Logging Error Details with ELMAH

Introduction

The preceding tutorial examined ASP.NET's health monitoring system, which offers an out of the box library for recording a wide array of Web events. Many developers use health monitoring to log and e-mail the details of unhandled exceptions. However, there are a few pain points with this system. First and foremost is the lack any sort of user interface for viewing information about the logged events. If you want to see a summary of the 10 last errors, or view the details of an error that happened last week, you must either poke through the database, scan through your e-mail Inbox, or build a web page that displays information from the `aspnet_WebEvent_Events` table.

Another pain point centers around health monitoring's complexity. Because health monitoring can be used to record a plethora of different events, and because there are a variety of options for instructing how and when events are logged, correctly configuring the health monitoring system can be an onerous task. Finally, there are compatibility issues. Because health monitoring was first added to the .NET Framework in version 2.0, it is not available for older web applications built using ASP.NET version 1.x. Moreover, the `SqlWebEventProvider` class, which we used in the preceding tutorial to logs error details to a database, only works with Microsoft SQL Server databases. You will need to create a custom log provider class should you need to log errors to an alternate data store, such as an XML file or Oracle database.

An alternative to the health monitoring system is Error Logging Modules And Handlers (ELMAH), a free, open-source error logging system created by Atif Aziz. The most notable difference between the two systems is ELAMH's ability to display a list of errors and the details of a specific error from a web page and as an RSS feed. ELMAH is easier to configure than health monitoring because it only logs errors. Furthermore, ELMAH includes support for ASP.NET 1.x, ASP.NET 2.0, and ASP.NET 3.5 applications, and ships with a variety of log source providers.

This tutorial walks through the steps involved in adding ELMAH to an ASP.NET application. Let's get started!

> **Note:** The health monitoring system and ELMAH both have their own sets of pros and cons. I encourage you to try both systems and decide what one best suits your needs.

Adding ELMAH to an ASP.NET Web Application

Integrating ELMAH into a new or existing ASP.NET application is an easy and straightforward process that takes under five minutes. In a nutshell, it involves four simple steps:

1. Download ELMAH and add the `Elmah.dll` assembly to your web application,
2. Register ELMAH's HTTP Modules and Handler in `Web.config`,
3. Specify ELMAH's configuration options, and
4. Create the error log source infrastructure, if needed.

Let's walk through each of these four steps, one at a time.

Step 1: Downloading the ELMAH Project Files and Adding `Elmah.dll` To Your Web Application

ELMAH 1.0 BETA 3 (Build 10617), the most recent version at the time of writing, is included in the download available with this tutorial. Alternatively, you may visit the ELMAH website to get the most recent version or to download the source code. Extract the ELMAH download to a folder on your desktop and locate the ELMAH assembly file (`Elmah.dll`).

> **Note:** The `Elmah.dll` file is located in the download's `Bin` folder, which has subfolders for different .NET Framework versions and for Release and Debug builds. Use the Release build for the appropriate framework version. For instance, if you are building an ASP.NET 3.5 web application, copy the `Elmah.dll` file from the `Bin\net-3.5\Release` folder.

Next, open Visual Studio and add the assembly to your project by right-clicking on the website name in the Solution Explorer and choosing Add Reference from the context menu. This brings up the Add Reference dialog box. Navigate to the Browse tab and choose the `Elmah.dll` file. This action adds the `Elmah.dll` file to the web application's `Bin` folder.

> **Note:** The Web Application Project (WAP) type does not show the `Bin` folder in the Solution Explorer. Instead, it lists these items under the References folder.

The `Elmah.dll` assembly includes the classes used by the ELMAH system. These classes fall into one of three categories:

- **HTTP Modules** - an HTTP Module is a class that defines event handlers for `HttpApplication` events, such as the `Error` event. ELMAH includes multiple HTTP Modules, the three most germane ones being:
    - `ErrorLogModule` - logs unhandled exceptions to a log source.
    - `ErrorMailModule` - sends the details of an unhandled exception in an e-mail message.
    - `ErrorFilterModule` - applies developer-specified filters to determine what exceptions are logged and what ones are ignored.
- **HTTP Handlers** - an HTTP Handler is a class that is responsible for generating the markup for a particular type of request. ELMAH includes HTTP Handlers that render error details as a web page, as an RSS feed, or as a comma-delimited file (CSV).

- **Error Log Sources** - out of the box ELMAH can log errors to memory, to a Microsoft SQL Server database, to a Microsoft Access database, to an Oracle database, to an XML file, to a SQLite database, or to a Vista DB database. Like the health monitoring system, ELMAH's architecture was built using the provider model, meaning that you can create and seamlessly integrate your own custom log source providers, if needed.

Step 2: Registering ELMAH's HTTP Module and Handler

While the `Elmah.dll` file contains the HTTP Modules and Handler needed to automatically log unhandled exceptions and to display error details from a web page, these must be explicitly registered in the web application's configuration. The `ErrorLogModule` HTTP Module, once registered, subscribes to the `HttpApplication`'s `Error` event. Whenever this event is raised the `ErrorLogModule` logs the details of the exception to a specified log source. We'll see how to define the log source provider in the next section, "Configuring ELMAH." The `ErrorLogPageFactory` HTTP Handler factory is responsible for generating the markup when viewing the error log from a web page.

The specific syntax for registering HTTP Modules and Handlers depends upon the web server that is powering the site. For the ASP.NET Development Server and Microsoft's IIS version 6.0 and earlier, HTTP Modules and Handlers are registered in the `<httpModules>` and `<httpHandlers>` sections, which appear within the `<system.web>` element. If you are using IIS 7.0 then they need to be registered in the `<system.webServer>` element's `<modules>` and `<handlers>` sections. Fortunately, you can define the HTTP Modules and Handlers in *both* places regardless of the web server being used. This option is the most portable one as it allows the same configuration to be used in the development and production environments regardless of the web server being used.

Start by registering the `ErrorLogModule` HTTP Module and the `ErrorLogPageFactory` HTTP Handler in the `<httpModules>` and `<httpHandlers>` section in `<system.web>`. If your configuration already defines these two elements then simply include the `<add>` element for ELMAH's HTTP Module and Handler.

```
<?xml version="1.0"?>
<configuration>
   ...

   <system.web>
      ...

      <httpHandlers>
         ...
```

```
            <add verb="POST,GET,HEAD" path="elmah.axd"
type="Elmah.ErrorLogPageFactory, Elmah" />
        </httpHandlers>

        <httpModules>
            ...

            <add name="ErrorLog" type="Elmah.ErrorLogModule, Elmah"/>
        </httpModules>

        ...
    </system.web>

    ...
</configuration>
```

Next, register ELMAH's HTTP Module and Handler in the `<system.webServer>` element. As before, if this element is not already present in your configuration then add it.

```
<?xml version="1.0"?>
<configuration>
    ...

    <system.webServer>
        <validation validateIntegratedModeConfiguration="false"/>
        <modules>
            ...

            <add name="Elmah.ErrorLog" type="Elmah.ErrorLogModule, Elmah"
preCondition="managedHandler" />
        </modules>
        <handlers>
            ...

            <add name="Elmah" path="elmah.axd" verb="POST,GET,HEAD"
type="Elmah.ErrorLogPageFactory, Elmah" preCondition="integratedMode" />
        </handlers>
    </system.webServer>
</configuration>
```

By default, IIS 7 complains if HTTP Modules and Handlers are registered in the `<system.web>` section. The `validateIntegratedModeConfiguration` attribute in the `<validation>` element instructs IIS 7 to suppress such error messages.

Note that the syntax for registering the `ErrorLogPageFactory` HTTP Handler includes a `path` attribute, which is set to `elmah.axd`. This attribute informs the web application that if a request arrives for a page named `elmah.axd` then the request should be processed by the `ErrorLogPageFactory` HTTP Handler. We'll see the `ErrorLogPageFactory` HTTP Handler in action later in this tutorial.

Step 3: Configuring ELMAH

ELMAH looks for its configuration options in the website's `Web.config` file in a custom configuration section named `<elmah>`. In order to use a custom section in `Web.config` it must first be defined in the `<configSections>` element. Open the `Web.config` file and add the following markup to the `<configSections>`:

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
    <configSections>
        ...

        <sectionGroup name="elmah">
            <section name="security" requirePermission="false"
type="Elmah.SecuritySectionHandler, Elmah"/>
            <section name="errorLog" requirePermission="false"
type="Elmah.ErrorLogSectionHandler, Elmah" />
            <section name="errorMail" requirePermission="false"
type="Elmah.ErrorMailSectionHandler, Elmah" />
            <section name="errorFilter" requirePermission="false"
type="Elmah.ErrorFilterSectionHandler, Elmah"/>
        </sectionGroup>
    </configSections>

    ...
</configuration>
```

Note: If you are configuring ELMAH for an ASP.NET 1.x application then remove the `requirePermission="false"` attribute from the `<section>` elements above.

The above syntax registers the custom `<elmah>` section and its subsections: `<security>`, `<errorLog>`, `<errorMail>`, and `<errorFilter>`.

Next, add the `<elmah>` section to `Web.config`. This section should appear at the same level as the `<system.web>` element. Inside the `<elmah>` section add the `<security>` and `<errorLog>` sections like so:

```
<?xml version="1.0"?>
<configuration>
    ...

    <elmah>
        <security allowRemoteAccess="0" />

        <errorLog type="Elmah.SqlErrorLog, Elmah"
connectionStringName="ReviewsConnectionString" />
    </elmah>

    ...
</configuration>
```

The `<security>` section's `allowRemoteAccess` attribute indicates whether remote access is allowed. If this value is set to 0, then the error log web page can only be viewed locally. If this attribute is set to 1 then the error log web page is enabled for both remote and local visitors. For now, let's disable the error log web page for remote visitors. We'll allow remote access later after we have an opportunity to discuss the security concerns of doing so.

The `<errorLog>` section defines the error log source, which dictates where the error details are recorded; it is similar to the `<providers>` section in the health monitoring system. The above syntax specifies the `SqlErrorLog` class as the error log source, which logs the errors to a Microsoft SQL Server database specified by the `connectionStringName` attribute value.

> **Note:** ELMAH ships with additional error log providers that can be used to log errors to an XML file, a Microsoft Access database, an Oracle database, and other data stores. Refer to the sample `Web.config` file that is included with the ELMAH download for information on how to use these alternate error log providers.

Step 4: Creating the Error Log Source Infrastructure

ELMAH's `SqlErrorLog` provider logs error details to a specified Microsoft SQL Server database. The `SqlErrorLog` provider expects this database to have a table named `ELMAH_Error` and three stored procedures: `ELMAH_GetErrorsXml`, `ELMAH_GetErrorXml`, and `ELMAH_LogError`. The ELMAH download includes a file named `SQLServer.sql` in the `db` folder that contains the T-SQL for creating this table and these stored procedures. You'll need to run these statements on your database to use the `SqlErrorLog` provider.

Figures 1 and 2 show the Database Explorer in Visual Studio after the database objects needed by the `SqlErrorLog` provider have been added.

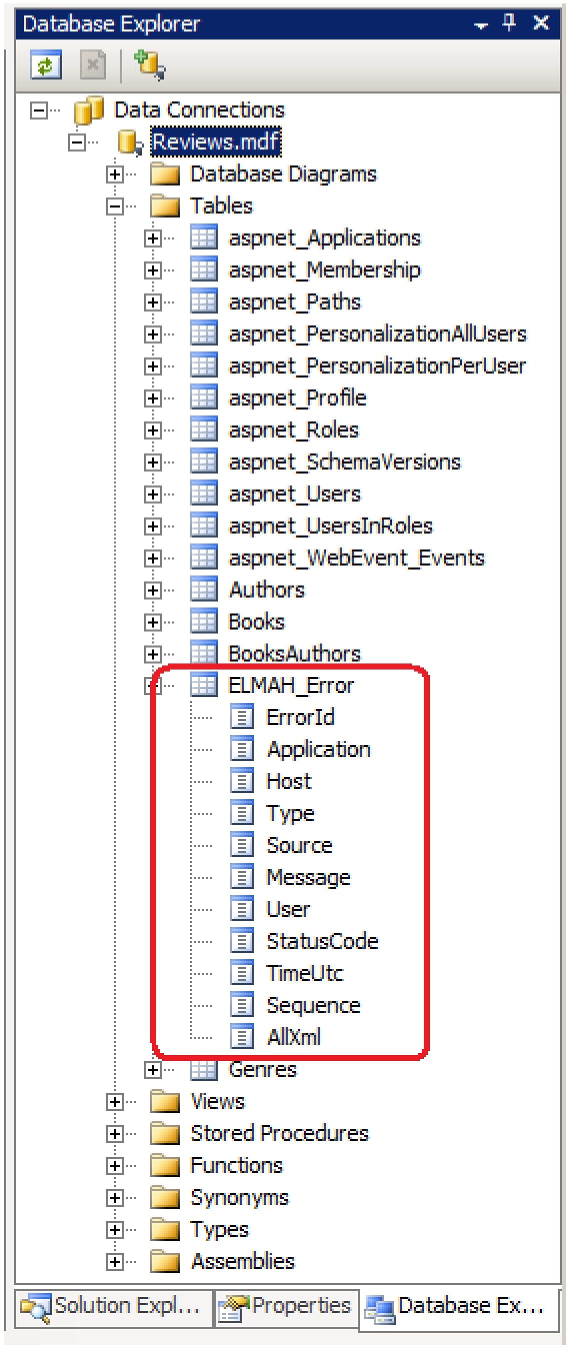


Figure 1: The `SqlErrorLog` Provider Logs Errors to the `ELMAH_Error` Table

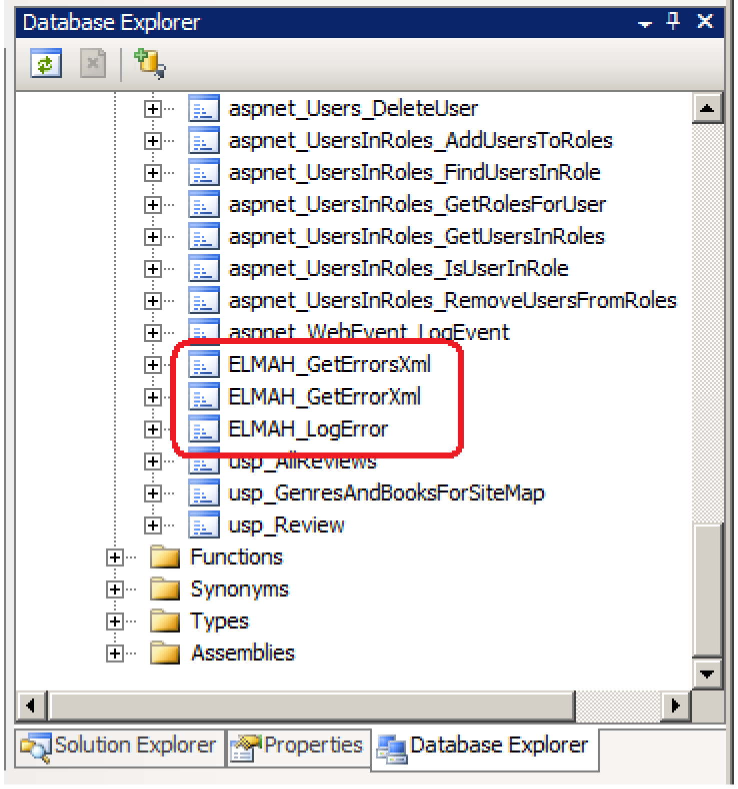

Database Explorer
aspnet_Users_DeleteUser
aspnet_UsersInRoles_AddUsersToRoles
aspnet_UsersInRoles_FindUsersInRole
aspnet_UsersInRoles_GetRolesForUser
aspnet_UsersInRoles_GetUsersInRoles
aspnet_UsersInRoles_IsUserInRole
aspnet_UsersInRoles_RemoveUsersFromRoles
aspnet_WebEvent_LogEvent
ELMAH_GetErrorsXml
ELMAH_GetErrorXml
ELMAH_LogError
usp_AllReviews
usp_GenresAndBooksForSiteMap
usp_Review
Functions
Synonyms
Types
Assemblies
Solution Explorer
Properties
Database Explorer

Figure 2: The `SqlErrorLog` Provider Uses Three Stored Procedures

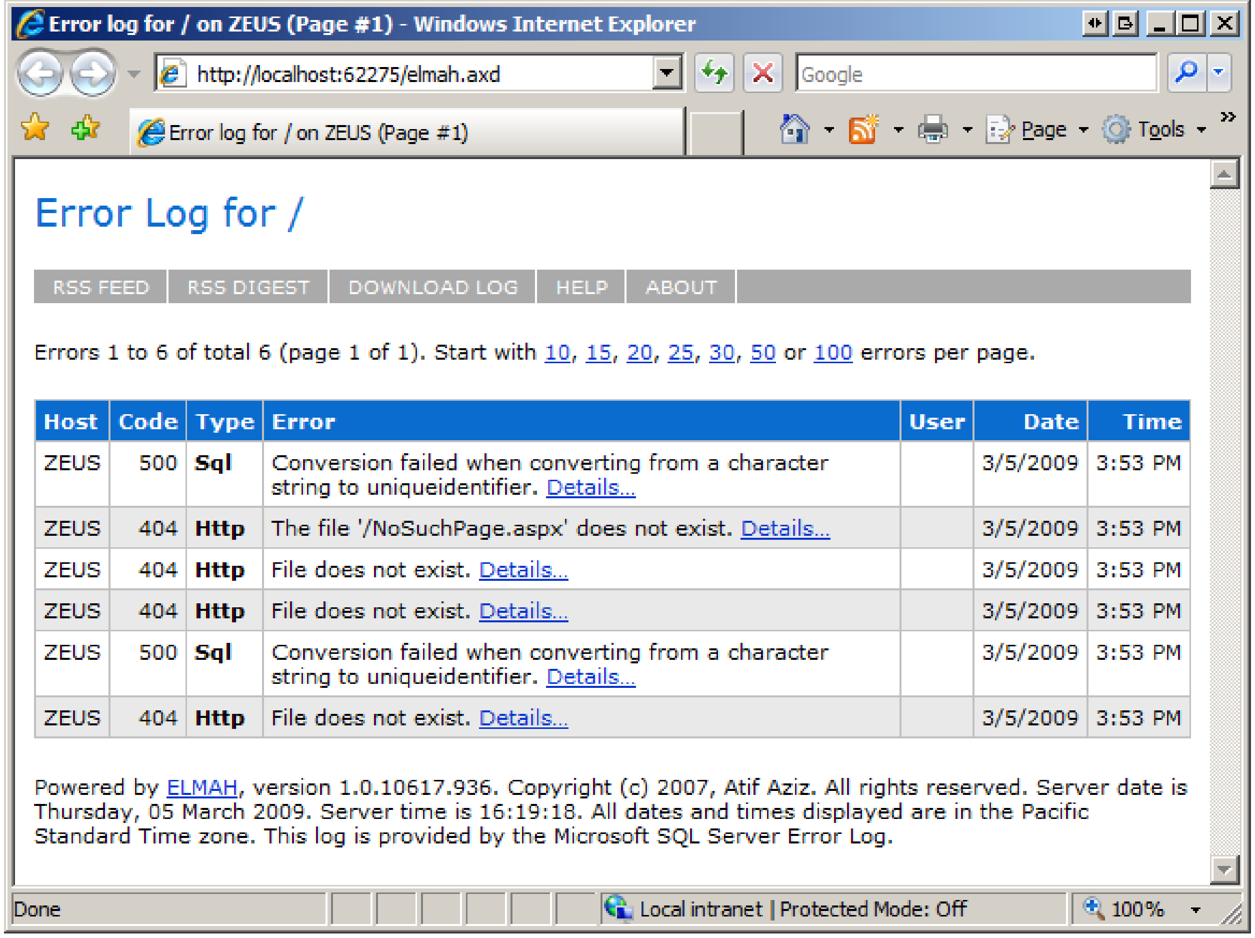
ELMAH In Action

At this point we have added ELMAH to the web application, registered the `ErrorLogModule` HTTP Module and the `ErrorLogPageFactory` HTTP Handler, specified ELMAH's configuration options in `Web.config`, and added the needed database objects for the `SqlErrorLog` error log provider. We are now ready to see ELMAH in action! Visit the Book Reviews website and visit a page that generates a runtime error, such as `Genre.aspx?ID=foo`, or a non-existent page, such as `NoSuchPage.aspx`. What you see when visiting these pages depends on the `<customErrors>` configuration and whether you're visiting locally or remotely. (Refer back to the *Displaying a Custom Error Page* tutorial for a refresher on this topic.)

ELMAH doesn't affect what content is shown to the user when an unhandled exception occurs; it just logs its details. This error log is accessible from the web page `elmah.axd` from the root of your website, such as `http://localhost/BookReviews/elmah.axd`. (This file does not physically exist in your project, but when a request comes in for `elmah.axd` the runtime dispatches it to the `ErrorLogPageFactory` HTTP Handler, which generates the markup sent back to the browser.)

> **Note:** You can also use the `elmah.axd` page to instruct ELMAH to generate a test error. Visiting `elmah.axd/test` (as in, `http://localhost/BookReviews/elmah.axd/test`) causes ELMAH to throw an exception of type `Elmah.TestException`, which has the error message: " This is a test exception that can be safely ignored."

Figure 3 shows the error log when visiting `elmah.axd` from the development environment.


Error log for / on ZEUS (Page #1) - Windows Internet Explorer
http://localhost:62275/elmah.axd
Google
Error log for / on ZEUS (Page #1)
Page
Tools
RSS FEED
RSS DIGEST
DOWNLOAD LOG
HELP
ABOUT
Done
Local intranet | Protected Mode: Off
100%

Error Log for /

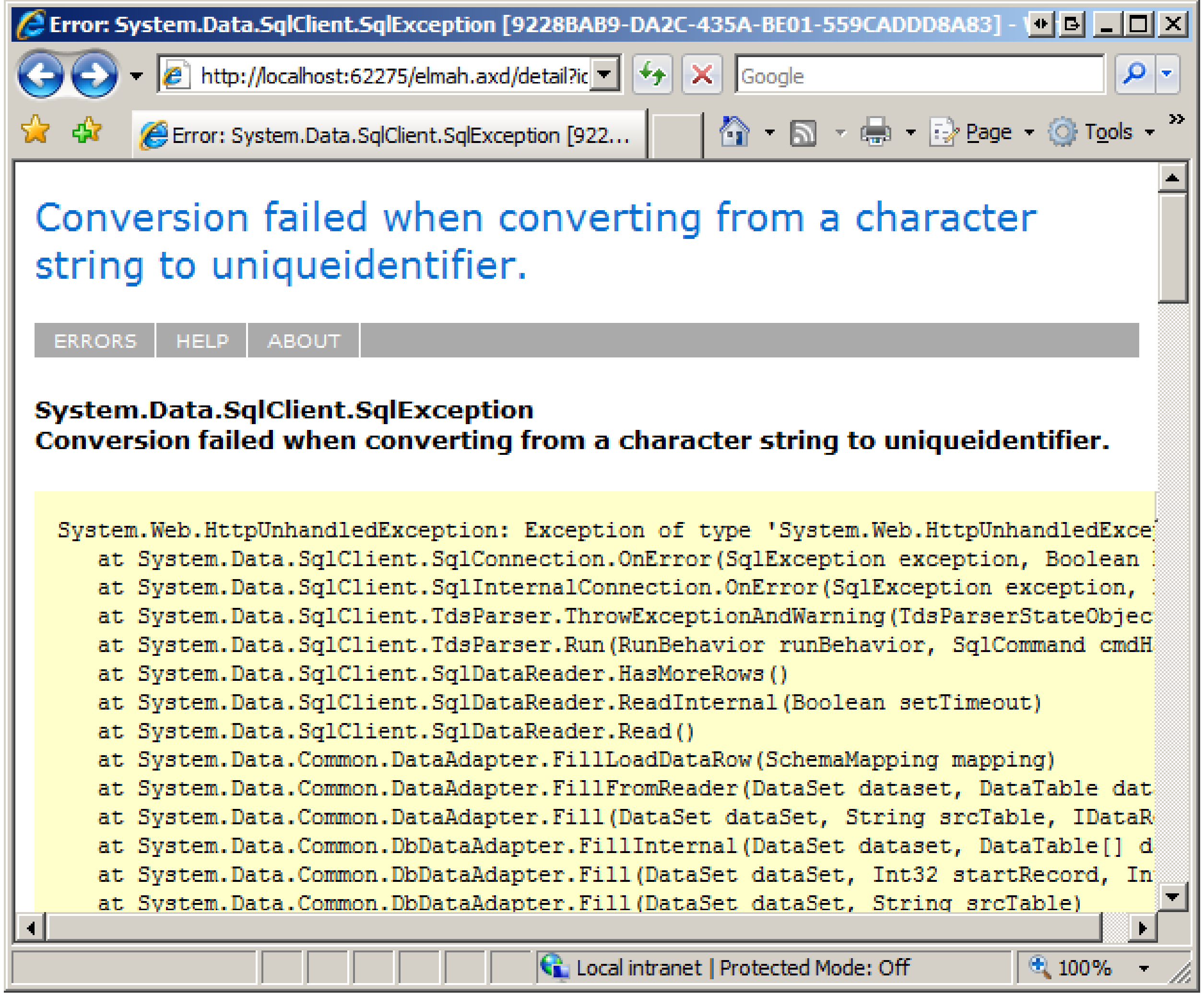
Errors 1 to 6 of total 6 (page 1 of 1). Start with 10, 15, 20, 25, 30, 50 or 100 errors per page.

Host	Code	Type	Error	User	Date	Time
ZEUS	500	**Sql**	Conversion failed when converting from a character string to uniqueidentifier. Details...		3/5/2009	3:53 PM
ZEUS	404	**Http**	The file '/NoSuchPage.aspx' does not exist. Details...		3/5/2009	3:53 PM
ZEUS	404	**Http**	File does not exist. Details...		3/5/2009	3:53 PM
ZEUS	404	**Http**	File does not exist. Details...		3/5/2009	3:53 PM
ZEUS	500	**Sql**	Conversion failed when converting from a character string to uniqueidentifier. Details...		3/5/2009	3:53 PM
ZEUS	404	**Http**	File does not exist. Details...		3/5/2009	3:53 PM

Powered by ELMAH, version 1.0.10617.936. Copyright (c) 2007, Atif Aziz. All rights reserved. Server date is Thursday, 05 March 2009. Server time is 16:19:18. All dates and times displayed are in the Pacific Standard Time zone. This log is provided by the Microsoft SQL Server Error Log.

Figure 3: `Elmah.axd` Displays the Error Log from a Web Page

The error log in Figure 3 contains six error entries. Each entry includes the HTTP status code (404 or 500, for these errors), the type, the description, the name of the logged on user when the error occurred, and the date and time. Clicking the Details link displays a page that includes the same error message shown in the Error Details Yellow Screen of Death (see Figure 4) along with the values of the server variables at the time of the error (see Figure 5). You can also view the raw XML in which the error details are saved, which includes additional information such as the values in the HTTP POST header.

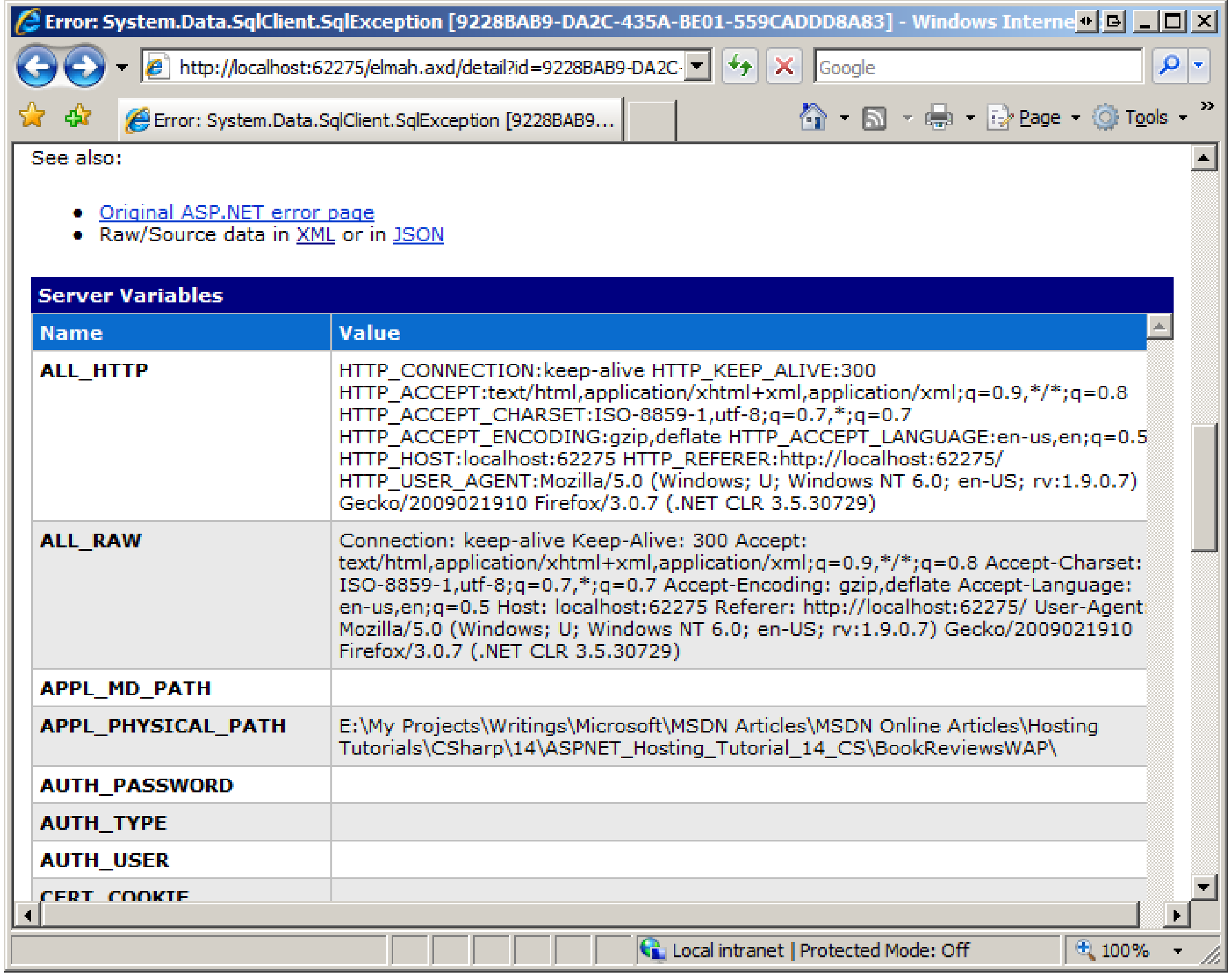

Error: System.Data.SqlClient.SqlException [9228BAB9-DA2C-435A-BE01-559CADDD8A83] -
http://localhost:62275/elmah.axd/detail?ic
Google
Error: System.Data.SqlClient.SqlException [922...
Page
Tools
ERRORS
HELP
ABOUT
Local intranet | Protected Mode: Off
100%

Conversion failed when converting from a character string to uniqueidentifier.

System.Data.SqlClient.SqlException
Conversion failed when converting from a character string to uniqueidentifier.

```
System.Web.HttpUnhandledException: Exception of type 'System.Web.HttpUnhandledExce
   at System.Data.SqlClient.SqlConnection.OnError(SqlException exception, Boolean 
   at System.Data.SqlClient.SqlInternalConnection.OnError(SqlException exception, 
   at System.Data.SqlClient.TdsParser.ThrowExceptionAndWarning(TdsParserStateObjec
   at System.Data.SqlClient.TdsParser.Run(RunBehavior runBehavior, SqlCommand cmdH
   at System.Data.SqlClient.SqlDataReader.HasMoreRows()
   at System.Data.SqlClient.SqlDataReader.ReadInternal(Boolean setTimeout)
   at System.Data.SqlClient.SqlDataReader.Read()
   at System.Data.Common.DataAdapter.FillLoadDataRow(SchemaMapping mapping)
   at System.Data.Common.DataAdapter.FillFromReader(DataSet dataset, DataTable dat
   at System.Data.Common.DataAdapter.Fill(DataSet dataSet, String srcTable, IDataR
   at System.Data.Common.DbDataAdapter.FillInternal(DataSet dataset, DataTable[] d
   at System.Data.Common.DbDataAdapter.Fill(DataSet dataSet, Int32 startRecord, In
   at System.Data.Common.DbDataAdapter.Fill(DataSet dataSet, String srcTable)
```

Figure 4: View the Error Details YSOD


Error: System.Data.SqlClient.SqlException [9228BAB9-DA2C-435A-BE01-559CADDD8A83] - Windows Interne
http://localhost:62275/elmah.axd/detail?id=9228BAB9-DA2C-
Google
Error: System.Data.SqlClient.SqlException [9228BAB9...
Page
Tools
Local intranet | Protected Mode: Off
100%

See also:

- Original ASP.NET error page
- Raw/Source data in XML or in JSON

Server Variables

Name	Value
ALL_HTTP	HTTP_CONNECTION:keep-alive HTTP_KEEP_ALIVE:300 HTTP_ACCEPT:text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8 HTTP_ACCEPT_CHARSET:ISO-8859-1,utf-8;q=0.7,*;q=0.7 HTTP_ACCEPT_ENCODING:gzip,deflate HTTP_ACCEPT_LANGUAGE:en-us,en;q=0.5 HTTP_HOST:localhost:62275 HTTP_REFERER:http://localhost:62275/ HTTP_USER_AGENT:Mozilla/5.0 (Windows; U; Windows NT 6.0; en-US; rv:1.9.0.7) Gecko/2009021910 Firefox/3.0.7 (.NET CLR 3.5.30729)
ALL_RAW	Connection: keep-alive Keep-Alive: 300 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8 Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7 Accept-Encoding: gzip,deflate Accept-Language: en-us,en;q=0.5 Host: localhost:62275 Referer: http://localhost:62275/ User-Agent Mozilla/5.0 (Windows; U; Windows NT 6.0; en-US; rv:1.9.0.7) Gecko/2009021910 Firefox/3.0.7 (.NET CLR 3.5.30729)
APPL_MD_PATH	
APPL_PHYSICAL_PATH	E:\My Projects\Writings\Microsoft\MSDN Articles\MSDN Online Articles\Hosting Tutorials\CSharp\14\ASPNET_Hosting_Tutorial_14_CS\BookReviewsWAP\
AUTH_PASSWORD	
AUTH_TYPE	
AUTH_USER	
CERT_COOKIE	

Figure 5: Explore the Values of the Server Variables Collection at the Time of the Error

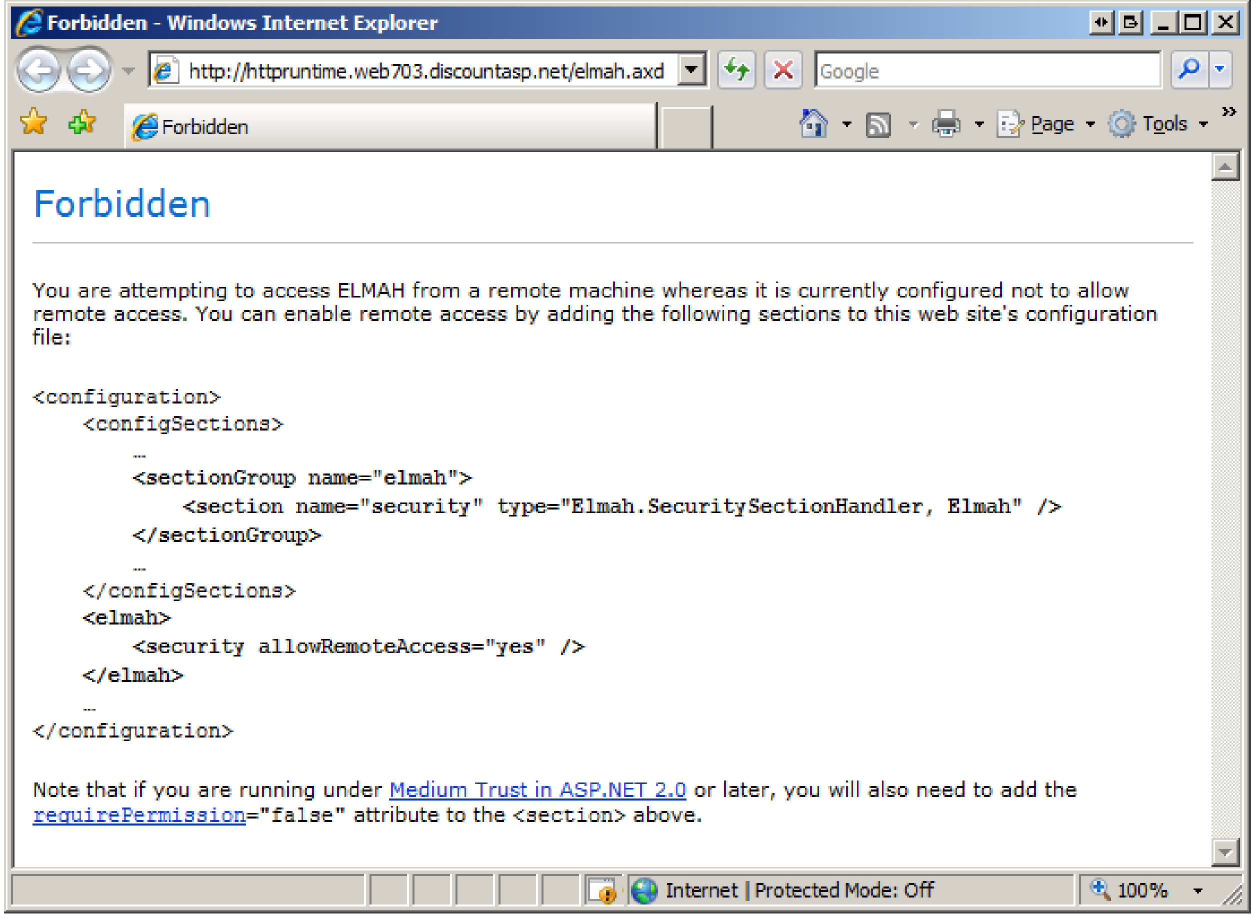
Deploying ELMAH to the production website entails:

- Copying the `Elmah.dll` file to the `Bin` folder on production,
- Copying the ELMAH-specific configuration settings to the `Web.config` file used on production, and
- Adding the error log source infrastructure to the production database.

We've explored techniques for copying files from development to production in previous tutorials. Perhaps the easiest way to get the error log source infrastructure on the production database is to use SQL Server Management Studio to connect to the production database and then execute the `SqlServer.sql` script file, which will create the needed table and stored procedures.

Viewing the Error Details Page on Production

After deploying your site to production, visit the production website and generate an unhandled exception. As in the development environment, ELMAH has no affect on the error page displayed when an unhandled exception occurs; instead, it merely logs the error. If you attempt to visit the error log page (`elmah.axd`) from the production environment, you will be greeted with the Forbidden page shown in Figure 6.


Forbidden - Windows Internet Explorer
http://httpruntime.web703.discountasp.net/elmah.axd
Google
Forbidden
Page
Tools
Internet | Protected Mode: Off
100%

Forbidden

You are attempting to access ELMAH from a remote machine whereas it is currently configured not to allow remote access. You can enable remote access by adding the following sections to this web site's configuration file:

```
<configuration>
    <configSections>
        ...
        <sectionGroup name="elmah">
            <section name="security" type="Elmah.SecuritySectionHandler, Elmah" />
        </sectionGroup>
        ...
    </configSections>
    <elmah>
        <security allowRemoteAccess="yes" />
    </elmah>
    ...
</configuration>
```

Note that if you are running under Medium Trust in ASP.NET 2.0 or later, you will also need to add the `requirePermission="false"` attribute to the <section> above.

Figure 6: By Default, Remote Visitors Cannot View the Error Log Web Page

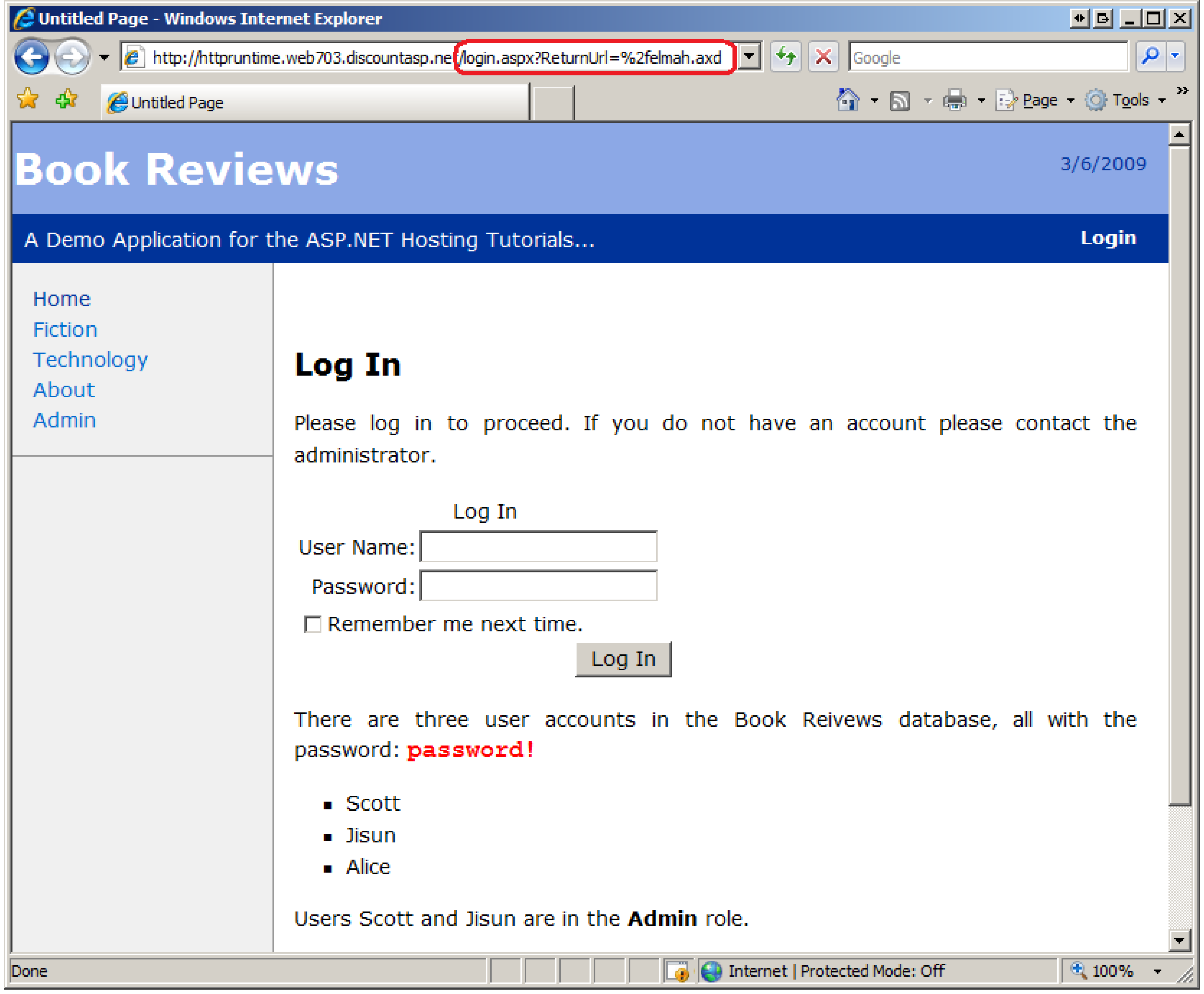
Recall that in the ELMAH configuration's `<security>` section we set the `allowRemoteAccess` attribute to 0, which prohibits remote users from viewing the error log. It's important to prohibit anonymous visitors from viewing the error log, as the error details might reveal security vulnerabilities or other sensitive information. If you decide to set this attribute to 1 and enable remote access to the error log then it is important to lock down the `elmah.axd` path so that only authorized visitors can access it. This can be achieved by adding a `<location>` element to the `Web.config` file.

The following configuration permits only users in the Admin role to access the error log web page:

```
<?xml version="1.0"?>
<configuration>
    ...

    <elmah>
        <security allowRemoteAccess="1" />

        ...
    </elmah>

    ...
    <location path="elmah.axd">
        <system.web>
            <authorization>
                <allow roles="Admin" />
                <deny users="*" />
            </authorization>
        </system.web>
    </location>
</configuration>
```

Note: The Admin role and the three users in the system - Scott, Jisun, and Alice - were added in the *Configuring a Website That Uses Application Services* tutorial. Users Scott and Jisun are members of the Admin role. For more information on authentication and authorization, refer to my Website Security Tutorials.

The error log on the production environment can now be viewed by remote users; refer back to Figures 3, 4, and 5 for screen shots of the error log web page. However, if an anonymous or non-Admin user attempts to view the error log page they are automatically redirected to the login page (`Login.aspx`), as Figure 7 shows.



Untitled Page - Windows Internet Explorer

http://httpruntime.web703.discountasp.net/login.aspx?ReturnUrl=%2felmah.axd

Book Reviews 3/6/2009

A Demo Application for the ASP.NET Hosting Tutorials... **Login**

Home
Fiction
Technology
About
Admin

Log In

Please log in to proceed. If you do not have an account please contact the administrator.

Log In

User Name:

Password:

☐ Remember me next time.

Log In

There are three user accounts in the Book Reivews database, all with the password: **password!**

- Scott
- Jisun
- Alice

Users Scott and Jisun are in the **Admin** role.

Done | Internet | Protected Mode: Off | 100%

Figure 7: Unauthorized Users are Automatically Redirected to the Login Page

Programmatically Logging Errors

ELMAH's `ErrorLogModule` HTTP Module automatically logs unhandled exceptions to the specified log source. Alternatively, you can log an error without having to raise an unhandled exception by using the `ErrorSignal` class and its `Raise` method. The `Raise` method is passed an `Exception` object and logs it as if that exception had been thrown and had reached the ASP.NET runtime without being handled. The difference, however, is that the request continues executing normally after the `Raise` method has been called, whereas a thrown, unhandled exception interrupts the request's normal execution and causes the ASP.NET runtime to display the configured error page.

The `ErrorSignal` class is useful in situations where there is some action that may fail, but its failure is not catastrophic to the overall operation being performed. For instance, a website may contain a form that takes the user's input, stores it in a database, and then sends the user an e-mail informing them that they information was processed. What should happen if the information is saved to the database successfully, but there is an error when sending the e-mail message? One option would be to throw an exception and send the user to the error page. However, this might confuse the user into thinking that the information they entered was not saved. Another approach would be to log the e-mail-related error, but not alter the user's experience in any way. This is where the `ErrorSignal` class is useful.

```
// ... Save user's information to the database ...
...

// Attempt to send the user a confirmation e-mail
try
{
    // ... Send an e-mail ...
}
catch (Exception e)
{
    // Error in sending e-mail. Log it!
    ErrorSignal.FromCurrentContext().Raise(e);
}
```

Error Notification Via E-Mail

Along with logging errors to a database, ELMAH can also be configured to e-mail error details to a specified recipient. This functionality is provided by the `ErrorMailModule` HTTP Module; therefore, you must register this HTTP Module in `Web.config` in order to send error details via e-mail.

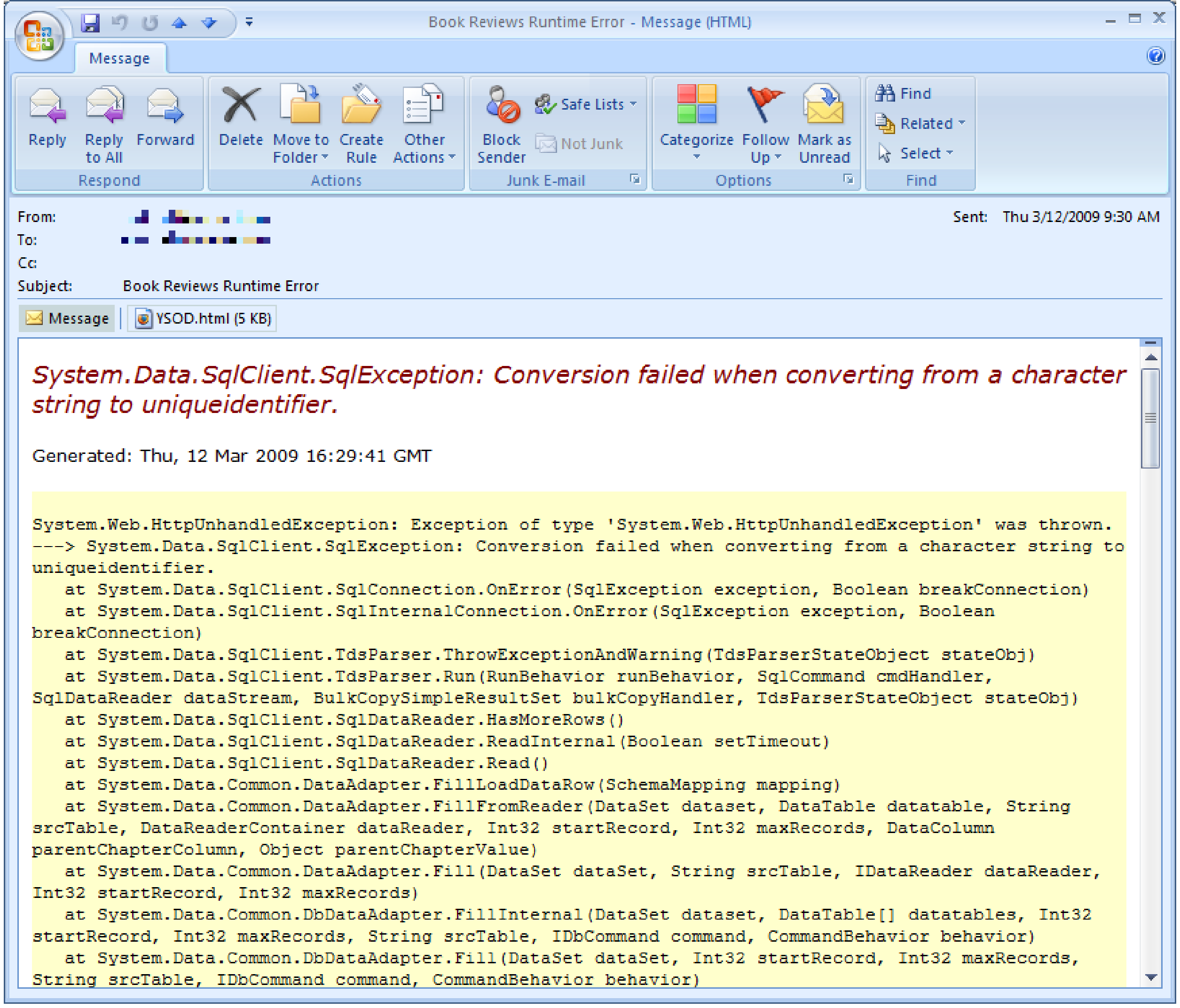
```
<?xml version="1.0"?>
<configuration>
    ...

    <system.web>
        ...

        <httpModules>
            ...

            <add name="ErrorLog" type="Elmah.ErrorLogModule, Elmah"/>
            <add name="ErrorMail" type="Elmah.ErrorMailModule, Elmah"/>
        </httpModules>
    </system.web>

    <system.webServer>
        <validation validateIntegratedModeConfiguration="false"/>
        <modules>
            ...
            <add name="Elmah.ErrorLog" type="Elmah.ErrorLogModule, Elmah"
preCondition="managedHandler" />
            <add name="Elmah.ErrorMail" type="Elmah.ErrorMailModule"
preCondition="managedHandler" />
        </modules>

        ...
    </system.webServer>
</configuration>
```

Next, specify information about the error e-mail in the `<elmah>` element's `<errorMail>` section, indicating the e-mail's sender and recipient, the subject, and whether the e-mail is sent asynchronously.

```
<?xml version="1.0"?>
<configuration>
    ...

    <elmah>
        ...

        <errorMail from="support@example.com"
                   to="support@example.com"
                   subject="Book Reviews Runtime Error"
                   async="true" />
    </elmah>

    ...
</configuration>
```

With the above settings in place, whenever a runtime error occurs ELMAH sends an e-mail to support@example.com with the error details. ELMAH's error e-mail includes the same information shown in the error details web page, namely the error message, the stack trace, and the server variables (refer back to Figures 4 and 5). The error e-mail also includes the Exception Details Yellow Screen of Death content as an attachment (`YSOD.html`).

Figure 8 shows ELMAH's error e-mail generated by visiting `Genre.aspx?ID=foo`. While Figure 8 shows only the error message and stack trace, the server variables are included further down in the e-mail's body.



From: [redacted]
To: [redacted]
Cc:
Subject: Book Reviews Runtime Error
Sent: Thu 3/12/2009 9:30 AM

System.Data.SqlClient.SqlException: Conversion failed when converting from a character string to uniqueidentifier.

Generated: Thu, 12 Mar 2009 16:29:41 GMT

```
System.Web.HttpUnhandledException: Exception of type 'System.Web.HttpUnhandledException' was thrown.
---> System.Data.SqlClient.SqlException: Conversion failed when converting from a character string to
uniqueidentifier.
   at System.Data.SqlClient.SqlConnection.OnError(SqlException exception, Boolean breakConnection)
   at System.Data.SqlClient.SqlInternalConnection.OnError(SqlException exception, Boolean
breakConnection)
   at System.Data.SqlClient.TdsParser.ThrowExceptionAndWarning(TdsParserStateObject stateObj)
   at System.Data.SqlClient.TdsParser.Run(RunBehavior runBehavior, SqlCommand cmdHandler,
SqlDataReader dataStream, BulkCopySimpleResultSet bulkCopyHandler, TdsParserStateObject stateObj)
   at System.Data.SqlClient.SqlDataReader.HasMoreRows()
   at System.Data.SqlClient.SqlDataReader.ReadInternal(Boolean setTimeout)
   at System.Data.SqlClient.SqlDataReader.Read()
   at System.Data.Common.DataAdapter.FillLoadDataRow(SchemaMapping mapping)
   at System.Data.Common.DataAdapter.FillFromReader(DataSet dataset, DataTable datatable, String
srcTable, DataReaderContainer dataReader, Int32 startRecord, Int32 maxRecords, DataColumn
parentChapterColumn, Object parentChapterValue)
   at System.Data.Common.DataAdapter.Fill(DataSet dataSet, String srcTable, IDataReader dataReader,
Int32 startRecord, Int32 maxRecords)
   at System.Data.Common.DbDataAdapter.FillInternal(DataSet dataset, DataTable[] datatables, Int32
startRecord, Int32 maxRecords, String srcTable, IDbCommand command, CommandBehavior behavior)
   at System.Data.Common.DbDataAdapter.Fill(DataSet dataSet, Int32 startRecord, Int32 maxRecords,
String srcTable, IDbCommand command, CommandBehavior behavior)
```

Figure 8: You Can Configure ELMAH To Send Error Details Via E-Mail

Only Logging Errors Of Interest

By default, ELMAH logs the details of every unhandled exception, including 404 and other HTTP errors. You can instruct ELMAH to ignore these or other types of errors using error filtering. The filtering logic is performed by ELMAH's `ErrorFilterModule` HTTP Module, which you'll need to register in `Web.config` in order to use the filtering logic. The rules for filtering are specified in the `<errorFilter>` section.

The following markup instructs ELMAH to not log 404 errors.

```
<?xml version="1.0"?>
<configuration>
    ...

    <elmah>
        ...
        <errorFilter>
            <test>
                <equal binding="HttpStatusCode" value="404" type="Int32" />
            </test>
        </errorFilter>
    </elmah>

    ...
</configuration>
```

> **Note:** Don't forget, in order to use error filtering you must register the `ErrorFilterModule` HTTP Module.

The `<equal>` element inside the `<test>` section is referred to as an assertion. If the assertion evaluates to true then the error is filtered from ELMAH's log. There are other assertions available, including: `<greater>`, `<greater-or-equal>`, `<not-equal>`, `<lesser>`, `<lesser-or-equal>`, and so on. You can also combine assertions using the `<and>` and `<or>` Boolean operators. What's more, you can even include a simple JavaScript expression as an assertion, or write your own assertions in C# or Visual Basic.

For more information on ELMAH's error filtering capabilities, refer to the Error Filtering section in the ELMAH wiki.

Summary

ELMAH provides a simple, yet powerful mechanism for logging errors in an ASP.NET web application. Like Microsoft's health monitoring system, ELMAH can log errors to a database and can send the error details to a developer via e-mail. Unlike the health monitoring system, ELMAH includes out of the box support for a wider range of error log data stores, including: Microsoft SQL Server, Microsoft Access, Oracle, XML files, and several others. Moreover, ELMAH offers a built-in mechanism for viewing the error log and details about a specific error from a web page, `elmah.axd`. The `elmah.axd` page can also render error information as an RSS feed or as a comma-separated value file (CSV), which you can read using Microsoft Excel. You can also instruct ELMAH to filter errors from the log using declarative or programmatic assertions. And ELMAH can be used with ASP.NET version 1.x applications.

Every deployed application should have some mechanism for automatically logging unhandled exceptions and sending notification to the development team. Whether this is accomplished using health monitoring or ELMAH is secondary. In other words, it doesn't really matter much whether you use health monitoring or ELMAH; evaluate both systems and then choose the one that best fits your needs. What is fundamentally important is that some mechanism be put in place to log unhandled exceptions in the production environment.

Happy Programming!

Further Reading

For more information on the topics discussed in this tutorial, refer to the following resources:

- ELMAH - Error Logging Modules and Handlers
- ELMAH Project Page (source code, samples, wiki)
- Plugging ELMAH Into a Web Application to Catch Unhandled Exceptions (video)
- Security Error Log Pages
- Using HTTP Modules and Handlers to Create Pluggable ASP.NET Components
- Website Security Tutorials

About the Author

Scott Mitchell, author of multiple ASP/ASP.NET books and founder of 4GuysFromRolla.com, has been working with Microsoft Web technologies since 1998. Scott works as an independent consultant, trainer, and writer. His latest book is *Sams Teach Yourself ASP.NET 3.5 in 24 Hours*. Scott can be reached at mitchell@4guysfromrolla.com or via his blog at http://ScottOnWriting.NET.

Special Thanks To…

This tutorial series was reviewed by many helpful reviewers. Interested in reviewing my upcoming MSDN articles? If so, drop me a line at mitchell@4GuysFromRolla.com.