Basics

Simple ASP.NET Web Page

- The following code sample includes one simple ASP.NET page that allows the user to provide his height and his weight and get his calculated BMI.

Simple ASP.NET Web Page

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="BMIForm.aspx.cs" Inherits="singleproj.BMIForm" %>
<!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></title>
</head>
<body>
    <form id="form1" runat="server">
    <div>
        <asp:Label ID="Label1" runat="server" Text="height (meters)"></asp:Label>
        <asp:TextBox ID="TextBox1" runat="server" style="margin-left: 20px"></asp:TextBox>
    </div>
    <div>
        <asp:Label ID="Label2" runat="server" Text="weight (kg)"></asp:Label>
        <asp:TextBox ID="TextBox2" runat="server" style="margin-left: 23px"></asp:TextBox>
    </div>
    <p>
        <asp:Button ID="Button1" runat="server" onclick="Button1_Click" Text="calculate" />
    </p>
    <div>
        <asp:Label ID="Label3" runat="server" Text="result"></asp:Label>
        <asp:TextBox ID="TextBox3" runat="server" style="margin-left: 59px"></asp:TextBox>
    </div>
    </form>
</body>
</html>
```

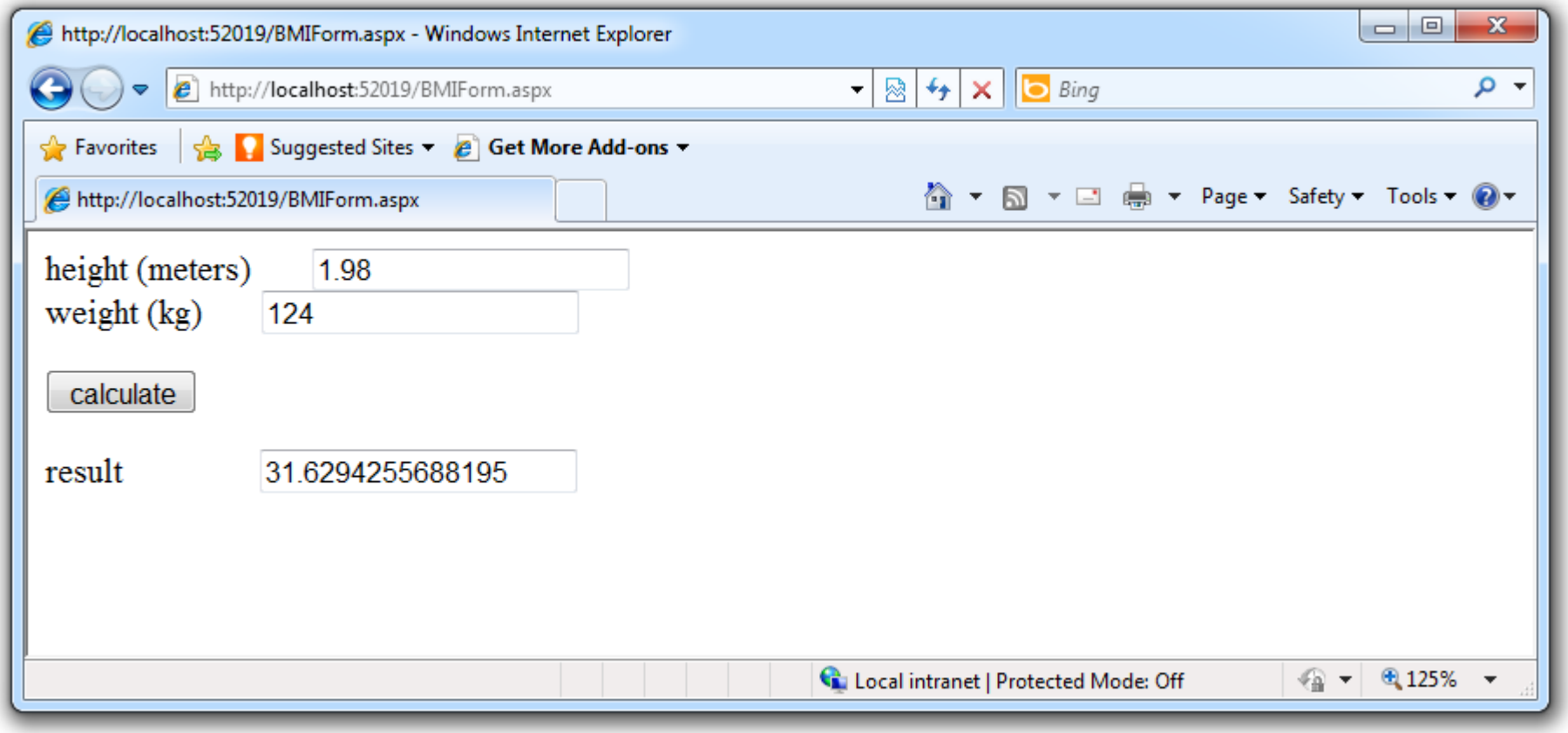
BMIForm.aspx

Simple ASP.NET Web Page

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

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

        }
        protected void Button1_Click(object sender, EventArgs e)
        {
            this.TextBox3.Text = "" + Double.Parse(this.TextBox2.Text) / ( (Double.Parse(this.TextBox1.Text)) * (Double.Parse(this.TextBox1.Text)) );
        }
    }
}
```

BMIForm.aspx.cs

Simple ASP.NET Web Page



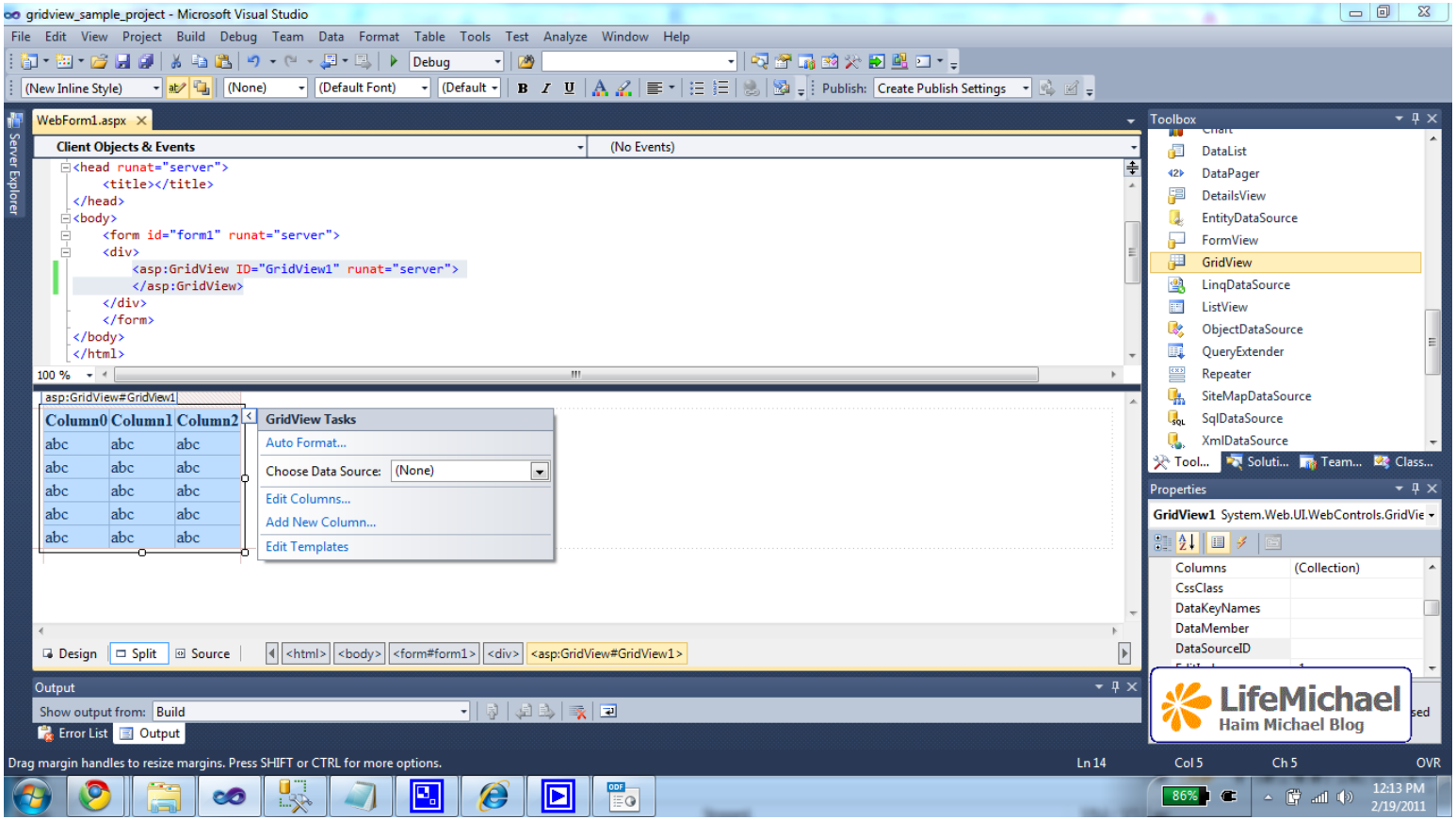
The Grid View Controller

- The grid view controller can spit out back to the browser a grid of data fetched from a data source.
- The following code sample includes a data source connected with a gird view. The data source is a simple table on the database.
- Once we drag the grid view control into our web form we can easily configure it using a built-in wizard for that purpose.

The Grid View Controller

- Once we drag the grid view control into our web form we can easily configure it using a built-in wizard for that purpose.

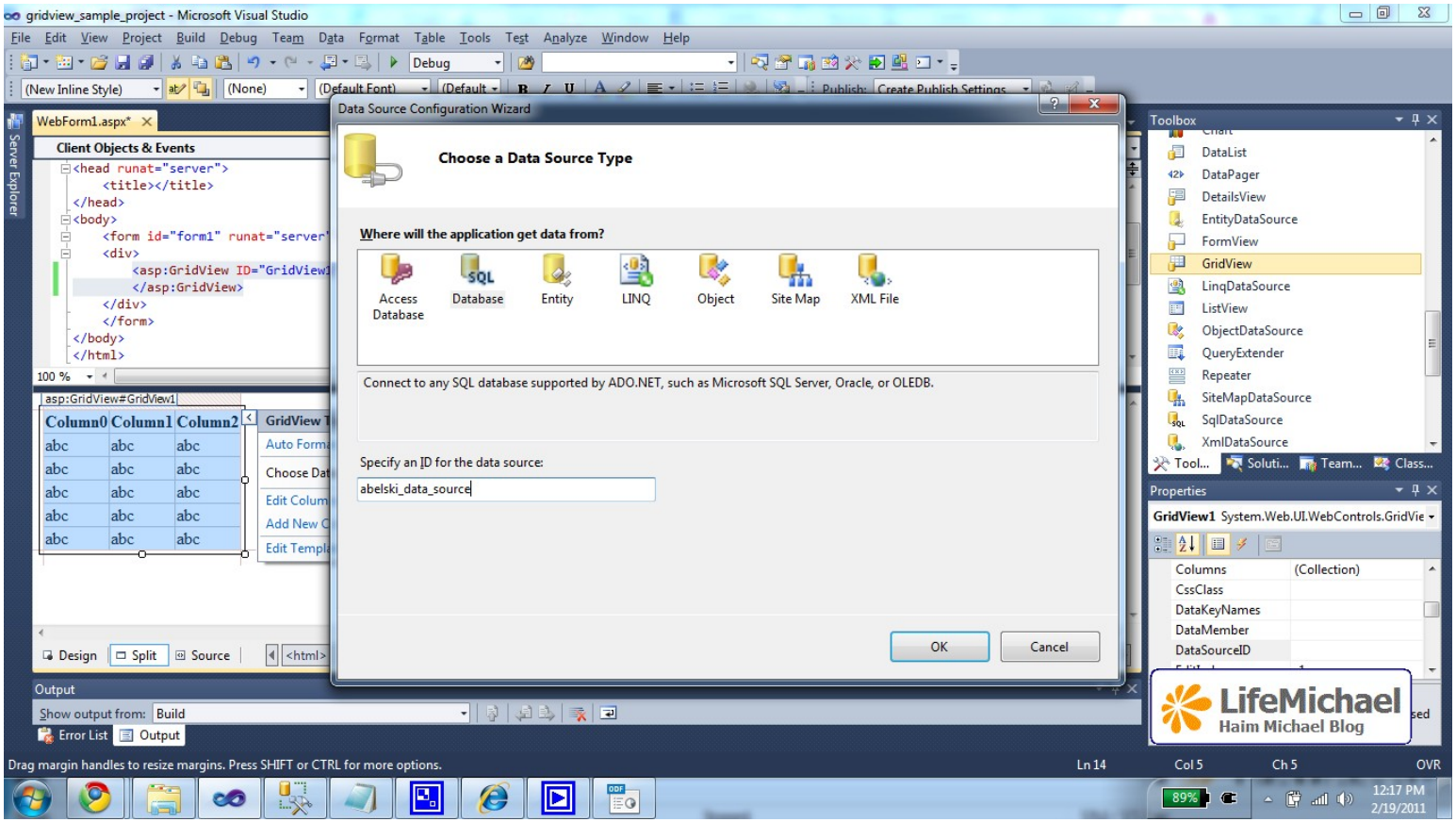
The Grid View Controller



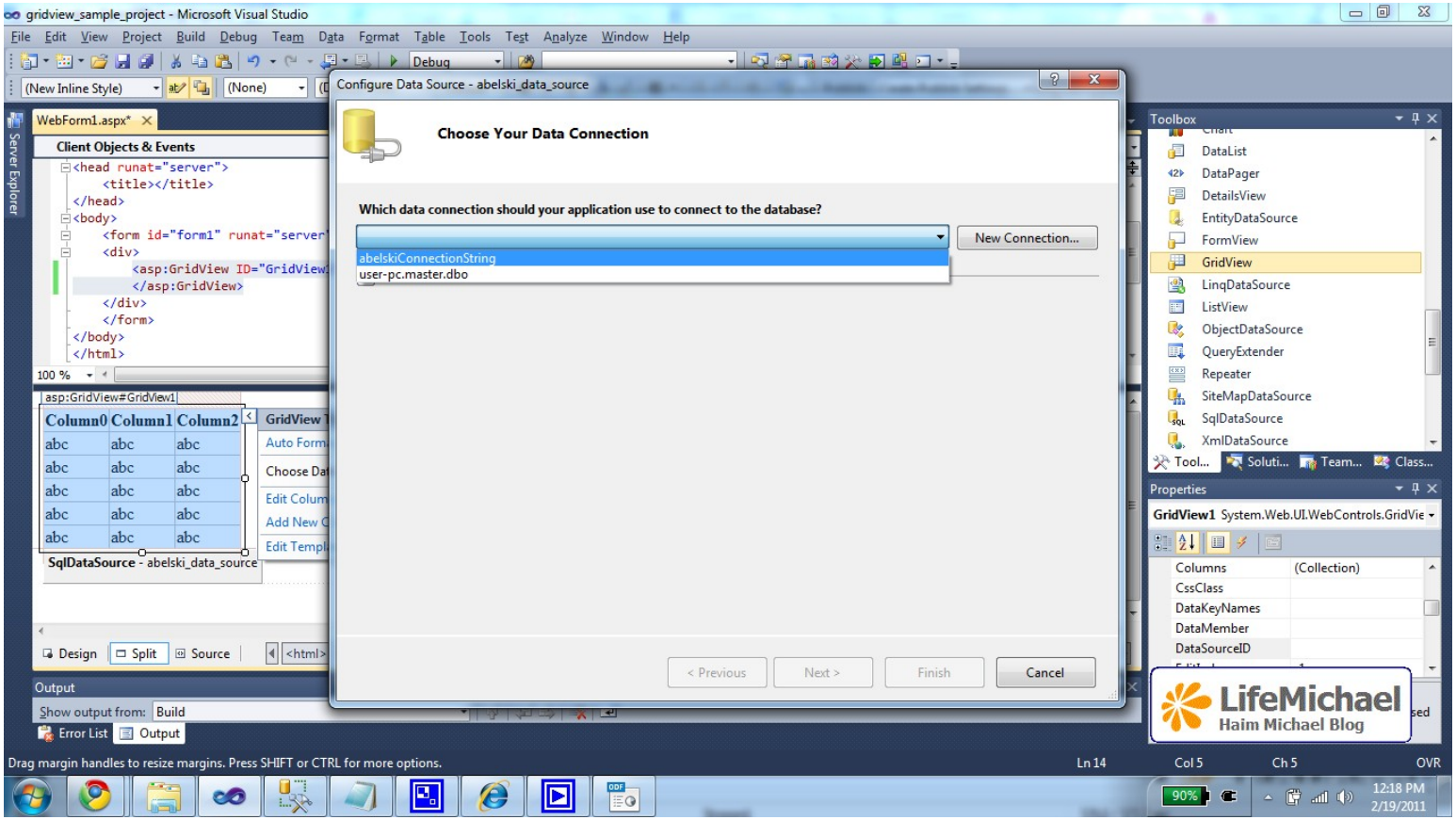
The Grid View Control

- The wizard allows us to specify the database we want to work with, the exact table from which we want to fetch the data and even configure the auto generated SQL statement that will be responsible for fetching the data the grid view displays.

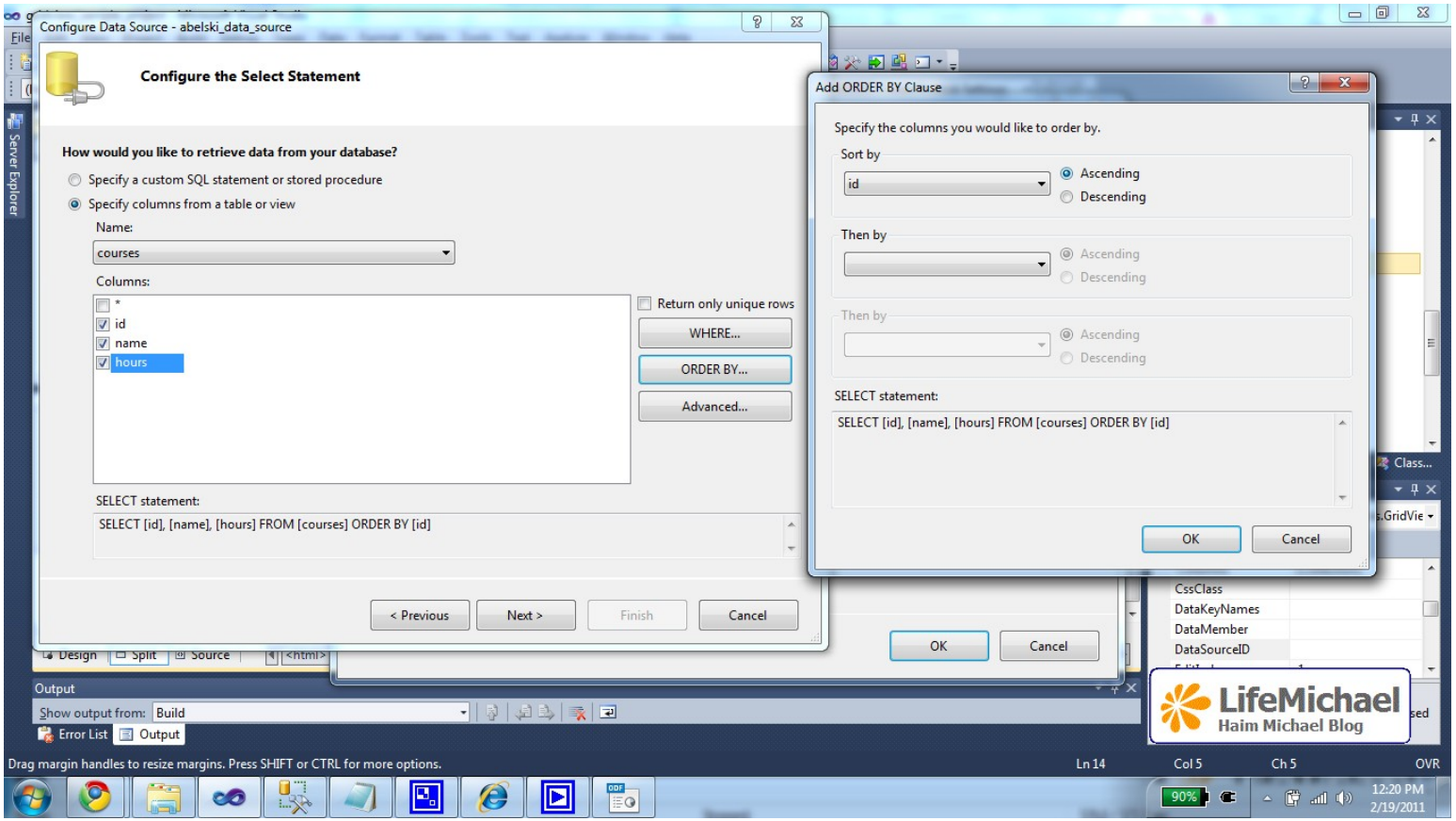
The Grid View Control



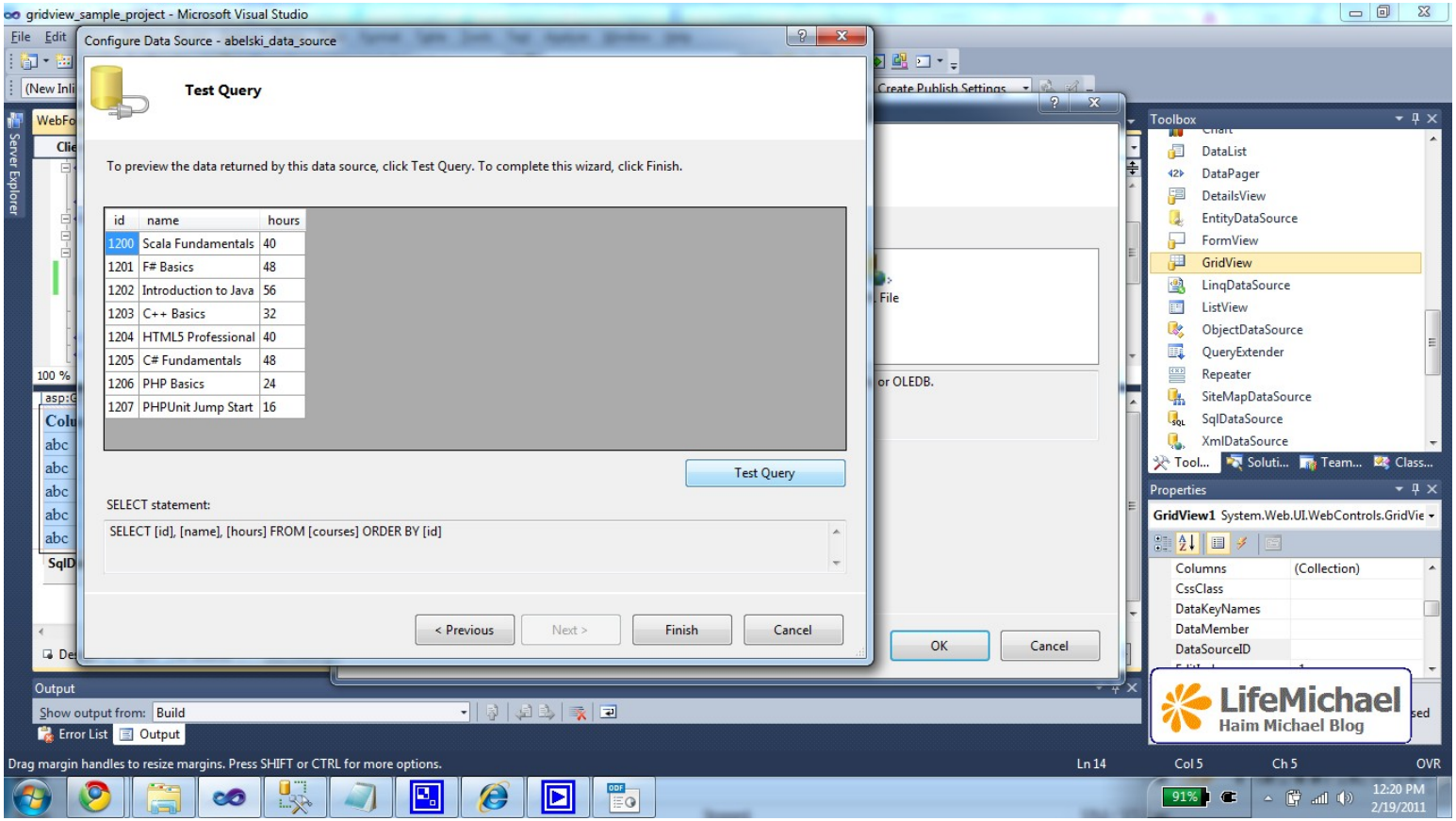
The Grid View Control



The Grid View Control



The Grid View Control



The Grid View Control

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm1.aspx.cs"
Inherits="gridview_sample_project.WebForm1" %>

<!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></title>
</head>
<body>
    <form id="form1" runat="server">
    <div>
```

This is The Auto Generated Code

The Grid View Control

```
        <asp:GridView ID="GridView1" runat="server" AutoGenerateColumns="False"
            DataKeyNames="id" DataSourceID="abelski_data_source">
            <Columns>
                <asp:BoundField DataField="id" HeaderText="id" ReadOnly="True"
                    SortExpression="id" />
                <asp:BoundField DataField="name" HeaderText="name"
                     SortExpression="name" />
                <asp:BoundField DataField="hours" HeaderText="hours"
                     SortExpression="hours" />
            </Columns>
        </asp:GridView>

        <asp:SqlDataSource ID="abelski_data_source" runat="server"
            ConnectionString="<%$ ConnectionStrings:abelskiConnectionString %>"
            SelectCommand="SELECT [id], [name], [hours] FROM [courses] ORDER BY [id]">
        </asp:SqlDataSource>

    </div>

    </form>

</body>
</html>
```

The Grid View Control

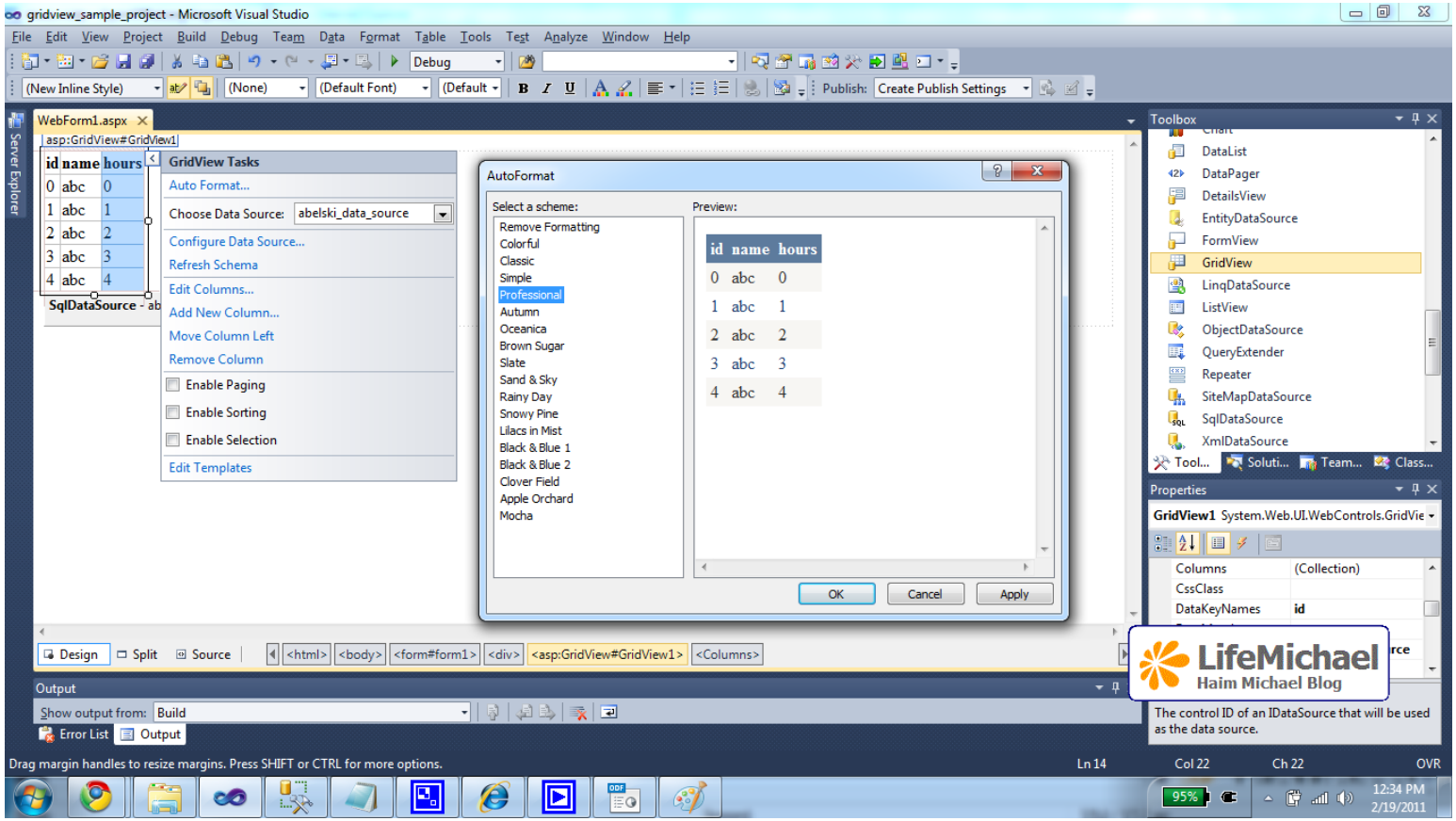
http://localhost:53439/WebForm1.aspx - Windows Internet Explorer

id	name	hours
1200	Scala Fundamentals	40
1201	F# Basics	48
1202	Introduction to Java	56
1203	C++ Basics	32
1204	HTML5 Professional	40
1205	C# Fundamentals	48
1206	PHP Basics	24
1207	PHPUnit Jump Start	16

The Grid View Control

- We can easily change the format of our grid view. Selecting the grid view control we can select the context menu and select the auto format option.

The Grid View Control



The Grid View Control

http://localhost:53439/WebForm1.aspx - Windows Internet Explorer

id	name	hours
1200	Scala Fundamentals	40
1201	F# Basics	48
1202	Introduction to Java	56
1203	C++ Basics	32
1204	HTML5 Professional	40
1205	C# Fundamentals	48
1206	PHP Basics	24
1207	PHPUnit Jump Start	16

LifeMichael
Haim Michael Blog

Single ASP.NET Web Page

- We can use the `Import` directive in the 'code behind' for specifying a specific namespace we want to use.

```
<% Import Namespace = "AbelskiDataLayer" %>
```

Single ASP.NET Web Page

- We can use the `Assembly` directive in the 'code behind' for specifying a specific assembly (`.dll` file) we want to use.

```
<% Assembly Name = "AbelskiDataAccessLayer" %>
```

ASP.NET Directives

- Each ASP.NET can start with a set of directives. The ASP.NET directives are denoted with <%@ %> markers. Within those markers we can place various attributes.
- The purpose of the ASP.NET directives is to inform the ASP.NET run-time about the way it should work.

The `Page` Directive

- At the minimum, each ASP.NET must have the `<%@Page%>` directive, that defines the programming language in use within our ASP.NET document. We specify that language by using the `Language` attribute.

The Page Directive

```
<%@ Page
    Language="C#"
    AutoEventWireup="true"
    CodeBehind="WebForm1.aspx.cs"
    Inherits="directives_sample.WebForm1" %>

<!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></title>
</head>
<body>
    <form id="form1" runat="server">
    <div>

    </div>
    </form>
</body>
</html>
```

The `Page` Directive

- The `Language` attribute specifies the programming language we use.

```
<%@ Page
    Language="C#"
    AutoEventWireup="true"
    CodeBehind="WebForm1.aspx.cs"
    Inherits="directives_sample.WebForm1" %>
```

The `Page` Directive

- The `CodeBehind` attribute specifies the name of the file that holds the code behind for our ASP.NET page.

```
<%@ Page
    Language="C#"
    AutoEventWireup="true"
    CodeBehind="WebForm1.aspx.cs"
    Inherits="directives_sample.WebForm1" %>
```

The `Page` Directive

- The `EnableTheming` attribute specifies whether the controls we use support ASP.NET themes or not.

```
<%@ Page
    Language="C#"
    AutoEventWireup="true"
    CodeBehind="WebForm1.aspx.cs"
    EnableTheming="true"
    Inherits="directives_sample.WebForm1" %>
```

The `Page` Directive

- The `EnableViewState` attribute specifies whether the view state is maintained across page requests.

```
<%@ Page
    Language="C#"
    AutoEventWireup="true"
    CodeBehind="WebForm1.aspx.cs"
    EnableTheming="true"
    EnableViewState="true"
    Inherits="directives_sample.WebForm1" %>
```

The `Page` Directive

- The `Inherits` attribute specifies the name of the class our code behind class inherits from.

```
<%@ Page
    Language="C#"
    AutoEventWireup="true"
    CodeBehind="WebForm1.aspx.cs"
    EnableTheming="true"
    EnableViewState="true"
    Inherits="directives_sample.WebForm1" %>
```

The `Page` Directive

- The `MasterPageFile` attribute specifies the name of the master page in use.

```
<%@ Page
    Language="C#"
    AutoEventWireup="true"
    CodeBehind="WebForm1.aspx.cs"
    MasterPage="MyWebsite.Master"
    EnableTheming="true"
    EnableViewState="true"
    Inherits="directives_sample.WebForm1" %>
```

The `Page` Directive

- The `Trace` attribute specifies whether the tracing feature is enabled or not.

```
<%@ Page
    Language="C#"
    AutoEventWireup="true"
    CodeBehind="WebForm1.aspx.cs"
    MasterPage="MyWebsite.Master"
    Trace="true"
    EnableViewState="true"
    Inherits="directives_sample.WebForm1" %>
```

The `Import` Directive

- We can use the Import directive for specifying a namespace required by our ASP.NET page.

```
<%@ Import Namespace="AbelskiDBLayer" %>
```

The `Assembly` Directive

- We can use the Assembly directive for specifying an assembly required by our ASP.NET page.

```
<%@ Assembly Name="AbelskiDBLayer" %>
```

- The assemblies we want to use should be placed within the `bin` folder, a sub folder of our web site.

The `web.config` File

- The purpose of this `XML` file is to set various definitions relevant for our web application.

The `web.config` File

directives_sample - Microsoft Visual Studio

File Edit View Project Build Debug Team Data Tools Test Analyze Window Help

Debug | Publish: Create Publish Settings

Web.config | WebForm1.aspx

```
<?xml version="1.0"?>

<!--
  For more information on how to configure your ASP.NET application, please visit
  http://go.microsoft.com/fwlink/?LinkId=169433
  -->

<configuration>
    <system.web>
        <compilation debug="true" targetFramework="4.0" />
    </system.web>

</configuration>
```

Solution Explorer

- Solution 'directives_sample' (1 project)
  - directives_sample
    - Properties
    - References
    - Web.config
      - Web.Debug.config
      - Web.Release.config
    - WebForm1.aspx

Properties

XML Document

Encoding	Unicode (UTF-8)
Output	
Schemas	C:\Program Files (x86)\M
Stylesheet	

Output
Result of XSL transform.

Output

Show output from:

Error List | Output

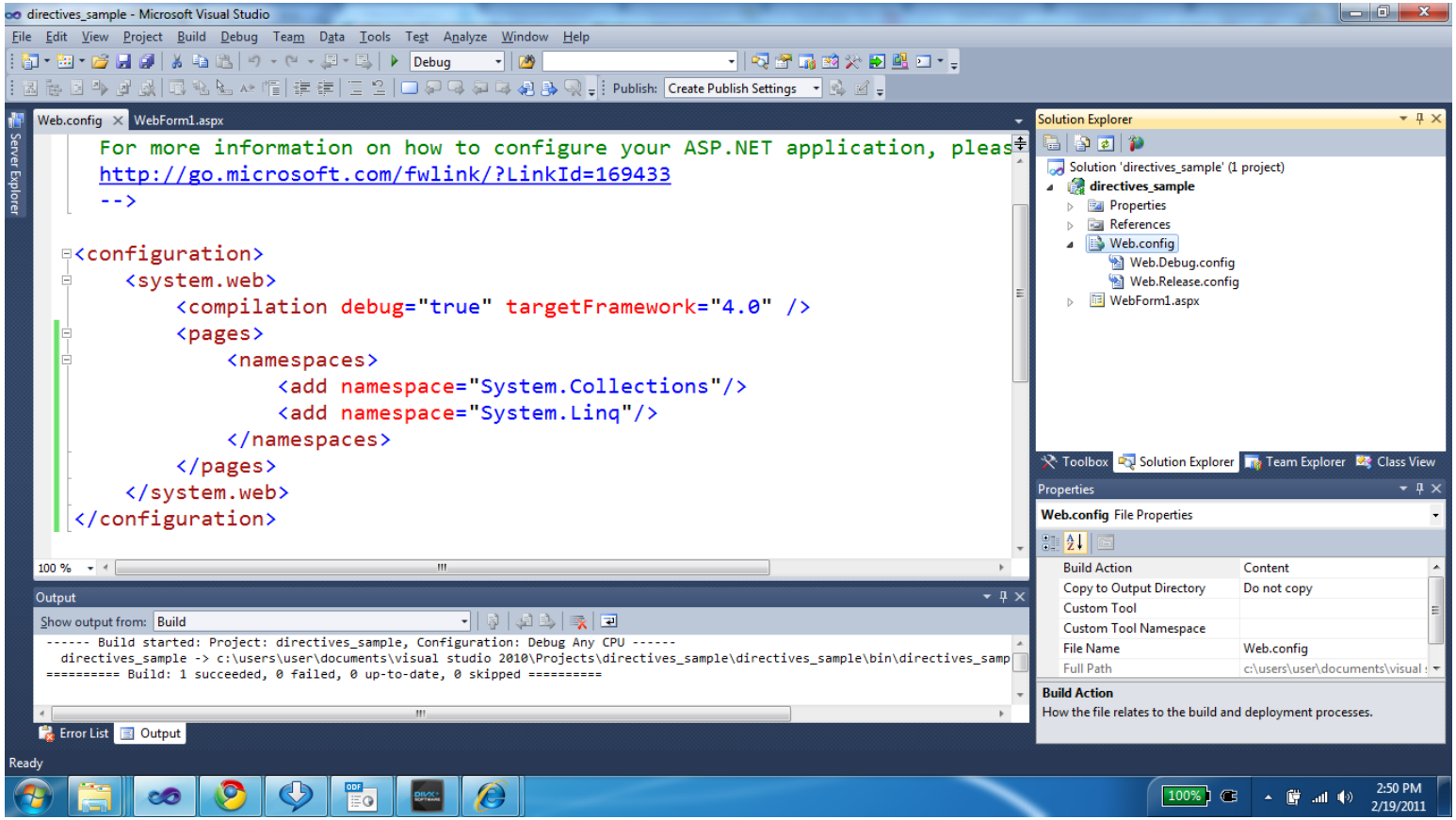
Ready

100 % | 2:14 PM 2/19/2011

The `web.config` File

- We can specify within this file the namespaces we would like to have access without having the need to use the `Namespace` directive.
- We do it using the `<pages>` and the `<namespaces>` XML elements.

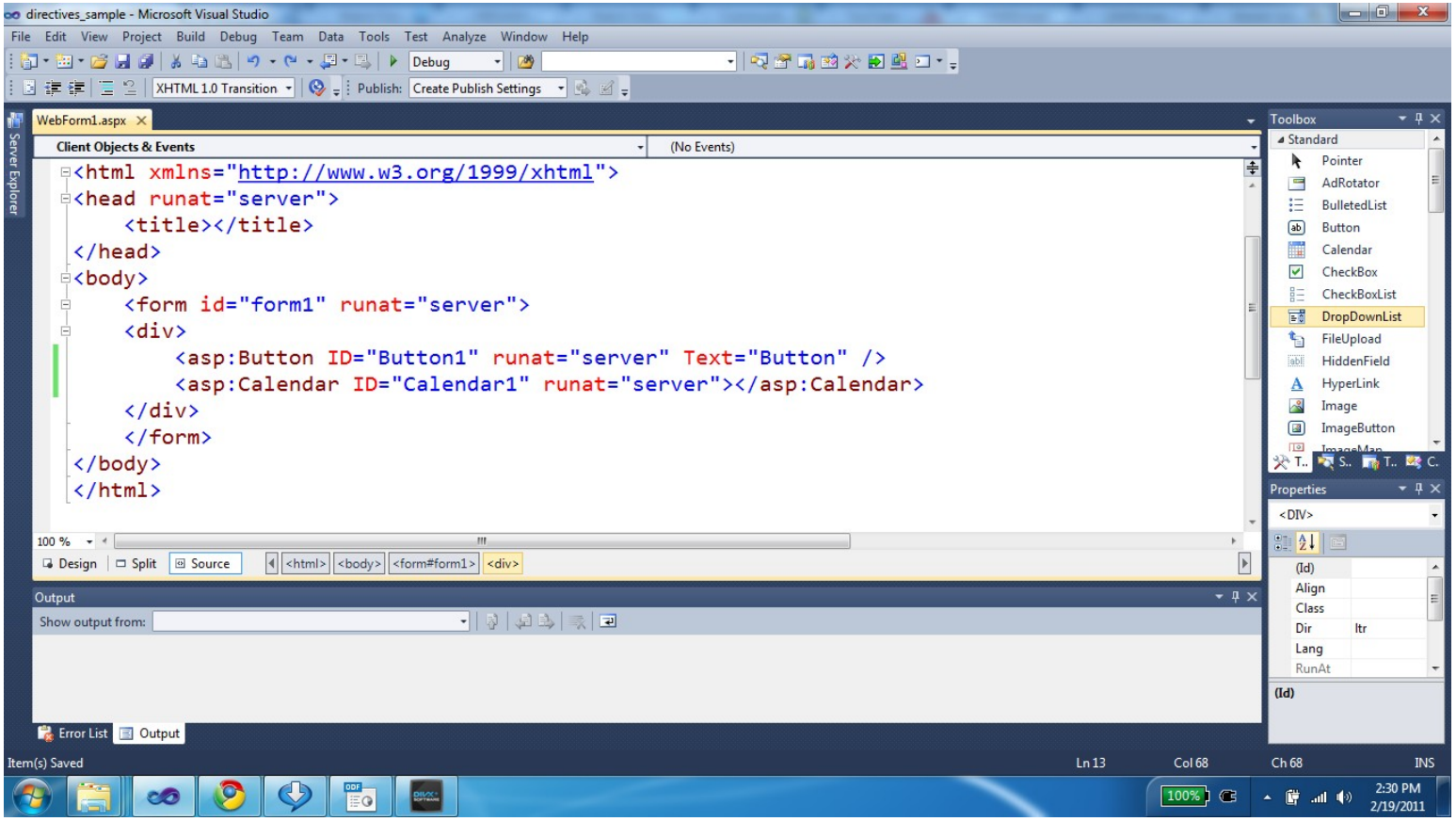
The `web.config` File



The `runat="server"` Attribute

- This attribute specifies our wish to have the code executed on the server side.
- We can always find it within the controls we place on our web page. It is located within the opening tag.

The `runat="server"` Attribute



Single ASP.NET Model

- One of the simplest approaches for developing an ASP.NET web application is the development of a single `*.aspx` page that includes the code behind as part of it.
- The advantages in this approach includes the relatively simple deployment and the relatively simple maintenance.
- The disadvantages might be an over complex `*.aspx` difficult for maintenance.

Single ASP.NET Web Page

- When our ASP.NET web application includes a single file only we can use the `Import` and the `Assembly` directives for specifying namespaces and assemblies we want to use in our code behind.

Single ASP.NET Web Page

- We can use the `Import` directive in the 'code behind' for specifying a specific namespace we want to use.

```
<% Import Namespace = "AbelskiDataLayer" %>
```

Single ASP.NET Web Page

- We can use the `Assembly` directive in the 'code behind' for specifying a specific assembly (`.dll` file) we want to use.

```
<% Assembly Name = "AbelskiDataAccessLayer" %>
```

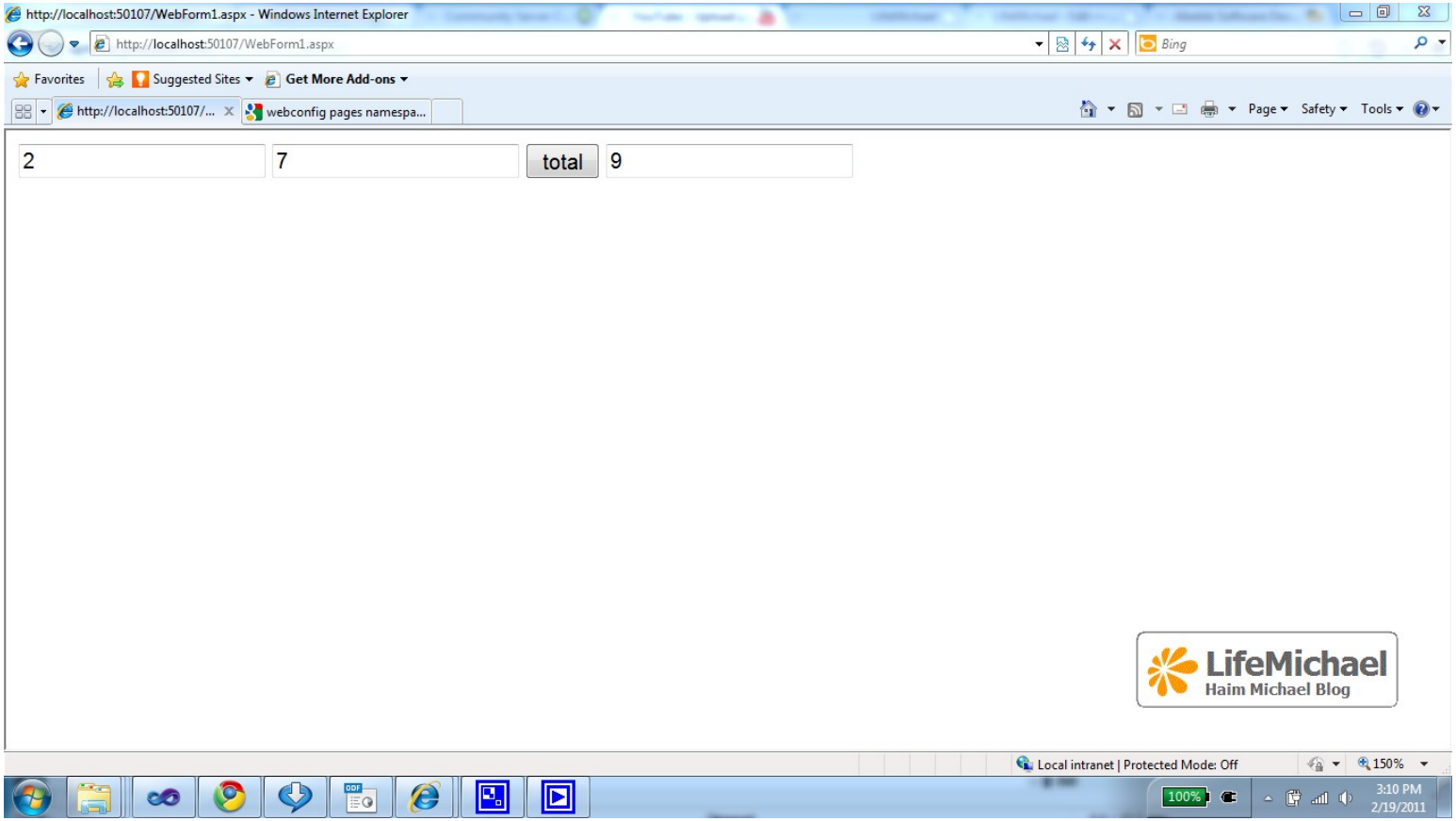
Single ASP.NET Web Page

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm1.aspx.cs"
Inherits="directives_sample.WebForm1" %>

<!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 id="Head1" runat="server">
    <title></title>
    <script runat="server">
        protected void CalculateTotal(object sender, EventArgs e)
        {
            try
            {
                double sum = (Double.Parse(this.TextBox1.Text)
                    + Double.Parse(this.TextBox2.Text));
                this.TextBox3.Text = "" + sum;
            }
            catch
            {
                this.TextBox3.Text = "error";
            }
        }
    </script>
</head>
```

Single ASP.NET Web Page

```
<body>
    <form id="form1" runat="server">
    <div>
        <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
        <asp:TextBox ID="TextBox2" runat="server"></asp:TextBox>
        <asp:Button ID="Button1" runat="server" Text="total"
            onclick="CalculateTotal" />
        <asp:TextBox ID="TextBox3" runat="server"></asp:TextBox>
    </div>
    </form>
</body>
</html>
```

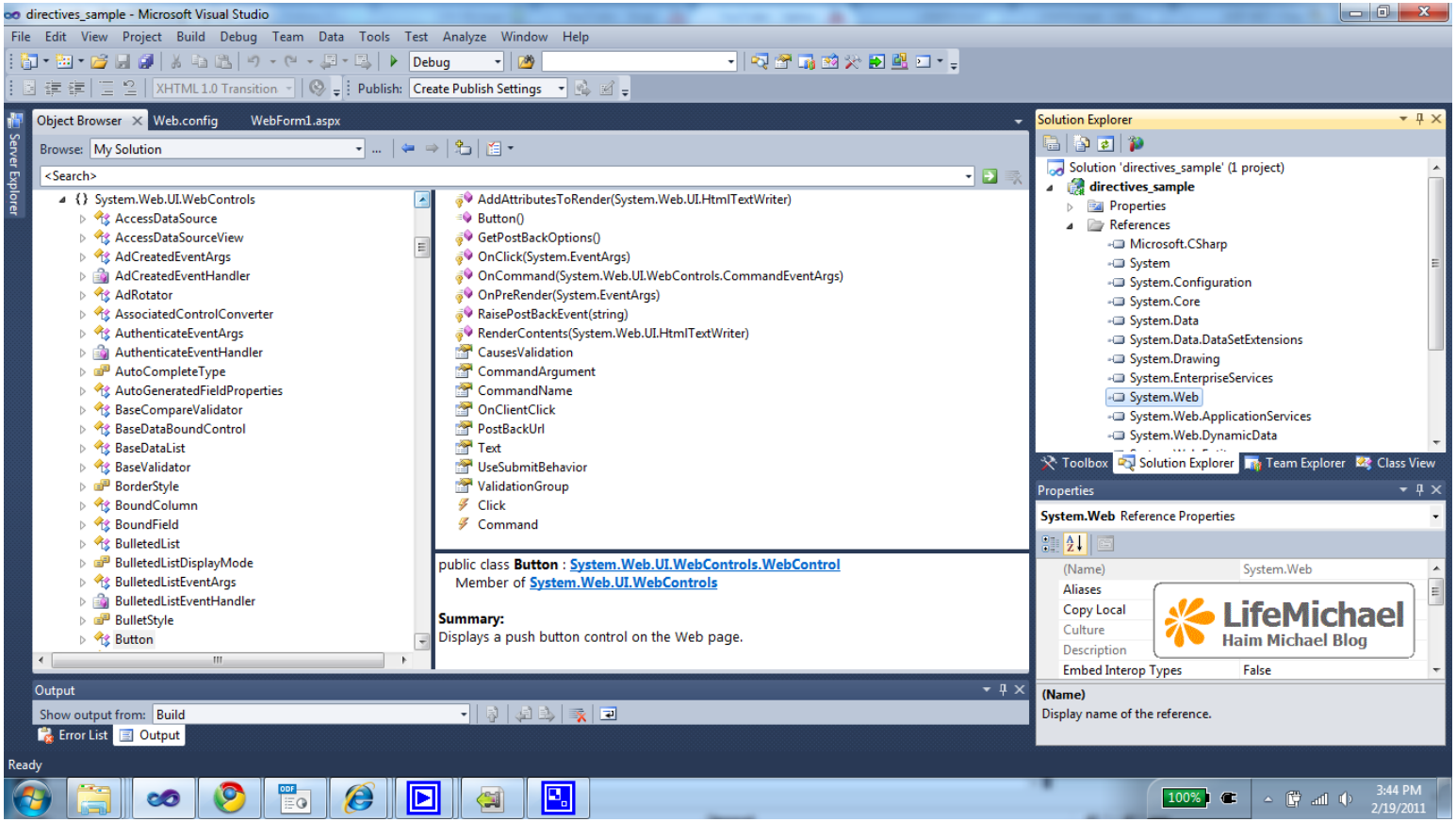
Single ASP.NET Web Page



The `WebControls` Namespace

- The `System.Web.UI.WebControls` namespace includes the definition for most of the ASP.NET web controls.
- This namespace is part of the `System.Web.dll` assembly.

The `WebControls` Namespace



The `WebControl` Class

- This is the parent class for all ASP.NET controls. It defines the common UI properties we can expect to find in each one of the available controls (Height, BackColor etc).

Single File Page Compilation Cycle

- When developing a single file page model the entire code in our page is dynamically compiled into a class type that extends `System.Web.UI.Page` when the first HTTP request arrives.
- Each control is defined as a member of this auto generated new class.

Single File Page Compilation Cycle

MultipleObjectsGraphs - Microsoft Visual Studio

WebForm1.aspx.cs

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

namespace MultipleObjectsGraphs
{
    public partial class WebForm1 : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {

        }
    }
}
```

Single File Page Compilation Cycle

- The auto generated assembly is then deployed on our server available for HTTP requests.
- Due to this compilation cycle the first request is handled slowly comparing with those that follow.

Multiple Files Compilation Cycle

- When taking the default approach and placing our code within separated files (code behind) we get a clean separation between the HTML markup and the code.
- This allows designers to work separately from the programmers.
- The compilation cycle when taking the multiple files default approach is similar to the compilation cycle of a single file web application.

Multiple Files Compilation Cycle

- The auto generated class is split into three separated files. The use of the partial modifier allows that to happen.

The `aspnet_compiler.exe` Utility

- We can pre compile all pages or just a subset of them so that the first request will be hand-held fast as those that follow.
- The `aspnet_compiler` utility allows us to do it. The use of this utility is done from the command line.

Debuggin ASP.NET Web Application

- We can debug our ASP.NET web application just as we debug any other .NET application.
- The debugging option is disabled (by default) in order to ensure the highest performance possible. In order to change that we should change the `Web.config` file to include the following code:

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

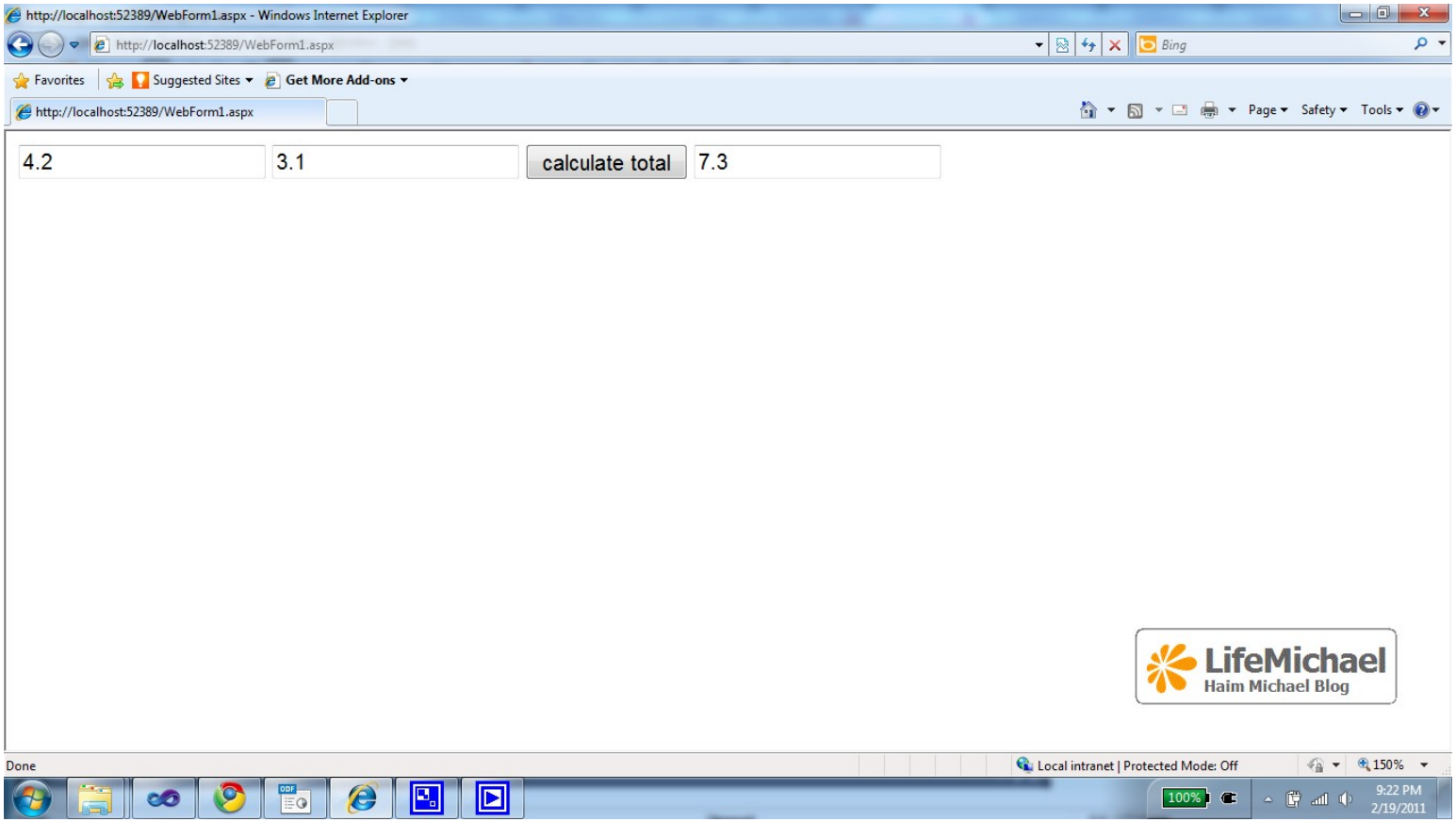
Debuggin ASP.NET Web Application

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm1.aspx.cs"
Inherits="multiple_files_model.WebForm1" %>

<!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></title>
</head>
<body>
    <form id="form1" runat="server">
    <div>
        <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
        <asp:TextBox ID="TextBox2" runat="server"></asp:TextBox>
        <asp:Button ID="Button1" runat="server" Text="calculate total"
            onclick="CalculateValues" />
        <asp:TextBox ID="TextBox3" runat="server"></asp:TextBox>
    </div>
    </form>
</body>
</html>
```

Debuggin ASP.NET Web Application

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

namespace multiple_files_model
{
    public partial class WebForm1 : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
        }

        protected void CalculateValues(object sender, EventArgs e)
        {
            double sum = Double.Parse(this.TextBox1.Text)
                + Double.Parse(this.TextBox2.Text);
            string str = sum.ToString();
            this.TextBox3.Text = str;
        }
    }
}
```

Debuggin ASP.NET Web Application



Tracing ASP.NET Web Application

- We can trace our ASP.NET web application and get the tracing data in the web browser.
- The tracing capability was added in ASP.NET 2.0. We can trace either in the page level or the application one.

Tracing ASP.NET Web Application

- We enable the tracing for the whole application by adding the trace element to the `Web.config` file.

```
<configuration>
    <appSettings/>
    <connectionStrings/>
    <system.web>
        <compilation debug="false" />
        <authentication mode="Windows" />
        <trace enabled ="true" pageOutput ="false"
            requestLimit ="30" traceMode ="SortByTime " />
    </system.web>
</configuration>
```

Tracing ASP.NET Web Application

- We enable the tracing for a specific page by adding `"Trace=true"` to the page directive.

```
<%@ Page Language="C#" Trace="true" TraceMode = "SortByCategory"
Inherits  = "System.Web.UI.Page" CodeFile="Default.aspx.cs" %>
```

- The page setting takes advantage over the application setting. If the application setting says that there should be tracing and the page setting says different then for that very same page the tracing won't take place.
- Browsing at `trace.axd` we will get to see all tracing data.

Tracing ASP.NET Web Application

http://localhost:52389/WebForm1.aspx - Windows Internet Explorer

4 | 4.5 | calculate total | 8.5

Request Details

Session Id:	ecmemjxgmawpymppprl52ctvn	**Request Type:**	POST
Time of Request:	2/19/2011 9:40:48 PM	**Status Code:**	200
Request Encoding:	Unicode (UTF-8)	**Response Encoding:**	Unicode (UTF-8)

Trace Information

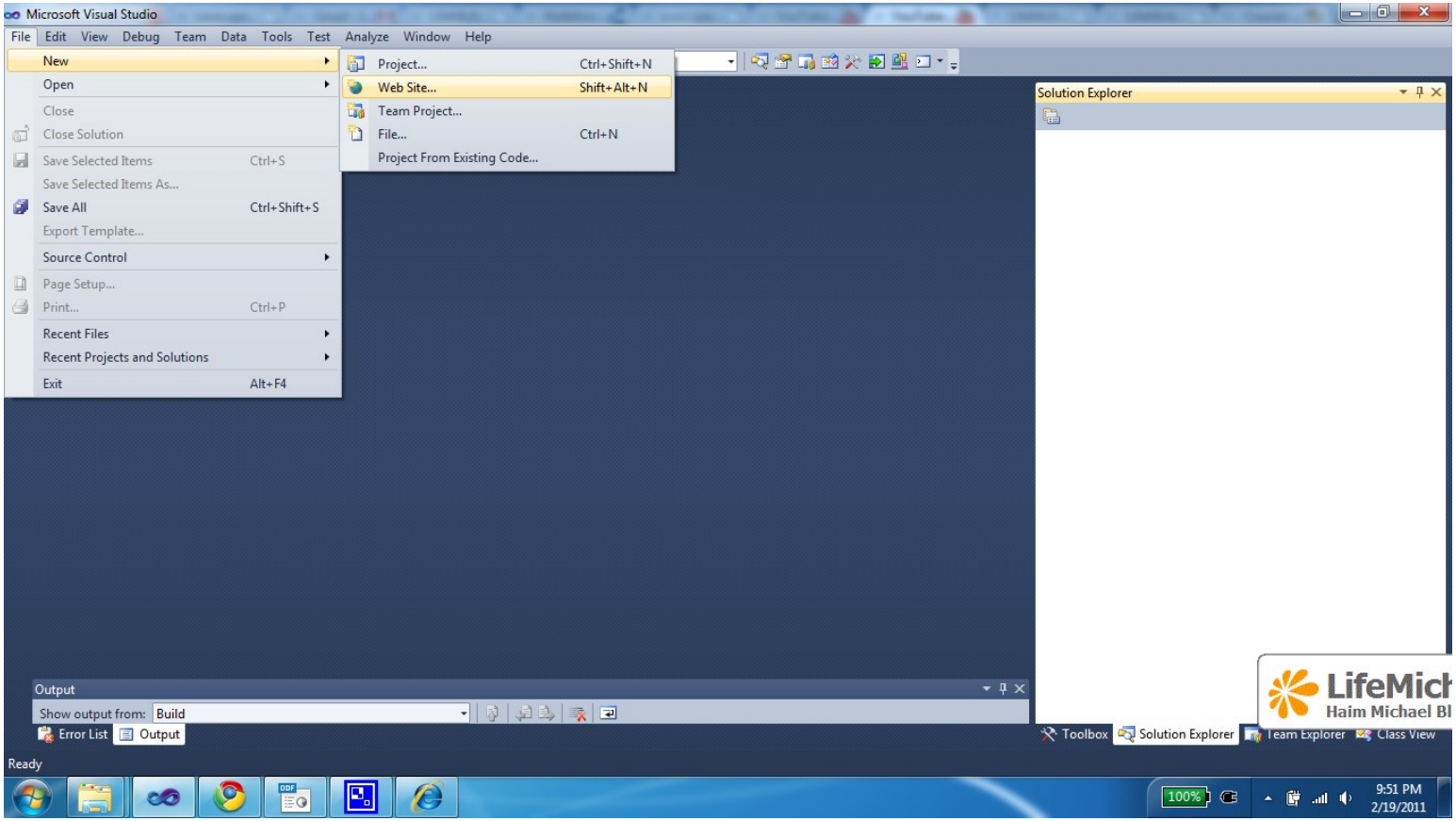
Category	Message	From First(s)	From Last(s)
aspx.page	Begin PreInit		
aspx.page	End PreInit	2.22421831558236E-05	0.000022
aspx.page	Begin Init	3.59296804824843E-05	0.000014
aspx.page	End Init	4.9617177809145E-05	0.000014
aspx.page	Begin InitComplete	5.81718636383079E-05	0.000009
aspx.page	End InitComplete	6.75820180503871E-05	0.000009
aspx.page	Begin LoadState	7.65644381710082E-05	0.000009
aspx.page	End LoadState	0.000149707002010351	
aspx.page	Begin ProcessPostData	0.000162539030754096	
aspx.page	End ProcessPostData	0.000264767526412593	
aspx.page	Begin PreLoad	0.000278455023739253	0.000014

LifeMichael
Haim Michael Blog

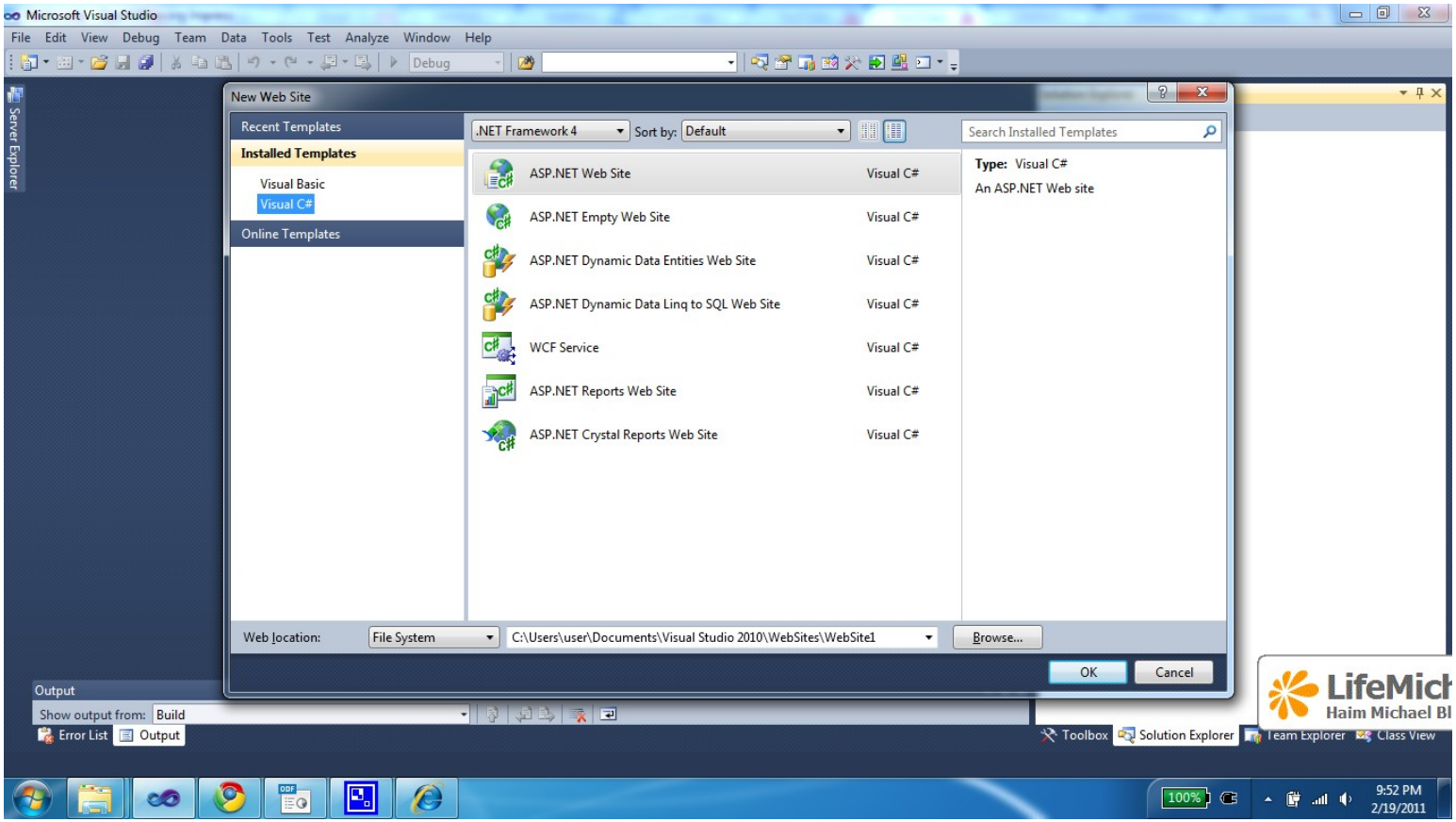
ASP.NET Web Site

- Instead of creating a new ASP.NET web application we can create an ASP.NET web site.

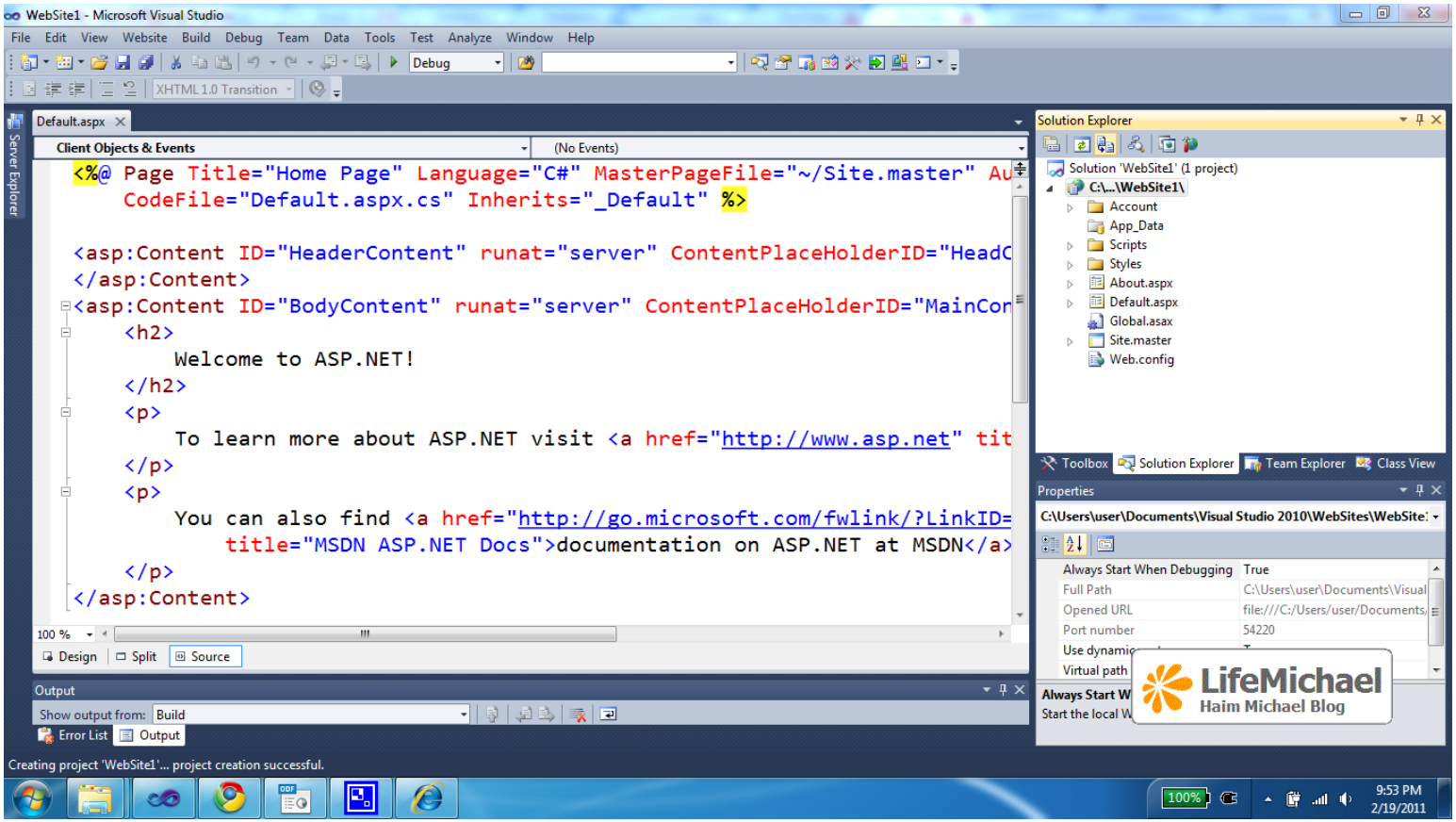
ASP.NET Web Site



ASP.NET Web Site



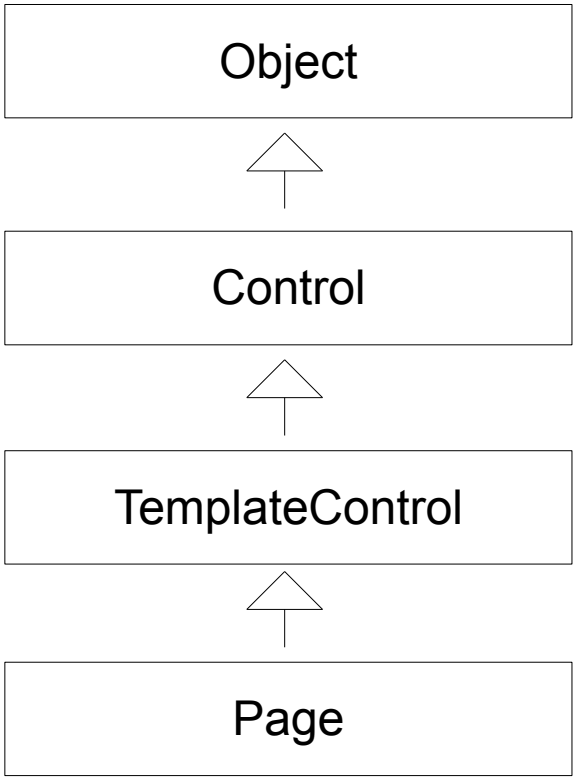
ASP.NET Web Site



ASP.NET Web Site

- The folders structure of ASP.NET web site project is different.
- Unlike ASP.NET web application that gets compiled into a single DLL file, the ASP.NET web site is compiled on the fly and the result is more than one DLL file.

The Page Inheritance Chain



The `Page.Application` Property

- This property allows us to share data across the entire web application between all users.

The `Page.Cashe` Property

- This property allows us to interact with the cache object. The cache object is responsible for the web site caching.

The `Page.ClientTarget` Property

- This property allows us to control the way a page is going to be rendered in according with the requesting web browser.
- When we avoid this property and don't set a value an automatic browser detection is enabled and the returned output is in according with the identified web browser.
- If we set a value for this property then the automatic web browser optimization won't take place.

The `Page.IsPostBack` Property

- This property tells us whether we are dealing with a response to user post back or whether the page is loaded for the first time.

The `Page.MasterPageFile` Property

- This property sets a master page for our page. Unless we assign this property with a value there won't be any master page in action.

The `Page.Request` Property

- This property provides access to the HTTP request. This property provides us with a reference for the `HttpRequest` object that represents the request.

The `Page.Response` Property

- This property provides access to the HTTP response. This property provides us with a reference for the `HttpResponse` object that represents the response.

The `Page.Server` Property

- This property provides access to the `HttpServerUtility` object on which we can invoke various useful functions related to the running server.

The `Page.Session` Property

- This property allows us to interact with the session data. Each and every user has his session. Each session has its data.

The `Page.Theme` Property

- This property allows us to get or set the name of the them we want our page to use.

The `Page.Trace` Property

- This property provides us with access to the `TraceContext` object. That object allows us to log custom messages during the debugging session.