Introduction

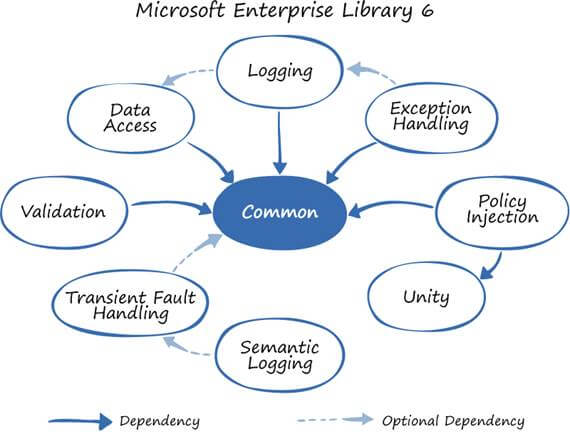
The Microsoft Enterprise Library is a collection of reusable software components (application blocks) designed to assist software developers with common enterprise development cross-cutting concerns, such as logging, validation, data access, exception handling, and many others.

Enterprise Library comes with an easy configuration tool that makes it easier to plug required application blocks into your application. Most developers, though, plug in these application blocks programmatically since this approach is also easy.

What are the application blocks?

The definition we use is “pluggable and reusable software compo­nents designed to assist developers with common enterprise development challenges.” Application blocks help address the kinds of problems developers commonly face from one line-of-business project to the next.

Their design encapsulates the Microsoft recommended practices for Microsoft .NET Framework-based applications, and developers can add them to .NET based applications so as to configure them quickly and easily.



**Application Blocks :**

Data Access Blocks :

Example 1 :

* Creating Application
* Creating App Library
* Adding Enterprise Library
* Working with Application

Open sql server Management studio and create Procedure under database db3 : (College table with some data must be available in the database)

create procedure ProcCollege

as

select \* from college

**Creating Application**

1. Create new C# console application
2. Right click on the solution explorer and add Class Library Project (Name :CollegeLibraryTracker)
3. Right click on the class library project > Manage Nuget Pakage > Install Enterprice Library-Data
4. To this class library project add two classes : CollegeModel,CollegeDAL
5. Add the below code in the files :

using Microsoft.Practices.EnterpriseLibrary.Data;

using Microsoft.Practices.EnterpriseLibrary.Data.Sql;

using System.Data;

using System.Data.Common;

namespace CollegeLibraryTracker

{

public class CollegeDAL

{

public static string constr = "initial catalog=db3 ;data source=M4L-FAC-RASHMI ; integrated security=true";

public DataSet GetData()

{

Database db = new SqlDatabase(constr);

DbCommand objcmd = db.GetStoredProcCommand("dbo.ProcCollege");

return db.ExecuteDataSet(objcmd);

//To Execute a sql command

//Database \_database = DatabaseFactory.CreateDatabase("AdventureWorksConnectionString");

// string sqlCommand = "SELECT TOP 5 FirstName, LastName, JobTitle FROM HumanResources.vEmployee";

//DbCommand dbCommand = \_database.GetSqlStringCommand(sqlCommand);

// DataSet \_dataSet = \_database.ExecuteDataSet(dbCommand);

}

}

}

1. Build both projects separately
2. Add a reference of dll in the console application
3. Open console application and write the code below :

using CollegeLibraryTracker;

using System.Data;

namespace EL\_Demo2

{

class Program

{

static void Main(string[] args)

{

CollegeDAL dalobj = new CollegeDAL();

DataSet ds= dalobj.GetData();

DataTable dt = ds.Tables[0];

DataRow dr = dt.Rows[0];

string s1 = dr["Col\_name"].ToString();

string s2 = dr["City"].ToString();

Console.WriteLine("Data Retrived");

Console.WriteLine(s1 + "\n" + s2);

Console.ReadLine();

}

}

}

Example 2 :

Data Access Application Blocks :

1) Open VSTS

2)Create new Web application

3) Manage Nuget Pakages -> Enterprice Library .commn

4) Manage Nuget Pakages -> Enterprice Library .data

5)Add new Webform to the application

6) Drag and Drop GridView on the form

7)Bind Gridview to some table in the database( Database name : collabra10 Table name : Userinfo)

8) Open sql server create a stored procedure

create procedure GetUser

as

select \* from UserInfo

return

9)Set DataSource property of the grid to none

10) write below code in Page\_Load

using Microsoft.Practices.EnterpriseLibrary.Data;

using Microsoft.Practices.EnterpriseLibrary.Common;

using System.Data;

Database objDataBase = new SqlDatabase(System.Configuration.ConfigurationManager.ConnectionStrings["collbra10ConnectionString"].ConnectionString);

DataSet ds = new DataSet();

try

{

ds = objDataBase.ExecuteDataSet("GetUsers");

GridView2.DataSource = ds.Tables[0];

GridView2.DataBind();

}

catch (Exception ex)

{

throw ex;

}

11) Execute the Webform1

Using Application logging :

1) Open VSTS

//Open the above application created

2) Nuget Page Manager -> Enterprice library.Logging

3) Write the below code in the web.config file :

<configSections>

<section name="loggingConfiguration" type="Microsoft.Practices.EnterpriseLibrary.Logging.Configuration.LoggingSettings, Microsoft.Practices.EnterpriseLibrary.Logging, Version=6.0.0.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35" requirePermission="true" />

</configSections>

<loggingConfiguration name="" tracingEnabled="false" defaultCategory="General"

logWarningsWhenNoCategoriesMatch="false">

<listeners>

<add name="Rolling Flat File Trace Listener" type="Microsoft.Practices.EnterpriseLibrary.Logging.TraceListeners.RollingFlatFileTraceListener, Microsoft.Practices.EnterpriseLibrary.Logging, Version=6.0.0.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35" listenerDataType="Microsoft.Practices.EnterpriseLibrary.Logging.Configuration.RollingFlatFileTraceListenerData, Microsoft.Practices.EnterpriseLibrary.Logging, Version=6.0.0.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35"

fileName="RollingFlatFile.log"

footer="----------------------------------" formatter="Text Formatter"

header="" rollInterval="Day"

traceOutputOptions="DateTime, Timestamp" filter="All" />

<add name="Event Log Trace Listener" type="Microsoft.Practices.EnterpriseLibrary.Logging.TraceListeners.FormattedEventLogTraceListener, Microsoft.Practices.EnterpriseLibrary.Logging, Version=6.0.0.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35" listenerDataType="Microsoft.Practices.EnterpriseLibrary.Logging.Configuration.FormattedEventLogTraceListenerData, Microsoft.Practices.EnterpriseLibrary.Logging, Version=6.0.0.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35" source="Application" formatter="Text Formatter" log="Application" machineName="." traceOutputOptions="None" />

</listeners>

<formatters>

<add type="Microsoft.Practices.EnterpriseLibrary.Logging.Formatters.TextFormatter, Microsoft.Practices.EnterpriseLibrary.Logging, Version=6.0.0.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35"

template="Timestamp: {timestamp(local)}{newline}

Message: {message}{newline}

Category: {category}{newline}

Priority: {priority}{newline}

Severity: {severity}"

name="Text Formatter" />

</formatters>

<categorySources>

<add switchValue="All" autoFlush="true" name="General">

<listeners>

<add name="Rolling Flat File Trace Listener" />

<add name="Event Log Trace Listener" />

</listeners>

</add>

</categorySources>

<specialSources>

<allEvents switchValue="All" name="All Events">

<listeners>

<add name="Rolling Flat File Trace Listener" />

</listeners>

</allEvents>

<notProcessed switchValue="All" name="Unprocessed Category">

<listeners>

<add name="Rolling Flat File Trace Listener" />

</listeners>

</notProcessed>

<errors switchValue="All" name="Logging Errors & Warnings">

<listeners>

<add name="Rolling Flat File Trace Listener" />

</listeners>

</errors>

</specialSources>

</loggingConfiguration>

4) Add New class to web application

public class LoggerBlock

{

protected LogWriter logWriter;

public LoggerBlock()

{

InitLogging();

}

private void InitLogging()

{

logWriter = new LogWriterFactory().Create();

Logger.SetLogWriter(logWriter, false);

}

public LogWriter LogWriter

{

get

{

return logWriter;

}

}

}

public class Executer

{

LoggerBlock loggerBlock = new LoggerBlock();

public static void LogMyMsgs()

{

new Executer().ReadFile();

}

public void ReadFile()

{

WriteTraceLog("Application Started!!!");

string[] lines;

var list = new List<string>();

var fileStream = new FileStream(@"D:\ATOS\Repos\ConsoleApplication1\WebApplication1\RollingFlatFile.log", FileMode.Open, FileAccess.Read);

using (var streamReader = new StreamReader(fileStream, Encoding.UTF8))

{

string line;

while ((line = streamReader.ReadLine()) != null)

{

WriteTraceLog(line);

list.Add(line);

}

}

lines = list.ToArray();

WriteTraceLog("Application Stopped!!!");

}

public void WriteTraceLog(String message)

{

loggerBlock.LogWriter.Write(message, "General", 5, 2000, TraceEventType.Information);

}

}

5) Open webform1.aspx and write the below code in PageLoad :

Executer.LogMyMsgs();

Response.Write("Log entry is done");

6) Run the application : this will create a log file in the application folder

7) Copy the log file in a different location and execute the application again

8) Go to Start -> Search -> Windows Event log ->

**Exception Handling Blocks :**

Exception Handling Blocks :

Exception Logging :process of logging an exception which might include sending formatted exceptions to the event log or sending an email

Exception Policies : allow you to control exception handling and logging behaviours using external configuration files instead of baking such rules into your code.

Create a new web application

add Ref to below From Nuget Pakage manager :

EnterpriceLibrary.Common,EnterPriceLibrary.Logging,ExceptionHandling,ExceptionHandling.Logging

Add new button and write below code in button1\_click

try

{

throw new Exception("This is test Exception");

}

catch(Exception ex)

{

bool rethrow = Microsoft.Practices.EnterpriseLibrary.ExceptionHandling.ExceptionPolicy.HandleException(ex, "Log only Policy");

if (rethrow)

throw;

}

Write the below code in web.config file :

<?xml version="1.0" encoding="utf-8"?>

<!--

For more information on how to configure your ASP.NET application, please visit

https://go.microsoft.com/fwlink/?LinkId=169433

-->

<configuration>

<configSections>

<section name="exceptionHandling" type="Microsoft.Practices.EnterpriseLibrary.ExceptionHandling.Configuration.ExceptionHandlingSettings,Microsoft.Practices.EnterpriseLibrary.ExceptionHandling" />

<section name="loggingConfiguration" type="Microsoft.Practices.EnterpriseLibrary.Logging.Configuration.LoggingSettings,Microsoft.Practices.EnterpriseLibrary.Logging" />

</configSections>

<exceptionHandling>

<exceptionPolicies>

<add name="Global Policy">

<exceptionTypes>

<add name="Exception" type="System.Exception, mscorlib, Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089" postHandlingAction="None">

<exceptionHandlers>

<add name="Application Message Handler" type="LayoutStudio.AppMessageExceptionHandler, LayoutStudio" />

</exceptionHandlers>

</add>

</exceptionTypes>

</add>

</exceptionPolicies>

</exceptionHandling>

<system.web>

<compilation debug="true" targetFramework="4.6.1"/>

<httpRuntime targetFramework="4.6.1"/>

</system.web>

<system.codedom>

<compilers>

<compiler language="c#;cs;csharp" extension=".cs"

type="Microsoft.CodeDom.Providers.DotNetCompilerPlatform.CSharpCodeProvider, Microsoft.CodeDom.Providers.DotNetCompilerPlatform, Version=1.0.8.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35"

warningLevel="4" compilerOptions="/langversion:default /nowarn:1659;1699;1701"/>

<compiler language="vb;vbs;visualbasic;vbscript" extension=".vb"

type="Microsoft.CodeDom.Providers.DotNetCompilerPlatform.VBCodeProvider, Microsoft.CodeDom.Providers.DotNetCompilerPlatform, Version=1.0.8.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35"

warningLevel="4" compilerOptions="/langversion:default /nowarn:41008 /define:\_MYTYPE=\&quot;Web\&quot; /optionInfer+"/>

</compilers>

</system.codedom>

</configuration>