Custom Error Pages in ASP.NET

Prabhu Sunderaraman
prabhu@durasoftcorp.com
http://www.durasoftcorp.com

Abstract

All web applications are prone to throw two kinds of errors, conditional and unconditional. Conditional errors are related to the business logic specific rules, whereas unconditional errors may occur due to a network slowdown, database crash, server breakdown, etc. All these errors, if left untouched may manifest in a clumsy way threatening the audience. ASP.NET provides various error handling techniques to elegantly trap and handle the errors. This article talks about the use of the web configuration file to corner the errors and prop up customized error pages in the application.

Web.Config file

In ASP, the configuration properties of the applications are stored in a binary encoded data format. They can be accessed either by the script or through the IIS Management Console. For example, if our website is hosted on a third party server and we want to provide customized error pages for our web site we need to contact that party, and pay to make them configure our web site using IIS. So one of the main problems with this particular way of configuration is portability. The configuration properties of the ASP applications have to be recreated and set manually when transferred from one machine to another using the IIS admin tool.

This issue was resolved with the advent of ASP.NET. ASP.NET took a XML-based configuration approach. ASP.NET provides 2 configuration files: **machine.config** and **web.config**. The machine.config is the server configuration file that is common to all ASP.NET applications. By default, this gets installed in the /WinNT/Microsoft.NET/Framework/*version*/config folder, where *version* is the version of the .NET framework. On the other hand, web.config is the configuration file for the individual applications. Each ASP.NET application has a web.config file installed in the project directory. This file manages the properties of that particular web application it is associated with. As machine.config is common to all the ASP.NET applications, it requires changes only in very rare circumstances. It is the web.config file that is engaged in configuring the individual applications. This web.config file may be used to provide custom error pages for our application.

Error tags in Web.config

The section of web.config that is under consideration for customizing the error pages is shown below:

```
<configuration>
        <system.web>
                <customErrors>
                        <!-- Modify this part -- >
                </customErrors>
        </system.web>
</configuration>
```

The customErrors element is to be modified for any application errors. The customErrors element has an attribute “mode” that sets the error mode for an application. The error mode can be on, off or RemoteOnly, the details of which are discussed below:

- <customErrors mode=”**On**”>… </customErrors>: Displays only user-friendly error messages and not the default error text.
- <customErrors mode=”**Off**”>… </customErrors>: Toggles off the display of user-friendly messages and only the default error messages appear.
- <customErrors mode=”**RemoteOnly**”>… </customErrors>: This is the default mode. Displays default/detailed error messages to users of the local web server machine. It displays user-friendly messages when accessed from elsewhere.

An Example

Create a project called “CustomErrorApp” using VS.NET and view the web.config file created for the project. Note that the mode for the customErrors element is set by default to RemoteOnly as shown below:

```
<customErrors
mode="RemoteOnly"
/>
```

Create a Web Form **main.aspx** in the project. Insert the following piece of code in the Page_Load function of the codebehind page.

```
private void Page_Load(object sender, System.EventArgs e)
{
     int a = 0;
     int b = 0;
     int i = a/b;
}
```

This code throws a DivideByZero exception. Compile and execute the page. The Internet Explorer opens up displaying a detailed descriptive error message as shown in Figure 1. When accessed from a different machine, however, the error message appears as shown in Figure 2.

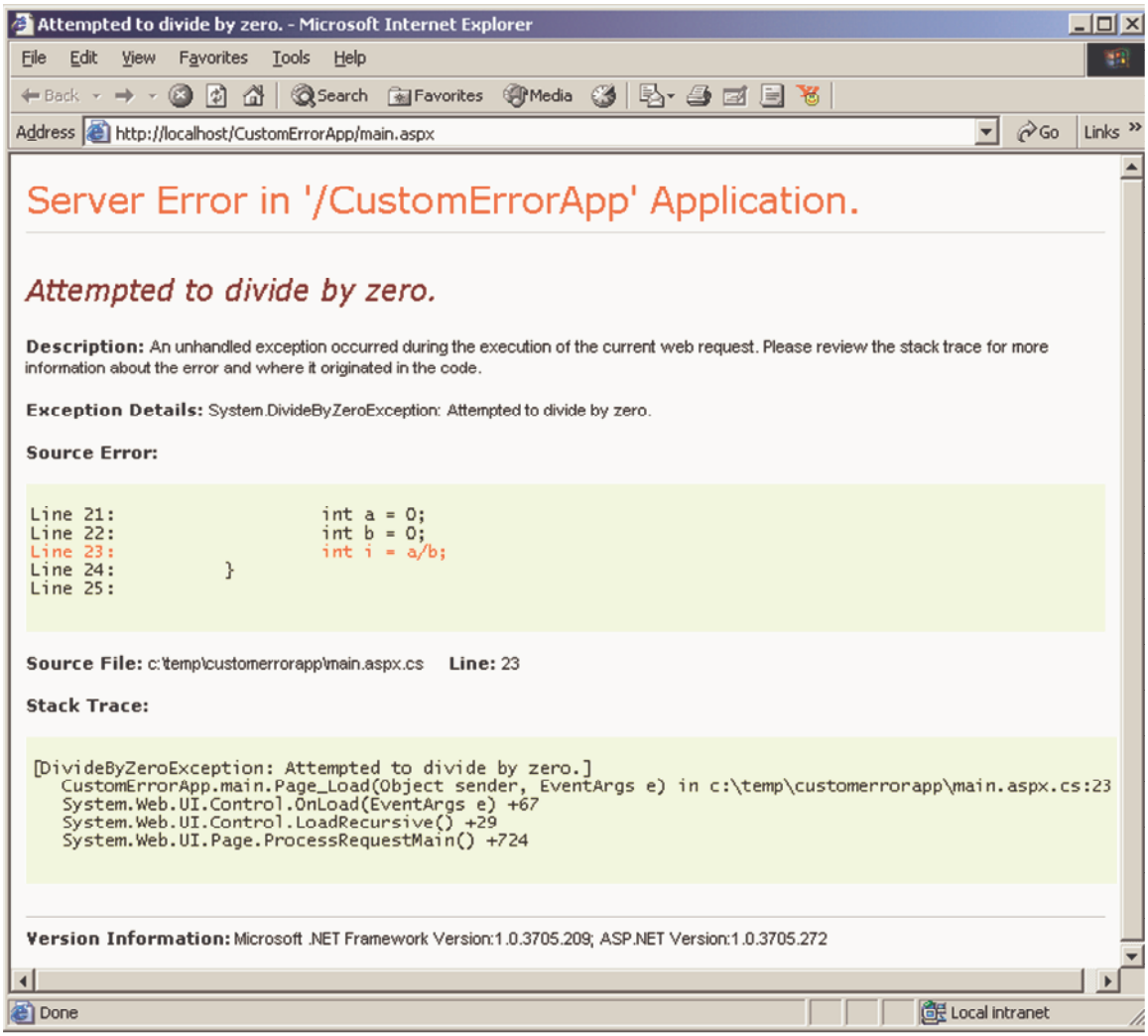


Figure 1. Default error message from ASP.NET for the DivisionByZeroException

Figure 2. Error message from ASP.NET when accessed from a different machine

We get two different error messages when accessed from the local web server machine and from a different machine. This is the property of the RemoteOnly mode of the custom error tag defined in the web.config file. RemoteOnly displays detailed messages when accessed from local machine. So administrators and developers who normally have access to the web server get a detailed report of the error message, stating the error occurrence location and type of the error. When the page is accessed from a remote machine, ASP.NET does not specify the details of the error. It just describes that a runtime error has occurred. The page also briefly explains the CustomErrors tags and its options.

If we need a detailed error report no matter wherever the page is accessed from, we can set the mode to be Off.

```
<customErrors mode="Off"/>
```

Turning the custom error page off displays only the ASP.NET error pages when accessed from everywhere.

Redirect attribute

As shown in Figure 2, the error message for a RemoteOnly mode, when main.aspx is executed remotely, the error message neither described the error completely nor was user-friendly. In order to redirect to a user-friendly error page "defaultRedirect" attribute has to be used in the CustomErrors tag. The defaultRedirect attribute can be used when the error mode is either "RemoteOnly" or "On."

Let's create a error page named "error.aspx" that displays the following message:

Oops!! There is an error

Let's modify the CustomErrors section as follows:

```
<customErrors
    mode="RemoteOnly" defaultRedirect="error.aspx" />
```

When main.aspx page is executed from the local server machine, we still get the same page as shown in Figure 1. The same page when visited from a remote machine, however, displays the error message as shown in Figure 3.

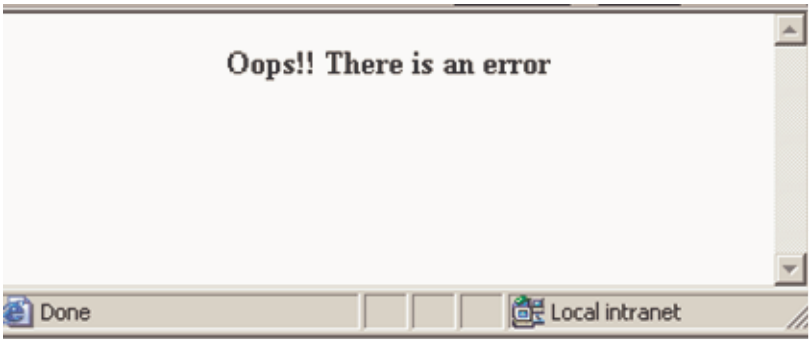


Figure 3. Effect of defaultRedirect attribute, when accessed from a remote site

Setting the mode to On will result in the error page to be displayed on all systems, local an remote:

```
<customErrors
 mode="On" defaultRedirect="error.aspx"/>
```

The redirected destination file may be an aspx file, an asp file or simply an html file.

The customErrors section works only for ASP.NET requests. If an error occurs while executing or requesting an asp file say, error.asp, IIS-defined error page still appear. To customize this, we have to lay our hands on IIS.

Modifying the error.aspx page to display the passed in query string, we get the information as shown in Figure 4.

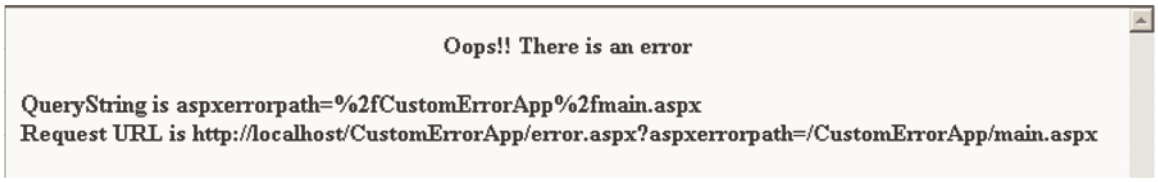
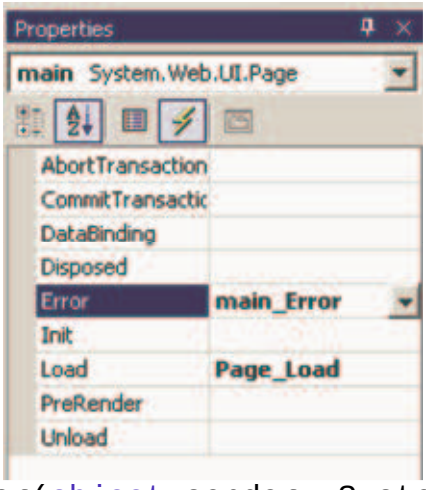


Figure 4. Information passed to the error page

The URL contains a querystring with a parameter “aspxerrorpath” that specifies the source of error. Figure 4 indicates main.aspx to be the error-prone page.

Apparently, this is the only information available to the redirected page. The type of error that occurred or the details of the error, etc. are not readily available to the error page. However, one could easily obtain these information by implementing an error handler on the error prone page or implementing the Application_Error on the global.asax.cs page. Here we will show how you can get the error information in the main.aspx. You may then log this information or send an email to the administrator, etc.

In main.aspx, implement an error handler for the unhandled exceptions as shown below:




```
private void main_Error(object sender, System.EventArgs e)
{
          Exception ex = Server.GetLastError();
          // Do whatever you want with this information,
          // like logging or sending email.
}
```

Handling specific errors

We can throw separate error pages for different types of HTTP errors. For example, a page not found error HTTP 404 can be trapped and handled differently and the same can be done for HTTP 403 forbidden error and so on.

This is achieved by adding <error> elements to the customErrors element. The error tags contain two attributes:

- statusCode : error status code like 404 /403/ 500
- redirect : redirected page

The customErrors section will look like this.

```
<customErrors  mode="On" defaultRedirect="error.aspx">

      <error statusCode="404" redirect="error404.aspx"/>
      <error statusCode="500" redirect="error500.aspx"/>

</customErrors>
```

Depending on the type of occurrence, the appropriate error page is thrown. If an error occurs, and the error status code does not fall under either 404 or 500, it brings up default error page specified, which is the error.aspx page in our case.

When should you use Web.config?

As we mentioned earlier there are different ways of handling errors in ASP.NET. The conditional errors are handled better using exception try/catch blocks or using global.asax file. These error handlers if offered are called first and the web.config section is only the last piece of defense mechanism invoked for unhandled errors in ASP.NET applications. Inspite of the presence of such efficient error handling techniques, web.config seems to be the better way to handle unconditional errors. Web.config is recommended to be put in use to fence in specific HTTP errors and to throw custom-made pages.

Conclusion

In this article, we have presented the details involved in creating custom error pages using the application configuration file in ASP.NET. In ASP, setting the custom error pages requires IIS configuration. The web.config file that is created for every ASP.NET application can be used to add the customized error pages for that application.