default.aspx'>" _
      & "Default Page</a>" & vbLf))
    ' Log the exception and notify system operators
    ExceptionUtility.LogException(exc, "DefaultPage")
    ExceptionUtility.NotifySystemOps(exc)
    ' Clear the error from the server
    Server.ClearError()
  Else
    ' Pass the error on to the default global handler
  End If
End Sub
```

Application-Level Example Using ASP.NET

- In ASP.NET, you use the `<customErrors>` handler to handle your application-level exceptions.
- You must access your Web application's Web.config file by adding the following lines of code to direct a user to your custom error page:

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
<system.web>
<customErrors mode=On
 defaultRedirect=yourCustomErrorPage.aspx />

</system.web>
</configuration>
```

The following lines add special handling of HTTP error codes:

```
<error statusCode="404" redirect="notFoundError.html">
<error statusCode="403" redirect="forbiddenError.html">
```

Server-Level Error Handling in ASP.NET

Microsoft Internet Information Services (IIS) allows custom error handling at the server level in four different ways:

1. If an error occurs, the server can display an automatically provided IIS error message.
2. The server also can output a customized error message if you have designed one for a specific error.
3. Instead of displaying an IIS error message, the server can redirect a user to a local Uniform Resource Locator (URL).
4. Alternatively, the server can redirect a user to an external URL.

Common HTTP Errors

Here are some common HTTP errors for which the developer should design error pages.

Error #	Error Code	IIS Error Information
403	Forbidden	HTTP Error 403 - Forbidden HTTP 403.1 - Forbidden: Execute Access Forbidden HTTP 403.2 - Forbidden: Read Access Forbidden
404	Page Not Found	HTTP 404 - File not found HTTP 404.1 - Web site not found
500	Internal Server Error	HTTP 500 - Internal server error HTTP Error 500-11 - Server shutting down
501	Not Implemented	Error 501 - Not implemented

Lesson Summary

- It is extremely important to catch errors and display appropriate messages to the user.
- This can be done by four levels of error checking:
    1. Code-block level
    2. Page level
    3. Application level
    4. Server level

LESSON 3

98-363 Web Development Fundamentals

Complete Quia Test:

MTA WebFund3 Test