

## Lesson 12-3: Log errors to a database

By adding some LINQ database code to your *Global.asax* file, you can log any errors that happen on your site to a database.

Keeping a log of errors that happen on your site is good practice, as it helps you to track down and fix any problems that occur, even if they are not reported by your users.

If you had several web applications (on a corporate intranet, for example), you could even use this technique to centralize your error logging into a single database, allowing you to see all application problems in one place.

- 1 Open *SmartMethodStore* from your sample files folder.
- 2 Open *Global.asax*.
- 3 Add code to log all errors to a database.

Although the error handling code you added in *Lesson 12-2: Handle errors with Global.asax* is useful, you could have done the same thing using the *ASP.NET Configuration* utility (you learned to use this in: *Lesson 9-2: Manage a site with ASP.NET Configuration*).

Manually adding code to *Global.asax* allows you to customize error handling to a far greater extent.

1. Add *LINQ to SQL Classes*, naming the LINQ file: **Store.dbml**

You learned how to do this in *Lesson 10-2: Add LINQ data classes to a project*.

2. Add all database tables from the *Store* database to your new LINQ classes.

You learned how to do this in *Lesson 10-2: Add LINQ data classes to a project*.

3. Open *Global.asax*.
4. Add the following code to the beginning of your *Application\_Error* event handler:

```
using (StoreDataContext Data = new StoreDataContext())
{
    Exception ExceptionToLog = Server.GetLastError();
    Error NewError = new Error();
    NewError.ErrorMessage = ExceptionToLog.Message;
    NewError.ErrorStackTrace = ExceptionToLog.StackTrace;
    NewError.ErrorURL = Request.Url.ToString();
    NewError.ErrorDate = DateTime.Now;
    Data.Errors.InsertOnSubmit(NewError);
    Data.SubmitChanges();
}
```

This code uses the *Server.GetLastError* method to retrieve an *Exception* object (called *ExceptionToLog*) containing details of the last error that occurred on the web server. You learned

about the *Exception* object in: *Lesson 3-5: Understand the Exception object*.

The code then needs to log the information in the *ExceptionToLog* object by adding a new row to the *Error* table in the *Store* database. It does this by creating a new *Error* object (representing a row in the *Error* table) called *NewError*.

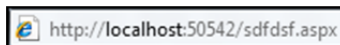
The error details are then transferred from the *ExceptionToLog* object to the *NewError* object and inserted into the database.

This process is covered in more depth in: *Lesson 10-8: Insert database records using LINQ*.

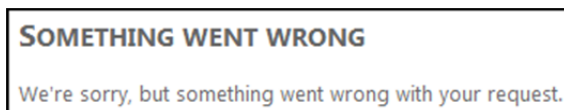
```
void Application_Error(object sender, EventArgs e)
{
    // Code that runs when an unhandled error occurs
    using (StoreDataContext Data = new StoreDataContext())
    {
        Exception ExceptionToLog = Server.GetLastError();
        Error NewError = new Error();
        NewError.ErrorMessage = ExceptionToLog.Message;
        NewError.ErrorStackTrace = ExceptionToLog.StackTrace;
        NewError.ErrorURL = Request.Url.ToString();
        NewError.ErrorDate = DateTime.Now;
        Data.Errors.InsertOnSubmit(NewError);
        Data.SubmitChanges();
    }
    Response.Redirect("~/error.aspx");
}
```

#### 4 Test your error logging.

1. View the site in your browser.
2. Attempt to navigate to a URL that doesn't exist.



You should be redirected to the error page, just as you were in: *Lesson 12-2: Handle errors with Global.asax*.



This time, however, the error was also written to the *Error* table of the *Store* database.

3. Close your web browser.
4. View the contents of the *Error* table using the *Database Explorer*.

You learned how to do this in: *Lesson 10-1: Work with SQL databases in Visual Studio*.

You will see that the details of the error have been recorded in the table.

ErrorID	ErrorURL	ErrorMessage
1	http://localhost:50542/sdfdsf.aspx	File does not exist.
NULL	NULL	NULL

#### 5 Close Visual Studio.