Data Binding

Introduction

- Data binding allows us to bind data objects to one or more web controls, that will show the data automatically.
- The data source control allows us to define a declarative link between our web page and a specific data source, such as a database or a custom data access component.
- Once a data source was configured we can hook it with web controls in our web page. The web controls will display data retrieved from the data source control.

Introduction

- The data binding model is extensible. We can extend it with new functionality
- The data binding is defined outside the source code. The data binding in ASP.NET is declarative. We define it within the ASP.NET web page.

Single Value Data Binding

- Most of the web controls (`TextBox`, `LinkButton`, `Image` etc) support a single value data binding.
- When the control supports single value data binding we can bind any of its properties with a data source.
- The data binding expression is enclosed between the `<%#` and `%>` delimiters.

```
<%# expression %>
```

Single Value Data Binding

- When calling the `Page.DataBind()` method in our code ASP.NET will go over all the binding expressions on our page and replace each one of them with the corresponding value.
- We will usually assign the data binding expressions to properties of controls on our page.

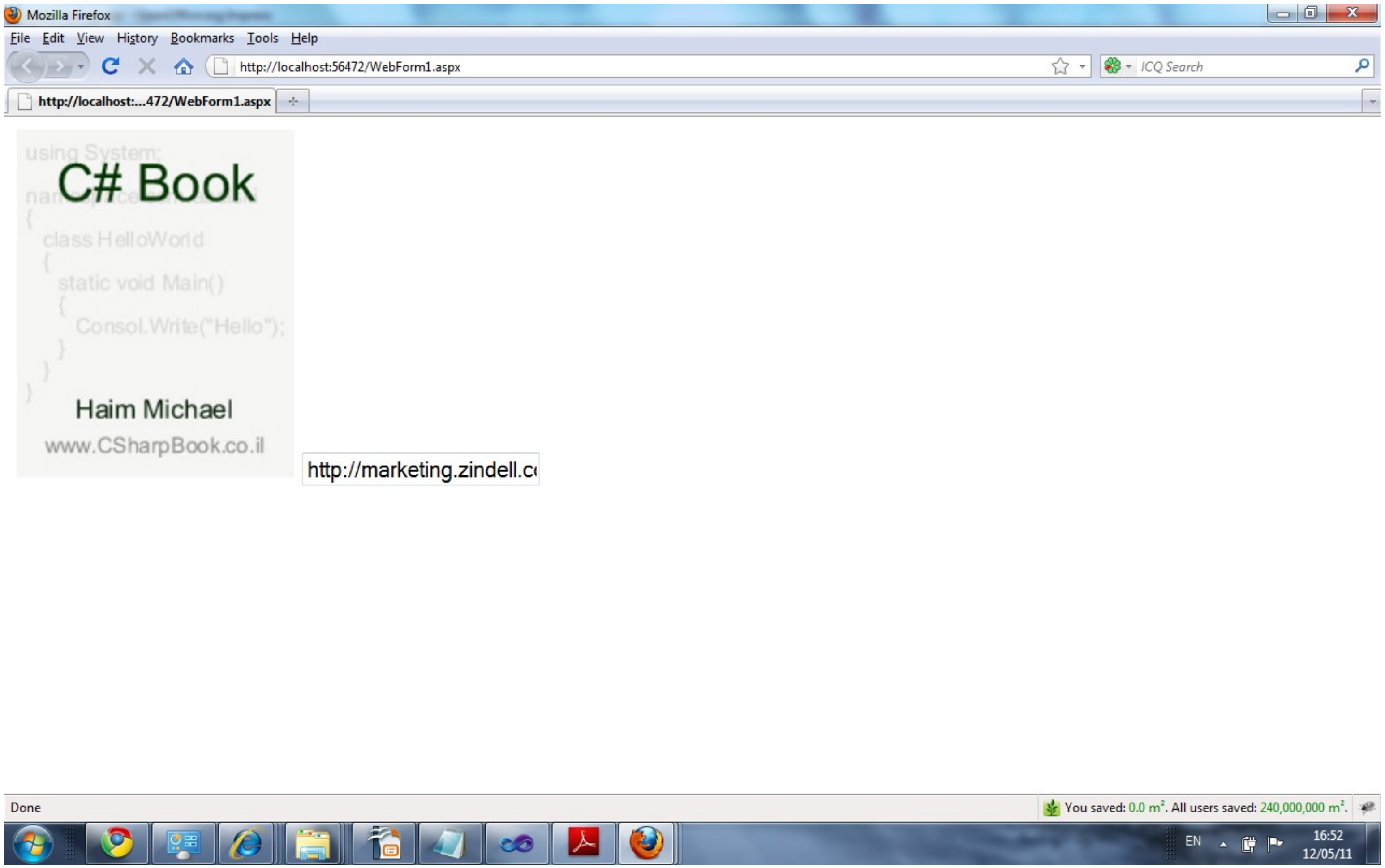
Single Value Data Binding

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm1.aspx.cs"
Inherits="WebApplication1.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>
        <img src="<%# GetImageURL() %>" alt="justpix" />
        <asp:TextBox ID="tb1" runat="server" Text="<%# GetImageURL() %>" />
    </div>
    </form>
</body>
</html>
```

Single Value Data Binding

```
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 WebForm1 : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            this.DataBind();
        }
        protected string GetImageURL()
        {
            return
                "http://marketing.zindell.com/banners/CSHARPBOOK_128x160.jpg";
        }
    }
}
```

Single Value Data Binding


Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://localhost:56472/WebForm1.aspx
ICQ Search
http://localhost:...472/WebForm1.aspx
C# Book
Haim Michael
www.CSharpBook.co.il
http://marketing.zindell.c
Done
You saved: 0.0 m². All users saved: 240,000,000 m².
EN
16:52
12/05/11

Expression Builder

- Instead of using the `#` character we can use `$`. Instead of writing expressions in the `<%#   ...   %>` format we can write expressions such as `<%$  ...  %>`.
- When using `$` instead of `#` we indirectly use the expression builder. The expression builder processes the expression and replaces it with a string in according with its own rules.
- There is no need in calling the `DataBind()` method. The expression builder automatically works.

Expression Builder

- The expression builder allows us to extract custom settings from the `web.config` file.

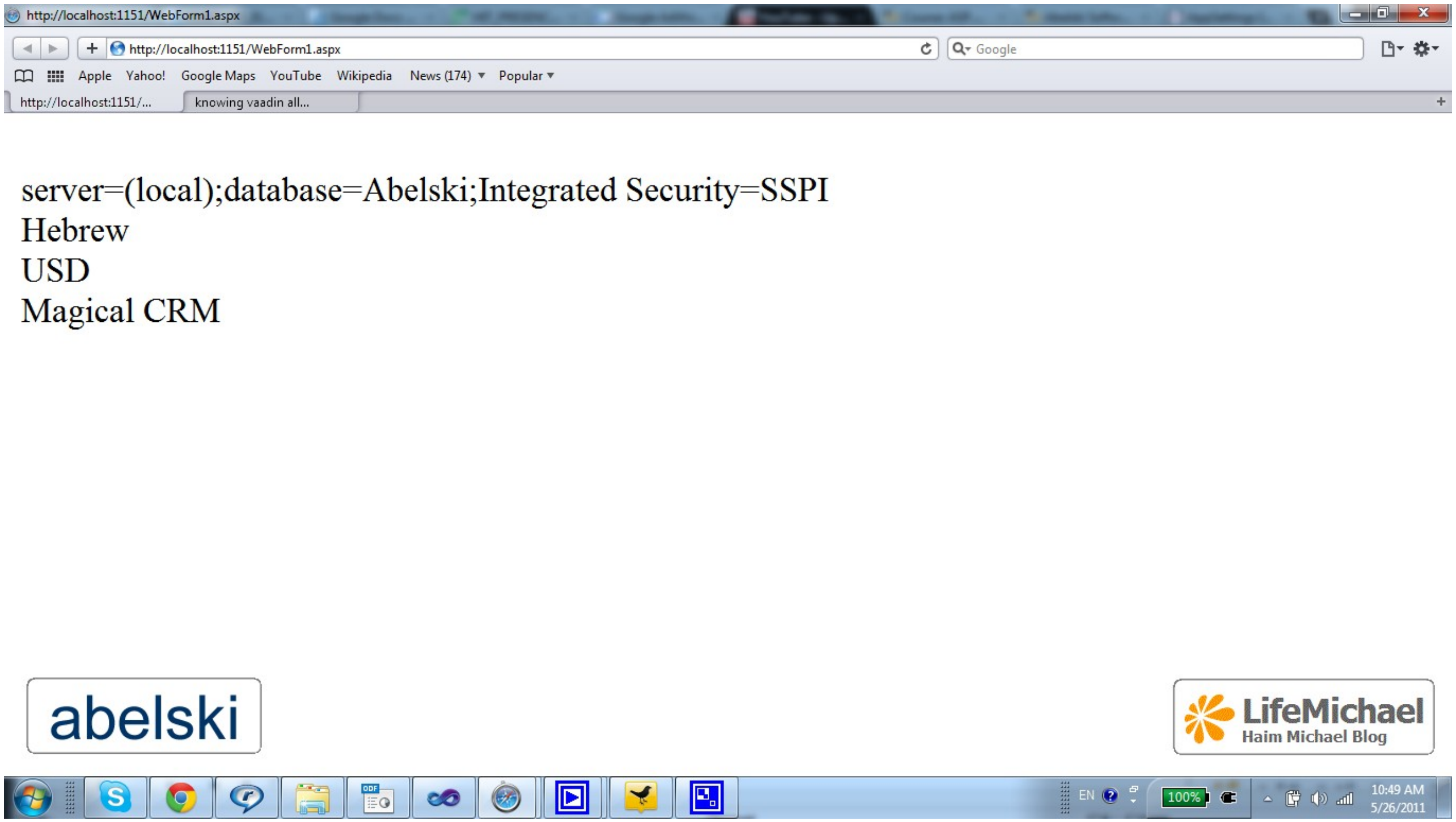
Expression Builder

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm1.aspx.cs"
Inherits="WebApplication47.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>
</head>
<body>
    <form id="form1" runat="server">

    <br/><asp:Literal ID="Literal1" Runat="server"
        Text="<%$ AppSettings:ConnectionInfo %>" />
    <br/><asp:Literal ID="Literal2" Runat="server"
        Text="<%$ AppSettings:Language %>" />
    <br/><asp:Literal ID="Literal3" Runat="server"
        Text="<%$ AppSettings:Currency %>" />
    <br/><asp:Literal ID="Literal4" Runat="server"
        Text="<%$ AppSettings:AppName %>" />
    </form>
</body>
</html>
```

Expression Builder

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

<configuration>

    ...

    <appSettings>
        <add key="ConnectionInfo"
        value="server=(local);database=Abelski;Integrated Security=SSPI" />
        <add key="Language" value="Hebrew" />
        <add key="Currency" value="USD" />
        <add key="AppName" value="Magical CRM" />
    </appSettings>

    ...

</configuration>
```

Expression Builder



```
server=(local);database=Abelski;Integrated Security=SSPI
Hebrew
USD
Magical CRM
```

Expression Builder

- Unlike data binding expressions, the $ expressions must be placed within a control tag.
- The first part of the $ expression indicates the name of the expression builder (e.g. `AppSettingsExpressionBuilder`).
- The `AppSettingsExpressionBuilder` was registered to handle all expressions that begin with AppSettings. The `ConnectionStringsExpressionBuilder` is another expression builder that was registered. It handles information retrieved from the `<connectionString>` section.

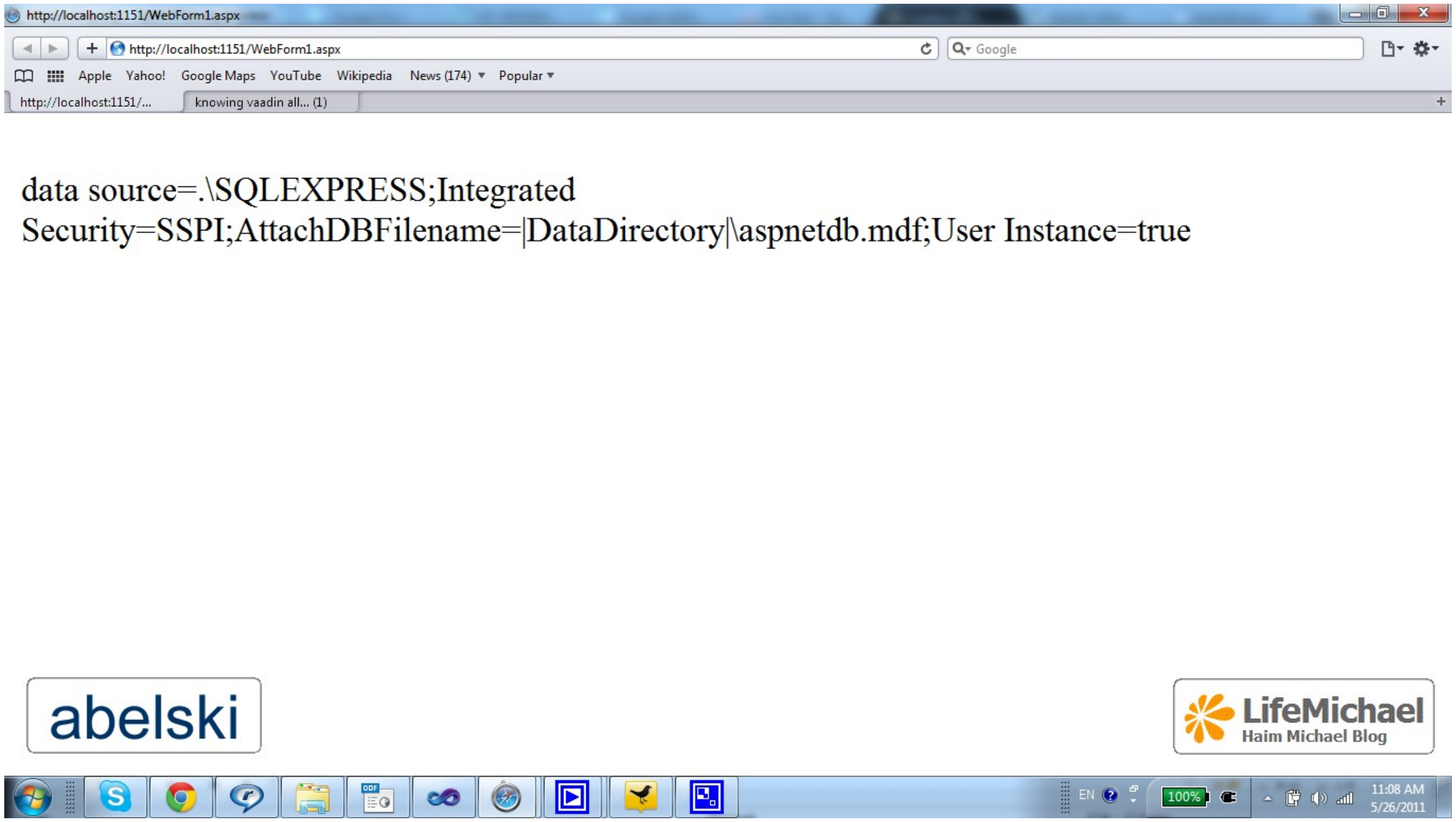
Expression Builder

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm1.aspx.cs"
Inherits="WebApplication47.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>
</head>
<body>
    <form id="form1" runat="server">

    <asp:Literal
            ID="Literal1"
            Runat="server"
            Text="<%$ ConnectionStrings:ApplicationServices %>" />

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

Expression Builder

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

<configuration>
    <connectionStrings>
        <add    name="ApplicationServices"
                connectionString="data source=.\SQLEXPRESS;Integrated
                Security=SSPI;AttachDBFilename=|
                DataDirectory|\aspnetdb.mdf;User Instance=true"
                providerName="System.Data.SqlClient" />
    </connectionStrings>
    ...
</configuration>
```

Expression Builder


http://localhost:1151/WebForm1.aspx
Apple
Yahoo!
Google Maps
YouTube
Wikipedia
News (174)
Popular
http://localhost:1151/...
knowing vaadin all... (1)

data source=.\SQLEXPRESS;Integrated Security=SSPI;AttachDBFilename=|DataDirectory|\aspnetdb.mdf;User Instance=true

Repeated Value Binding

- The repeated value binding allows us to bind an entire list of information to control on our web page. The information is represented by an object that wraps a collection of objects.
- The ASP.NET controls that support repeated value binding include the ones that render themselves using the `<select>` tag (`HtmlSelect`, `ListBox` and `DropDownList`), the `CheckBoxList`, the `RadioButtonList` and the `BulletedList` control.

Repeated Value Binding

- The ASP.NET controls that support repeated value binding have the following properties: `DataSource`, `DataSourceID`, `DataTextField`, `DataTextFormatString` and `DataValueField`.

The `DataSource` Property

- This property holds the reference for the object that holds the collection of data items we want to display.
- This property will usually implement the `ICollection` interface.

The `DataSourceID` Property

- Using this property we can link the list control to a data source control. The data source control generates the required data object automatically.
- We can use either the `DataSource` or the `DataSourceID`. We cannot use both of them.

The `DataTextField` Property

- This property specifies the field (or property in case of an object) of the data item that contains the value we want to display on page.

The `DataTextFormatString` Property

- This property specifies the format string to be use when displaying the data.

The `DataValueField` Property

- This property purpose is similar to the purpose of the `DataTextField` property. Unlike `DataTextField`, when using `DataValueField` the value from the data item won't be displayed in the page. It will be stored in the value attribute of the underlying HTML tag.

Repeated Value Binding

- We can bind the same data source with as many controls as we want.
- Each and every control that supports repeated value data binding includes the `DataBind()` method. We can either call it directly in order to bind the very specific control or call the `Page.DataBind()` method, which will indirectly call the `DataBind()` method on each and every control it containes.

Repeated Value Binding

```
namespace WebApplication47
{
    public partial class WebForm1 : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            if (!Page.IsPostBack)
            {
                Hashtable hashtable = new Hashtable();
                hashtable.Add("key1", "value1");
                hashtable.Add("key2", "value2");
                hashtable.Add("key3", "value3");
                hashtable.Add("key4", "value4");
                hashtable.Add("key5", "value5");
                Select1.DataSource = hashtable;
                Select2.DataSource = hashtable;
                CheckBoxList1.DataSource = hashtable;
                RadioButtonList1.DataSource = hashtable;
                ListBox1.DataSource = hashtable;
                this.DataBind();
            }
        }
    }
}
```

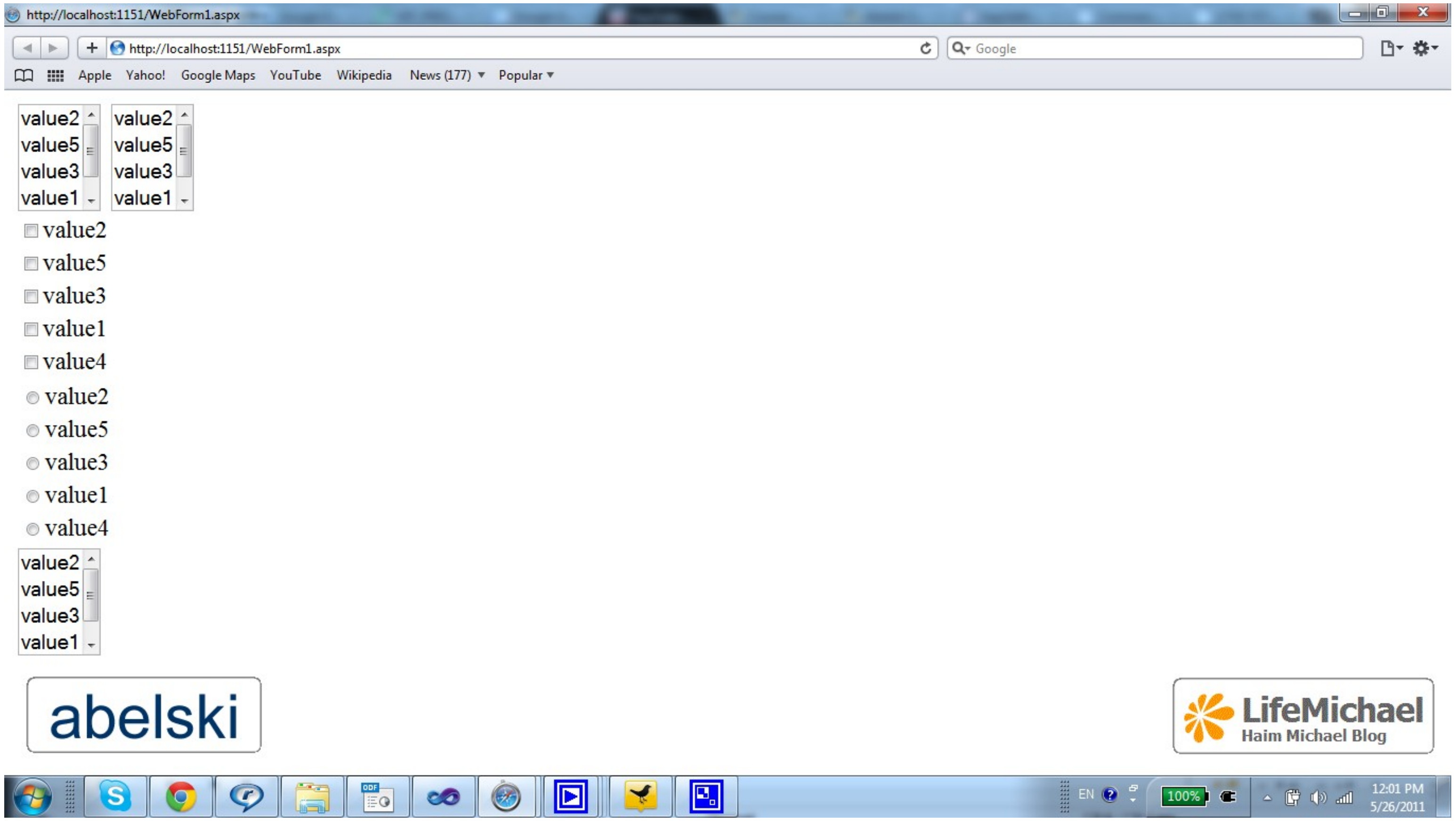
Repeated Value Binding

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm1.aspx.cs"
Inherits="WebApplication47.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>
</head>
<body>
    <form id="form1" runat="server">
    <select runat="server"
        ID="Select1" size="3" DataTextField="Value" DataValueField="Key" />
    <select runat="server"
        ID="Select2" size="3" DataTextField="Value" DataValueField="Key" />
```

Repeated Value Binding

```
    <asp:CheckBoxList runat="server"
        ID="CheckBoxList1" DataTextField="Value" DataValueField="Key" />
    <asp:RadioButtonList runat="server"
        ID="RadioButtonList1" DataTextField="Value" DataValueField="Key" />
    <asp:ListBox runat="server"
        ID="ListBox1" DataTextField="Value" DataValueField="Key" />
    </form>
</body>
</html>
```

Repeated Value Binding


http://localhost:1151/WebForm1.aspx
Google
Apple Yahoo! Google Maps YouTube Wikipedia News (177) Popular

value2 value5 value3 value1 | value2 value5 value3 value1

- [ ] value2
- [ ] value5
- [ ] value3
- [ ] value1
- [ ] value4

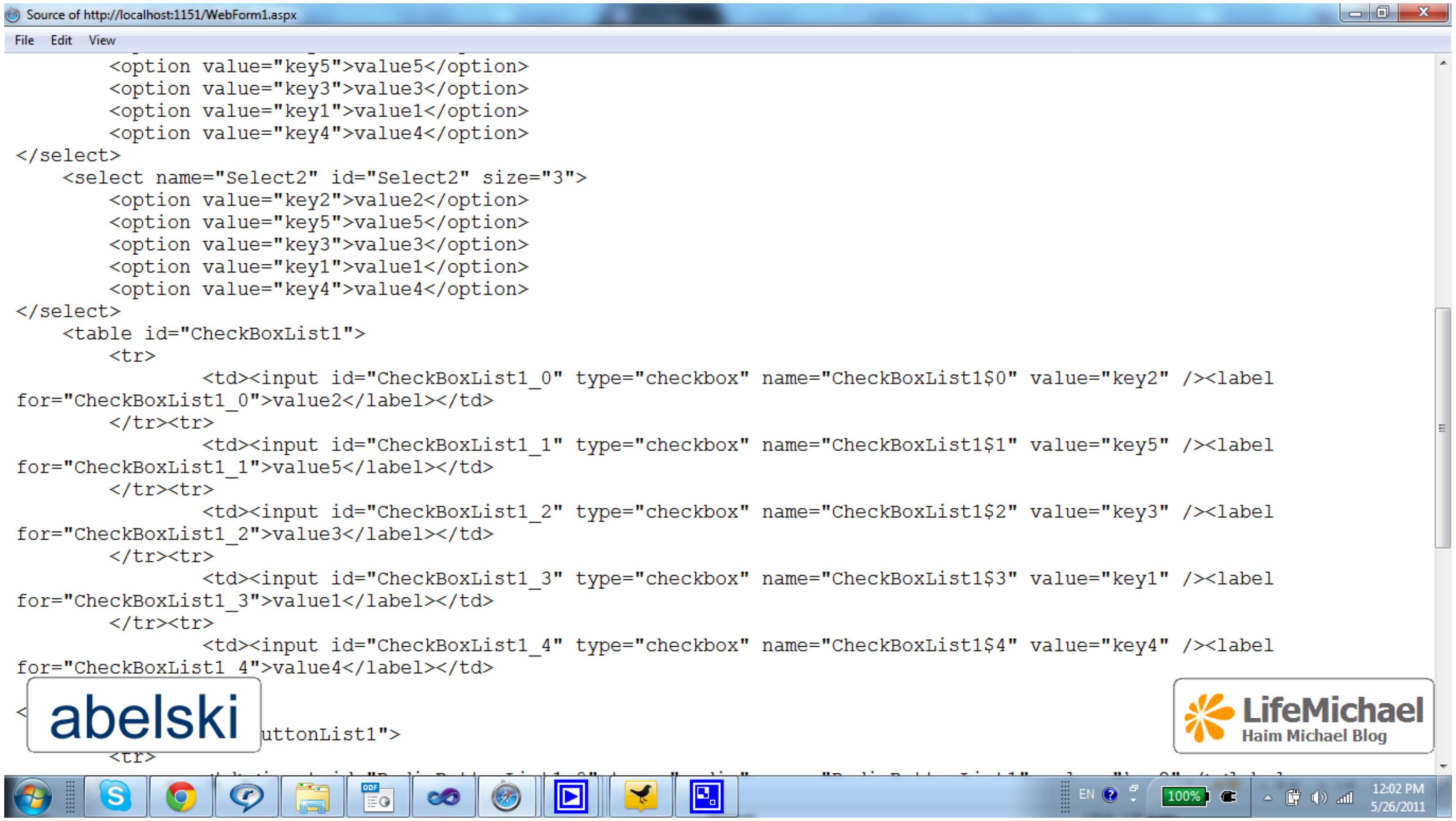
○ value2
○ value5
○ value3
○ value1
○ value4

value2 value5 value3 value1

abelski

LifeMichael Haim Michael Blog

Repeated Value Binding



Source of http://localhost:1151/WebForm1.aspx

File Edit View

```
        <option value="key5">value5</option>
        <option value="key3">value3</option>
        <option value="key1">value1</option>
        <option value="key4">value4</option>
</select>
    <select name="Select2" id="Select2" size="3">
        <option value="key2">value2</option>
        <option value="key5">value5</option>
        <option value="key3">value3</option>
        <option value="key1">value1</option>
        <option value="key4">value4</option>
</select>
    <table id="CheckBoxList1">
        <tr>
                <td><input id="CheckBoxList1_0" type="checkbox" name="CheckBoxList1$0" value="key2" /><label
for="CheckBoxList1_0">value2</label></td>
        </tr><tr>
                <td><input id="CheckBoxList1_1" type="checkbox" name="CheckBoxList1$1" value="key5" /><label
for="CheckBoxList1_1">value5</label></td>
        </tr><tr>
                <td><input id="CheckBoxList1_2" type="checkbox" name="CheckBoxList1$2" value="key3" /><label
for="CheckBoxList1_2">value3</label></td>
        </tr><tr>
                <td><input id="CheckBoxList1_3" type="checkbox" name="CheckBoxList1$3" value="key1" /><label
for="CheckBoxList1_3">value1</label></td>
        </tr><tr>
                <td><input id="CheckBoxList1_4" type="checkbox" name="CheckBoxList1$4" value="key4" /><label
for="CheckBoxList1_4">value4</label></td>
<
                     uttonList1">
        <tr>
```

12:02 PM
5/26/2011

Repeated Value Binding

- We can bind any data structure that implement the `ICollection` interface or one of its derivatives. We can bind all collection classes (such as `Hashtable`, `ArrayList` and `Dictionary`), we can bind the ADO.NET `DataReader` and `DataView` objects and we can bind any other object that was instantiated from a class that implements the `ICollection` interface.

Data Reader Binding

```
namespace WebApplication47
{
    public partial class WebForm1 : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            if (!Page.IsPostBack)
            {
                string connectionString =
                     WebConfigurationManager.ConnectionStrings["abelski"].
                     ConnectionString;
                string sqlStatement = "SELECT id, name FROM courses";
                SqlDataReader reader = null;
                SqlConnection connection = null;
                SqlCommand command = null;
```

Data Reader Binding

```
                try
                {
                    connection = new SqlConnection(connectionString);
                    connection.Open();
                    command = new SqlCommand(sqlStatement, connection);
                    reader = command.ExecuteReader();
                    ListBox1.DataSource = reader;
                    ListBox1.DataBind();
                }
                finally
                {
                    reader.Close();
                    connection.Close();
                }
            }
        }
    }
}
```

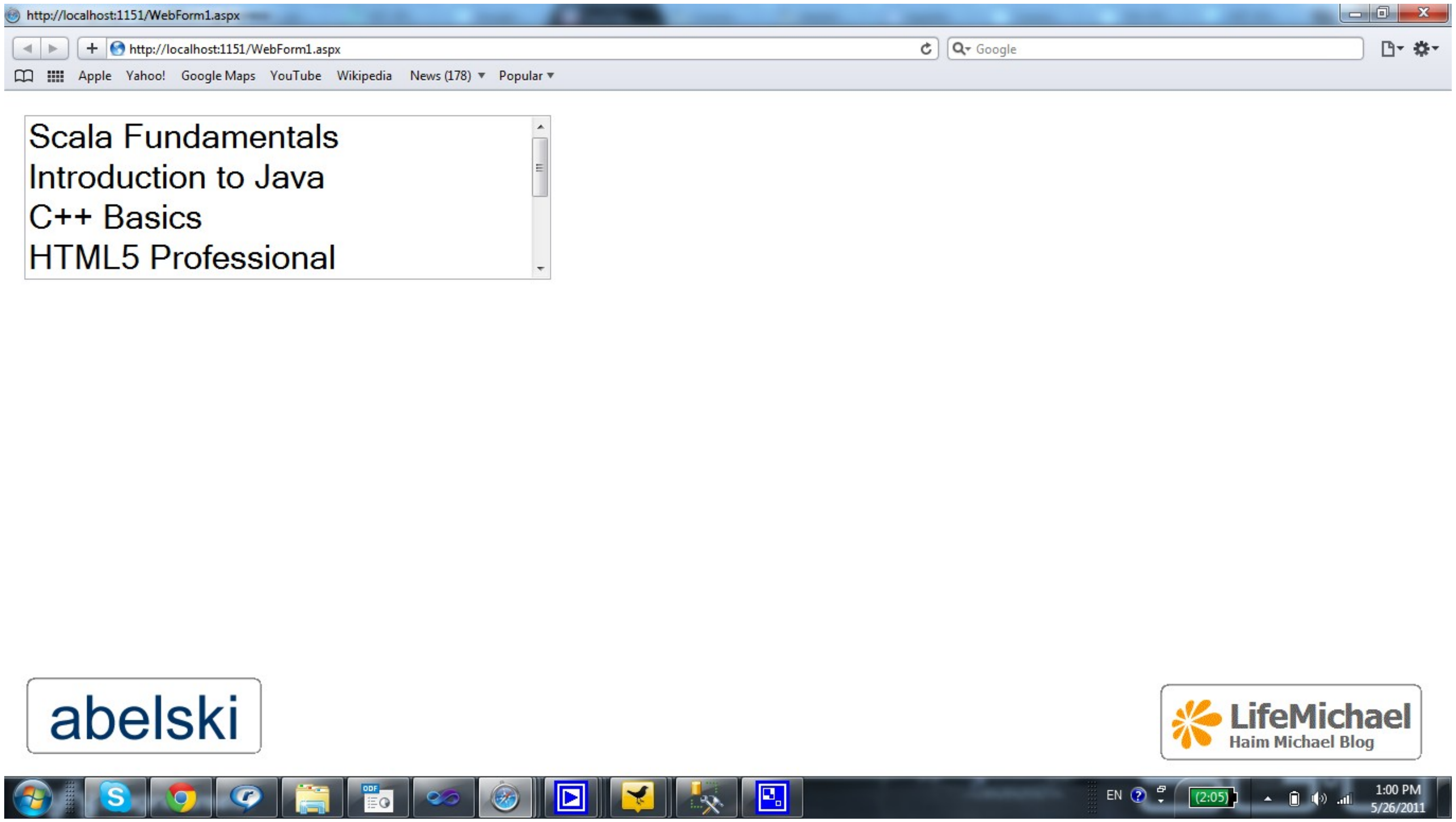
Data Reader Binding

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm1.aspx.cs"
Inherits="WebApplication47.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>
</head>
<body>
    <form id="form1" runat="server">
    <asp:ListBox runat="server" ID="ListBox1" DataTextField="name"
        DataValueField="id" Width="200px" />
    </form>
</body>
</html>
```

Data Reader Binding

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

<configuration>
    <connectionStrings>
    ...
        <add    name="abelski"
                connectionString="Data Source=(local);Initial Catalog=abelski;
                    Integrated Security=True"
                providerName="System.Data.SqlClient" />
    </connectionStrings>
    ...
</configuration>
```

Data Reader Binding



```
Scala Fundamentals
Introduction to Java
C++ Basics
HTML5 Professional
```

abelski

LifeMichael
Haim Michael Blog

Rich Data Controls

- The rich data controls support repeated value binding. Unlike the simple list controls they were designed exclusively for data binding and they allow us to display several properties of fields from each data item. The rich data controls often use a table based layout.
- The rich data controls support higher level features such as editing and they support various events that allow us to hook with.

Rich Data Controls

- The rich data controls include the `GridView`, `DetailsView` and `FormView`.

Rich Data Controls

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm1.aspx.cs"
Inherits="WebApplication47.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>
</head>
<body>
    <form id="form1" runat="server">
    <asp:GridView ID="GridView1" runat="server" AutoGenerateColumns="true" />
    </form>
</body>
</html>
```

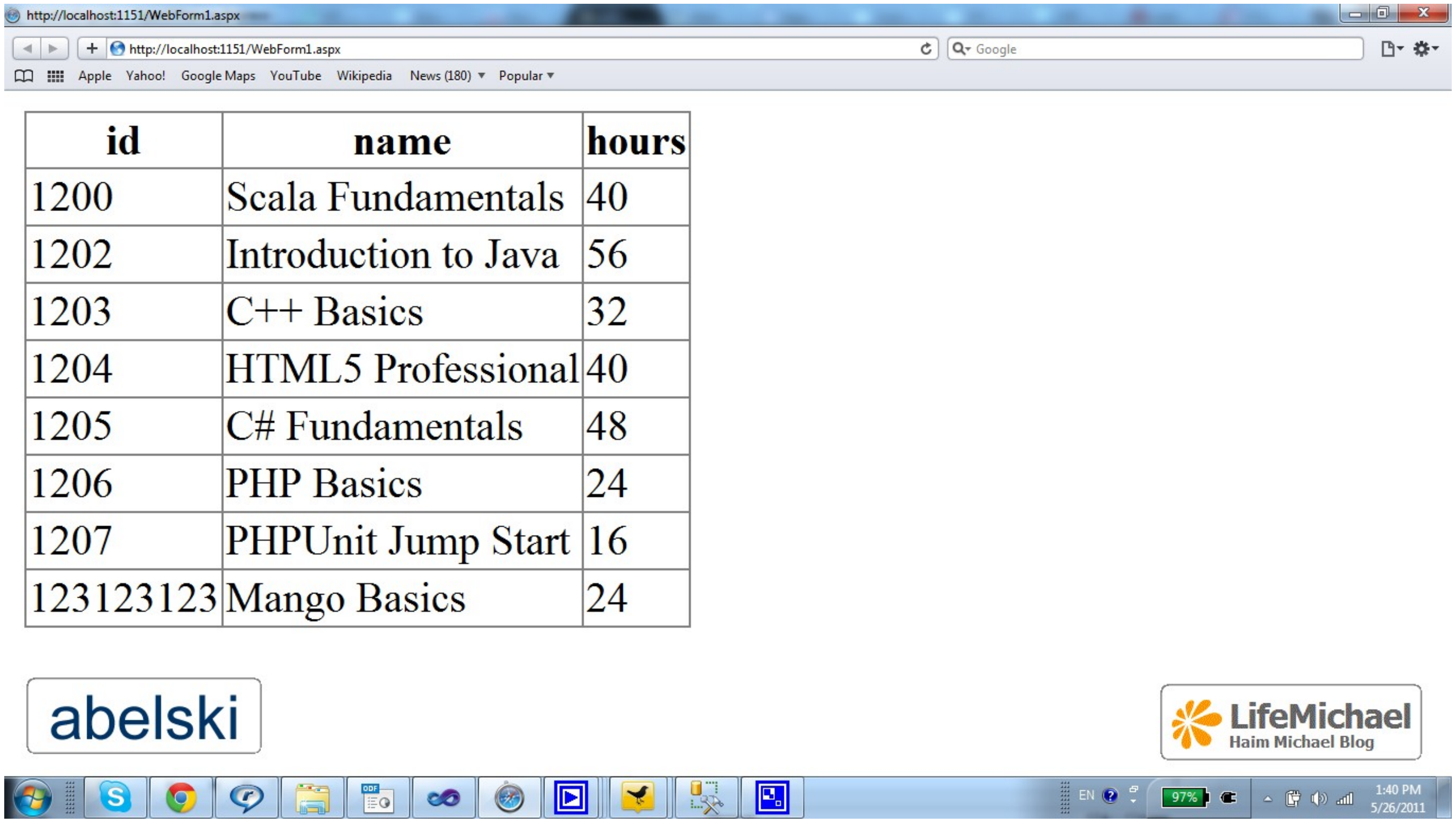
Rich Data Controls

```
namespace WebApplication47
{
    public partial class WebForm1 : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            if (!Page.IsPostBack)
            {
                string connectionString = WebConfigurationManager.
                      ConnectionStrings["abelski"].ConnectionString;
                string sqlStatement = "SELECT id, name, hours FROM courses";
                SqlDataReader reader = null;
                SqlConnection connection = null;
                SqlCommand command = null;
                try
                {
                    connection = new SqlConnection(connectionString);
                    connection.Open();
                    command = new SqlCommand(sqlStatement, connection);
```

Rich Data Controls

```
                                    reader = command.ExecuteReader();
                                    GridView1.DataSource = reader;
                                    GridView1.DataBind();
                                }
                                finally
                                {
                                    reader.Close();
                                    connection.Close();
                                }
                        }
                   }
              }
         }
```

Rich Data Controls



http://localhost:1151/WebForm1.aspx

id	name	hours
1200	Scala Fundamentals	40
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
123123123	Mango Basics	24

Data Source Controls

- The data source controls implement the IDataSource interface. The .NET framework include the following data source controls: `SqlDataSource`, `ObjectDataSource`, `AccessDataSource`, `XmlDataSource` and `SiteMapDataSource`.
- The data source controls can retrieve data from their source and provide it to the controls they are linked with and they can update the data source when the user edits the linked control.

Data Source Controls

- The data source control life cycle includes the following phases:
  - (1) The page object is created.
  - (2) The page life cycle starts. Page.Init() and Page.Load() are fired.
  - (3) The events of the controls (apart of the data source controls) are fired.
  - (4) The data source controls perform the required updated (deletes, inserts and updates).
  - (5) The Page.PreRender even is fired.
  - (6) The data source controls perform the required queries and insert the new retrieved data into the linked controls. The linked controls Selecting and Selected events are fired.
  - (7) The page is rendered back to the user and the Page object is disposed.