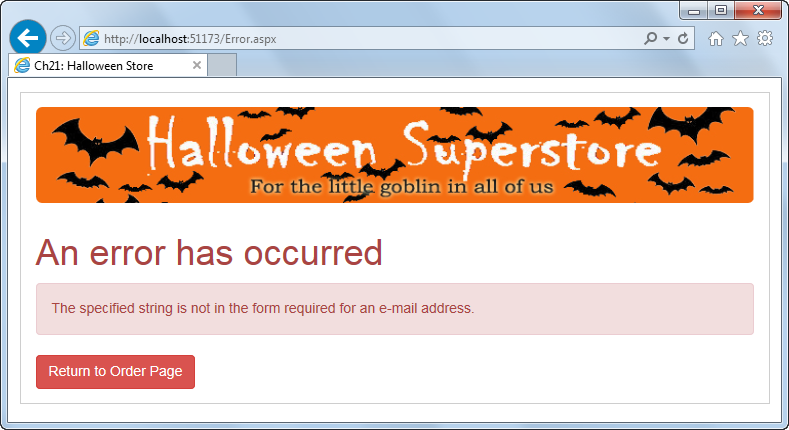
Extra 21-2 Add custom error handling to the Halloween Store application

In this exercise, you’ll use all four techniques for displaying custom error pages in an ASP.NET application. Note that when you’re running an application and an exception occurs, the application may go into break mode. If this happens, click on the Continue button to continue program execution.



Cause an error and review the default ASP.NET error page

1. Open the XEx21HandleErrors application in your exercises\_extra directory. Notice that this application includes pages named Error, E404, and EGeneric.
2. Run the application and place an order, but enter the email address in an incorrect format like “email”.
3. Review the error page that’s displayed by default. Although this page provides helpful information, you might not want to expose this kind of information to a user.

Use a try-catch statement to catch exceptions

1. Display the aspx code for the Error page, and notice that it contains a heading, a label for displaying an error, and a button for returning to the Order page.
2. Display the code-behind file for the Confirmation page. Then, add a try-catch statement to the SendConfirmation method that will catch any exceptions that are thrown by the statements in that method. If an exception occurs, the Exception object should be stored in session state and the Error page should be displayed.
3. Display the code-behind file for the Error page. Then, add code that gets the Exception object from session state and displays the Message property of the exception in the label on the page to the Page\_Load event handler.
4. Run the application and repeat step 2 to test it. This time, the Error page shown above should be displayed.

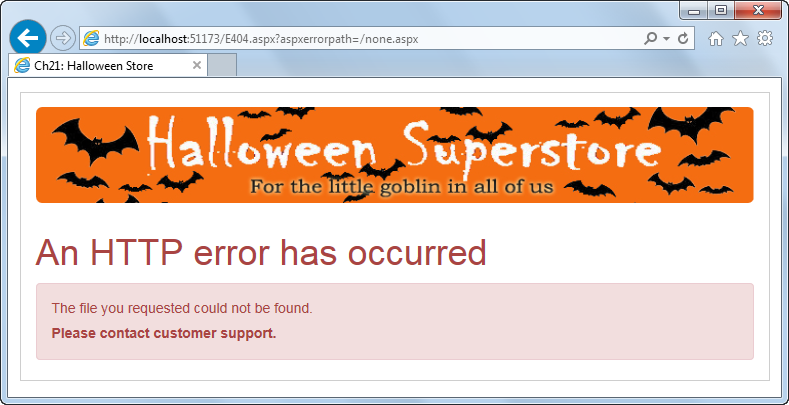
Use a Page\_Error method to catch exceptions

1. Comment out the try-catch statement you added in step 5.
2. Add a Page\_Error method to the Confirmation page. This method should get the Exception object, store it in session state, and display the Error page.
3. Run the application and repeat step 2. Just as it was previously, the Error page should be displayed.

Use an Application\_Error method to catch exceptions

1. Comment out the Page\_Error event handler you added in step 9.
2. Add a Global.asax file to the application. Then, add code to the Application\_Error method that gets the Exception object, stores it in session state, and displays the Error page.
3. Run the application and repeat step 2. Just as it was previously, the Error page should be displayed. Notice, however, that this time the error message is different.
4. In the Load event handler for the Error page, change the code so it checks if the InnerException property of the Exception object is null. If it is, display the Message property of the Exception object. Otherwise, display the Message property of the Exception object that’s returned by the InnerException property.
5. Run the application and repeat step 2. This time, the error message should be the same as the one you saw in the try-catch and page error examples.

Handle HTTP errors



1. Comment out the code you added to the Application\_Error method in step 12.
2. Modify the Web.config file so handles HTTP errors. By default, the EGeneric page should be displayed. But if a 404 error occurs, the E404 page should be displayed.
3. Run the application and repeat step 2. Note that the EGeneric page displays rather than the Error page you saw in previous steps.
4. In the address bar of the browser, change the file name in the URL to a file that doesn’t exist, like none.aspx. Then, click on the browser’s Go button. This time, the E404 page shown above should be displayed.