Add-> Global applacation class – Global.asax

Application life style

public class Global : System.Web.HttpApplication

{

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

// Обработчики событий приложения.

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

// Вызывается при старте веб приложения.

void Application\_Start(object sender, EventArgs e)

{

Debug.WriteLine("Start");

}

// Вызывается при завершении приложения.

// Приложение завершается, если к нему, в течении 20 минут (по умолчанию), не было ни одного запроса.

void Application\_End(object sender, EventArgs e)

{

Debug.WriteLine("End");

}

// В приложении возникло исключение, которое не обработали

void Application\_Error(object sender, EventArgs e)

{

Debug.WriteLine("Error");

// Получение исключения, которое произошло на странице.

Exception ex = Server.GetLastError().GetBaseException();

}

// Срабатывает, когда начинается новая сессия.

void Session\_Start(object sender, EventArgs e)

{

Debug.WriteLine("Session\_Start");

}

// Срабатывает при завершении сессии.

// Этот метод будет вызываться только если в web.config, в секции sessionState, атрибут mode установлен, как InProc

// InProc - это значение по умолчанию, означает, что данные сеанса будут храниться в текущем рабочем процессе.

void Session\_End(object sender, EventArgs e)

{

Debug.WriteLine("Session\_End");

}

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

// Обработчики событий запроса. (В хронологическом порядке.)

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

// Вызывается при каждом запросе.

protected void Application\_BeginRequest(object sender, EventArgs e)

{

Debug.WriteLine("BeginRequest");

}

// Срабатывает после аутентификации пользователя, выполняющего запрос. (аутентификация - определение принадлежности к ресурсу, поиск учетной записи)

void Application\_AuthenticateRequest(object sender, EventArgs e)

{

Debug.WriteLine("AuthenticateRequest");

}

// Срабатывает после авторизации пользователя, выполняющего запрос. (авторизация - определение наличия прав доступа)

void Application\_AuthorizeRequest(object sender, EventArgs e)

{

Debug.WriteLine("AuthorizeRequest");

}

// Модули отвечающие за кэширование, извлекают ранее обработанный запрос, минуя вызов обработчика события.

void Application\_ResolveRequestCache(object sender, EventArgs e)

{

Debug.WriteLine("ResolveRequestCache");

}

// Срабатывает после получения состояния сессии (Session state) для текущего запроса.

protected void Application\_AcquireRequestState(object sender, EventArgs e)

{

Debug.WriteLine("AcquireRequestState");

}

// Срабатывает до того как начнется вызов обработчика.

protected void Application\_PreRequestHandlerExecute(object sender, EventArgs e)

{

Debug.WriteLine("PreRequestHandlerExecute");

}

// ВЫЗОВ ОБРАБОТЧИКА.

// Срабатывает после завершения обработчика события.

protected void Application\_PostRequestHandlerExecute(object sender, EventArgs e)

{

Debug.WriteLine("PostRequestHandlerExecute");

}

// Вызывается тогда, когда информация сеанса сериализуется из коллекции Session, чтобы стать доступной в следующем запросе

protected void Application\_ReleaseRequestState(object sender, EventArgs e)

{

Debug.WriteLine("ReleaseRequestState");

}

// Вызывается перед добавлением информации в кэш выходных данных, если он был разрешен

protected void Application\_UpdateRequestCache(object sender, EventArgs e)

{

Debug.WriteLine("UpdateRequestCache");

}

// Последнее событие, которое происходит при обработке запроса.

protected void Application\_EndRequest(object sender, EventArgs e)

{

Debug.WriteLine("EndRequest");

}

}

\*\*\*

Page life style

public partial class Default : System.Web.UI.Page

{

// Событие используется для динамической установки MasterPage, смены темы.

protected void Page\_PreInit(object sender, EventArgs e)

{

Response.Write(PrepareMessage("Page\_PreInit"));

}

// Для чтения или инициализации свойств элементов управления.

protected void Page\_Init(object sender, EventArgs e)

{

Response.Write(PrepareMessage("Page\_Init"));

}

// Для действий, которые должны произвестись после всех инициализаций. До этого момента данные сохраненные во ViewState не сериализуються.

protected void Page\_InitComplete(object sender, EventArgs e)

{

Response.Write(PrepareMessage("Page\_InitComplete"));

}

// Для выполнения действий после загрузки ViewState и до события Load.

protected void Page\_PreLoad(object sender, EventArgs e)

{

Response.Write(PrepareMessage("Page\_PreLoad"));

}

// Используется для установки и инициализаций свойств открытия подключений к базам данных.

protected void Page\_Load(object sender, EventArgs e)

{

Response.Write(PrepareMessage("Page\_Load"));

}

// Обработчик нажатия по кнопке.

protected void Button\_Click(object sender, EventArgs e)

{

Response.Write(PrepareEmphasizedMessage("Button\_Click"));

}

// Для действий, которые должны происходить после выполнения обработчиков событий элементов управления.

protected void Page\_LoadComplete(object sender, EventArgs e)

{

Response.Write(PrepareMessage("Page\_LoadComplete"));

}

// Для внесения окончательных изменений перед рендерингом страницы.

protected void Page\_PreRender(object sender, EventArgs e)

{

Response.Write(PrepareMessage("Page\_PreRender"));

}

// Используется при разработке асинхронных страниц.

protected void Page\_PreRenderComplete(object sender, EventArgs e)

{

Response.Write(PrepareMessage("Page\_PreRenderComplete"));

}

// Сохранение ViewState. Последнее событие перед рендерингом.

protected void Page\_SaveStateComplete(object sender, EventArgs e)

{

Response.Write(PrepareMessage("Page\_SaveStateComplete"));

}

// Рендеринг страницы.

// Для освобождения ресурсов.

protected void Page\_Unload(object sender, EventArgs e)

{

//Response.Write(PrepareMessage("Page\_Unload"));

}

#region Help methods

private string PrepareMessage(string message)

{

return string.Format("<div style='background-color:lightgreen; border:1px solid black; padding:12px;margin:4px;'>{0}</div>", message);

}

private string PrepareEmphasizedMessage(string message)

{

return string.Format("<div style='background-color:purple; border:1px solid black; padding:12px;margin:4px;'>{0}</div>", message);

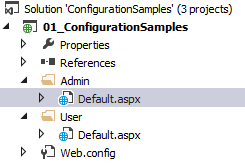
}

#endregion

}

\*\*\*

Конфигурация



<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<head runat="server">

<title>Web.Config location</title>

</head>

<body>

<form id="form1" runat="server">

<div>

Connection String from web.config: <asp:Label Font-Bold="true" Text="<%$ ConnectionStrings:AdminConnection %>" runat="server" />

</div>

</form>

</body>

</html>

WEB CONFIG

<configuration>

<location path="Admin">

<connectionStrings>

<add name="AdminConnection" connectionString="DataSource=AdminDb ...."/>

</connectionStrings>

</location>

<location path="User">

<connectionStrings>

<add name="UserConnection" connectionString="DataSource=UsersDb ...."/>

</connectionStrings>

</location>

<system.web>

<compilation debug="true" targetFramework="4.0" />

</system.web>

</configuration>

\*\*\*

Чтение и запись в web config

<head runat="server">

<title></title>

</head>

<body>

<form id="form1" runat="server">

<div>

<asp:Button Text="Прочитать данные из web.config" runat="server" OnClick="ReadButton\_Click" />

<br />

<asp:Label ID="Label1" runat="server" />

<br />

<br />

<asp:Button Text="Добавить запись в web.config" runat="server" OnClick="AddButton\_Click" />

</div>

</form>

</body>

</html>

public partial class Default : System.Web.UI.Page

{

// Чтение данных из файла web.config

protected void ReadButton\_Click(object sender, EventArgs e)

{

string path = @"system.web/customErrors";

CustomErrorsSection section = WebConfigurationManager.GetSection(path) as CustomErrorsSection;

foreach (CustomError item in section.Errors)

{

Label1.Text += item.StatusCode;

Label1.Text += " ";

Label1.Text += item.Redirect;

Label1.Text += "<br />";

}

}

// Запись данных в web.config

protected void AddButton\_Click(object sender, EventArgs e)

{

Configuration cfg = WebConfigurationManager.OpenWebConfiguration("/");

CustomErrorsSection section = cfg.GetSection("system.web/customErrors") as CustomErrorsSection;

CustomError tempError = new CustomError(501, "http://example.com/not\_implemented.aspx");

section.Errors.Add(tempError);

cfg.Save();

}

}

WEBCONFIG

<configuration>

<system.web>

<compilation debug="true" targetFramework="4.0" />

<customErrors mode="RemoteOnly">

<error statusCode="404" redirect="http://example.com/notfond.aspx" />

<error statusCode="500" redirect="http://example.com/error.aspx" />

<error statusCode="501" redirect="http://example.com/not\_implemented.aspx" />

</customErrors>

</system.web>

</configuration>