Building a dynamic website using ASP.NET in visual studion

By
Rebin Birzo SALEH
2017

ASP ?

- ASP stands for **A**ctive **S**erver **P**ages.
- ASP.NET is a web application framework developed by Microsoft to allow programmers to build dynamic web sites.
- An ASP file can contain text, HTML tags and scripts. Scripts in an ASP file are executed on the server.
- ASP is a Microsoft Technology that runs inside **IIS.**
- **IIS** is the web server created by Microsoft for use with Windows NT family.
- To run IIS you must have Windows NT 4.0 or later.
- ChiliASP and InstantASP are two technology's which runs ASP without Windows.

History

- After four years of development, and a series of beta releases in 2000 and 2001, ASP.NET 1.0 was released on January 5, 2002 as part of version 1.0 of the **.NET Framework**.
- ASP.NET is a new ASP generation.
- ASP.NET is the successor to **Microsoft's Active Server Pages (ASP) technology**. ASP.NET is built on the Common Language Runtime (CLR), allowing programmers to write ASP.NET code using any supported .NET language.

ASP.NET Versions

ASP.NET Version	Introduced with .NET & IDE
4.5.1	4.5.1 and Visual Studio 2013
4.5	4.5 and Visual Studio 2012
4.0	4.0 and Visual Studio 2010
3.5	3.5 and Visual Studio 2008
2.0	2.0 and Visual Studio 2005
1.1	1.1 and Visual Studio .NET 2003
1.0	1.0 and Visual Studio .NET

Compilers

- ASP.NET Intellisense Generator
- Microsoft Visual Studio
- Microsoft Visual Web Developer Express
- Microsoft SharePoint Designer
- MonoDevelop
- SharpDevelop
- Adobe Dreamweaver
- CodeGear Delphi

What can ASP do for you?

- Websites that require user requests to be processed at server side can be developed using asp.net.
- Access any data or databases and return the results to a browser.
- To build an Internet application that supports adding, editing, deleting, and listing of information stored in a database.
- Customize a Web page to make it more useful for individual users.
- Applications such as:
  - ✓Hotel Reservation web application
  - ✓Super market Billing System etc.

ASP.NET Models

- ASP.NET supports three different development models:
- Web Pages:
  - Web Pages is the easiest development model for developing ASP.NET web sites.
- MVC (Model View Controller):
  - MVC is a model for building web applications using a MVC (Model View Controller) design.
- Web Forms:
  - Web Forms is the traditional ASP.NET model, based on event driven Web Forms and post backs.

Code-behind model

- It encourages developers to build applications with separation of presentation and content in mind.
- In theory, this would allow a web designer, for example, to focus on the design markup with less potential for disturbing the programming code that drives it.
- This is similar to the separation of the controller from the view in Model–View–Controller (MVC) frameworks.
- Using "code behind" separates the presentation logic from UI visualization.

Program Structure

- ASP.NET pages have the extension .aspx, and are normally written in VB (Visual Basic) or C# (C sharp).
- Razor is a new and simple markup syntax for embedding server code into ASP.NET web pages.

Data Types **and Data Types**

You don't have to specify a type for a variable.

Most of the time, ASP.NET can figure out the type based on how the data in the variable is being used.

```
// Assigning a string to a variable.
var greeting = "Welcome!";
// Assigning a number to a variable.
var theCount = 3;
// Assigning an expression to a variable.
var monthlyTotal = theCount + 5;
// Assigning a date value to a variable.
var today = DateTime.Today;
// Declaring variables using explicit data types.
string name = "Joe";
int count = 5;
DateTime tomorrow = DateTime.Now.AddDays(1);
```

Razor Syntax Rules for C#

- Razor code blocks are enclosed in @{ ... }
- Inline expressions (variables and functions) start with @
- Code statements end with semicolon
- Variables are declared with the var keyword
- Strings are enclosed with quotation marks
- C# code is case sensitive
- C# files have the extension .cshtml

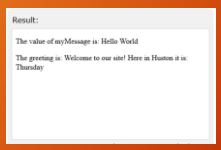
C# Code

```
<html>
<body>
<!-- Single statement block -->
@{ var myMessage = "Hello World"; }

<!-- Inline expression or variable -->
<p>The value of myMessage is: @myMessage</p>

<!-- Multi-statement block -->
@{
var greeting = "Welcome to our site!";
var weekDay = DateTime.Now.DayOfWeek;
var greetingMessage = greeting + " Today is: " + weekDay; }
<p>The greeting is: @greetingMessage</p>
</body>
</html>
```

Output



Razor Syntax Rules for VB

- Razor code blocks are enclosed in @Code … End Code
- Inline expressions (variables and functions) start with @
- Variables are declared with the Dim keyword
- Strings are enclosed with quotation marks
- VB code is not case sensitive
- VB files have the extension .vbhtml

VB Code

```
<html>
<body>
<!-- Single statement block -->
@Code
dim myMessage = "Hello World"
End Code

<!-- Inline expression or variable -->
<p>The value of myMessage is: @myMessage</p>

<!-- Multi-statement block -->
@Code
dim greeting = "Welcome to our site!"
dim weekDay = DateTime.Now.DayOfWeek
dim greetingMessage = greeting & " Today is: " & weekDay
End Code

<p>The greeting is: @greetingMessage</p>
</body>
</html>
```

Expressions, Assignment Statements

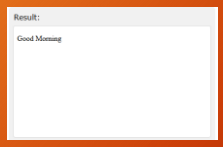
- Expressions
- @(5 + 13) @{ var netWorth = 150000; }
- @{ var newTotal = netWorth * 2; }
- @(newTotal / 2)

- Assignment Statements
- var age = 17;

Conditional Statements

```
@ {
var txt = "";
if(DateTime.Now.Hour > 12)
  {txt = "Good Evening";}
else
  {txt = "Good Morning";}
}
<html>
<body>
<p>The message is @txt</p>
</body>
</html>
```

Output



Objects, Methods

- "Date" object is a typical built-in ASP.NET object.
- Objects can also be self-defined.
- Examples: a web page, a text box, a file, a database record, etc.
- Objects may have methods they can perform.
- Examples: A database record might have a "Save" method, an image object might have a "Rotate" method, an email object might have a "Send" method, and so on.
- Objects also have properties that describe their characteristics.
- Examples: A database record might have a FirstName and a LastName property (amongst others).

Example:

```
<table border="1">
<tr>
<th width="100px">Name</th>
<td width="100px">Value</td>
</tr>
<tr>
<td>Day</td><td>@DateTime.Now.Day</td>
</tr>
<tr>
<td>Hour</td><td>@DateTime.Now.Hour</td>
</tr>
<tr>
<td>Minute</td><td>@DateTime.Now.Minute</td>
</tr>
<tr>
<td>Second</td><td>@DateTime.Now.Second</td>
</tr>
</td>
</table>
```

Output

Result:

Name	Value
Day	20
Hour	11
Minute	39
Second	56

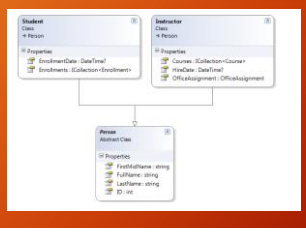
Inheritance

- All managed languages in the .NET Framework, such as Visual Basic and C#, provide full support for object-oriented programming including encapsulation, inheritance, and polymorphism.
- Inheritance describes the ability to create new classes based on an existing class.

```
public class A
{
    public A() {{  }
}

public class B : A
{
    public B()   {  }{ }
}
}
```

Inheritance Example



Encapsulation

- Encapsulation means that a group of related properties, methods, and other members are treated as a single unit or object.
- Encapsulation is implemented by using **access specifiers**.
- An **access specifier** defines the scope and visibility of a class member.
- C# supports the following access specifiers:

Access Modifier	Description (who can access)
private	Only members within the same type. (default for type members)
protected	Only derived types or members of the same type.
internal	Only code within the same assembly. Can also be code external to object as long as it is in the same assembly. (default for types)
protected internal	Either code from derived type or code in the same assembly. Combination of protected OR internal.
public	Any code. No inheritance, external type, or external assembly restrictions.

Example

```
using System;
class BankAccountPublic
{
    public decimal GetAmount()
    {
        return 1000.00m;
    }
}
```

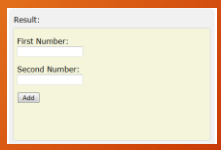
The GetAmount() method is public meaning that it can be called by code that is external to this class. elsewhere in your program, to use the method.

```
BankAccountPublic bankAcctPub = new BankAccountPublic();
// call a public method
decimal amount = bankAcctPub.GetAmount();
```

Add Two Numbers

```
@{
    var totalMessage = "";
    if(IsPost)
        {
        var num1 = Request["text1"];
        var num2 = Request["text2"];
        var total = num1.AsInt() + num2.AsInt();
        totalMessage = "Total = " + total;
    } }
}
<!DOCTYPE html>
<html>
<body style="background-color: beige; font-family: Verdana, Arial;">
<form action="" method="post">
<p><label for="text1">First Number:</label><br>
<input type="text" name="text1"></p>
<p><label for="text2">Second Number:</label><br>
<input type="text" name="text2"></p>
<p><input type="submit" value="Add"></p>
</form>
<p>@totalMessage</p>
</body>
</html>
```

Output



Resources

- http://www.w3schools.com
- http://www.dotnet-tricks.com/Tutorial/aspnet/3JEV171213-A-brief-version-history-of-ASP.NET.html
- http://forums.asp.net
- http://www.asp.net/web-pages/overview/getting-started/introducing-razor-syntax-%28c%29

THANK YOU !