Elmah Everywhere Documentation

Elmah Everywhere is an exception logging library for .NET, Silverlight, WPF, ASP.NET MVC and WCF that uses an ELMAH (Error Logging Modules and Handlers for ASP.NET).

For more detailed information how to configure and use Elmah Everywhere see source code and samples at <https://github.com/vincoss/vinco-logging-toolkit>.

Error Database

Error database is located on project directory. “Vinco.Elmah.Everywhere\Source\ErrorWebSite\App\_Data”

Error web site

# Error website default login details

You can access sample error web site at http://localhost:11079/.

* UserName : administrator
* Password: p@ssword

Configuration

# Silverlight configuration

You can configure error logging as indicated in the following example.

private static void SetUpExceptionHandler()

{

// Configure

var writter = new ClientHttpExceptionWritter

{

// **NOTE: Possible to pass URI by startup arguments.**

RequestUri = new Uri("http://localhost:11079/error/log", UriKind.Absolute)

};

Uri uri = Application.Current.Host.Source;

var defaults = new ExceptionDefaults

{

Token = "Silverlight-Test-Token",

ApplicationName = "Silverlight-Sample",

Host = string.Format("{0}{1}{2}:{3}", uri.Scheme, Uri.SchemeDelimiter, uri.Host, uri.Port)

};

ExceptionHandler.Configure(writter, defaults);

}

**In Application constructor call handler setup method.**

public App()

{

SetUpExceptionHandler();

}

**Add handler log into Application\_UnhandledException method**

private void Application\_UnhandledException(object sender, ApplicationUnhandledExceptionEventArgs e)

{

ExceptionHandler.Report(e.ExceptionObject, null);

}

# WPF configuration

You can configure error logging as indicated in the following example.

**In Application constructor call handler setup method.**

public App()

{

// Configure error handler from configuration file

ExceptionHandler.ConfigureFromConfigurationFile(new HttpExceptionWritter());

}

**Attach handler.**

protected override void OnStartup(StartupEventArgs e)

{

ExceptionHandler.Attach(AppDomain.CurrentDomain);

}

**Detach handler.**

protected override void OnExit(ExitEventArgs e)

{

ExceptionHandler.Detach(AppDomain.CurrentDomain);

}

**Configuration details in App.config file**

<everywhere>

<!-- Configure Elmah.Everywhere

URL: Remote web site url to log an error

Token: Token to identify client

ApplicationName: Error source

Host: Error host

-->

<settings remoteLogUri="http://localhost:11079/error/log"

token="Token-Test"

host="Wpf-Sample"

applicationName="Exceptions-Handler"/>

</everywhere>

# Code configuration

You can configure error logging in code as indicated in the following example.

static void Main(string[] args)

{

var writter = new HttpExceptionWritter

{

RequestUri = new Uri("http://localhost:11079/error/log", UriKind.Absolute)

};

var defaults = new ExceptionDefaults

{

Token = "Test-Token",

ApplicationName = "Console-Sample",

Host = Environment.MachineName

};

ExceptionHandler.Configure(writter, defaults);

}

# WCF configuration with Web.config or App.config file

To use WCF service error logging, you should configure service behaviour in configuration file or in code. You can configure error logging as indicated in the following example.

<system.serviceModel>

<serviceHostingEnvironment aspNetCompatibilityEnabled="true" multipleSiteBindingsEnabled="true" />

<behaviors>

<serviceBehaviors>

<behavior name="">

<ElmahErrorLog />

</behavior>

</serviceBehaviors>

</behaviors>

<extensions>

<behaviorExtensions>

<add name="ElmahErrorLog" type="Elmah.Everywhere.ServiceModel.ErrorBehaviorExtensionElement, Elmah.Everywhere"/>

</behaviorExtensions>

</extensions>

</system.serviceModel>

# WCF configuration with behaviour attribute from code

You can configure error logging as indicated in the following example.

[ServiceHttpErrorBehavior(typeof(HttpErrorHandler))]

public class MyService : IMyService

{

Tracing

# Trace diagnostics configuration

Elmah.Everywhere tracing is built on top of System.Diagnostics. To use tracing, you should define trace sources in configuration file or in code. You can configure trace logging as indicated in the following example.

<configuration>

<system.diagnostics>

<trace autoflush="true">

<listeners>

<clear/>

<add name="fileListener" type="System.Diagnostics.TextWriterTraceListener" initializeData="E:\\_Log\Console\_Sample\_FileListener.log" />

</listeners>

</trace>

</system.diagnostics>

</configuration>