



PROGRAMACIÓN EN .NET C#

CLASE 10

ING. HAZAEL FERNANDO MOJICA GARCIA

- INTRODUCCIÓN A LA PROGRAMACION WEB CON ASP.NET
- USO DE BASE DE DATOS CON ASP.NET
- AJAX Y ASP.NET PARA ENRIQUECER LA EXPERIENCIA DEL USUARIO: **UPDATEPANEL**
- MANEJANDO AJAX DE MANERA ROBUSTA USANDO **WEBMETHODS**

INTRODUCCIÓN A WEBFORMS CON ASP.NET



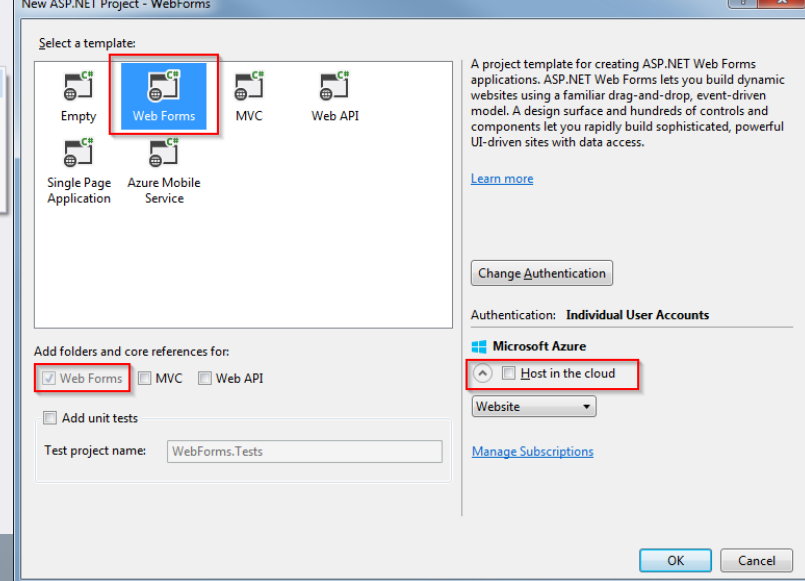
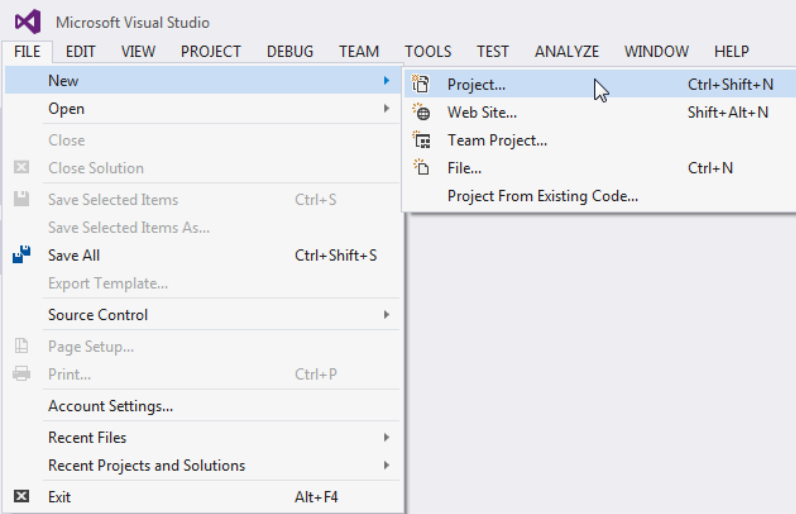
ASP.NET

Es un **framework** para aplicaciones WEB controladas desde servidor diseñada para producir página WEB dinámicas. La primera versión venía en .Net Framework 1.0 en el 2002 y actualmente es OpenSource.

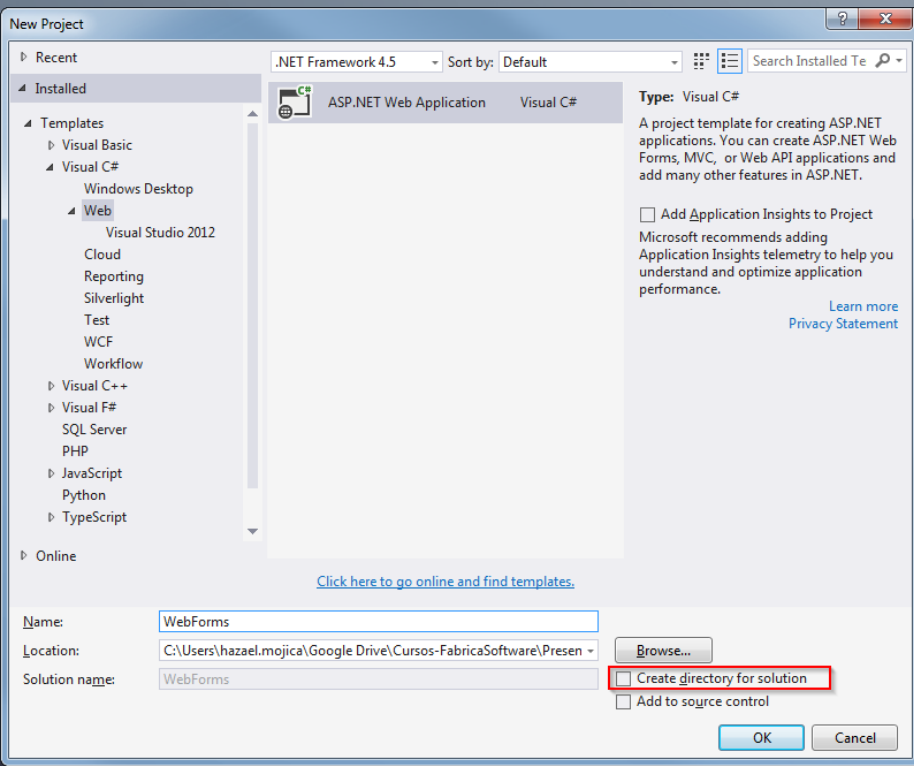
ASP.NET es un lenguaje de etiquetado muy parecido a HTML o XML pero la diferencia principal reside en que uno es capaz de controlar el comportamiento, métodos y propiedades de dichos controles desde el servidor, con mucho o poco conocimiento que se tenga de tecnologías que corren del lado del cliente (como javascript o html puro).

ASP.NET no solo es etiquetado, ya que posee una parte de codificación llamada coloquialmente “Code Behind” el cual es código **.Net (C# o VB)** el cual es usado para llevar a cabo el comportamiento deseado del sitio web, tal como consultar bases de datos, conectarse a Web Services o hacer cualquiera de las acciones posibles con el Framework de .Net pero corriendo en el servidor IIS.

Las páginas web desarrolladas en ASP.Net son oficialmente llamadas **Web Forms**, y existen muchas formas de poder actualizar con **AJAX** el contenido de una página WEB incluyendo la clase **UpdatePanel**, sin embargo para este curso veremos también una forma un tanto más complicada pero que permite el manejo del comportamiento y conexiones a servidor de una manera más robusta y controlada: **WebMethods**.

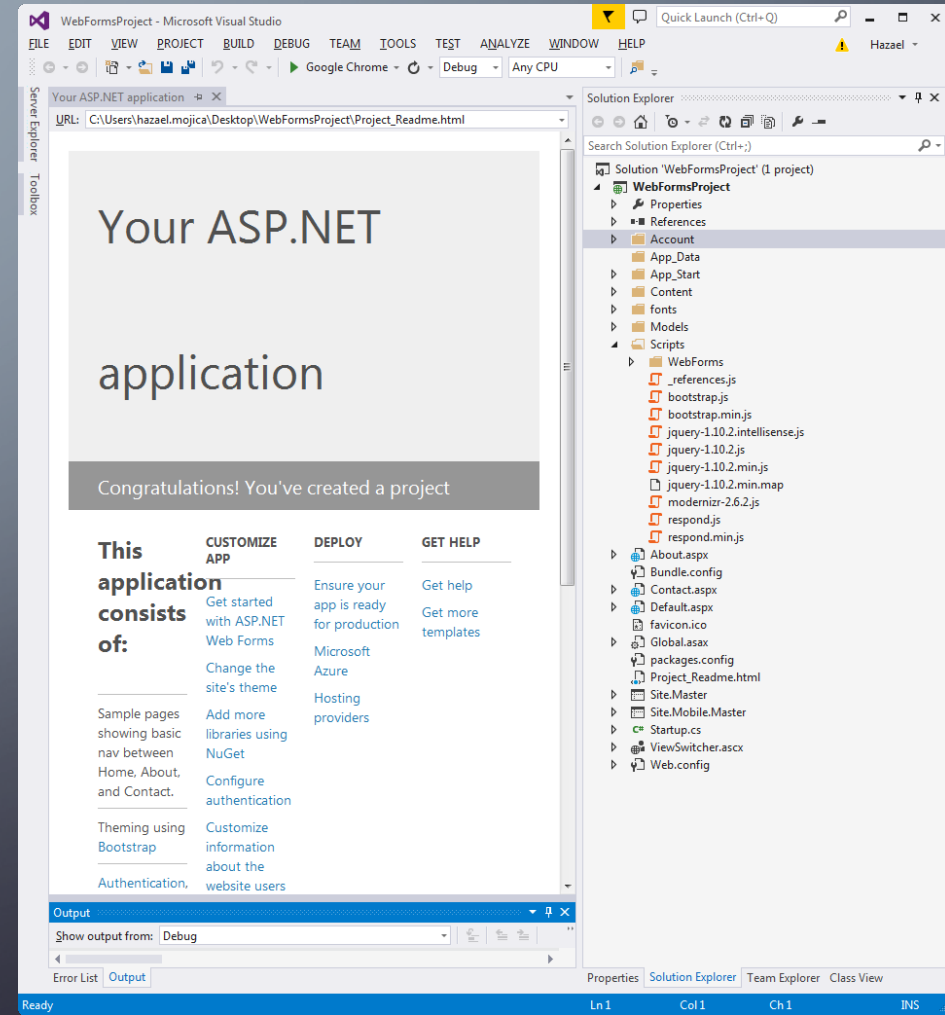


Crearemos un nuevo proyecto de WebForms usando el template que proporciona VisualStudio.



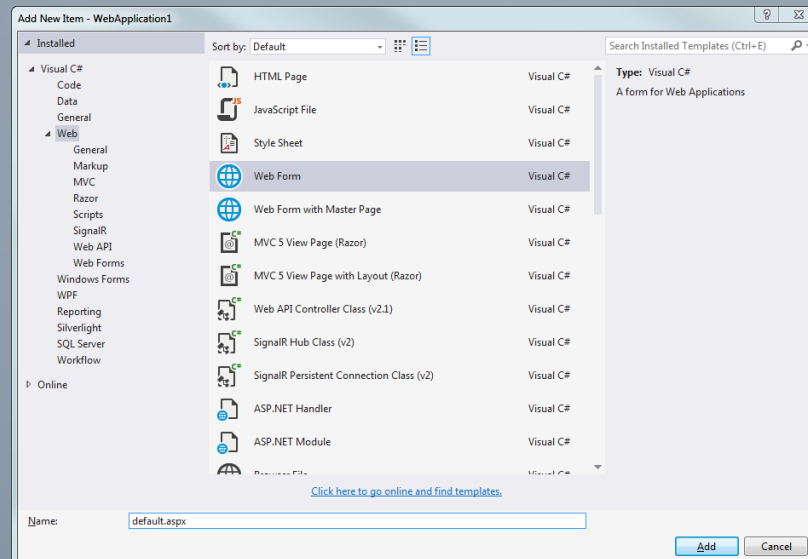
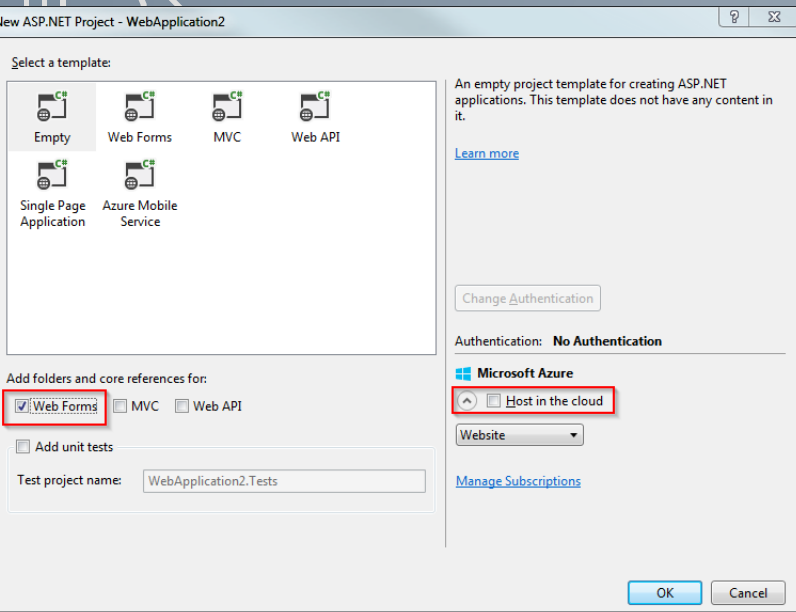
Una vez creada la solución, F5 para correr...

10.1.WebForms



Comenzando a crear una solución WebForms desde 0.

Ejemplo2



default.aspx.cs - CodeBehind

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

namespace WebApplication1
{
    public partial class _default : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {

        }

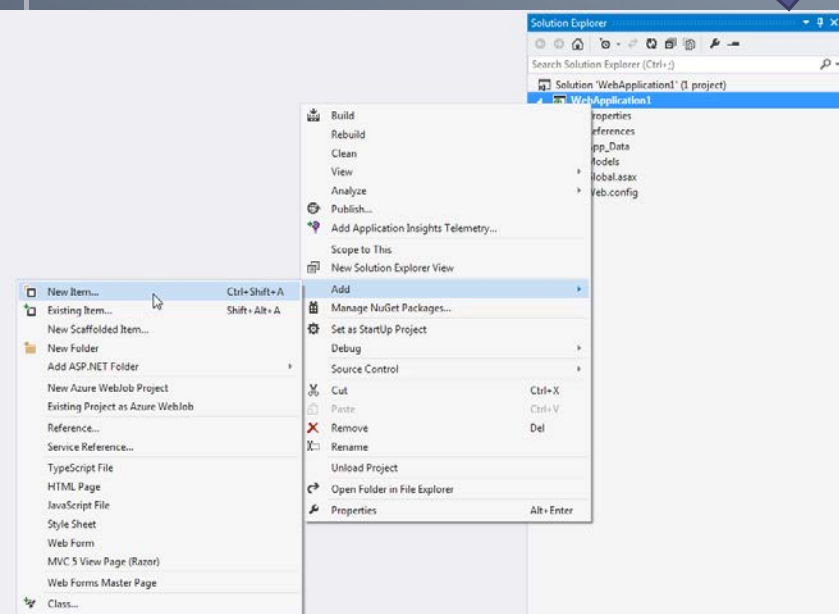
        protected void button_Hello_Click(object sender, EventArgs e)
        {
            this.label_Hello.Text = "Hola FIME";
        }
    }
}
```

default.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="default.aspx.cs"
Inherits="WebApplication1._default" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
        <div>
            <asp:Button ID="button_Hello" runat="server" OnClick="button_Hello_Click" Text="Hola FIME. CLICK ME" />
            <br />
            <asp:Label ID="label_Hello" runat="server"></asp:Label>
        </div>
    </form>
</body>
</html>
```



default.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="default.aspx.cs" Inherits="WebApplication1._default" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
        <div>
            <asp:Button ID="button_Hello" runat="server" OnClick="button_Hello_Click" Text="Add Item" />
            <br />
            <asp:TextBox ID="textBox_Hello" runat="server"></asp:TextBox>
            <br />
            <asp:DropDownList ID="dropDownList_Hello" runat="server"></asp:DropDownList>
            <br />
            <asp:Label ID="label_Hello" runat="server"></asp:Label>
            <br /><br /><br /><br />
            <asp:Button ID="buttons_Redirect" runat="server" Text="Redirecciona a otra pagina"
OnClick="buttons_Redirect_Click" />
        </div>
    </form>
</body>
</html>
```

default.aspx.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

namespace WebApplication1
{
    public partial class _default : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {

        }

        protected void button_Hello_Click(object sender, EventArgs e)
        {
            //Guarda todos los elementos en el ComboBox/DropDownList
            string text = this.textBox_Hello.Text;
            this.dropDownList_Hello.Items.Add(text);
            this.label_Hello.Text = text;
            this.textBox_Hello.Text = "";
        }

        protected void buttons_Redirect_Click(object sender, EventArgs e)
        {
            //Redirecciona a otra pagina mandando datos
            //Obtiene los elementos a guardar
            ListItemCollection items = this.dropDownList_Hello.Items;
            Session.Add("DropDown_Items", items);
            this.Response.Redirect("OtherPage.aspx");
        }
    }
}
```

OtherPage.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="OtherPage.aspx.cs" Inherits="WebApplication1.OtherPage" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
        <div>
            <asp:Table ID="Table1" runat="server" BorderColor="Black" BorderStyle="Solid" BorderWidth="1px"
GridLines="Both"></asp:Table>
        </div>
    </form>
</body>
</html>
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

namespace WebApplication1
{
    public partial class OtherPage : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            if(!IsPostBack)
            {
                //Si la llamada no es un PostBack
                try
                {
                    ListItemCollection items = (ListItemCollection)this.Session["DropDown_Items"];
                    if (items != null)
                        this.addElements(items);
                    else
                        Response.Redirect("default.aspx");
                }
                catch
                {
                    Response.Redirect("default.aspx");
                }
            }
        }

        private void addElements(ListItemCollection items)
        {
            for (int i = 0; i < items.Count; i++)
            {
                string text = items[i].Value;
                TableRow row = new TableRow();
                TableCell cellNum = new TableCell();
                TableCell cellVal = new TableCell();
                cellNum.Text = i.ToString();
                cellVal.Text = text;
                row.Cells.Add(cellNum);
                row.Cells.Add(cellVal);
                this.Table1.Rows.Add(row);
            }
        }
    }
}
```

En **default.aspx** agregamos como item al DropDownList el texto en la TextBox.

Al hacer click en el botón “Redirecciona a otra pagina” se guardara en sesión el objeto Items del DropDownList y se mandará llamar el otro WebForm **OtherPage.aspx**.

En OtherPage.aspx intentaremos tomar dichos items y llenar una tabla HTML con ellos.

EJERCICIO 1

Usar el código de **10.2.WebForms** para crear un nuevo proyecto que haga lo siguiente:

- El sistema emulará un **Login de Usuario**
- La página Web contendrá **dos textbox**, una para usuario y otra para el password
- También contendrá un **botón y label WebForms** que verificará en **ServerSide** que el usuario y contraseña sean correctas mandando un mensaje por medio de la label.
- El usuario y contraseña estarán “hardcodeados” en el código C#

CONEXIÓN A BASE DE DATOS CON ADO.NET

ADO.NET es un conjunto de clases que exponen servicios de acceso a datos para programadores de .NET Framework.

System.Data.SqlClient proporciona acceso a versiones de SQL Server, que encapsula los protocolos específicos de base de datos. La funcionalidad del proveedor de datos se ha diseñado para que sea similar a la de los proveedores de datos .NET Framework para OLE DB, ODBC y Oracle. System.Data.SqlClient incluye un analizador de flujo TDS para comunicarse directamente con SQL Server.

[https://msdn.microsoft.com/es-es/library/e80y5yhx\(v=vs.110\).aspx](https://msdn.microsoft.com/es-es/library/e80y5yhx(v=vs.110).aspx)

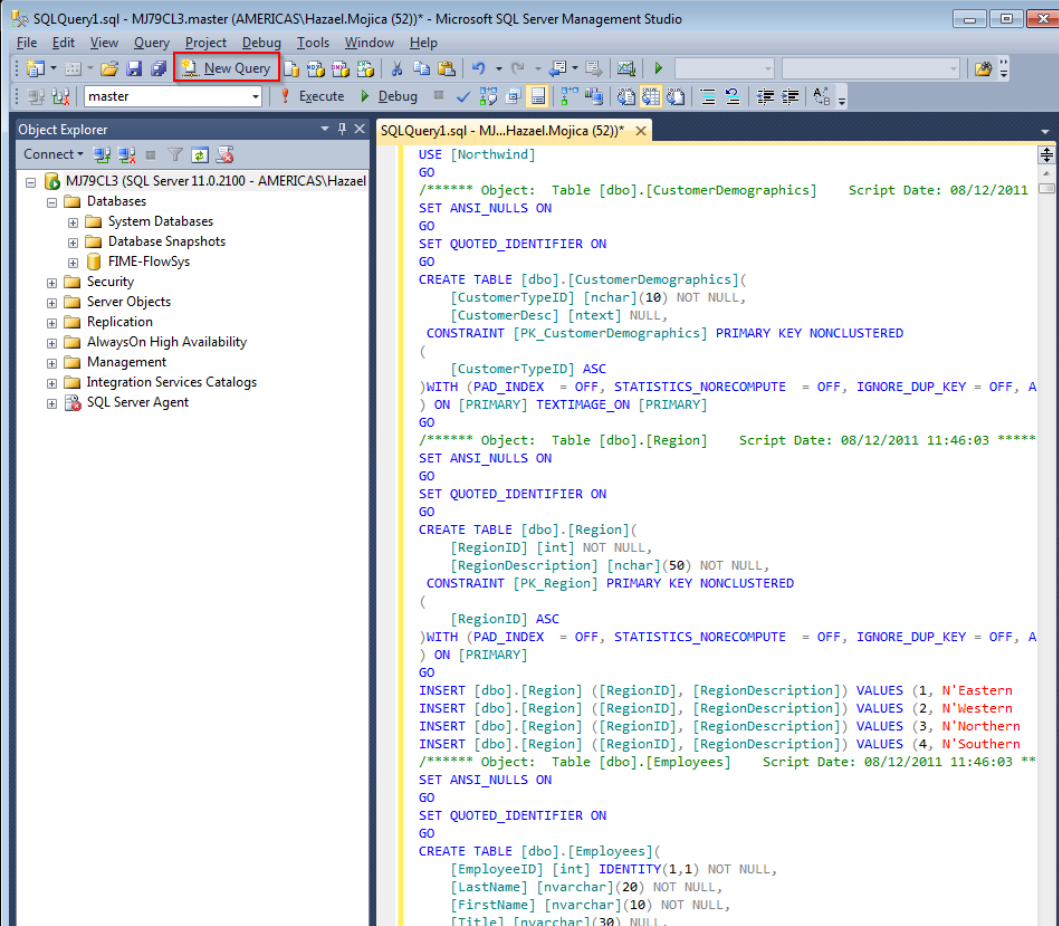


ADO.NET

ENVIRONMENT SETUP

Crear la base de datos en nuestro servidor SQL Server.

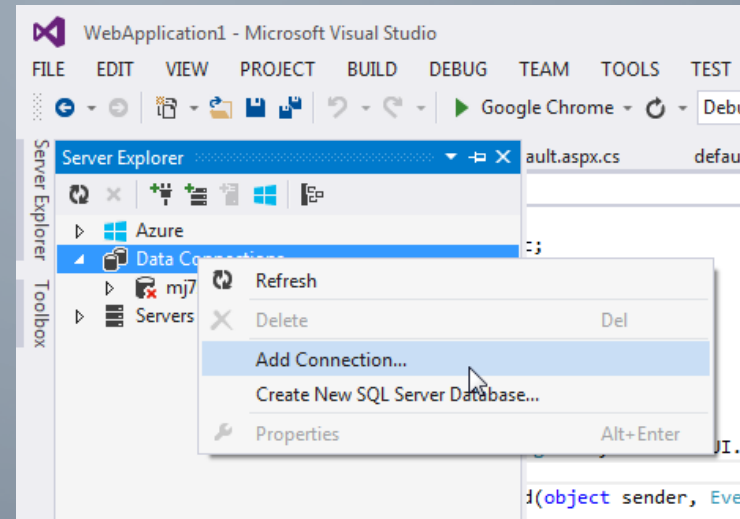
En **SQL Management Studio** crear una nueva base de datos llamada **Northwind** (click derecho en Databases y New Database).
Click en **New Query**, pegar el contenido de **northwind.sql** y Ejecutar.



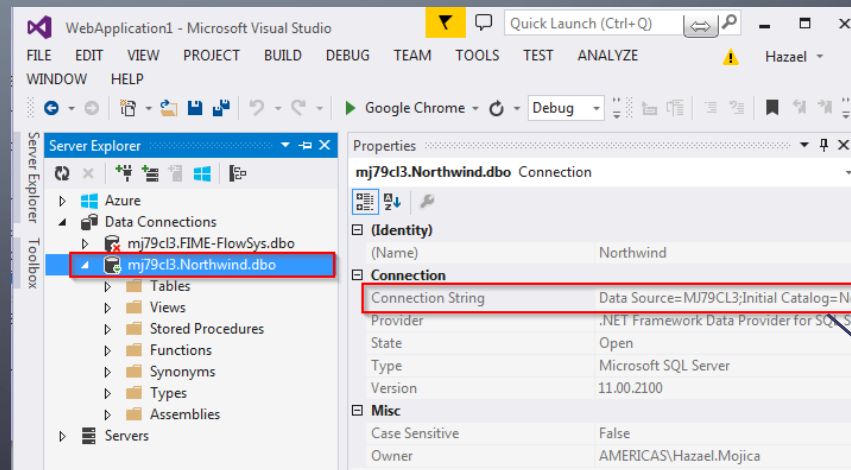
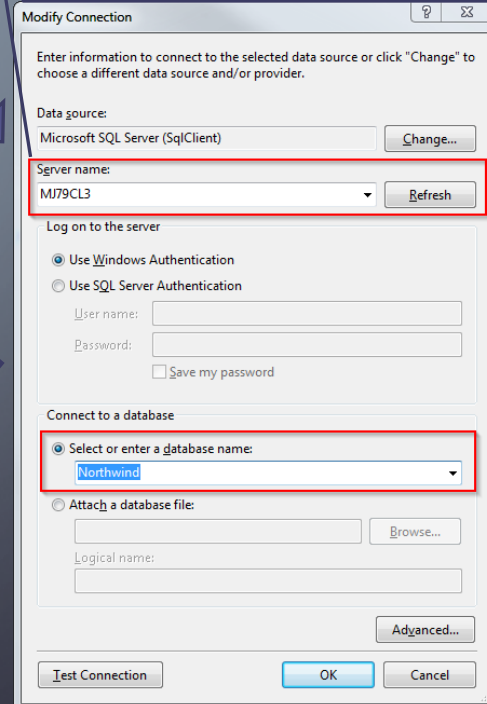
Conectar la Connection String.

En **Visual Studio** abrir el **Server Explorer** y en **Data Connections** hacer click derecho y **Add Connection**.
Llena los datos y crea la conexión.

Abre la barra de propiedades y anota la Connection String en algún lugar que recuerdes, por ejemplo, para mi WorkSpace mi connection string es:
"Data Source=MJ79CL3;Initial Catalog=Northwind;Integrated Security=True"



Recuerda que el Servidor es tu propia PC (para este caso)



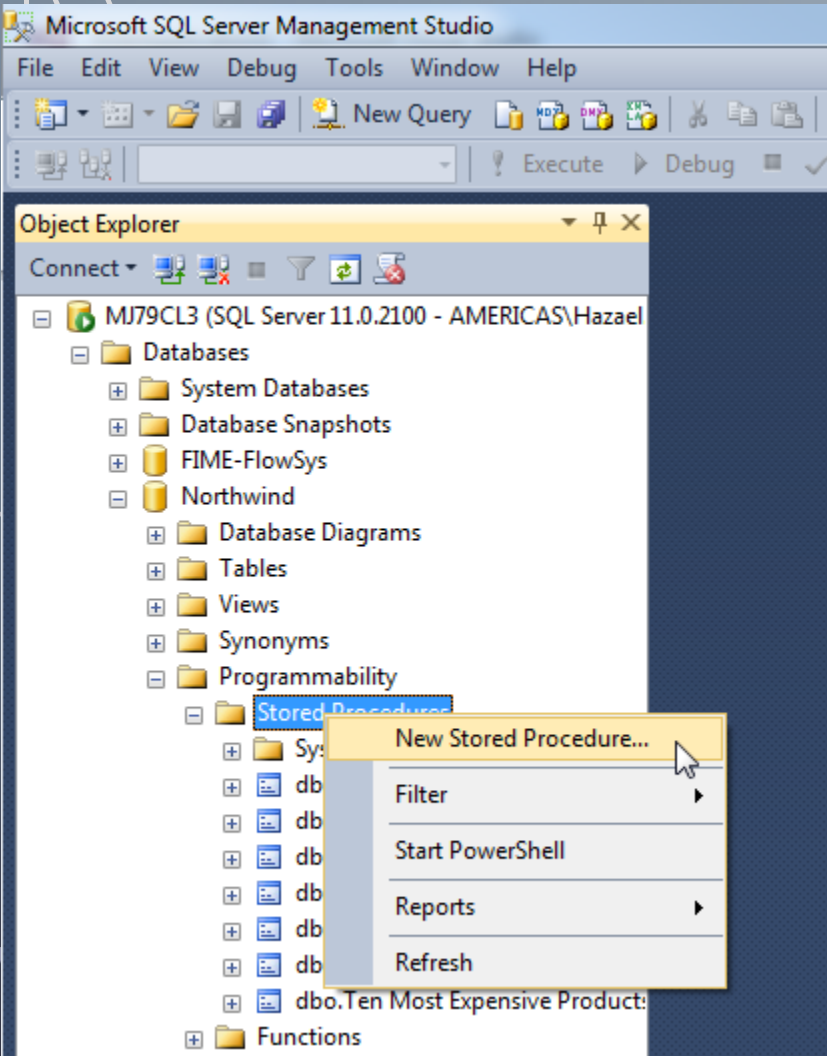
Anota la Connection String

Crear un **Stored Procedure** que sea capaz de consultar la tabla de Regiones y colocar el resultado de dicha consulta en una **GridView**.

1.- Vamos a crear el Stored Procedure en la base de datos:

Ejemplo4

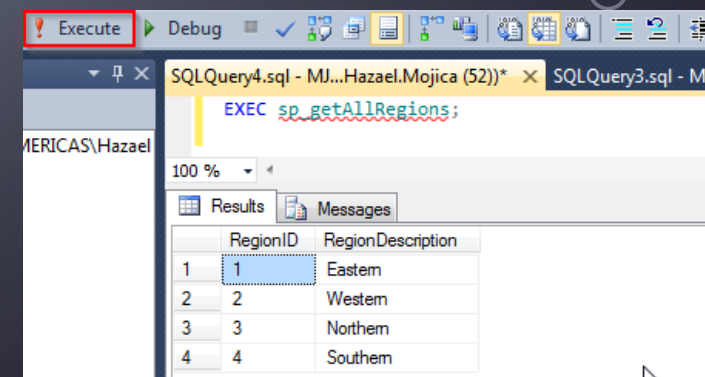
10.4.WebForms



En la ventana de query que nos saldrá modificamos el código para que haga un **SELECT**:

```
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- =====
-- Author:<Hazael Mojica>
-- Create date: <1/10/2015>
-- Description:<Stored Procedure que obtiene todas las regiones
de NorthWind>
-- =====
CREATE PROCEDURE sp_getAllRegions
AS
BEGIN
SELECT * FROM Region
END
GO
```

Puedes probar el **StoredProcedure** directo en **M SQL Server Management Studio** abriendo un **nuevo query** y ejecutando:
EXEC sp_getAllRegions;



Creamos un nuevo **StoredProcedure** llamado **sp_getAllRegions**.

Creamos el código en **C#** por medio de **ADO.NET** que nos permitirá ejecutar dicho **Stored Procedure** y regresar los datos en un **DataSet**.

SQLManager.cs

```
using System;
using System.Data;
using System.Web;
using System.Web.Configuration;
using System.Web.UI;
using System.Data.SqlClient;
using System.Diagnostics;
using System.Collections.Generic;

public static class SQLManager
{
    /// <summary>
    /// Metodo que ejecuta un Stored Procedure en base de datos y que posee un manejador Excepciones
    /// </summary>
    /// <param name="storedProcedure">Nombre del Stored Procedure</param>
    /// <param name="parametros">Parametros el Stored Procedure</param>
    /// <returns></returns>
    public static DataSet EjecutarQueryConSP(string storedProcedure, params SqlParameter[] parametros)
    {
        DataSet dataSet = null;
        SqlCommand command;
        SqlDataAdapter sqlDataAdapter;
        string connectionString = "Data Source=MJ79CL3;Initial Catalog=Northwind;Integrated Security=True";
        //obtenemos la string de conexion para el tipo de servidor especificado
        SqlConnection cnn = new SqlConnection(connectionString); //creamos una conexion a la base de datos
        //intentamos abrir la conexion y obtener los datos de la consulta
        try
        {
            cnn.Open();
            command = new SqlCommand(storedProcedure, cnn);
            command.CommandType = CommandType.StoredProcedure;

            for (int i = 0; i < parametros.Length; i++)
            {
                command.Parameters.Add(parametros[i]);
            }

            sqlDataAdapter = new SqlDataAdapter(command);
            dataSet = new DataSet();
            sqlDataAdapter.Fill(dataSet);
            cnn.Close();
        }
        catch (Exception ex)
        {
            Debug.WriteLine(ex);
            Debug.WriteLine(ex.StackTrace);
            dataSet = null;
        }
        return dataSet;
    }
}
```

default.aspx.cs

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

namespace WebApplication1
{
    public partial class _default : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {

        }

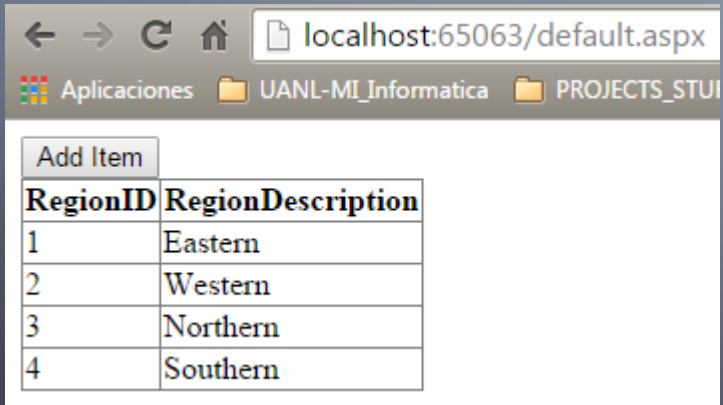
        protected void button_Hello_Click(object sender, EventArgs e)
        {
            //Ejecuta el Stored Procedure
            DataSet dataSet = SQLManager.EjecutarQueryConSP("sp_getAllRegions");
            if (dataSet != null)
            {
                this.GridView_Hello.DataSource = dataSet;
                this.GridView_Hello.DataBind();
            }
            else
            {
                //El DataSet es nulo, ocurrio algun error
                Type cstype = this.GetType(); //Obtiene la pagina
                string key = "alertScript"; //Da un ID a tu script
                string script = "<script type='text/javascript'>";
                script += "alert('Error consultando la base de datos');";
                script += "</script>";
                ClientScript.RegisterStartupScript(cstype, key, script);
            }
        }
    }
}
```

default.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="default.aspx.cs"
Inherits="WebApplication1._default" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
        <div>
            <asp:Button ID="button_Hello" runat="server" OnClick="button_Hello_Click" Text="Add Item" />
            <br />
            <asp:GridView ID="GridView_Hello" runat="server"></asp:GridView>
        </div>
    </form>
</body>
</html>
```



El método ***SQLManager.EjecutarQueryConSP(...)*** nos permitirá de una manera rápida y ordenada de ejecutar Stored Procedures usando el ***SQLDataAdapter***.

Una vez tenemos el ***DataSet*** resultado de la consulta, podemos ligarla con el ***GridView*** usando la propiedad ***DataSource*** y después procesar los datos haciendo un ***DataBind()***.

Crear un sistema WEB capaz de consultar la base de datos NorthWind y que regrese el CustomerID, Company Name, Contact Name y Address de un Customer identificado por medio de su ID.

Dicho sistema WEB debe ser completamente responsivo y tener una interfaz web amigable al usuario.

Crear el Stored Procedure sp_getCustomerByID.

```
USE [NorthWind]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- =====
-- Author:<Hazael Mojica>
-- Create date: <01/10/2015>
-- Description:<Get a Customer by its ID>
-- =====
CREATE PROCEDURE [dbo].[sp_getCustomerByID]
@customerID AS varchar(50)
AS
BEGIN
-- SET NOCOUNT ON added to prevent extra result sets from
-- interfering with SELECT statements.
SET NOCOUNT ON;

    SELECT CustomerID,CompanyName,ContactName,Address
    FROM Customers
    WHERE CustomerID=@customerID;
END
GO
```

CustomerID	CompanyName	ContactName	Address
BONAP	Bon app'	Laurence Lebihan	12, rue des Bouchers

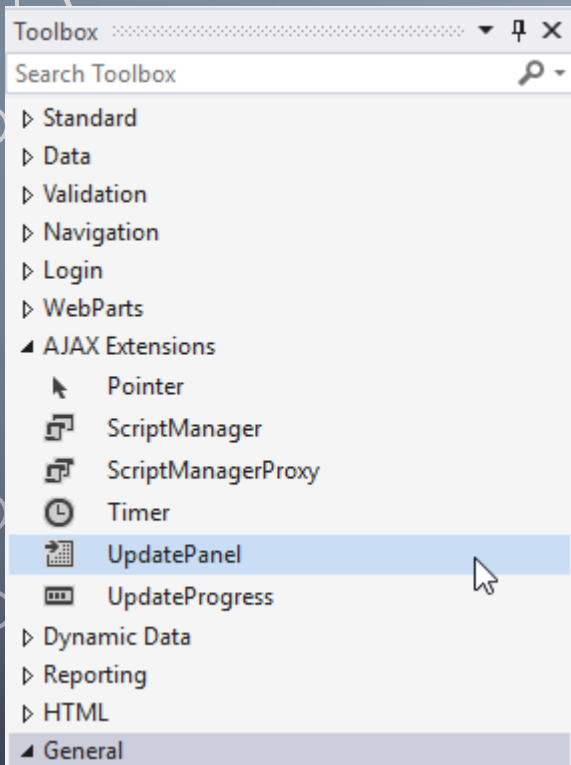
Después de crear el Stored Procedure correr **10.5.WebForms** y analizar el código.

Ahora se necesita proporcionar un parámetro al Stored Procedure, dicho parámetro es el ID del Customer a buscar.

La página WEB ha sido “bootstrapeada” para enriquecer la experiencia del usuario.

USO DE **UPDATEPANEL** PARA MEJORAR LA EXPERIENCIA DEL USUARIO CON AJAX.

(FACILIDAD DE PROGRAMACIÓN PERO BAJO RENDIMIENTO).



Los controles de **ASP.NET UpdatePanel** permiten generar aplicaciones web enriquecidas y centradas en el cliente. Los controles UpdatePanel permiten actualizar las partes seleccionadas de una página en lugar de actualizar toda la página con una devolución de datos. Esto se conoce como actualización parcial de la página. Una página web ASP.NET que contiene un control **ScriptManager** y uno o varios controles UpdatePanel puede participar automáticamente en actualizaciones parciales de la página, sin un script de cliente personalizado.

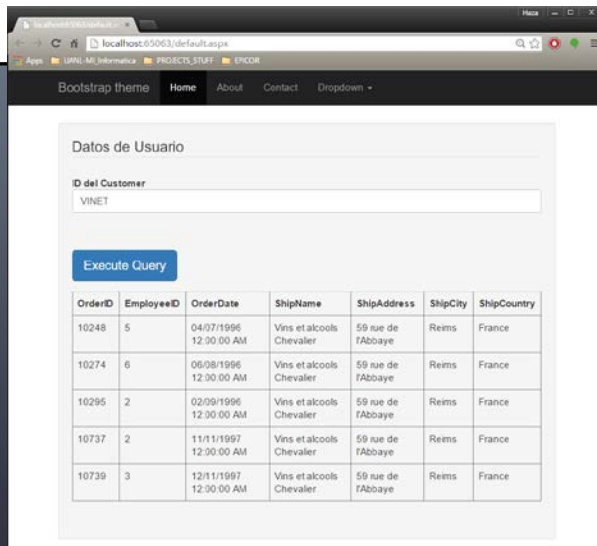
UpdatePanel es un control de servidor que ayuda a desarrollar páginas web con comportamientos de cliente complejos que hacen que una página web parezca más interactiva al usuario final. Normalmente, escribir código que se coordine entre el servidor y el cliente para actualizar sólo las partes especificadas de una página web requiere un conocimiento detallado de **ECMAScript (JavaScript)**. Sin embargo, con el control UpdatePanel puede hacer que una página web participe de las actualizaciones parciales de la página sin escribir ningún script de cliente.

Consulta las ordenes hechas por un mismo cliente usando AJAX, el usuario no desea experimentar horribles tiempos de refresco de página.

Crear el Stored Procedure `sp_getOrderByCustomerID`.

```
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- =====
-- Author:<Hazael Mojica>
-- Create date: <01/10/2015>
-- Description:<Get all the orders of a Customer>
-- =====
CREATE PROCEDURE sp_getOrderByCustomerID
-- Add the parameters for the stored procedure here
@customerID AS varchar(50)
AS
BEGIN
-- SET NOCOUNT ON added to prevent extra result sets from
-- interfering with SELECT statements.
SET NOCOUNT ON;

-- Insert statements for procedure here
SELECT OrderID, EmployeeID, OrderDate, ShipName,
ShipAddress, ShipCity, ShipCountry
FROM Orders
WHERE CustomerID = @customerID
END
GO
```



The screenshot shows a web browser window with a URL of localhost:5003/default.aspx. The page has a Bootstrap theme and a navigation bar with links for Home, About, Contact, and a Dropdown menu. The main content area is titled 'Datos de Usuario' and contains a form with a label 'ID del Customer' and a text input field containing 'VINET'. Below the form is a blue button labeled 'Execute Query'. Underneath the button is a table displaying order data for the customer VINET.

OrderID	EmployeeID	OrderDate	ShipName	ShipAddress	ShipCity	ShipCountry
10248	5	04/07/1996 12:00:00 AM	Vins et alcools Chevalier	59 rue de l'Abbaye	Reims	France
10274	6	06/08/1996 12:00:00 AM	Vins et alcools Chevalier	59 rue de l'Abbaye	Reims	France
10295	2	02/09/1996 12:00:00 AM	Vins et alcools Chevalier	59 rue de l'Abbaye	Reims	France
10737	2	11/11/1997 12:00:00 AM	Vins et alcools Chevalier	59 rue de l'Abbaye	Reims	France
10739	3	12/11/1997 12:00:00 AM	Vins et alcools Chevalier	59 rue de l'Abbaye	Reims	France

```
<asp:ScriptManager ID="ScriptManager1" runat="server" />
<asp:UpdatePanel ID="UpdatePanel1" runat="server">
    <ContentTemplate>
        <div class="form-group">
            <label for="inputEmail" class="col-lg-2 control-label">ID del Customer</label>
            <div class="col-lg-10">
                <asp:TextBox ID="TextBox1" CssClass="form-control"
runat="server"></asp:TextBox>
            </div>
        </div>
        <br /><br />
        <div class="form-group">
            <div class="col-lg-10 col-lg-offset-2">
                <asp:Button ID="button_Hello" runat="server" OnClick="button_Hello_Click"
CssClass="btn btn-lg btn-primary" Text="Execute Query" />
            </div>
        </div>
        <div class="form-group">
            <div class="col-lg-10 col-lg-offset-2">
                <asp:GridView ID="GridView_Hello" CssClass="table"
runat="server"></asp:GridView>
            </div>
        </div>
    </ContentTemplate>
</asp:UpdatePanel>
```

Por medio del control **UpdatePanel** y del **ScriptManager** uno es capaz de poder hacer refrescar el DOM contenido en el primero sin necesidad de refrescar toda la página haciendo un **Postback**. UpdatePanel usa AJAX aunque no nos damos cuenta de como funciona.

Ejemplo6

10.6.WebForms

EJERCICIO 2

Usar el código de **10.6.WebForms** para crear un nuevo proyecto que haga lo siguiente:

- El sistema emulará un **Login de Usuario**
- Crear la tabla **Users**
 - Columnas:
 - UserId: int – auto incremental
 - Name varchar(50)
 - Pass varchar(20)
 - TimeStamp: DateTime
 - Insertar 3 usuarios con datos cualquiera
- Crear el StoredProcedure **sp_getUserByUserID**
 - Este Stored Procedure tomará de entrada: UserID y regresará: UserID, Name y Password del usuario encontrado o un dataset nulo en caso contrario.
- La página Web contendrá dos textbox, una para usuario y otra para el password
- También contendrá un **botón** de login el cual al ser presionado consultará la base de datos en busca del usuario.
 - Mandará un **alert() javascript** diciendo si dicho usuario fue encontrado o no y el nombre.

AJAX Y ASP.NET

(UNA SOLUCIÓN INTELIGENTE)

AJAX con JQuery

```
$.ajax({
  type: "GET",
  url: "responsePage.aspx",
  dataType: "html",
  data:
  {
    var1: "var1val",
    var2: "var3val",
    var3: "var2val",
  },
  success:
    function (data)
    {

    },
  error:
    function ()
    {

    }
});
```

Por medio de **AJAX** podemos ejecutar acciones en el servidor en background tales como llamar a una base de datos, consumir un Webservice, ejecutar un proceso en un Application Server etc.

Todo esto es posible por medio de sencillas llamadas **GET, POST** ejecutadas en el fondo.

Recuerda que una llamada **GET/POST** leerá como texto plano lo que el servidor regrese, si este es un archivo txt, html, css, js lo más seguro es que su contenido nunca cambia sin embargo el poder esto es observable cuando llamamos a una página dinámica como .aspx o php la cual puede entregar diferente contenido y ejecutar acciones en el servidor.

Crear un login de usuario por medio de AJAX (sin uso de **UpdatePanel**), se requiere que los mensajes que se vayan a mostrar al usuario se vean decentes, por tanto se recomienda usar [bootbox](#).

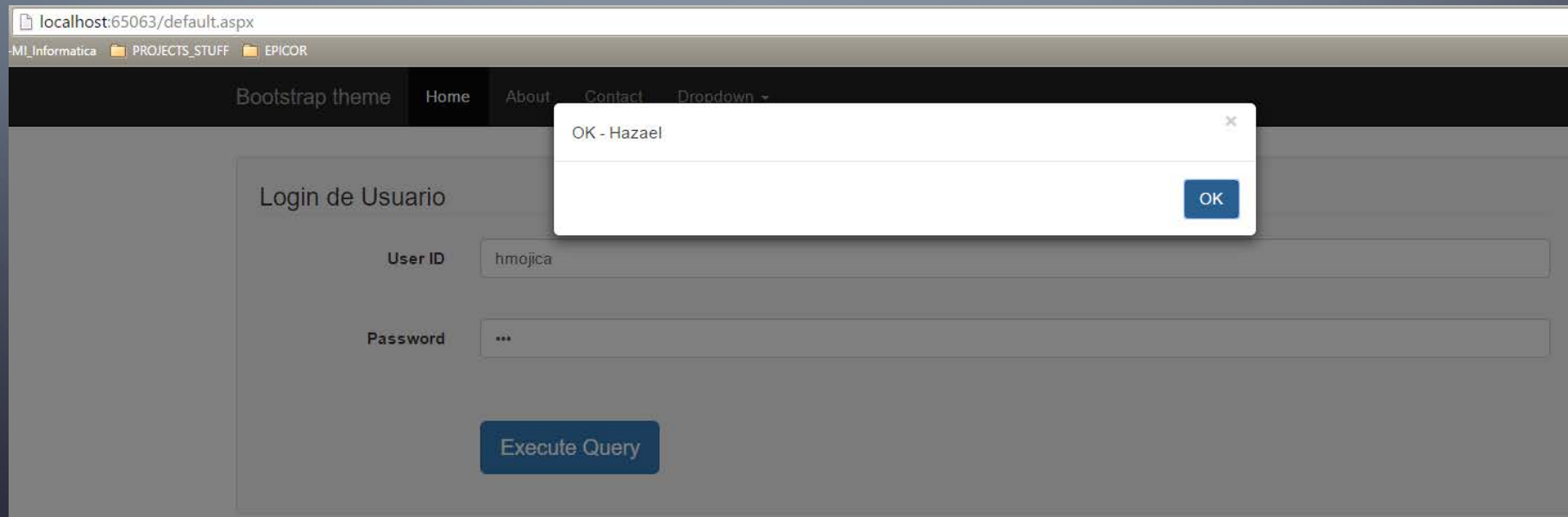
Stored Procedure *sp_getUserByID*

```
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- =====
-- Author:<Hazael Mojica>
-- Create date: <03/10/2015>
-- Description:<Get a User by its ID>
-- =====
CREATE PROCEDURE sp_getUserByID
@userId as varchar(20)
AS
BEGIN
-- SET NOCOUNT ON added to prevent extra result sets from
-- interfering with SELECT statements.
SET NOCOUNT ON;

SELECT UserId,Name,Pass FROM Users WHERE UserId = @userId
END
GO
```

default.aspx

```
<legend>Login de Usuario</legend>
<div class="form-group">
  <label for="text_userID" class="col-lg-2 control-label">User ID</label>
  <div class="col-lg-10">
    <input type="text" class="form-control" id="text_userID" />
  </div>
</div>
<br />
<div class="form-group">
  <label for="text_userPass" class="col-lg-2 control-label">Password</label>
  <div class="col-lg-10">
    <input type="password" class="form-control" id="text_userPass" />
  </div>
</div>
<br /><br />
<div class="form-group">
  <div class="col-lg-10 col-lg-offset-2">
    <input type="button" class="btn btn-lg btn-primary" id="buttonExec"
onclick="doLogin();" value="Execute Query" />
  </div>
</div>
```



scripts.js

```
function doLogin()
{
    var useridVal = $("#text_userID").val();
    var passVal = $("#text_userPass").val();

    $.ajax({
        type: "GET",
        url: "doLogin.aspx",
        dataType: "html",
        data:
        {
            userid: useridVal,
            pass: passVal
        },
        success:
        function (data)
        {
            if(data == "ERROR")
            {
                var msg = "Error consultando la base de datos";
                displayMsg(msg);
            }
            else if(data != "")
            {
                //OK, show the user data
                displayMsg(data);
            }
            else
            {
                //No data, error
                var msg = "Error desconocido";
                displayMsg(msg);
            }
        },
        error:
        function ()
        {
            var msg = "Error en la llamada al servidor";
            msg += "estas seguro que tienes internet?";
            displayMsg(msg);
        }
    });
}

function displayMsg(msg)
{
    //Usamos js/Bootbox para desplegar mensajes hermosos
    bootbox.alert(msg);
}
```

doLogin.aspx.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Data;
using System.Data.SqlClient;

namespace WebApplication1
{
    public partial class doLogin : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            if (!IsPostBack)
            {
                string userID = Request.QueryString["userid"];
                string pass = Request.QueryString["pass"];
                if (userID != "" && pass != "")
                {
                    SqlParameter parameter1 = new SqlParameter("@userID", userID);
                    DataSet dataSet = SQLManager.EjecutarQueryConSP("sp_getUserByID", parameter1);
                    if (dataSet != null)
                    {
                        try
                        {
                            string msg = "";
                            //Obtenemos el pass y nombre
                            //Tables[index] - index 0 puesto que se regresa una tabla
                            //Rows[0][1] - Primera fila segunda columna
                            //Rows[0][2] - Primera fila tercera columna
                            string bdName = dataSet.Tables[0].Rows[0][1].ToString();
                            string bdPass = dataSet.Tables[0].Rows[0][2].ToString();

                            if (pass.Trim() == bdPass.Trim())
                                msg = "OK - " + bdName;
                            else
                                msg = "El usuario y password no coinciden";
                            Response.Write(msg);
                        }
                        catch (Exception ex){Response.Write("ERROR");}
                    }
                    else
                    {
                        //El DataSet es nulo, no hay datos regresados
                        Response.Write("El usuario no existe en nuestra base de datos");
                    }
                }
            }
        }
    }
}
```

WEBMETHODS DE .NET

(PORQUE MICROSOFT TE QUIERE HACER LA VIDA MAS FÁCIL)

En el ejemplo pasado se observó como poder llamar a una página aspx por medio de AJAX, esto sin duda alguna incrementa la experiencia del usuario ya que no existen recargas de página cada vez que se hacen click en algún botón, pero nos deja con el trabajo de crear una página aspx que procese cada una de las diferentes peticiones que se deseen hacer al servidor o crear nuestro propio manejador de peticiones GET/POST contenido en un solo aspx (esto supongo usaría un enorme switch case).

Microsoft para afrontar esto creo algo llamado WebMethods.

Los **WebMethods** son métodos C#/VB que pueden ser llamados desde AJAX directamente es decir, ahora podemos tener una página aspx dedicada a recibir peticiones desde el cliente y especificar qué método deseamos llamar.

Crear un login de usuario por medio de AJAX (sin uso de **UpdatePanel**), se requiere que los mensajes que se vayan a mostrar al usuario se vean decentes, por tanto se recomienda usar [bootbox](#). Se recomienda ampliamente el uso de **WebMethods**.

Ejemplo8

10.8.WebForms

default.js

```
function doLogin()
{
    var useridVal = $("#text_userID").val();
    var passVal = $("#text_userPass").val();
    var dataJSON = { userid: useridVal, pass: passVal };
    var dataJSONStr = JSON.stringify(dataJSON);

    $.ajax({
        type: "POST",
        url: "default.aspx/doLogin", //La url ahora es ResponsePage.aspx/MethodName
        dataType: "json", //Los WebMethods regresan un JSON por default
        contentType: "application/json; charset=utf-8",
        data: dataJSONStr,
        success:
            function (data)
            {
                //En un WebMethod se regresa la info en el attrb. "d" del json
                var response = data.d;
                if (response == "ERROR") {
                    var msg = "Error consultando la base de datos";
                    displayMsg(msg);
                }
                else if (response != "") { //OK, show the user data
                    displayMsg(response);
                }
                else { //No data, error
                    var msg = "Error desconocido";
                    displayMsg(msg);
                }
            },
        error:
            function ()
            {
                var msg = "Error en la llamada al servidor";
                msg += " estas seguro que tienes internet?";
                displayMsg(msg);
            }
    });
}

function displayMsg(msg) { //Usamos js/Bootbox para desplegar mensajes hermosos
    bootbox.alert(msg);
}
```

default.aspx.cs

```
[WebMethod]
public static string doLogin(string userid, string pass)
{
    string result = "";

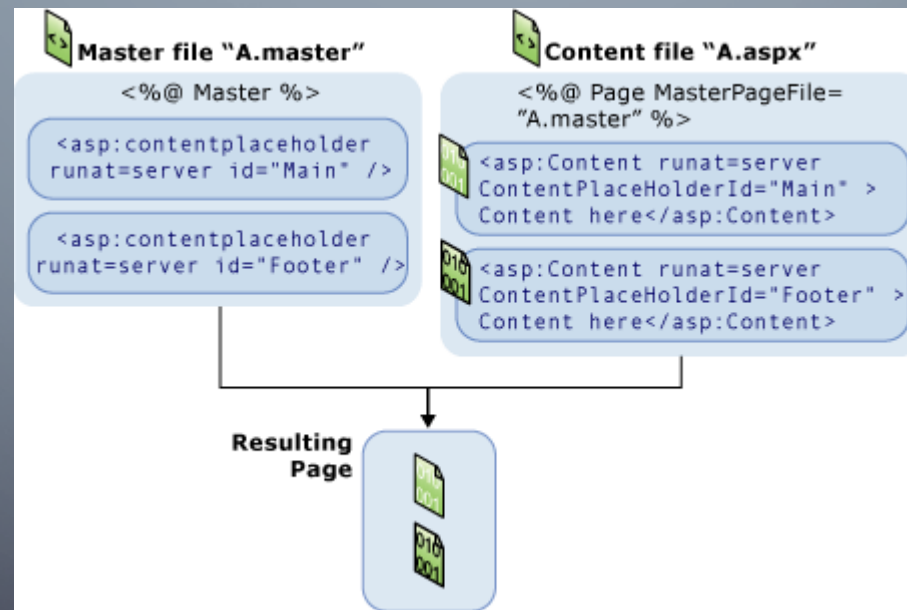
    if (userid != "" && pass != "")
    {
        SqlParameter parameter1 = new SqlParameter("@userId", userid);
        DataSet dataSet = SQLManager.EjecutarQueryConSP("sp_getUserByID", parameter1);
        if (dataSet != null)
        {
            try
            {
                string msg = "";
                //Obtenemos el pass y nombre
                //Tables[index] - index 0 puesto que se regresa una tabla
                //Rows[0][1] - Primera fila segunda columna
                //Rows[0][2] - Primera fila tercera columna
                string bdName = dataSet.Tables[0].Rows[0][1].ToString();
                string bdPass = dataSet.Tables[0].Rows[0][2].ToString();

                if (pass.Trim() == bdPass.Trim())
                    msg = "OK - " + bdName;
                else
                    msg = "El usuario y password no coinciden";
                result = msg;
            }
            catch (Exception ex)
            {
                result = "ERROR";
            }
        }
        else
        {
            //El DataSet es nulo, no hay datos regresados
            result = "El usuario no existe en nuestra base de datos";
        }
    }

    return result;
}
```

MASTER PAGE (UNA ÚNICA PLANTILLA PARA TODO PROYECTO)

ASP.NET master pages allow you to create a consistent layout for the pages in your application. A single master page defines the look and feel and standard behavior that you want for all of the pages (or a group of pages) in your application. You can then create individual content pages that contain the content you want to display. When users request the content pages, they merge with the master page to produce output that combines the layout of the master page with the content from the content page.



Proyecto sencillo usando una Master Page

A un proyecto de WebSite nuevo agregar una Master Page.

Y agregar tantos WebForms habilitados para MasterPage se ocupen

The image is a composite of three screenshots from the Visual Studio IDE, illustrating the steps to create a web application with a Master Page.

Top Left Screenshot: Shows the 'Add' menu open in a new project named 'WebApplication1'. The 'Add' menu is expanded, showing options like 'New Item...', 'Existing Item...', 'New Scaffolded Item...', 'New Folder', 'Add ASP.NET Folder', 'New Azure WebJob Project', 'Existing Project as Azure WebJob', 'Reference...', 'Service Reference...', 'TypeScript File', 'HTML Page', 'JavaScript File', 'Style Sheet', 'Web Form', 'MVC 5 View Page (Razor)', 'Web Forms Master Page', 'Web Form with Master Page', 'Text File', and 'Class...'. The 'Web Form with Master Page' option is highlighted.

Top Right Screenshot: Shows the 'Add New Item - WebApplication1' dialog box. The 'Web' category is selected in the left pane. The 'Web Form with Master Page' item is selected in the right pane. The 'Name' field is empty, and the 'Type' is 'Visual C#'. A blue arrow points from this dialog to the bottom right screenshot.

Bottom Screenshot: Shows the 'Add New Item - WebApplication1' dialog box again, but with the 'Web Forms Master Page' item selected. The 'Name' field is filled with 'SiteMaster'. A blue arrow points from this dialog to the bottom right screenshot.

Bottom Right Screenshot: Shows the 'Add New Item - WebApplication1' dialog box with the 'Web Forms Master Page' item selected. The 'Name' field is filled with 'WebForm4.aspx'. A blue arrow points from this dialog to the bottom right screenshot.

Bottom Right Screenshot: Shows the 'Add New Item - WebApplication1' dialog box with the 'Web Forms Master Page' item selected. The 'Name' field is filled with 'WebForm4.aspx'. A blue arrow points from this dialog to the bottom right screenshot.

Bottom Right Screenshot: Shows the 'Add New Item - WebApplication1' dialog box with the 'Web Forms Master Page' item selected. The 'Name' field is filled with 'WebForm4.aspx'. A blue arrow points from this dialog to the bottom right screenshot.

Site.Master

```
<%@ Master Language="C#" AutoEventWireup="true" CodeBehind="Site.master.cs"
Inherits="WebApplication1.Site" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
  <title>Web Site usando Master Pages</title>
  <asp:ContentPlaceHolder ID="head" runat="server">
  </asp:ContentPlaceHolder>
</head>
<body>
  <form id="form1" runat="server">
  <div>

    <h1>Este código es de la Master Page</h1>
    <asp:ContentPlaceHolder ID="ContentPlaceHolder1" runat="server">

    </asp:ContentPlaceHolder>
    <h2>Este código también es de la Master Page</h2>
  </div>
</form>
</body>
```

Los **ContentPlaceHolder** son segmentos designados a contener el código de los WebForms derivados/hijos y es posible agregar tantos como se deseen.

Normalmente siempre se tienen dos, uno para el Head y otro para el Body.

WebForm1.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm1.aspx.cs"
MasterPageFile="~/Site.Master" Inherits="WebApplication1.WebForm1" %>
<asp:Content ID="Content1" ContentPlaceHolderID="head" runat="server">
  <script type="text/javascript">
    alert("WebForm1.aspx");
  </script>
</asp:Content>
<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1" runat="server">
  <strong>Este texto corresponde al WebForm1</strong>
</asp:Content>
```

El código dentro del **ContentPlaceHolder** será mezclado junto con el de su **MasterPageFile** para así formar un sitio web uniforme.

Las librerías javascript a usar y los temas css son colocados normalmente en la **MasterPage** y cada página derivada únicamente se encarga de agregar su propio código de funcionamiento.