5.1 Introducing ASP.NET

ASP.NET is a web development platform, which provides a programming model, a comprehensive software infrastructure and various services required to build up robust web applications for PC, as well as mobile devices.

ASP.NET works on top of the HTTP protocol, and uses the HTTP commands and policies to set a browser-to-server bilateral communication and cooperation.

ASP.NET is a part of Microsoft .Net platform. ASP.NET applications are compiled codes, written using the extensible and reusable components or objects present in .Net framework. These codes can use the entire hierarchy of classes in .Net framework.

The ASP.NET application codes can be written in any of the following languages:

- C#
- Visual Basic.Net
- Jscript
- J#

ASP.NET is used to produce interactive, data-driven web applications over the internet. It consists of a large number of controls such as text boxes, buttons, and labels for assembling, configuring, and manipulating code to create HTML pages.

ASP.NET Web Forms Model

ASP.NET web forms extend the event-driven model of interaction to the web applications. The browser submits a web form to the web server and the server returns a full markup page or HTML page in response.

All client side user activities are forwarded to the server for stateful processing. The server processes the output of the client actions and triggers the reactions.

Now, HTTP is a stateless protocol. ASP.NET framework helps in storing the information regarding the state of the application, which consists of:

- Page state
- Session state

The page state is the state of the client, i.e., the content of various input fields in the web form. The session state is the collective information obtained from various pages the user visited and worked with, i.e., the overall session state. To clear the concept, let us take an example of a shopping cart.

User adds items to a shopping cart. Items are selected from a page, say the items page, and the total collected items and price are shown on a different page, say the cart page. Only HTTP cannot keep track of all the information coming from various pages. ASP.NET session state and server side infrastructure keeps track of the information collected globally over a session.

The ASP.NET runtime carries the page state to and from the server across page requests while generating ASP.NET runtime codes, and incorporates the state of the server side components in hidden fields.

This way, the server becomes aware of the overall application state and operates in a two-tiered connected way.

The ASP.NET Component Model

The ASP.NET component model provides various building blocks of ASP.NET pages. Basically it is an object model, which describes:

- Server side counterparts of almost all HTML elements or tags, such as <form> and <input>.
- Server controls, which help in developing complex user-interface. For example, the Calendar control or the Gridview control.

ASP.NET is a technology, which works on the .Net framework that contains all web-related functionalities. The .Net framework is made of an object-oriented hierarchy. An ASP.NET web application is made of pages. When a user requests an ASP.NET page, the IIS delegates the processing of the page to the ASP.NET runtime system.

The ASP.NET runtime transforms the .aspx page into an instance of a class, which inherits from the base class page of the .Net framework. Therefore, each ASP.NET page is an object and all its components i.e., the server-side controls are also objects.

Components of .Net Framework 3.5

Let us go through at the various components of the .Net framework 3.5. The following table describes the components of the .Net framework 3.5 and the job they perform:

Components and their Description
(1) Common Language Runtime or CLR It performs memory management, exception handling, debugging, security checking, thread execution, code execution, code safety, verification, and compilation. The code that is directly managed by the CLR is called the managed code. When the managed code is compiled, the compiler converts the source code into a CPU independent intermediate language (IL) code. A Just In Time(JIT) compiler compiles the IL code into native code, which is CPU specific.
(2) .Net Framework Class Library It contains a huge library of reusable types. classes, interfaces, structures, and enumerated values, which are collectively called types.
(3) Common Language Specification It contains the specifications for the .Net supported languages and implementation of language integration.
(4) Common Type System It provides guidelines for declaring, using, and managing types at runtime, and cross-language communication.
(5) Metadata and Assemblies Metadata is the binary information describing the program, which is either stored in a portable executable file (PE) or in the memory. Assembly is a logical unit consisting of the assembly manifest, type metadata, IL code, and a set of resources like image files.
(6) Windows Forms Windows Forms contain the graphical representation of any window displayed in the

application.

(7) ASP.NET and ASP.NET AJAX

ASP.NET is the web development model and AJAX is an extension of ASP.NET for developing and implementing AJAX functionality. ASP.NET AJAX contains the components that allow the developer to update data on a website without a complete reload of the page.

(8) ADO.NET

It is the technology used for working with data and databases. It provides access to data sources like SQL server, OLE DB, XML etc. The ADO.NET allows connection to data sources for retrieving, manipulating, and updating data.

(9) Windows Workflow Foundation (WF)

It helps in building workflow-based applications in Windows. It contains activities, workflow runtime, workflow designer, and a rules engine.

(10) Windows Presentation Foundation

It provides a separation between the user interface and the business logic. It helps in developing visually stunning interfaces using documents, media, two and three dimensional graphics, animations, and more.

(11) Windows Communication Foundation (WCF)

It is the technology used for building and executing connected systems.

(12) Windows CardSpace

It provides safety for accessing resources and sharing personal information on the internet.

(13) LINQ

It imparts data querying capabilities to .Net languages using a syntax which is similar to the tradition query language SQL.

5.2 ASP.NET namespaces

1) The System.Web namespace

System.web namespace holds some basic ingredients which includes classes and built-in Objects like

a. Server
b. Application
c. Request
d. Response

And Classes used for managing

a. Cookies
b. Configuring page caching
c. Tracing Implementation
d. Retrieving Information of Web Server and client browser

We uses built-in object frequently .This namespace are very important for ASP.net application, it is also required for User Interface , Web forms and Web services.

2) The System.Web.services namespace

The System.Web.services namespace is a initial point for creating the web services. It contains web service classes , which allow to create XML web services using Asp.Net . XML web services provides feature to exchange messages in loosely coupled environment. using protocol like SOAP , HTTP , XML .
Some classes are as follows

a. WebMethodAttribute
b. Webservice
c. WebServiceAttribute
d. WebServiceBindingAttribute

3) The System.web.UI.WebControls namespace

The System.web.UI.WebControls namespace contains classes that provides to create web server controls on web pages. These controls run on the server so we can programmaticaly control elements . It also include form controls like text boxes and buttons. special purpose controls such as calender , and Data Grid . Web controls are more abstract than HTML controls . They also provide a shophisticated formatting properties and events . The System.web.UI.WebControls namespace contains web control class which is the base class for all web controls

Some classes are as follows

a. Calendar
b. Check Box
c. Button
d. Base Data Bound Control
e. Data Control Field etc

4) The System.web.UI namespace

The System.web.UI namespace includes classes that allows to create server controls with data-binding functionality , which has ability to save view state of given control and pages (.aspx pages) . Many of these types allows to support for controls System.web.UI .Html controls. It is a base class for all HTML ,Web, and User Controls. Every aspx pages comes under it. Control classes provides common set of functionslity

There are following controls available

- HTML server controls
- Web server controls
- User controls

Some classes are as follows

a. Attribute Collection
b. BaseParser
c. BaseTemplateParser
d. AsyncPostBackTrigger etc

5) The System.web.sessionstate namespace

Session state management comes under The System.web.sessionstate namespace . That enable storage of data specific to a single client with in a web application on the server. Session state is ideal for sensitive data such as mailing address , credit card number , password , important numbers etc. Sesssion state can be used with clients that do not support cookies.

Some classes are as follows

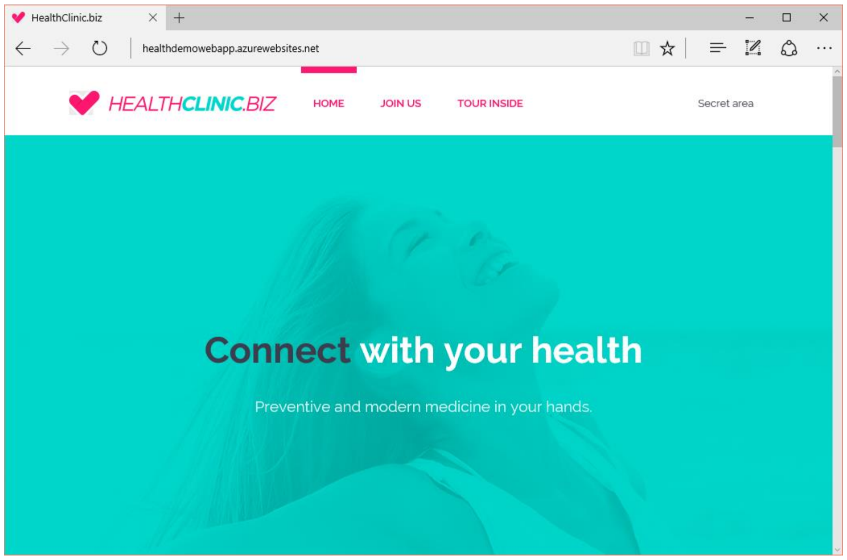
a. HttpSessionState
b. HttpSessionStateContainer
c. SessionIDManager
d. SessionStateModule etc

5.3 Creating and deploying ASP.NET applications

Azure App Service is the only cloud service that integrates everything you need to quickly and easily build web and mobile apps for any platform and any device. Built for developers, App Service is a fully managed platform that has powerful capabilities such as built-in DevOps, continuous integration with Visual Studio Team Services and GitHub, staging and production support, and automatic patching.

This walkthrough shows how to deploy an ASP.NET web application to a web app in Azure App Service by using Visual Studio 2015 or Visual Studio 2013. The walkthrough assumes that you are an ASP.NET developer who has no prior experience with Azure. After you complete the tutorial, you’ll have a web app that’s deployed to Azure from Visual Studio. Samples and tutorials in this article are based on the Connect(); //2015 event demos.

The following illustration shows the completed application:



This article will help you learn the following:

- How to prepare your computer for Azure development by installing the Azure SDK for .NET.
- How to set up Visual Studio to create a new web app in Azure App Service and an ASP.NET MVC 5 web project.
- How to compose the Demo project that is presented in Microsoft Connect(); //2015.
- How to deploy a web app project to App Service by using Visual Studio.
- How to use the Azure portal to monitor and manage your web app.

Sign up for Microsoft Azure

You need an Azure account to complete this tutorial. You can:

- Open an Azure account for free. You get credits that you can use to try paid Azure services. Even after the credits are used up, you can keep the account and use free Azure services and features, such as the Web Apps feature of Azure App Service.
- Activate Visual Studio subscriber benefits. Your Visual Studio subscription gives you credits every month that you can use for paid Azure services.
- Get credits every month by joining to Visual Studio Dev Essentials.

If you want to get started with Azure App Service before you sign up for an Azure account, go to Try App Service. There, you can immediately create a short-lived starter web app in App Service without a credit card or commitments.

Set up the development environment

To start, set up your development environment by installing the latest version of the Azure SDK.

Visual Studio 2015▸

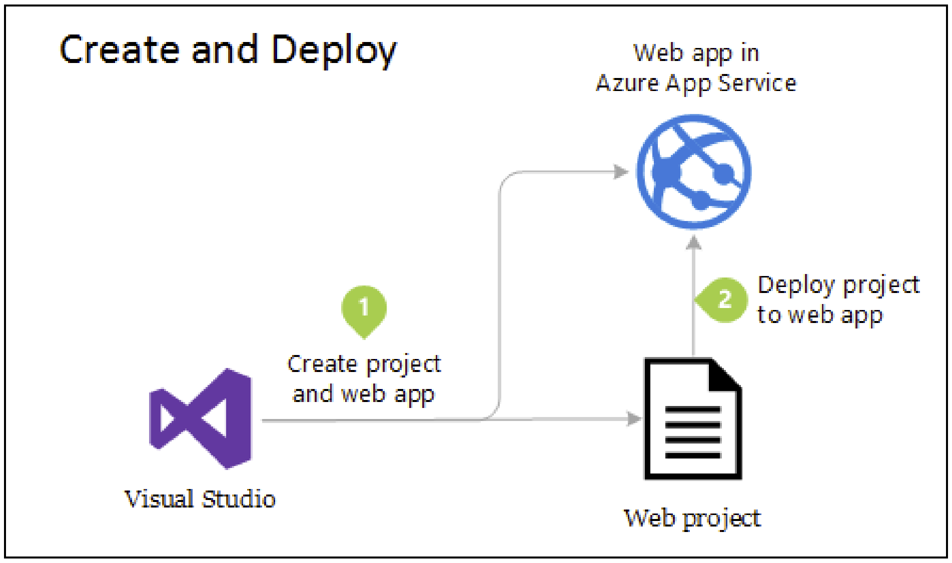
Visual Studio 2013▸

If you don't have Visual Studio installed, use the link for Visual Studio 2015, and Visual Studio will be installed along with the SDK.

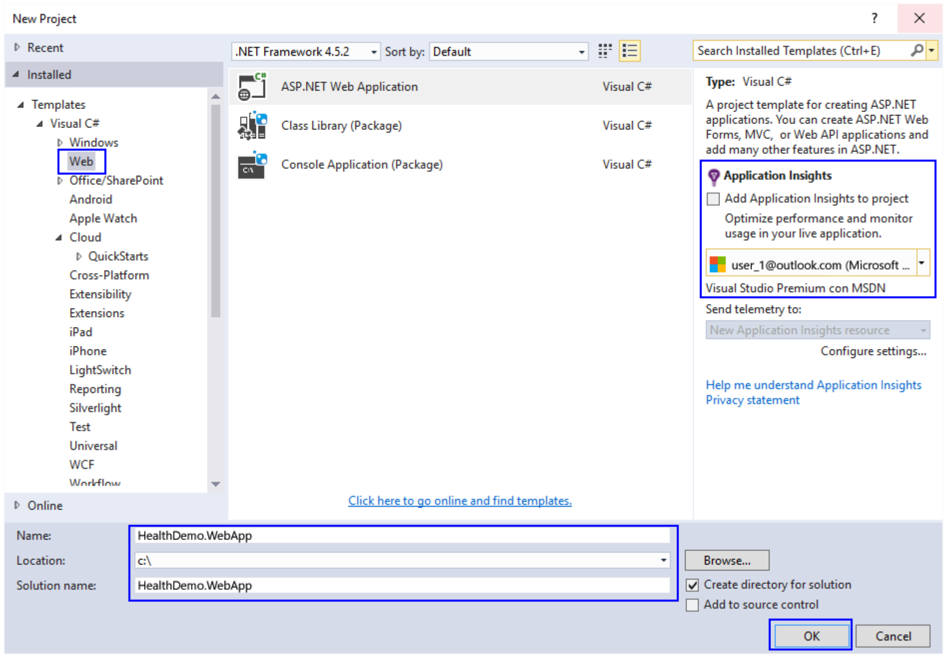
Create a new project and a web app

Your first step is to create a web project in Visual Studio and a web app in Azure App Service. When that's done, you'll deploy the project to the web app to make it available on the Internet.

The diagram below illustrates what you're doing in the create and deploy steps.


Create and Deploy
Web app in Azure App Service
1
Create project and web app
2
Deploy project to web app
Visual Studio
Web project

1. Open Visual Studio 2015 or Visual Studio 2013.
2. On the **File** menu, click **New** > **Project**.
3. In the **New Project** dialog box, click **C#** > **Web** > **ASP.NET Web Application**.
4. Choose a good name for the project, for example, HealthDemo.WebApp.
5. You can deactivate or activate **Application Insights** if you want your application to be monitored in terms of availability and performance by using Azure Application Insights.
6. Click **OK**.


New Project

Type: Visual C#

A project template for creating ASP.NET applications. You can create ASP.NET Web Forms, MVC, or Web API applications and add many other features in ASP.NET.

Application Insights

Add Application Insights to project

Optimize performance and monitor usage in your live application.

Name: HealthDemo.WebApp

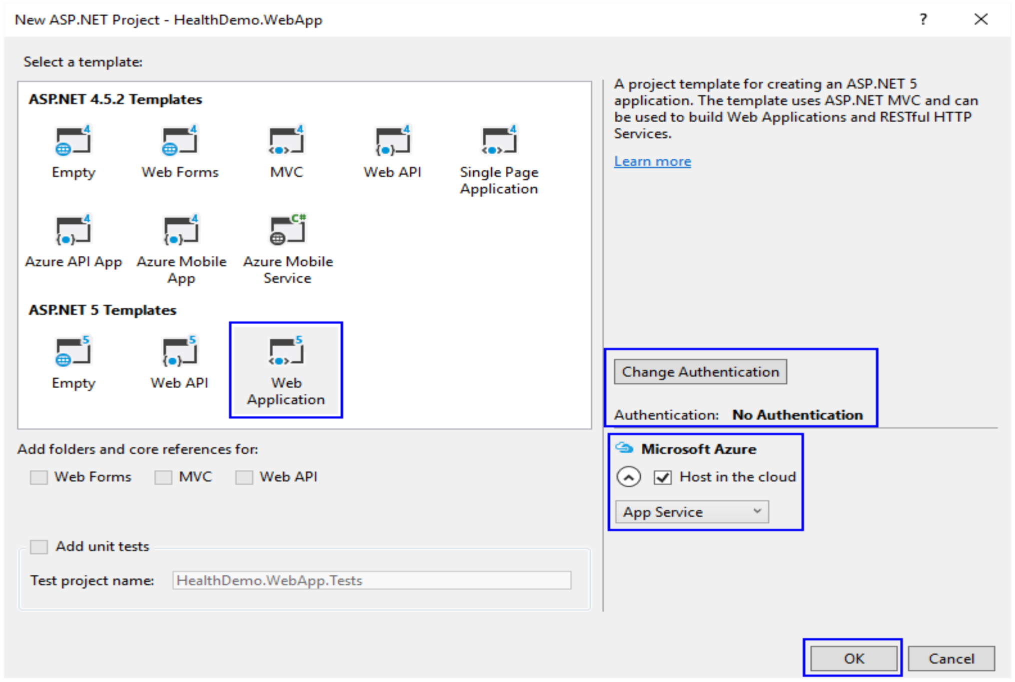
Location: c:\

Solution name: HealthDemo.WebApp

Create directory for solution

{width="6.5in" height="4.510416666666667in"}

1. In the **Select a template** window, select **ASP.NET 5 Web Application**. On the other side, indicate that you want to host the **Web App** in **Azure**. Finally, indicate that you don't want authentication for the application.


New ASP.NET Project - HealthDemo.WebApp

Select a template:

A project template for creating an ASP.NET 5 application. The template uses ASP.NET MVC and can be used to build Web Applications and RESTful HTTP Services.

Authentication: No Authentication

Microsoft Azure

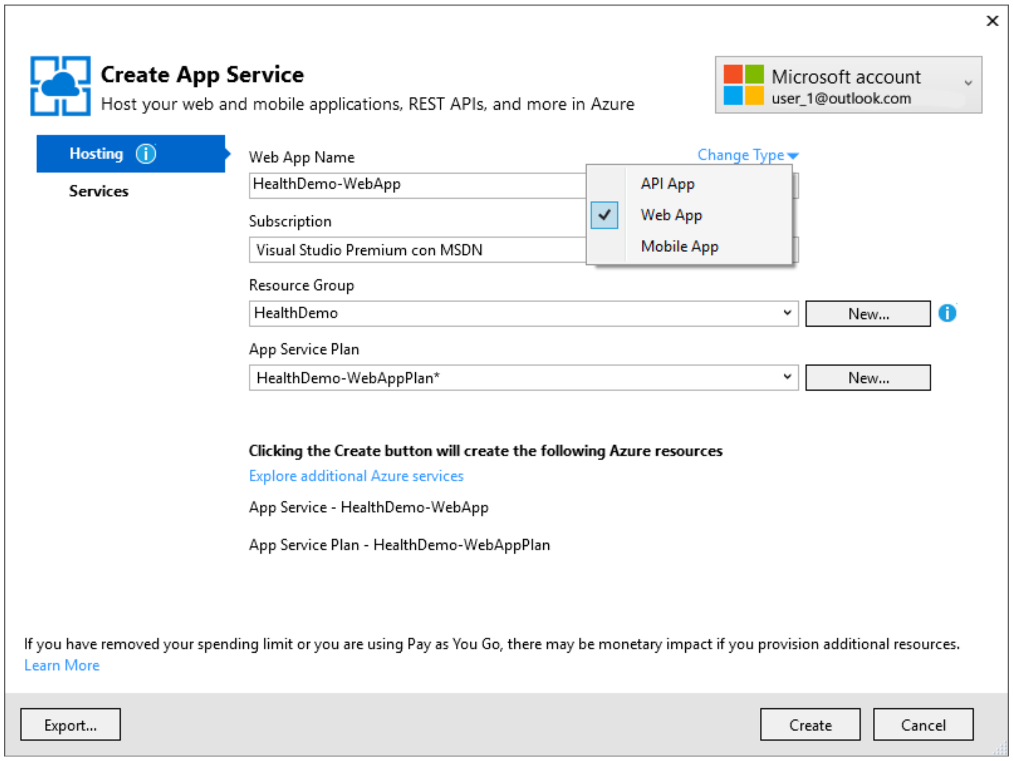
Host in the cloud

App Service

Test project name: HealthDemo.WebApp.Tests

{width="6.489583333333333in" height="5.0625in"}

1. When you select Azure as the host, a new window will automatically open to indicate that the resources that will be created in Azure, which will host all application resources. Fill in the required information and make sure to indicate the type of deployment as **Web App**.


Create App Service
Host your web and mobile applications, REST APIs, and more in Azure

Microsoft account
user_1@outlook.com

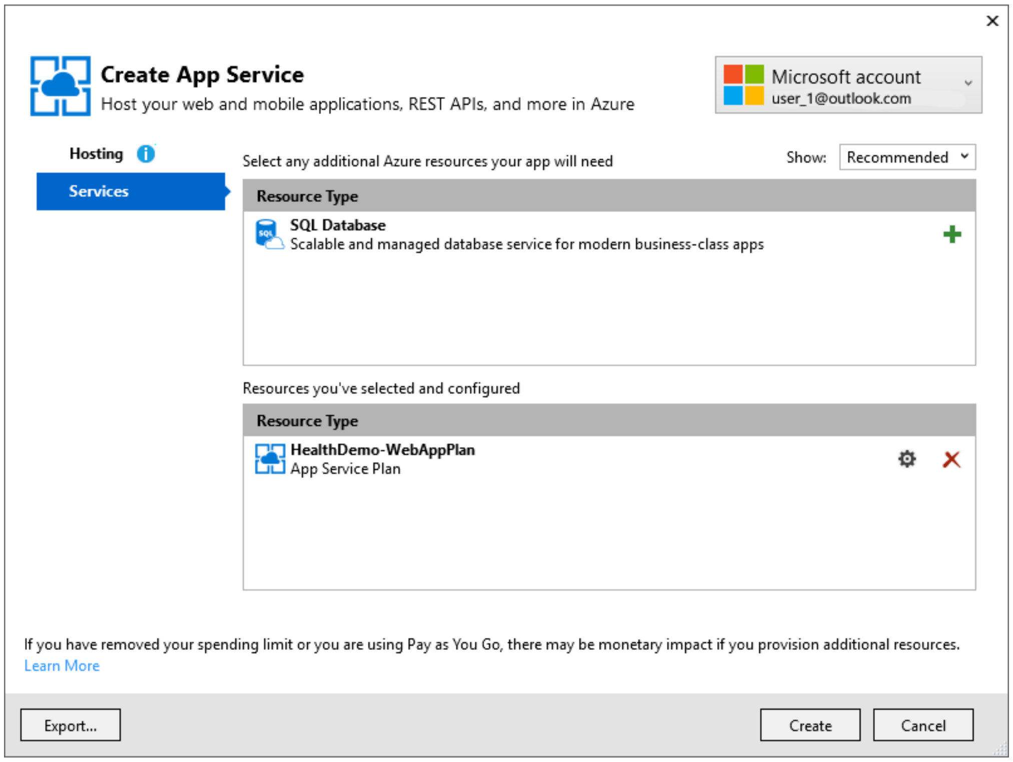
Web App Name: HealthDemo-WebApp
Subscription: Visual Studio Premium con MSDN
Resource Group: HealthDemo
App Service Plan: HealthDemo-WebAppPlan*

Clicking the Create button will create the following Azure resources
Explore additional Azure services
App Service - HealthDemo-WebApp
App Service Plan - HealthDemo-WebAppPlan

If you have removed your spending limit or you are using Pay as You Go, there may be monetary impact if you provision additional resources.
Learn More

{width="6.489583333333333in" height="4.864583333333333in"}

1. Because your web application will have a SQL database in the future, you can add one as a service in the configuration window.


Create App Service
Host your web and mobile applications, REST APIs, and more in Azure

Microsoft account
user_1@outlook.com

Select any additional Azure resources your app will need

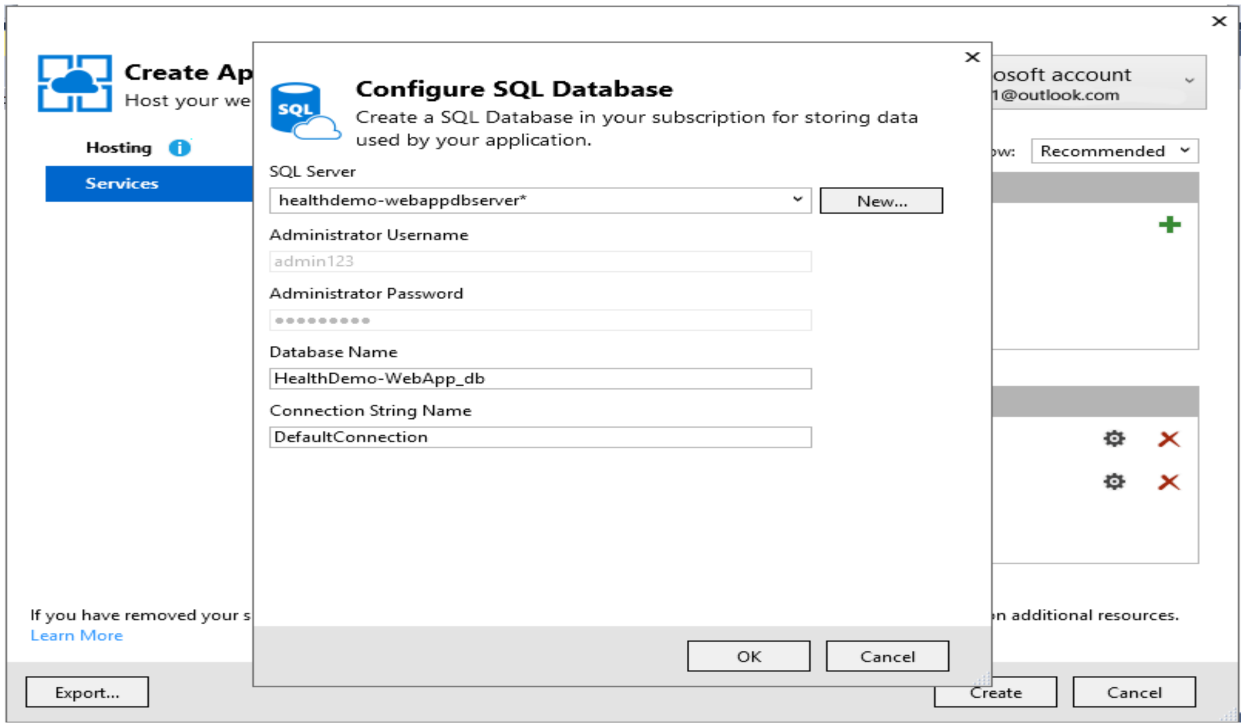
Resource Type
SQL Database Scalable and managed database service for modern business-class apps

Resources you've selected and configured

Resource Type
HealthDemo-WebAppPlan App Service Plan

If you have removed your spending limit or you are using Pay as You Go, there may be monetary impact if you provision additional resources.
Learn More

{width="6.489583333333333in" height="4.864583333333333in"}



Configure SQL Database

Create a SQL Database in your subscription for storing data used by your application.

SQL Server: healthdemo-webappdbserver*   New...

Administrator Username: admin123

Administrator Password: ●●●●●●●●●

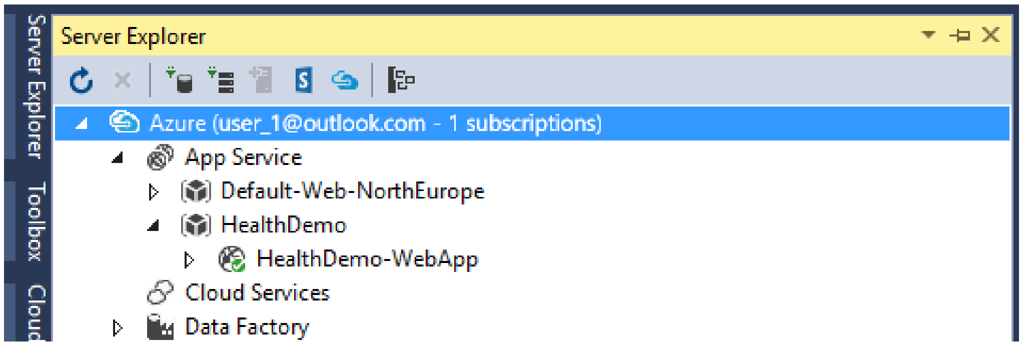
Database Name: HealthDemo-WebApp_db

Connection String Name: DefaultConnection

OK   Cancel

{width="6.489583333333333in" height="4.864583333333333in"}

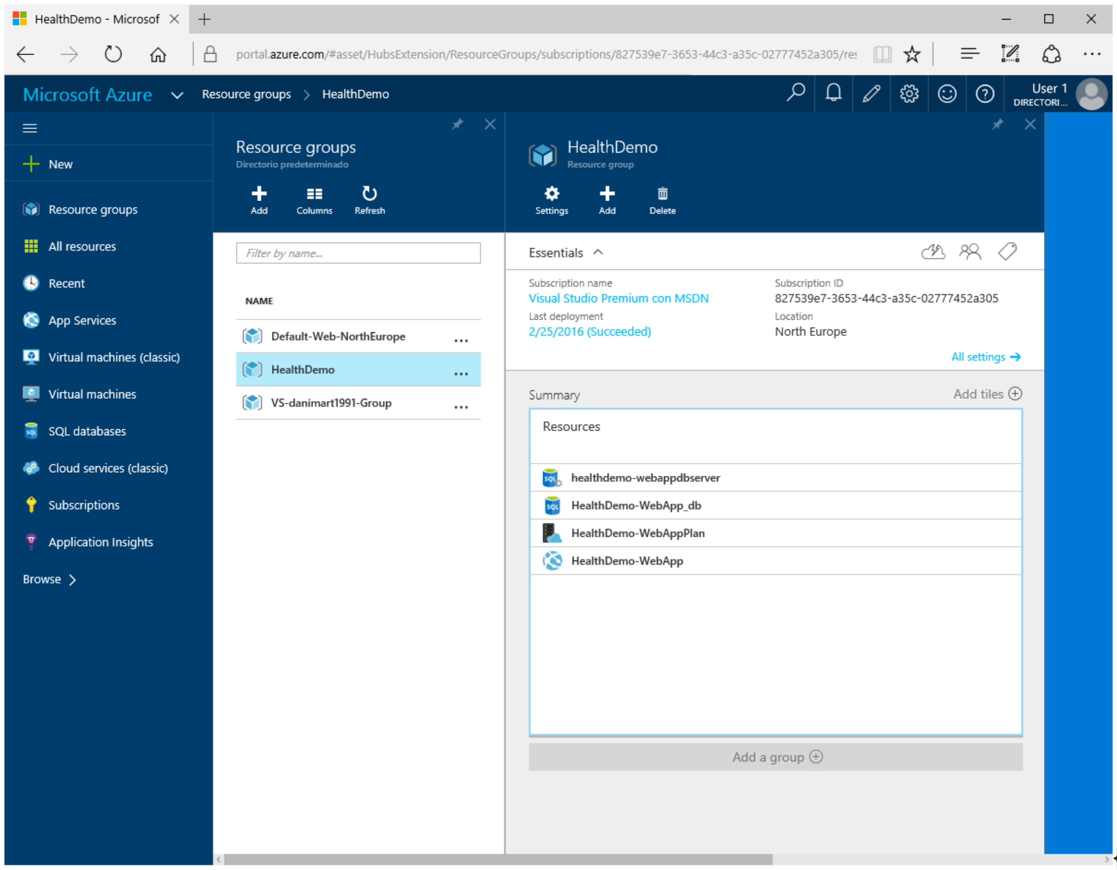
1. When you click the **OK** button, Azure will begin to create the web app in Azure. After it is deployed, you can see all the information in the Azure portal and in Server Explorer in Visual Studio.



Server Explorer

- Azure (user_1@outlook.com - 1 subscriptions)
  - App Service
    - Default-Web-NorthEurope
    - HealthDemo
      - HealthDemo-WebApp
  - Cloud Services
  - Data Factory

{width="5.614583333333333in" height="1.8645833333333333in"}



Subscription name: Visual Studio Premium con MSDN
Subscription ID: 827539e7-3653-44c3-a35c-02777452a305
Last deployment: 2/25/2016 (Succeeded)
Location: North Europe

Resources:
- healthdemo-webappdbserver
- HealthDemo-WebApp_db
- HealthDemo-WebAppPlan
- HealthDemo-WebApp

{width="6.5in" height="5.041666666666667in"}

Azure Resource Manager templates

You can use an Azure Resource Manager template to provision the services that the applications need. In a single template, you can deploy multiple services along with their dependencies. In that way, you can use the same template to repeatedly deploy your application in different environments during every stage of the application lifecycle.

Azure Quickstart Templates are popular deployments from the community that you can adapt to the needs of the developed application.

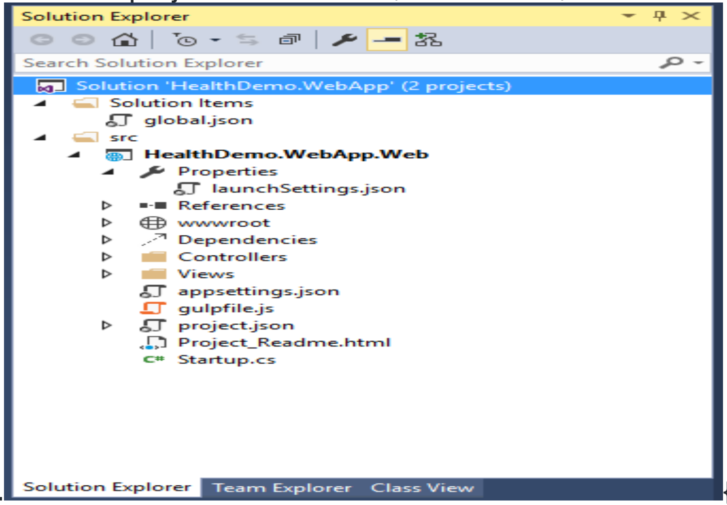
Build the Connect(); //2015 Demo web app

In this part of the tutorial, you learn how to develop the ASP.NET 5 Demo shown at Connect(); //2015. In this walkthrough, we show you how to create the skeleton and the structure of the solution. The completed project is in the following location:

Download the Demo solution, which is the result of this walkthrough, to see the structure and organization.

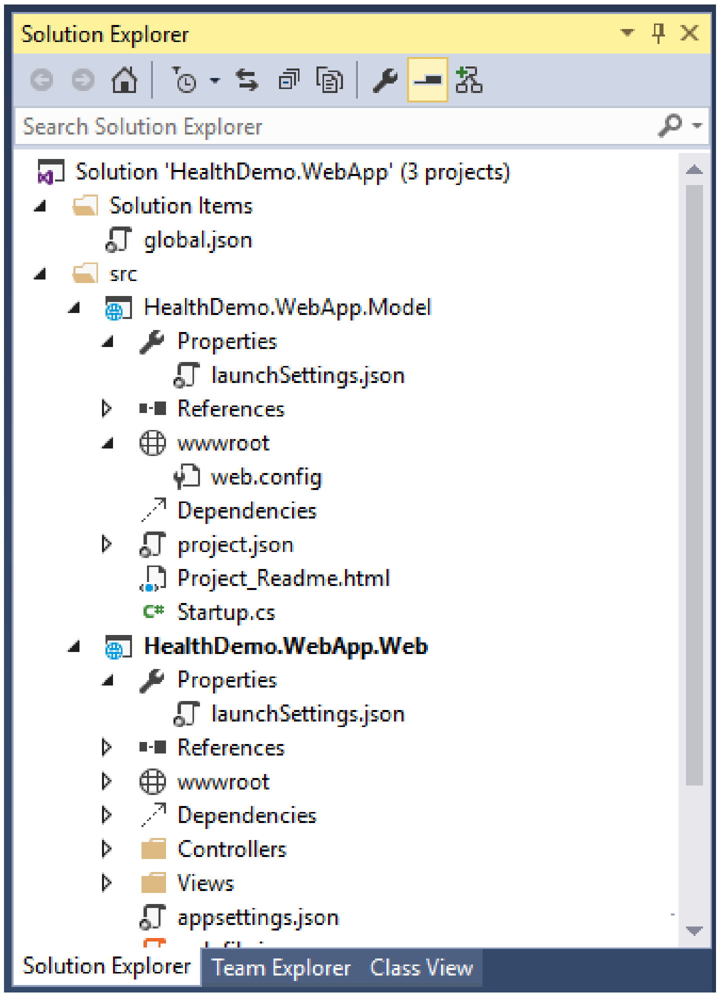
HealthDemo.WebApp.Web

Use the solution that you created in the previous section of this article, and rename the project as HealthDemo.WebApp.Web. The project contains the website that users see, the client-side code of the web application, and the definition of the view.

Make sure that you define this as a startup project and a project that will be deployed in Azure. This project will contain references to other projects including the one that's included in the solution. You will use the projects to add data, information, and context to the web application.{width="3.96875in" height="5.458333333333333in"}

HealthDemo.WebApp.Model

The following steps help you add an ASP.NET 5 empty project. You should call the project HealthDemo.WebApp.Model. The project contains the model files of the application. Model files are a list of classes that define the data model of the web application that the other projects will use.

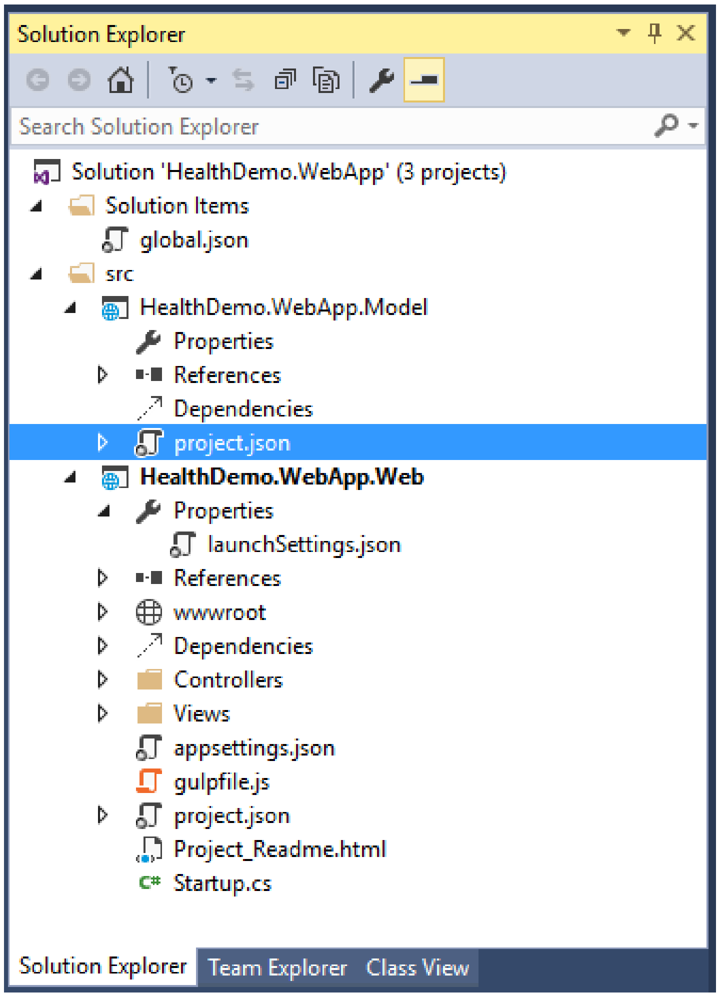


```
Solution 'HealthDemo.WebApp' (3 projects)
  Solution Items
    global.json
  src
    HealthDemo.WebApp.Model
      Properties
        launchSettings.json
      References
      wwwroot
        web.config
      Dependencies
      project.json
      Project_Readme.html
      Startup.cs
    HealthDemo.WebApp.Web
      Properties
        launchSettings.json
      References
      wwwroot
      Dependencies
      Controllers
      Views
      appsettings.json
```

{width="3.9375in" height="5.46875in"}

The following files that will be created must be deleted because they aren’t necessary or useful:

- File Properties > launchSettings.json.
- Node wwwroot.
- File Project_Readme.html
- File Startup.cs



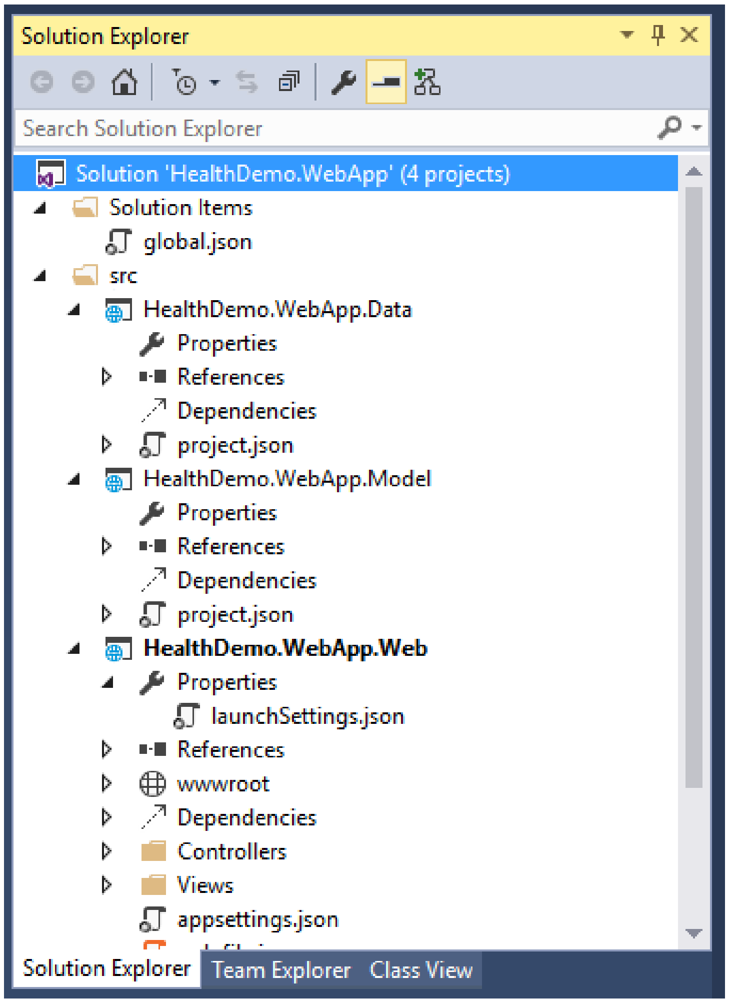
```
Solution 'HealthDemo.WebApp' (3 projects)
  Solution Items
    global.json
  src
    HealthDemo.WebApp.Model
      Properties
      References
      Dependencies
      project.json
    HealthDemo.WebApp.Web
      Properties
        launchSettings.json
      References
      wwwroot
      Dependencies
      Controllers
      Views
      appsettings.json
      gulpfile.js
      project.json
      Project_Readme.html
      Startup.cs
```

{width="3.9375in" height="5.46875in"}

Although you could create a Class Library project type , we recommend an ASP.NET project type to contain all references, dependencies, and components that a web project needs.

HealthDemo.WebApp.Data

Create an empty ASP.NET 5 project named HealthDemo.WebApp.Data. Like the previous project, you need to delete the same files and leave an empty project. This project will contain repositories, initial application data, and the infrastructure that is used in the database to store the information about the web application.



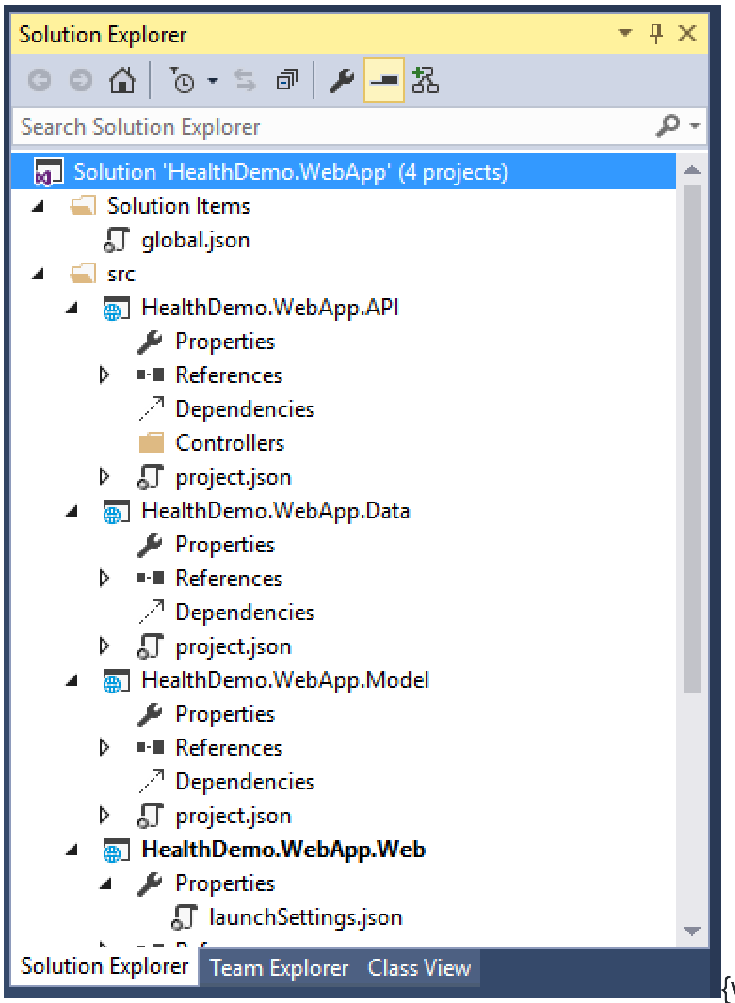
{width="3.9895833333333335in" height="5.5in"}

The MyHealthContext file in the project contains the database context as well as two directories:

- **Infrastructure**: Contains the data infrastructure that the web application uses, the data initialization, and the sample images that are used in the application itself.
- **Repositories**: Contains the database repositories that the web application uses. You can find the repositories that access the data in the database and that will be used by the controllers of the application.

HealthDemo.WebApp.API

Finally, create an ASP.NET 5 empty project named HealthDemo.WebApp.API. Delete the files that are indicated in the previous steps so that the final structure of the solution looks like the following illustration:



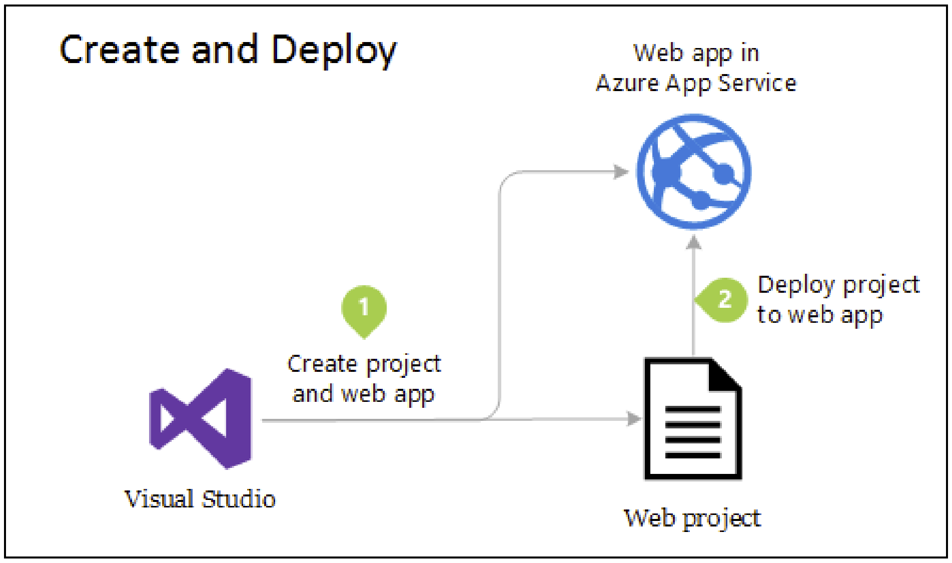
{width="3.96875in" height="5.5in"}

This project will contain the Web Application API. This is the web service that will provide the data to the web application. The apps have the controllers that have the calls that the application needs to access. The project also uses the repositories that are located in the Data project and the data model that's in the Model project.

From this point, you have the basic structure of the web application and the availability to deploy it in Azure. Details about the content of each of the projects can be consulted in the repository of the demo shown in Microsoft Connect(); //2015.

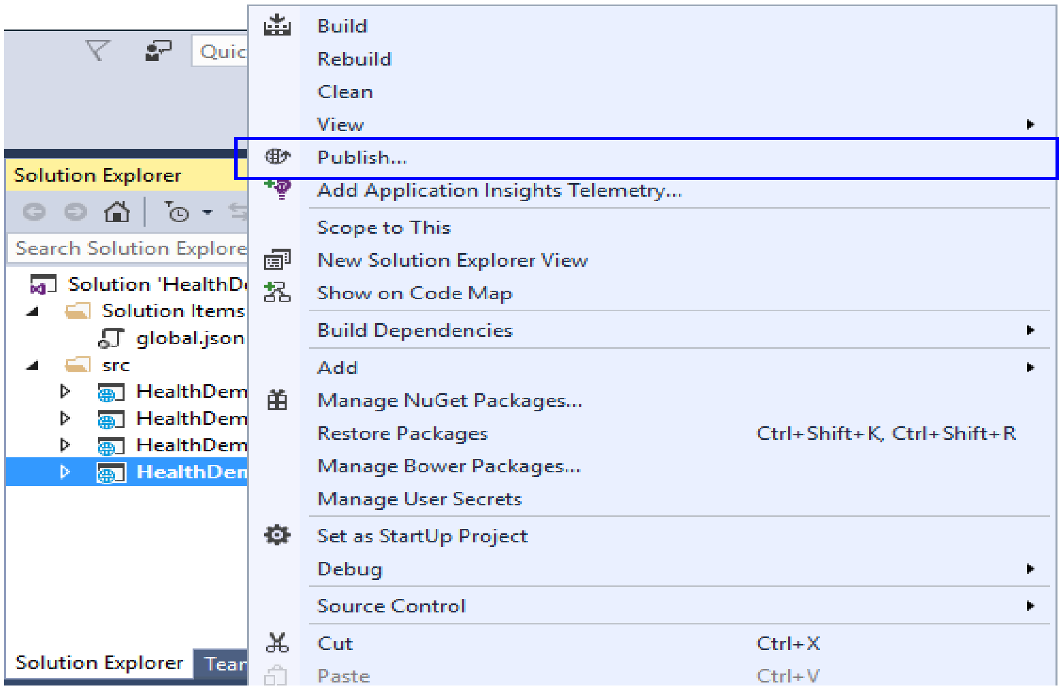
Deploy the project to a web app in Azure

To have the web application in an App Service in Azure, just follow a few steps.

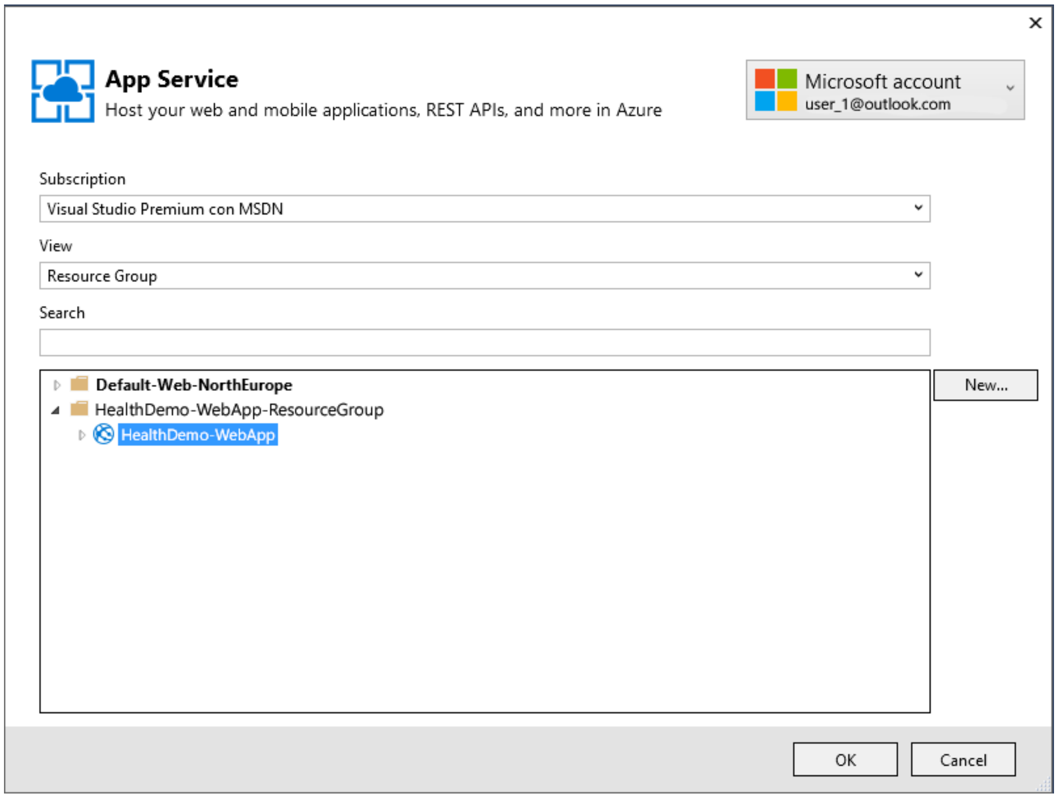

Create and Deploy
Web app in Azure App Service
1
Create project and web app
2
Deploy project to web app
Visual Studio
Web project

With the previous steps, you already generated the Azure resources that can help create more deployments.

1. In Solution Explorer, right-click the HealthDemo.WebApp.Web project, and then click **Publish**.



1. You will see the **Publish Web** dialog box. The wizard shows you a list of available **Publish**profiles. If you select **Microsoft Azure App Service**, you can see a list of available Azure subscriptions and the resource groups that were previously created. Among them is the resource group that you used to deploy the web application that you configured in the previous steps.



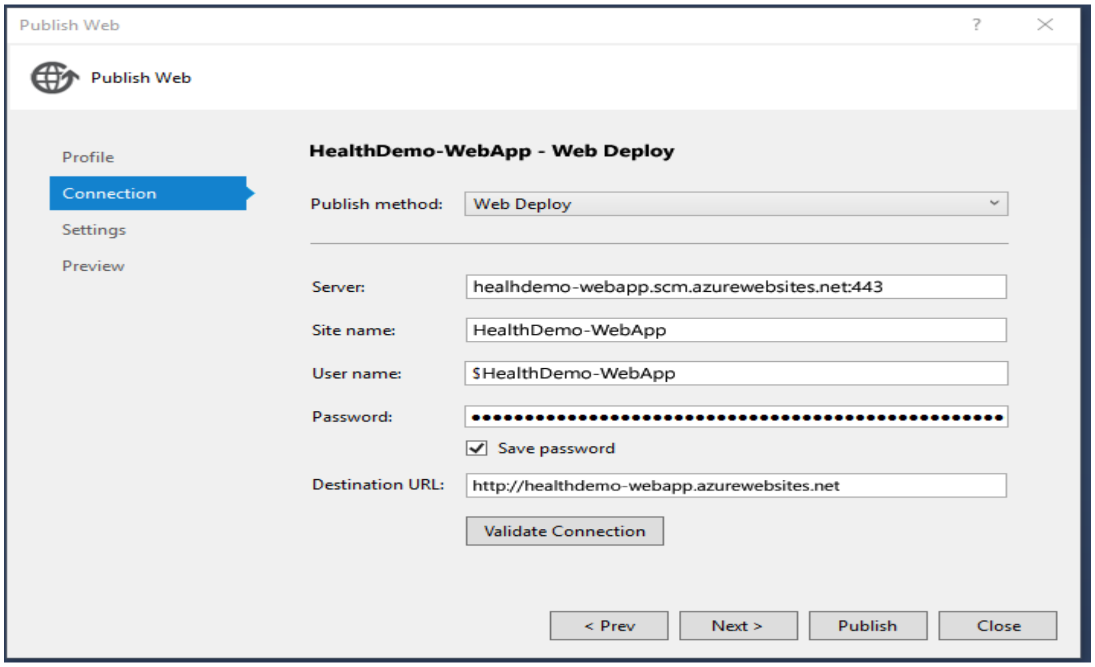
App Service
Host your web and mobile applications, REST APIs, and more in Azure

Microsoft account
user_1@outlook.com

Subscription: Visual Studio Premium con MSDN
View: Resource Group
Search:

- Default-Web-NorthEurope
- HealthDemo-WebApp-ResourceGroup
  - HealthDemo-WebApp

1. After you select the resource group, the page that opens shows the connection information. The default parameters will populate the fields. You can modify the fields if necessary. To check that the connection works correctly, click **Validate Connection**.

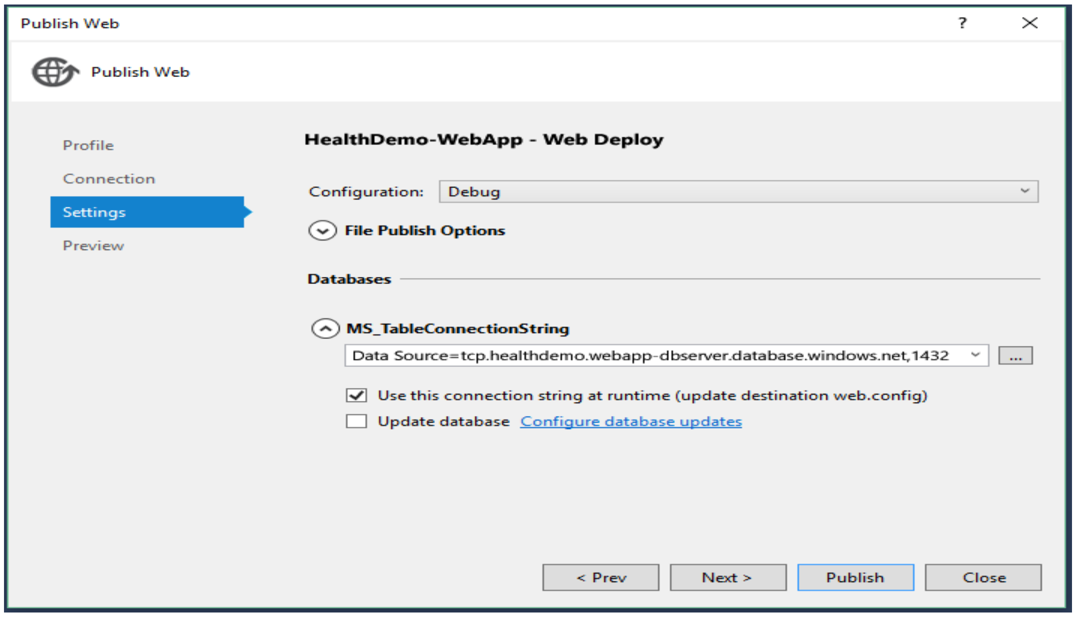


Publish Web

HealthDemo-WebApp - Web Deploy

Publish method: Web Deploy
Server: healhdemo-webapp.scm.azurewebsites.net:443
Site name: HealthDemo-WebApp
User name: $HealthDemo-WebApp
Password: ••
[✓] Save password
Destination URL: http://healthdemo-webapp.azurewebsites.net

1. In the **Settings** page, you can configure the deployment type depending on whether you require a deployment in a production environment or in a debug environment. You can see the connection to the database.



HealthDemo-WebApp - Web Deploy

Configuration: Debug

File Publish Options

Databases

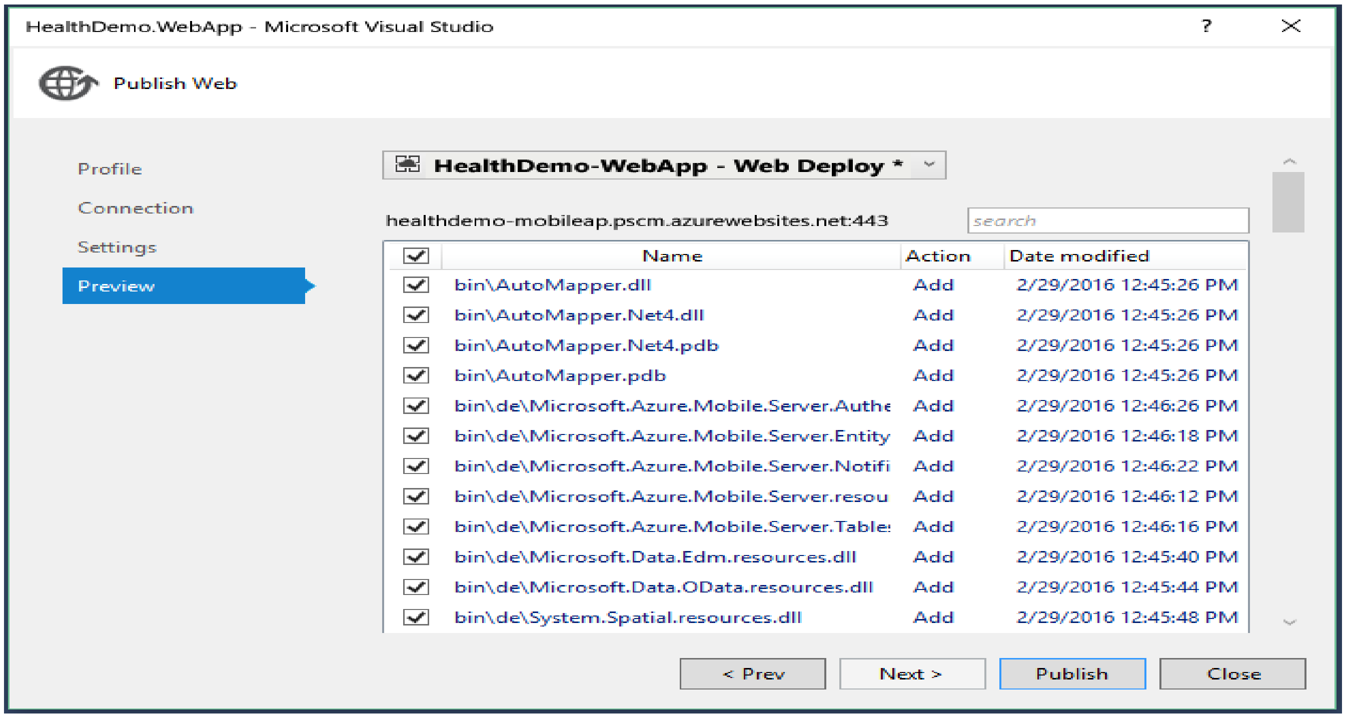
MS_TableConnectionString

Data Source=tcp.healthdemo.webapp-dbserver.database.windows.net,1432

[✓] Use this connection string at runtime (update destination web.config)

[ ] Update database Configure database updates

1. On the last page, **Preview**, you can determine the changes that affect the Azure environment.



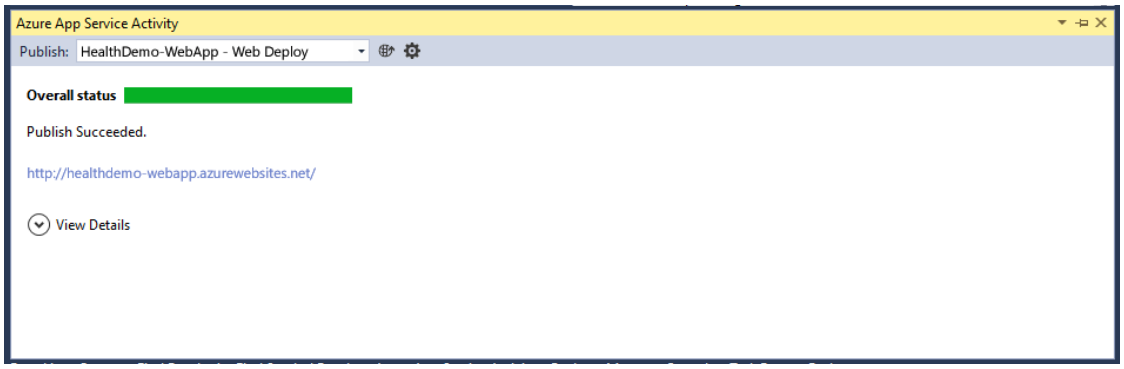
HealthDemo-WebApp - Web Deploy *

healthdemo-mobileap.pscm.azurewebsites.net:443

✓	Name	Action	Date modified
✓	bin\AutoMapper.dll	Add	2/29/2016 12:45:26 PM
✓	bin\AutoMapper.Net4.dll	Add	2/29/2016 12:45:26 PM
✓	bin\AutoMapper.Net4.pdb	Add	2/29/2016 12:45:26 PM
✓	bin\AutoMapper.pdb	Add	2/29/2016 12:45:26 PM
✓	bin\de\Microsoft.Azure.Mobile.Server.Authe	Add	2/29/2016 12:46:26 PM
✓	bin\de\Microsoft.Azure.Mobile.Server.Entity	Add	2/29/2016 12:46:18 PM
✓	bin\de\Microsoft.Azure.Mobile.Server.Notifi	Add	2/29/2016 12:46:22 PM
✓	bin\de\Microsoft.Azure.Mobile.Server.resou	Add	2/29/2016 12:46:12 PM
✓	bin\de\Microsoft.Azure.Mobile.Server.Table:	Add	2/29/2016 12:46:16 PM
✓	bin\de\Microsoft.Data.Edm.resources.dll	Add	2/29/2016 12:45:40 PM
✓	bin\de\Microsoft.Data.OData.resources.dll	Add	2/29/2016 12:45:44 PM
✓	bin\de\System.Spatial.resources.dll	Add	2/29/2016 12:45:48 PM

1. At last, click **Publish**.

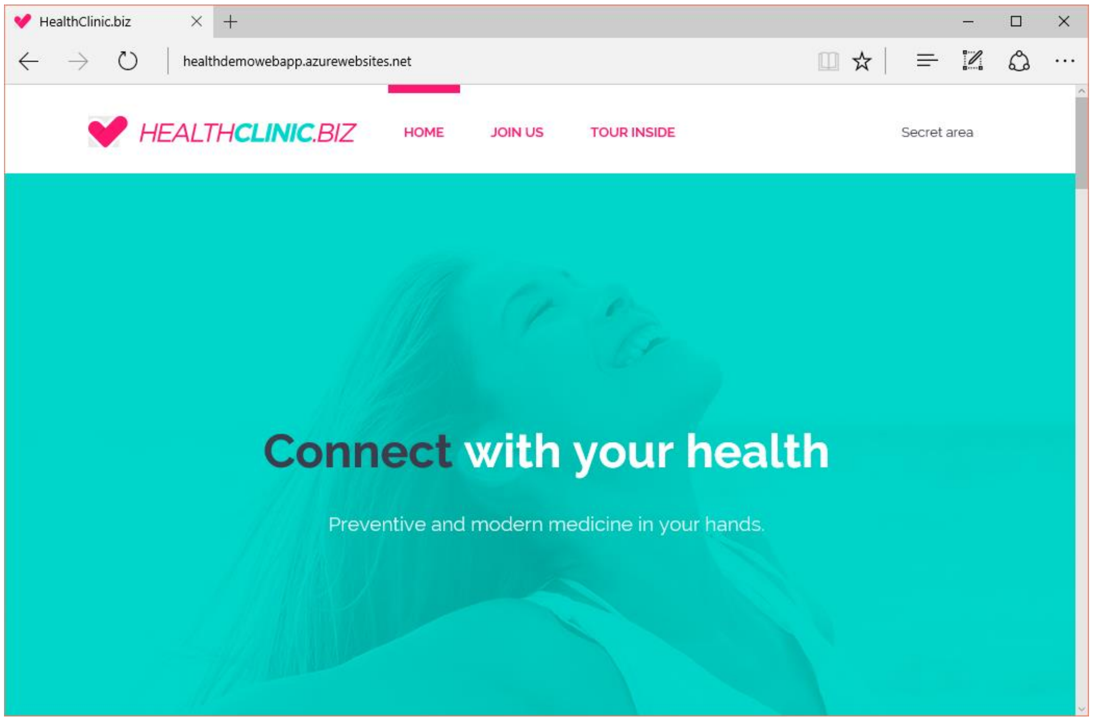
The **Output** window displays information about the deployment. When it’s finished, a message that everything has been completed successfully is displayed.


Azure App Service Activity
Publish: HealthDemo-WebApp - Web Deploy
Overall status
View Details

Publish Succeeded.

http://healthdemo-webapp.azurewebsites.net/

You can see the final result in the browser that will open.

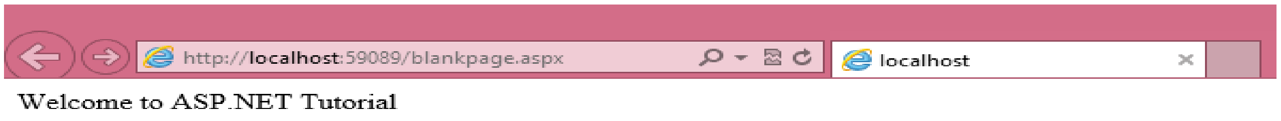


5.4 Web forms - Basic Web Controls - Working with events

Basic Web Controls

1. ASP.NET Label Control

The Label control used to display text in a set location on a Web page, also it can customize the displayed text through the Text property.



Welcome to ASP.NET Tutorial

HTML Code

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

<!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> ASP.NET </title>

</head>

<body>

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

    <div>

        <asp:Label ID="Label1" runat="server" Text="Welcome to ASP.NET Tutorial">

    </div>

    </form>

</body>

</html>
```

Properties

Property	Description

runat	Specifies that the control is a server control. Must be set to "server"
Text	The text to display in the label

ASP.NET Control Standard Properties

AppRelativeTemplateSourceDirectory, BindingContainer, ClientID, Controls, EnableTheming, EnableViewState, ID, NamingContainer, Page, Parent, Site, TemplateControl, TemplateSourceDirectory, UniqueID, Visible

For a full description, go to Web Control Standard Attributes.

ASP.NET Web Control Standard Properties

AccessKey, Attributes, BackColor, BorderColor, BorderStyle, BorderWidth, CssClass, Enabled, Font, EnableTheming, ForeColor, Height, IsEnabled, SkinID, Style, TabIndex, ToolTip, Width

For a full description, go to Web Control Standard Attributes.

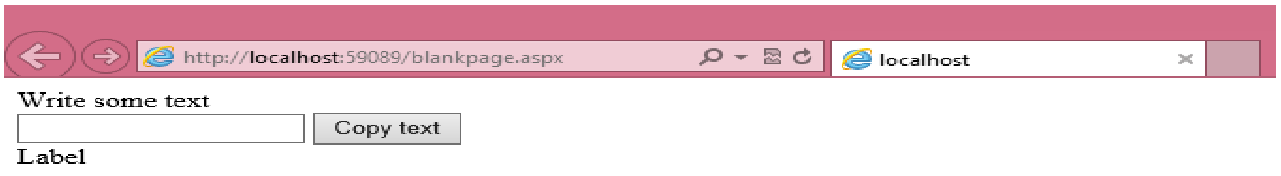
Example of Label Control

Declare one Label control, one TextBox control, and one Button control in an .aspx file. When the user clicks on the button, the submit subroutine is executed. The subroutine copies the content of the TextBox control to the Label control.

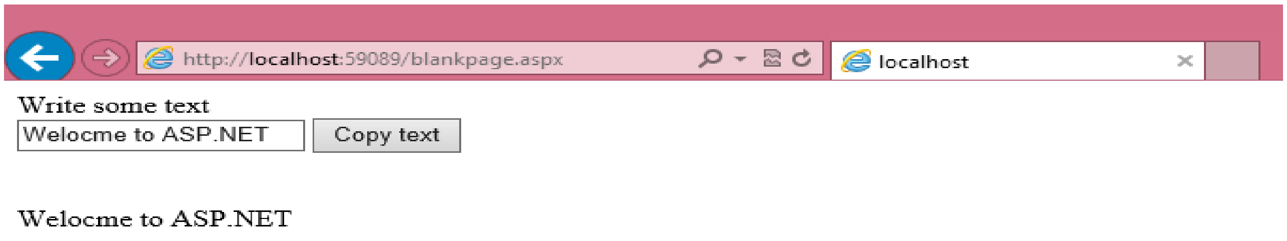
HTML Code

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

<head runat="server">

    <title> ASP.NET </title>

</head>

<body>

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

    <div>

        <asp:Label ID="Label1" runat="server" Text="Welcome to ASP.NET Tutorial">

        <br/>
        <asp:TextBox ID="TextBox1" runat="server">

        <asp:Button ID="Button1" runat="server" onclick="Button1_Click"Text="Button" />

        <br />

        <asp:Label ID="Label2" runat="server" Text="Label">

    </div>

    </form>

 </body>

 </html>
```

Output



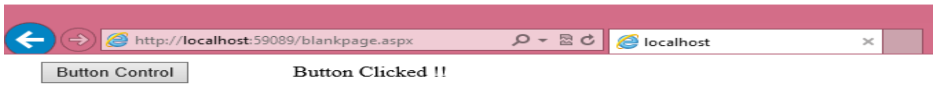
```
Write some text
[                    ] [Copy text]
Label
```



```
Write some text
[Welocme to ASP.NET  ] [Copy text]

Welocme to ASP.NET
```

2. ASP.NET Button Control

- The Button control is used to display a push button. The push button may be a submit button or a command button. By default, this control is a submit button.
- A submit button does not have a command name and it posts the Web page back to the server when it is clicked. It is possible to write an event handler to control the actions performed when the submit button is clicked.
- A command button has a command name and allows you to create multiple Button controls on a page. It is possible to write an event handler to control the actions performed when the command button is clicked.



Button Clicked !!

HTML Code

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title> Button Control </title>
</head>
<body>
    <form id="form1" runat="server">
    <div>
        <asp:Button ID="Button1" runat="server" onclick="Button1_Click"Text="Button" />
        <asp:Label ID="Label1" runat="server" Text="Label1"/>
    </div>
    </form>
</body>
</html>
```

Properties

Property	Description
CausesValidation	Specifies if a page is validated when a button is clicked
CommandArgument	Specifies additional information about the command to perform

CommandName	Specifies the command associated with the Command event
OnClientClick	Specifies the name of the function to be executed when a button is clicked
PostBackUrl	Specifies the URL of the page to post to from the current page when a button is clicked
runat	Specifies that the control is a server control. Must be set to "server"
Text	Specifies the text on a button
UseSubmitBehavior	Specifies whether or not a button uses the browser's submit mechanism or the ASP.NET postback mechanism
ValidationGroup	Specifies the group of controls a button causes validation, when it posts back to the server

ASP.NET Control Standard Properties

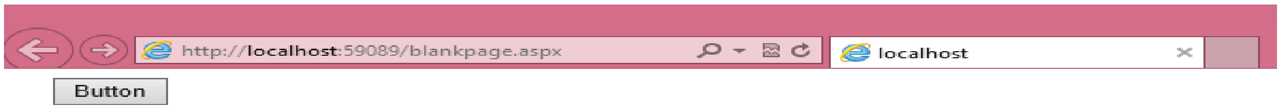
AppRelativeTemplateSourceDirectory, BindingContainer, ClientID, Controls, EnableTheming, EnableViewState, ID, NamingContainer, Page, Parent, Site, TemplateControl, TemplateSourceDirectory, UniqueID, Visible

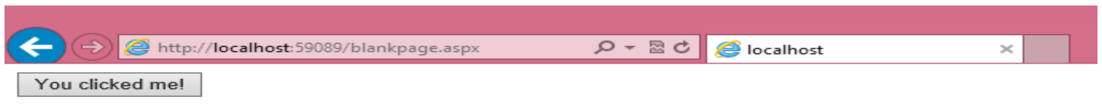
HTML Code

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

<head runat="server">

    <title> Button Control </title>

</head>

<body>
```

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

    <div>

        <asp:Button ID="Button1" runat="server" onclick="Button1_Click"Text="Button" />

    </div>

    </form>

 </body>

 </html>
```

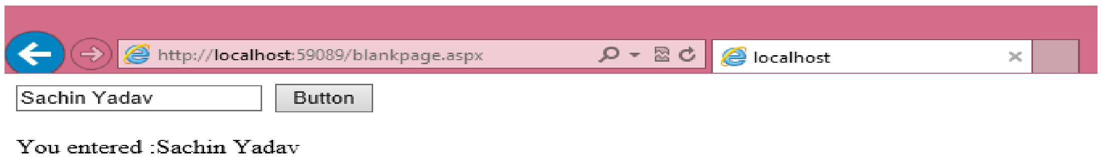
Output





3. **ASP.NET Textbox Control**

The TextBox control is used to create a text box where the user can input text.



```
You entered :Sachin Yadav
```

HTML Code

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

```
<head runat="server">
   <title> Button Control </title>
</head>
<body>
   <form id="form1" runat="server">
   <div>
      <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
      <asp:Button ID="Button1" runat="server" onclick="Button1_Click"Text="Button" />
      <br />
      <br />
      <asp:Label ID="Label1" runat="server" Text="Label"></asp:Label>
      <br />
   </div>
   </form>
</body>
</html>
```

ASP.NET - Event Handling

An event is an action or occurrence such as a mouse click, a key press, mouse movements, or any system-generated notification. A process communicates through events. For example, interrupts are system-generated events. When events occur, the application should be able to respond to it and manage it.

Events in ASP.NET raised at the client machine, and handled at the server machine. For example, a user clicks a button displayed in the browser. A Click event is raised. The browser handles this client-side event by posting it to the server.

The server has a subroutine describing what to do when the event is raised; it is called the event-handler. Therefore, when the event message is transmitted to the server, it checks whether the Click event has an associated event handler. If it has, the event handler is executed.

Event Arguments

ASP.NET event handlers generally take two parameters and return void. The first parameter represents the object raising the event and the second parameter is event argument.

The general syntax of an event is:

```
private void EventName (object sender, EventArgs e);
```

Application and Session Events

The most important application events are:

- **Application_Start** - It is raised when the application/website is started.
- **Application_End** - It is raised when the application/website is stopped.

Similarly, the most used Session events are:

- **Session_Start** - It is raised when a user first requests a page from the application.
- **Session_End** - It is raised when the session ends.

Page and Control Events

Common page and control events are:

- **DataBinding** - It is raised when a control binds to a data source.
- **Disposed** - It is raised when the page or the control is released.
- **Error** - It is a page event, occurs when an unhandled exception is thrown.
- **Init** - It is raised when the page or the control is initialized.
- **Load** - It is raised when the page or a control is loaded.
- **PreRender** - It is raised when the page or the control is to be rendered.

- **Unload** - It is raised when the page or control is unloaded from memory.

Event Handling Using Controls

All ASP.NET controls are implemented as classes, and they have events which are fired when a user performs a certain action on them. For example, when a user clicks a button the 'Click' event is generated. For handling events, there are in-built attributes and event handlers. Event handler is coded to respond to an event, and take appropriate action on it.

By default, Visual Studio creates an event handler by including a Handles clause on the Sub procedure. This clause names the control and event that the procedure handles.

The ASP tag for a button control:

```
<asp:Button ID="btnCancel" runat="server" Text="Cancel" />
```

The event handler for the Click event:

```
Protected Sub btnCancel_Click(ByVal sender As Object, ByVal e As System.EventArgs)

  Handles btnCancel.Click

  End Sub
```

An event can also be coded without Handles clause. Then, the handler must be named according to the appropriate event attribute of the control.

The ASP tag for a button control:

```
<asp:Button ID="btnCancel" runat="server" Text="Cancel" Onclick="btnCancel_Click" />
```

The event handler for the Click event:

```
Protected Sub btnCancel_Click(ByVal sender As Object, ByVal e As System.EventArgs)

End Sub
```

The common control events are:

Event	**Attribute**	**Controls**
Click	OnClick	Button, image button, link button, image map
Command	OnCommand	Button, image button, link button
TextChanged	OnTextChanged	Text box
SelectedIndexChanged	OnSelectedIndexChanged	Drop-down list, list box, radio button list, check box list.
CheckedChanged	OnCheckedChanged	Check box, radio button

Some events cause the form to be posted back to the server immediately, these are called the postback events. For example, the click event such as, Button.Click.

Some events are not posted back to the server immediately, these are called non-postback events.

For example, the change events or selection events such as TextBox.TextChanged or CheckBox.CheckedChanged. The nonpostback events could be made to post back immediately by setting their AutoPostBack property to true.

Default Events

The default event for the Page object is Load event. Similarly, every control has a default event. For example, default event for the button control is the Click event.

The default event handler could be created in Visual Studio, just by double clicking the control in design view. The following table shows some of the default events for common controls:

Control	**Default Event**
AdRotator	AdCreated
BulletedList	Click
Button	Click
Calender	SelectionChanged
CheckBox	CheckedChanged
CheckBoxList	SelectedIndexChanged
DataGrid	SelectedIndexChanged
DataList	SelectedIndexChanged
DropDownList	SelectedIndexChanged
HyperLink	Click
ImageButton	Click
ImageMap	Click
LinkButton	Click
ListBox	SelectedIndexChanged
Menu	MenuItemClick
RadioButton	CheckedChanged
RadioButtonList	SelectedIndexChanged

5.5 Rich Web Controls : AdRotator Control - Calendar Control - Custom web Controls - Validation controls

ASP.NET - Ad Rotator

The AdRotator control randomly selects banner graphics from a list, which is specified in an external XML schedule file. This external XML schedule file is called the advertisement file.

The AdRotator control allows you to specify the advertisement file and the type of window that the link should follow in the AdvertisementFile and the Target property respectively.

The basic syntax of adding an AdRotator is as follows:

```
<asp:AdRotator  runat = "server" AdvertisementFile = "adfile.xml"  Target =  "_blank" />
```

Before going into the details of the AdRotator control and its properties, let us look into the construction of the advertisement file.

The Advertisement File

The advertisement file is an XML file, which contains the information about the advertisements to be displayed.

Extensible Markup Language (XML) is a W3C standard for text document markup. It is a text-based markup language that enables you to store data in a structured format by using meaningful tags. The term 'extensible' implies that you can extend your ability to describe a document by defining meaningful tags for the application.

XML is not a language in itself, like HTML, but a set of rules for creating new markup languages. It is a meta-markup language. It allows developers to create custom tag sets for special uses. It structures, stores, and transports the information.

Following is an example of XML file:

```
<BOOK>
  <NAME> Learn XML </NAME>
  <AUTHOR> Samuel Peterson </AUTHOR>
  <PUBLISHER> NSS Publications </PUBLISHER>
  <PRICE> $30.00</PRICE>
</BOOK>
```

Like all XML files, the advertisement file needs to be a structured text file with well-defined tags delineating the data. There are the following standard XML elements that are commonly used in the advertisement file:

Element	Description
Advertisements	Encloses the advertisement file.
Ad	Delineates separate ad.
ImageUrl	The path of image that will be displayed.
NavigateUrl	The link that will be followed when the user clicks the ad.
AlternateText	The text that will be displayed instead of the picture if it cannot be displayed.
Keyword	Keyword identifying a group of advertisements. This is used for filtering.
Impressions	The number indicating how often an advertisement will appear.
Height	Height of the image to be displayed.
Width	Width of the image to be displayed.

Apart from these tags, customs tags with custom attributes could also be included. The following code illustrates an advertisement file ads.xml:

```
<Advertisements>
  <Ad>
    <ImageUrl>rose1.jpg</ImageUrl>
    <NavigateUrl>http://www.1800flowers.com</NavigateUrl>
    <AlternateText>
      Order flowers, roses, gifts and more
    </AlternateText>
    <Impressions>20</Impressions>
    <Keyword>flowers</Keyword>
  </Ad>
  <Ad>
    <ImageUrl>rose2.jpg</ImageUrl>
    <NavigateUrl>http://www.babybouquets.com.au</NavigateUrl>
    <AlternateText>Order roses and flowers</AlternateText>
    <Impressions>20</Impressions>
    <Keyword>gifts</Keyword>
  </Ad>
  <Ad>
    <ImageUrl>rose3.jpg</ImageUrl>
    <NavigateUrl>http://www.flowers2moscow.com</NavigateUrl>
    <AlternateText>Send flowers to Russia</AlternateText>
    <Impressions>20</Impressions>
    <Keyword>russia</Keyword>
  </Ad>
  <Ad>
    <ImageUrl>rose4.jpg</ImageUrl>
    <NavigateUrl>http://www.edibleblooms.com</NavigateUrl>
    <AlternateText>Edible Blooms</AlternateText>
    <Impressions>20</Impressions>
    <Keyword>gifts</Keyword>
  </Ad>
</Advertisements>
```

Properties and Events of the AdRotator Class

The AdRotator class is derived from the WebControl class and inherits its properties. Apart from those, the AdRotator class has the following properties:

Properties	Description
AdvertisementFile	The path to the advertisement file.
AlternateTextFeild	The element name of the field where alternate text is provided. The default value is AlternateText.
DataMember	The name of the specific list of data to be bound when advertisement file is not used.
DataSource	Control from where it would retrieve data.
DataSourceID	Id of the control from where it would retrieve data.
Font	Specifies the font properties associated with the advertisement banner control.
ImageUrlField	The element name of the field where the URL for the image is provided. The default value is ImageUrl.
KeywordFilter	For displaying the keyword based ads only.
NavigateUrlField	The element name of the field where the URL to navigate to is provided. The default value is NavigateUrl.
Target	The browser window or frame that displays the content of the page linked.
UniqueID	Obtains the unique, hierarchically qualified identifier for the AdRotator control.

Following are the important events of the AdRotator class:

Events	Description
AdCreated	It is raised once per round trip to the server after creation

	of the control, but before the page is rendered
DataBinding	Occurs when the server control binds to a data source.
DataBound	Occurs after the server control binds to a data source.
Disposed	Occurs when a server control is released from memory, which is the last stage of the server control lifecycle when an ASP.NET page is requested
Init	Occurs when the server control is initialized, which is the first step in its lifecycle.
Load	Occurs when the server control is loaded into the Page object.
PreRender	Occurs after the Control object is loaded but prior to rendering.
Unload	Occurs when the server control is unloaded from memory.

Working with AdRotator Control

Create a new web page and place an AdRotator control on it.

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

   <div>

      <asp:AdRotator ID="AdRotator1" runat="server" AdvertisementFile ="~/ads.xml"
onadcreated="AdRotator1_AdCreated" />

   </div>

</form>
```

The ads.xml file and the image files should be located in the root directory of the web site.

ASP.NET – Calendars

The calendar control is a functionally rich web control, which provides the following capabilities:

- Displaying one month at a time
- Selecting a day, a week or a month
- Selecting a range of days
- Moving from month to month
- Controlling the display of the days programmatically

The basic syntax of a calendar control is:

```
<asp:Calender ID = "Calendar1" runat = "server">

</asp:Calender>
```

Properties and Events of the Calendar Control

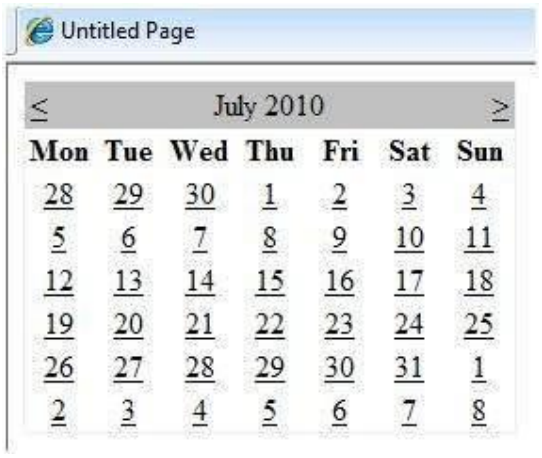
The calendar control has many properties and events, using which you can customize the actions and display of the control. The following table provides some important properties of the Calendar control:

The Calendar control has the following three most important events that allow the developers to program the calendar control. They are:

Events	Description
SelectionChanged	It is raised when a day, a week or an entire month is selected.
DayRender	It is raised when each data cell of the calendar control is rendered.
VisibleMonthChanged	It is raised when user changes a month.

Working with the Calendar Control

Putting a bare-bone calendar control without any code behind file provides a workable calendar to a site, which shows the months and days of the year. It also allows navigation to next and previous months.



<			July 2010			>
Mon	Tue	Wed	Thu	Fri	Sat	Sun
28	29	30	1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	1
2	3	4	5	6	7	8

Calendar controls allow the users to select a single day, a week, or an entire month. This is done by using the SelectionMode property. This property has the following values:

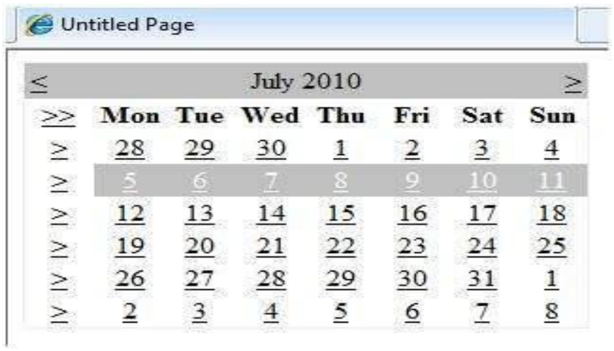
Properties	Description
Day	To select a single day.

DayWeek	To select a single day or an entire week.
DayWeekMonth	To select a single day, a week, or an entire month.
None	Nothing can be selected.

The syntax for selecting days:

```
<asp:Calender ID = "Calendar1" runat = "server" SelectionMode="DayWeekMonth">

</asp:Calender>
```

When the selection mode is set to the value DayWeekMonth, an extra column with the > symbol appears for selecting the week, and a >> symbol appears to the left of the days name for selecting the month.



<			July 2010				>
>>	Mon	Tue	Wed	Thu	Fri	Sat	Sun
>	28	29	30	1	2	3	4
>	5	6	7	8	9	10	11
>	12	13	14	15	16	17	18
>	19	20	21	22	23	24	25
>	26	27	28	29	30	31	1
>	2	3	4	5	6	7	8

Example

The following example demonstrates selecting a date and displays the date in a label:

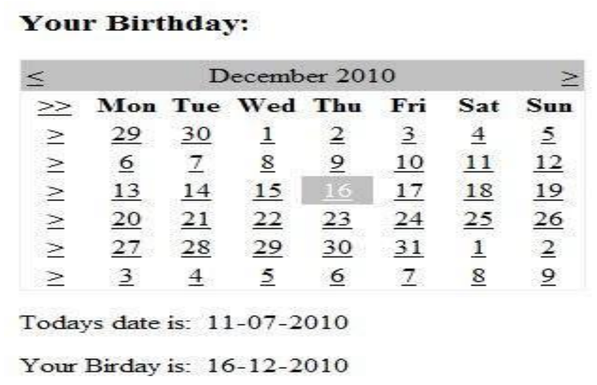
The content file code is as follows:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Default.aspx.cs"
Inherits="calendardemo._Default" %>

<!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>

      Untitled Page
    </title>

  </head>

    <body>

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

          <div>

        <h3> Your Birthday:</h3>

        <asp:Calendar ID="Calendar1" runat="server  SelectionMode="DayWeekMonth"
onselectionchanged="Calendar1_SelectionChanged">

        </asp:Calendar>

      </div>

           <p>Todays date is:

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

      </p>

           <p>Your Birday is:

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

      </p>

       </form>

  </body>

</html>
```

The event handler for the event SelectionChanged:

```
protected void Calendar1_SelectionChanged(object sender, EventArgs e)

{

  lblday.Text = Calendar1.TodaysDate.ToShortDateString();

  lblbday.Text = Calendar1.SelectedDate.ToShortDateString();

}
```

When the file is run, it should produce the following output:


Your Birthday:

<			December 2010				>
>>	Mon	Tue	Wed	Thu	Fri	Sat	Sun
>	29	30	1	2	3	4	5
>	6	7	8	9	10	11	12
>	13	14	15	16	17	18	19
>	20	21	22	23	24	25	26
>	27	28	29	30	31	1	2
>	3	4	5	6	7	8	9

Todays date is: 11-07-2010

Your Birday is: 16-12-2010

Asp.Net – Validators

ASP.NET validation controls validate the user input data to ensure that useless, unauthenticated, or contradictory data don't get stored.

ASP.NET provides the following validation controls:

1. RequiredFieldValidator
2. RangeValidator
3. CompareValidator
4. RegularExpressionValidator
5. CustomValidator
6. ValidationSummary

BaseValidator Class

The validation control classes are inherited from the BaseValidator class hence they inherit its properties and methods. Therefore, it would help to take a look at the properties and the methods of this base class, which are common for all the validation controls:

Members	Description
ControlToValidate	Indicates the input control to validate.
Display	Indicates how the error message is shown.
EnableClientScript	Indicates whether client side validation will take.
Enabled	Enables or disables the validator.
ErrorMessage	Indicates error string.
Text	Error text to be shown if validation fails.
IsValid	Indicates whether the value of the control is valid.
SetFocusOnError	It indicates whether in case of an invalid control, the focus should

	switch to the related input control.
ValidationGroup	The logical group of multiple validators, where this control belongs.
Validate	This method revalidates the control and updates the IsValid property.

RequiredFieldValidator Control

The RequiredFieldValidator control ensures that the required field is not empty. It is generally tied to a text box to force input into the text box.

The syntax of the control is as given:

```
<asp:RequiredFieldValidator ID="rfvcandidate"
   runat="server" ControlToValidate ="ddlcandidate"
   ErrorMessage="Please choose a candidate"
   InitialValue="Please choose a candidate">
</asp:RequiredFieldValidator>
```

RangeValidator Control

The RangeValidator control verifies that the input value falls within a predetermined range.

It has three specific properties:

Properties	Description
Type	It defines the type of the data. The available values are: Currency, Date, Double, Integer, and String.
MinimumValue	It specifies the minimum value of the range.
MaximumValue	It specifies the maximum value of the range.

The syntax of the control is as given:

```
<asp:RangeValidator ID="rvclass" runat="server" ControlToValidate="txtclass"
   ErrorMessage="Enter your class (6 - 12)" MaximumValue="12" MinimumValue="6"
   Type="Integer">
</asp:RangeValidator>
```

CompareValidator Control

The CompareValidator control compares a value in one control with a fixed value or a value in another control.

It has the following specific properties:

Properties	Description
Type	It specifies the data type.
ControlToCompare	It specifies the value of the input control to compare with.
ValueToCompare	It specifies the constant value to compare with.
Operator	It specifies the comparison operator, the available values are: Equal, NotEqual, GreaterThan, GreaterThanEqual, LessThan, LessThanEqual, and DataTypeCheck.

The basic syntax of the control is as follows:

```
<asp:CompareValidator ID="CompareValidator1" runat="server"
   ErrorMessage="CompareValidator">
</asp:CompareValidator>
```

CustomValidator

The CustomValidator control allows writing application specific custom validation routines for both the client side and the server side validation.

The client side validation is accomplished through the ClientValidationFunction property. The client side validation routine should be written in a scripting language, such as JavaScript or VBScript, which the browser can understand.

The server side validation routine must be called from the control's ServerValidate event handler.

The server side validation routine should be written in any .Net language, like C# or VB.Net.

The basic syntax for the control is as given:

```
<asp:CustomValidator ID="CustomValidator1" runat="server"
   ClientValidationFunction=.cvf_func. ErrorMessage="CustomValidator">
</asp:CustomValidator>
```

ValidationSummary

The ValidationSummary control does not perform any validation but shows a summary of all errors in the page. The summary displays the values of the ErrorMessage property of all validation controls that failed validation.

The following two mutually inclusive properties list out the error message:

ShowSummary : shows the error messages in specified format.

ShowMessageBox : shows the error messages in a separate window.

The syntax for the control is as given:

```
<asp:ValidationSummary ID="ValidationSummary1" runat="server"
   DisplayMode = "BulletList" ShowSummary = "true" HeaderText="Errors:" />
```

Validation Groups

Complex pages have different groups of information provided in different panels. In such situation, a need might arise for performing validation separately for separate group. This kind of situation is handled using validation groups.

To create a validation group, you should put the input controls and the validation controls into the same logical group by setting their *ValidationGroup* property.

5.6 ASP.NET APPLICATION LOCALIZATION

ASP.NET is one of the most common web application development platforms. As with any other .NET framework application, ASP.NET provides great support for localization and globalization straight out of the box. Following best practices from the beginning of development will result in a properly internationalized application, making ASP.NET application localization an enjoyable process. In order to save money, reduce localization/internationalization defects and shorten the time to market, it’s important to consider localization as part of the development process.

ASP.NET Application Localization: What Needs to be Localized/Translated?

In a typical ASP.NET application, the following four items may differ depending on the user’s language and regional preferences:

Text resources – The text that resides in the aspx page. Both asp server tags and traditional html tags may contain text on an aspx page. This includes the UI text that may appear in C-Sharp(C#) and Visual Basic code behind files.

Database content – Most of the translatable text resides in a database for a typical ASP.NET application particularly in content management systems (product information, articles etc.).

Images and graphics – Graphics and images may contain translatable text.

Regional options – Date/time, currency, number/decimal formatting will vary depending on the user’s region and/or preferences.

Text Resources

All text resources should be externalized to resx files in order to make translation and maintenance easier. Also known as “resources file”, a resx file consists of XML entries which specify objects and strings. Visual Studio does provide an automated way to externalize the strings after the page has been created but this function is limited. It only works on the server tags on aspx pages. Visual Studio cannot externalize the text in CS/VB code or process regular html tags.

The best approach is to externalize the tags as the pages are being developed. More information on the use of resx files for ASP.NET localization can be found on MSDN.

Database Content

The database schema and relations will have to be carefully designed to externalize translatable text to a table(s) which maps to languages and parent tables. Although adding additional columns to existing tables for each language may seem easier; maintenance of this solution is extremely difficult and not recommended.

Images and Graphics

If possible, try avoiding images and graphics with embedded text. Using background images and retrieving the text from a resource file will make the localization effort much easier. If that is not possible, point the source of the image to the resx file which will let you easily change it depending on the language. This applies to other external assets such as pdf and doc files as well.

Regional Options

As in any other programming language and platform, avoid manually formatting and hard coding date/time and number formats. .NET offers a CultureInfo class which provides access to culture-specific instances of objects such as: DateTimeFormatInfo and NumberFormatInfo. Using these objects, culture-specific operations can be performed easily such as formatting dates and numbers, casing and comparing strings.

Asp.net localization and globalization

Most websites on the internet today are designed and developed to support a single language and culture, but making a website or web application to support different languages and cultures is not difficult and is relatively easy to implement in ASP.Net.

When a web application supports globalization, it means that the application can render it contents based on the language preference set by a user in his or her browser. Localization involves translating a web application into different languages and cultures.

Making an already deployed website to support globalization is not an easy task, thus it is always advised to make an application support globalization right from the development stage, even if there is no current plan to make it support more than one language. If later there is a decision to translate the application to another language, it will be easy to do, since the structure has already been put in place.

ASP.Net makes it easy to localize an application through the use of resource files. Resource files are xml files with .resx extension. Resources can either be a global resource, in which it is accessible throughout the site and placed in App_GlobalResources folder in a website or a local resource which is specific to a page only and is placed in App_LocalResources folder.

Now, lets get started, we are going to have a page that displays Good Morning in four different Languages(French, Spanish, German and Yoruba). Create a new ASP.Net website in Visual Studio and name it localizationexample, right click the website in solution explorer and select Add ASP.Net Folder then select App_GlobalResources. After that, right click the App_GlobalResources and select Add New Item in the list of items displayed in the dialog box then select Resource File , give it any name you like, but in this case I name it content.resx this will serve as the default resource file that will be selected if the language selected in the user browser is not available. Note that of great importance is the languageid and cultureid because ASP.Net checks the languageid and cultureid combination on the browser to determinge the resource file to be used in rendering contents to that browser, to see the detailed list of language and culture identification click here.

The next thing is to create our language specific resource file, for the four languages that the localizationexample website will be supporting, the culture names are

fr for French

de for German

es for Spanish and

yo-NG for Yoruba

Apart from the content.resx file in the App_GlobalResources four other resource files with the format resourcename.languageId-cultureId.resx are to be created content.de.resx, content.es.resx, content.fr.resx and content.yo-NG.resx

Global resource files

A resource file takes a key value pair, so all the resource files will contain the same key but the different values as the language translation may be, so I have a key Greeting in all the files with their values.

Resource file	**Key**	**Value**
content.resx	Greeting -	Good Morning
content.de.resx	Greeting -	Guten Morgen
content.es.resx	Greeting -	buenos días
content.fr.resx	Greeting -	bonjour
content.yo-NG.resx	Greeting -	E kaa ro

The next thing to do is add a Label control that will be used to display the localized text to the Default.aspx page, then switch to the page source view in visual studio and set the Text property of the Label control to Text="< %$ Resources:content, Greeting % >"

Where Resource:content specifies that we are retrieving the content from a resource file named content and the value to be retrieved from the resource file has the name or key Greeting. Note that content was used and not content.de or any of the remaining three resource files, because this is our default resource file that ASP.Net will always use should in case we do not have translation for the user's browser language preference.

We need to override the page's InitializeCulture() method and set the page UICulture property, this property determines which global or local resource to be loaded for the page. The browser's topmost language preference will be obtainted by Request.UserLanguages[0] and then call the page’s base InitializeCulture method.

```
protected override void InitializeCulture()
{
  UICulture = Request.UserLanguages[0];
  base.InitializeCulture();
}
```