COMP 3111: Software Engineering

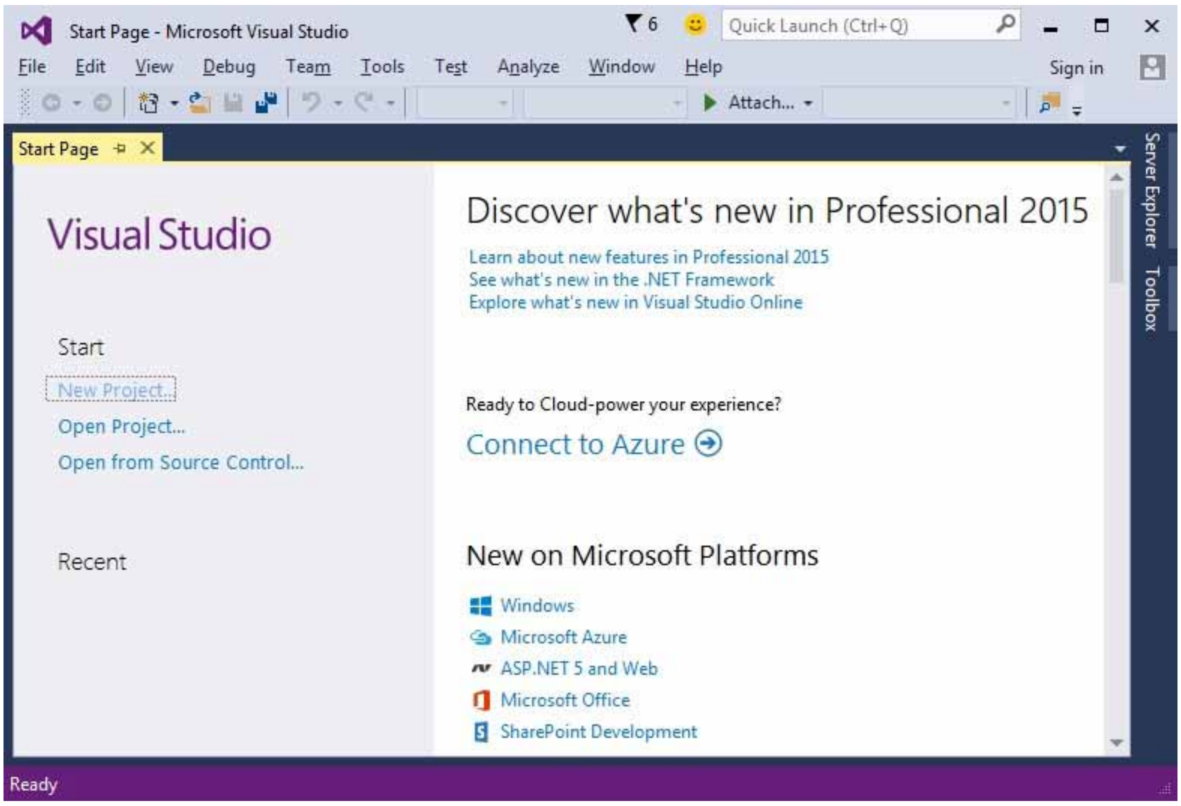
Developing ASP.NET Applications
Tutorial and Lab Notes

MICROSOFT .NET

Microsoft .NET is a set of technologies that includes the .NET Framework and the .NET programming languages and language tools. The .NET Framework provides the basic system support services for the .NET technologies, such as ASP.NET, and consists of the common language runtime (CLR) and the .NET Framework class library. The CLR provides a run-time environment that manages the execution of .NET code written in any .NET language, including memory and object lifetime, and allows debugging, exception handling and inheritance across multiple languages. The .NET Framework class library is a hierarchical set of base classes that can be used in developing applications. These classes, which are organized by containers referred to as *namespaces*[1], include classes that support basic common data types, classes that provide access to data and classes that support system services such as drawing, network functionality, etc. The .NET class library also contains the classes that form the basis of ASP.NET.

VISUAL STUDIO

Visual Studio is the integrated development environment (IDE) for all the .NET languages. When you start the Visual Studio IDE, the Start Page tab is displayed as shown in Figure 1[2]. In the Start Page tab you can open recent projects or create new projects. Projects are used for creating Windows applications or websites. The Start Page also allows you to access online documentation and information, such as recent MSDN articles. Much of the Visual Studio IDE can be customized from the `Customize` and `Options` windows, which can be accessed under Tools in the Visual Studio menu bar.



Start Page - Microsoft Visual Studio
Quick Launch (Ctrl+Q)
File
Edit
View
Debug
Team
Tools
Test
Analyze
Window
Help
Sign in
Attach...
Start Page
Server Explorer
Toolbox
Visual Studio
Start
New Project...
Open Project...
Open from Source Control...
Recent
Discover what's new in Professional 2015
Learn about new features in Professional 2015
See what's new in the .NET Framework
Explore what's new in Visual Studio Online
Ready to Cloud-power your experience?
Connect to Azure
New on Microsoft Platforms
Windows
Microsoft Azure
ASP.NET 5 and Web
Microsoft Office
SharePoint Development
Ready

Figure 1: Visual Studio Start Page.

[1] A namespace is a container used for scoping and can contain classes, structures, enumerations, interfaces, delegates and other namespaces. Namespaces are declared using the *Namespace…End Namespace* syntax.

[2] The first time you start Visual Studio you will be prompted to choose the development settings that you want to use. Although several settings are available, only Visual Basic and Visual C# are supported in the CSE labs. The development settings can be changed by selecting “Tools→Import and Export Settings” in the Visual Studio menu bar. The tutorial and lab notes use Visual C# code examples.

ASP.NET

ASP.NET[3] is a programming framework built on the CLR that is used on a web server to construct web applications[4]. A toolbox and visual web page designer in the Visual Studio IDE provide WYSIWYG editing, drag-and-drop server controls[5] and automatic deployment that complement the ASP.NET programming framework. Because ASP.NET is based on the CLR, the .NET Framework class library, messaging and data access solutions are all accessible from the Web. ASP.NET web forms[6] allow building web-based user interfaces that cleanly separate application logic from presentation code and that handle events[7] in a simple, forms-processing model. Additionally, the CLR simplifies development with managed code services such as automatic reference counting and garbage collection.

ASP.NET is a server-side technology that enables code and server controls to run on a web server. An ASP.NET application resides on a web server where it receives requests from a browser, runs code associated with the request and returns responses to the browser. This allows browsers to display dynamic content and to request information. The web server, instead of returning a static page, runs an application and returns HTML that reflects up-to-date, accurate information. ASP.NET applications are like conventional stand-alone applications—they can retain user information between sessions, or users, of the application.

Visual Studio supports two types of ASP.NET projects, *Web Applications* and *Web Sites*, both of which provide HTML-based user interfaces. These notes will cover only web application projects.

ASP.NET Web Application Projects

An ASP.NET web application project consists of the following components.

1. A directory that holds the files of the website.
2. One or more `.master` files (master pages) containing layout templates for web pages.
3. One or more `.aspx` files (web forms) that contain the web page contents and that provide an HTML-based user interface through which users access the website's functionality.
4. An optional `Global.asax` file that deals with Session and Application startup and cleanup logic.
5. An optional `Web.config` file used to store application configuration settings.

These files are created as needed when an ASP.NET web application is developed in Visual Studio.

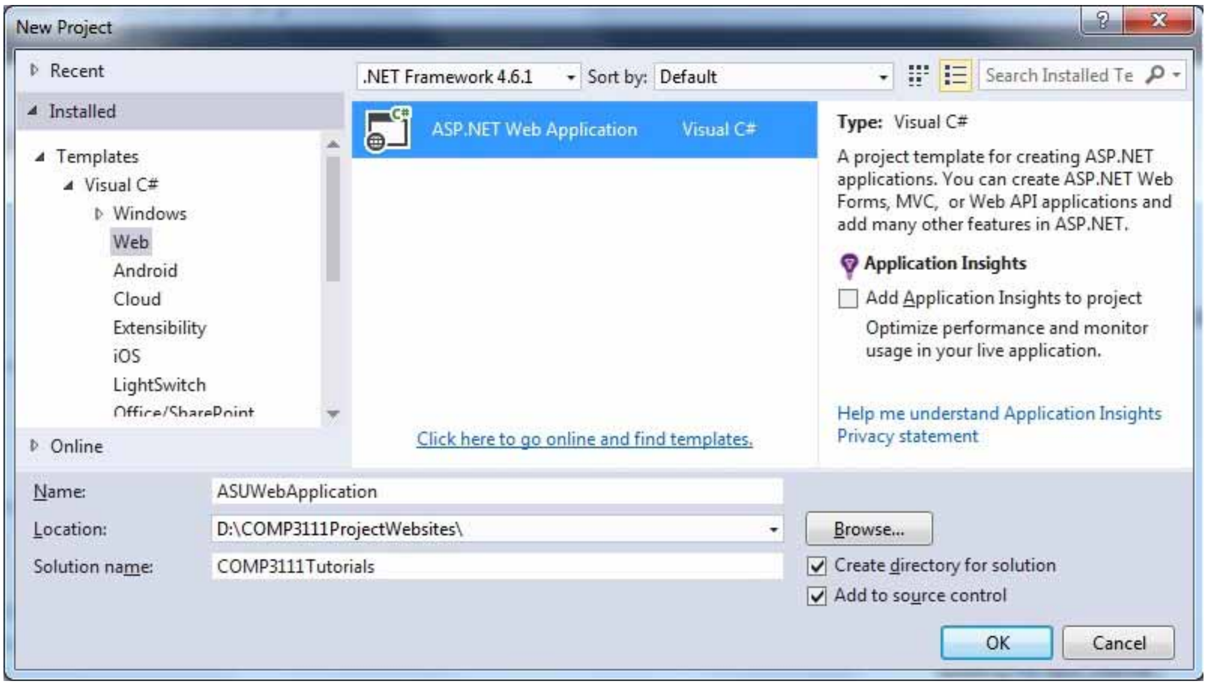
Creating a Web Application

We will create a new ASP.NET web application project in Visual Studio for a fictitious university called ASU.

1. Launch Visual Studio from the Windows Start menu[8].
2. Click "New Project…" in the Start Page[9].
3. Make the following selections in the `New Project` window shown in Figure 2.
   a. In the left pane under "Installed→Templates":
      i. Select the language to use. (Our examples will use Visual C#.)
      ii. Select "Web" as the project type.
   b. In the centre pane:
      i. If not already selected, select ".NET Framework 4.6.1" from the dropdown list at the top left.
      ii. Select "ASP.NET Web Application".
   c. In the bottom pane:
      i. In the "Name" textbox, enter the name of the project. (For our example enter "ASUWebSite".)
      ii. In the "Location" textbox, either enter the directory path or browse to the directory where you want to create the web application project.
      iii. In the "Solution name" textbox, enter the name of the solution. The solution name defaults to the website name. A solution can contain several projects.
      iv. Check the "Create directory for solution" and "Add to source control"[10] checkboxes.
4. Click the OK button.

[3] You can learn more about ASP.NET at http://www.asp.net/.
[4] Since ASP.NET uses compiled CLR code running on the server, it can take advantage of early binding, just-in-time compilation, native optimization and caching services.
[5] A control is a pre-specified piece of functionality, such as a button or input box, used to build applications. Controls are discussed in the ASP.NET Controls section of these notes.
[6] A web form is one of the ASP.NET programming models used to develop applications. The other programming models are ASP.NET MVC and ASP.NET Web Pages.
[7] An event is a user (or system) action, such as clicking the mouse, pressing a key, a timer expiring, etc.
[8] The first time you launch Visual Studio it will take some time to set up the environment; be patient.
[9] You can also create a new web application by entering Ctrl+Shift+N.
[10] Source control will be discussed in a subsequent tutorial and lab.

The `New ASP.NET Project` window opens, which allows you to select the type of web application project to create as shown in Figure 3.


New Project

Recent
Installed
Templates
Visual C#
Windows
Web
Android
Cloud
Extensibility
iOS
LightSwitch
Office/SharePoint
Online

.NET Framework 4.6.1 | Sort by: Default | Search Installed Te

ASP.NET Web Application — Visual C#

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.

Click here to go online and find templates.

Help me understand Application Insights

Privacy statement

Name: ASUWebApplication

Location: D:\COMP3111ProjectWebsites\

Solution name: COMP3111Tutorials

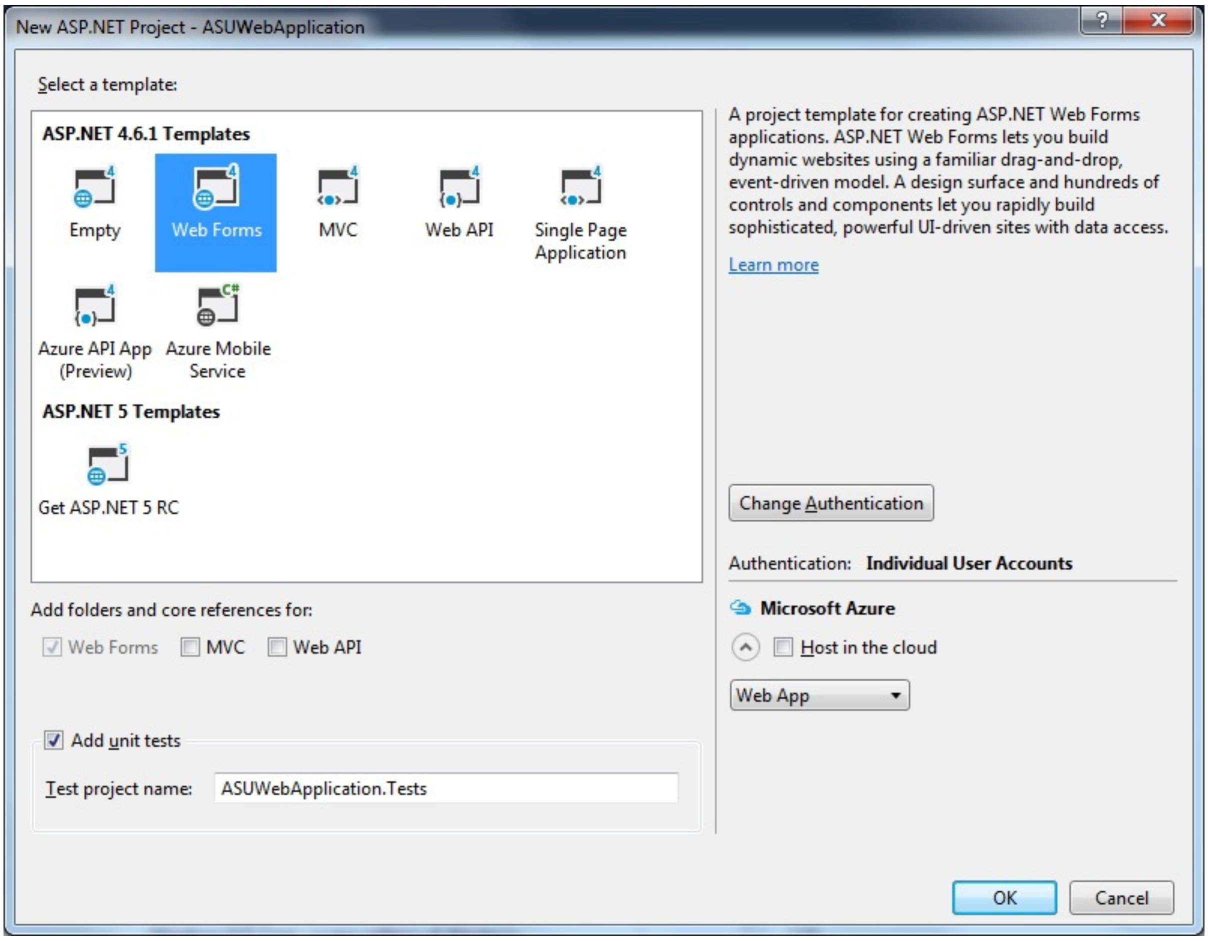
Browse...

☑ Create directory for solution

☑ Add to source control

OK | Cancel

Figure 2: `New Project` window.


New ASP.NET Project - ASUWebApplication

Select a template:

ASP.NET 4.6.1 Templates

Empty | Web Forms | MVC | Web API | Single Page Application | Azure API App (Preview) | Azure Mobile Service

ASP.NET 5 Templates

Get ASP.NET 5 RC

A project template for creating ASP.NET Web Forms applications. ASP.NET Web Forms lets you build dynamic websites using a familiar drag-and-drop, event-driven model. A design surface and hundreds of controls and components let you rapidly build sophisticated, powerful UI-driven sites with data access.

Learn more

Change Authentication

Authentication: **Individual User Accounts**

Microsoft Azure

☐ Host in the cloud

Web App

Add folders and core references for:

☑ Web Forms ☐ MVC ☐ Web API

☑ Add unit tests

Test project name: ASUWebApplication.Tests

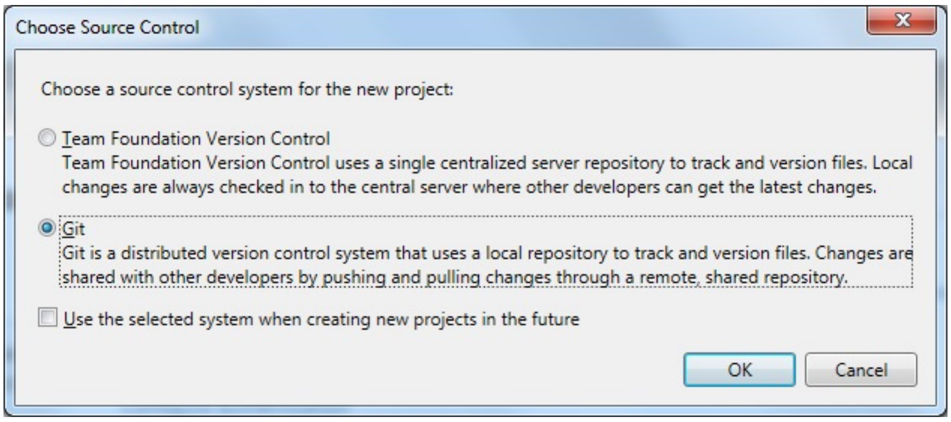
OK | Cancel

Figure 3: `New ASP.NET Project` window.

In the `New ASP.NET Project` window make the following selections.

1. Select “Web Forms” in the “Select a template” pane.
2. Check the “Add unit tests” checkbox[11] at the bottom of the left pane.
3. The “Test project name” should default to the name of the web application.
4. Uncheck the “Host in the cloud” checkbox at the bottom of the right pane.
5. Click the OK button.

After some time (be patient) the `Choose Source Control` window opens, which allows you to choose a source control system for the project as shown in Figure 4. Git will be used in the course project for source control, so you should choose Git in the `Choose Source Control` window and click OK.


Choose Source Control

Choose a source control system for the new project:

( ) Team Foundation Version Control
Team Foundation Version Control uses a single centralized server repository to track and version files. Local changes are always checked in to the central server where other developers can get the latest changes.

(●) Git
Git is a distributed version control system that uses a local repository to track and version files. Changes are shared with other developers by pushing and pulling changes through a remote, shared repository.

☐ Use the selected system when creating new projects in the future

Figure 4: Adding source control to a project.

After Visual Studio creates the files needed for a project, which can take some time, the Web Forms Designer interface of Visual Studio appears as shown in Figure 5. This interface includes a number of windows[12], toolbars and menus.

- The `Document` window, shown at the top of the center pane in Figure 5 with tab label “Your ASP.NET application”, is used to design the interface of a web page and to develop the source code. The `Document` window integrates almost all of the design and source-code editors used in Visual Studio. The view tabs at the bottom of the `Document` window allow switching between the Design and Source (HTML) views of web pages, as well as between different modes for other document types. When in Design view, the bottom right side of the `Document` window also allows you to select HTML tags in a web form and to set their properties in the `Properties` window.
- The `Solution Explorer`[13] window, shown in the left pane in Figure 5, allows management of files and resources, including adding, removing, opening, renaming and moving files. It also allows setting a start-up web page, switching between Design and Code view for a file and viewing status information on files.
- The `Properties` window, not shown in Figure 5, provides access to the properties of the element currently selected in the `Document` window.
- The `Toolbox` window, shown in the right pane in Figure 5, provides access to many server, third-party and custom controls as well as HTML tags. Controls can be added to web forms by either double-clicking or dragging-and-dropping them, can be written in any language and used in any language.
- The `SQL Server Object Explorer` window, whose tab can be seen at the bottom of the right pane in Figure 5, allows resources on both the local machine and remote servers to be viewed, including configured data connections, event logs, message queues and performance counters.
- The `Class View` window, whose tab can be seen at the bottom of the left pane Figure 5, lists all classes in an application and the methods, properties and interfaces that the classes implement.
- The `Error List` window, shown at the bottom of the middle pane in Figure 5, is used to display messages related to unresolved errors in your project.
- The `Task List` window, whose tab can be seen at the bottom of the bottom centre pane in Figure 5, allows creation, sorting and tracking of unfinished tasks for the current website. It can contain tasks automatically generated by Visual Studio that help locate and correct errors.

[11] Unit testing will be discussed in a subsequent tutorial and lab.
[12] Dockable windows can be shown or hidden by clicking the Auto Hide button, which looks like a pushpin, on the window’s menu bar.
[13] You can make visible any window by selecting it in the Visual Studio menu bar View menu.

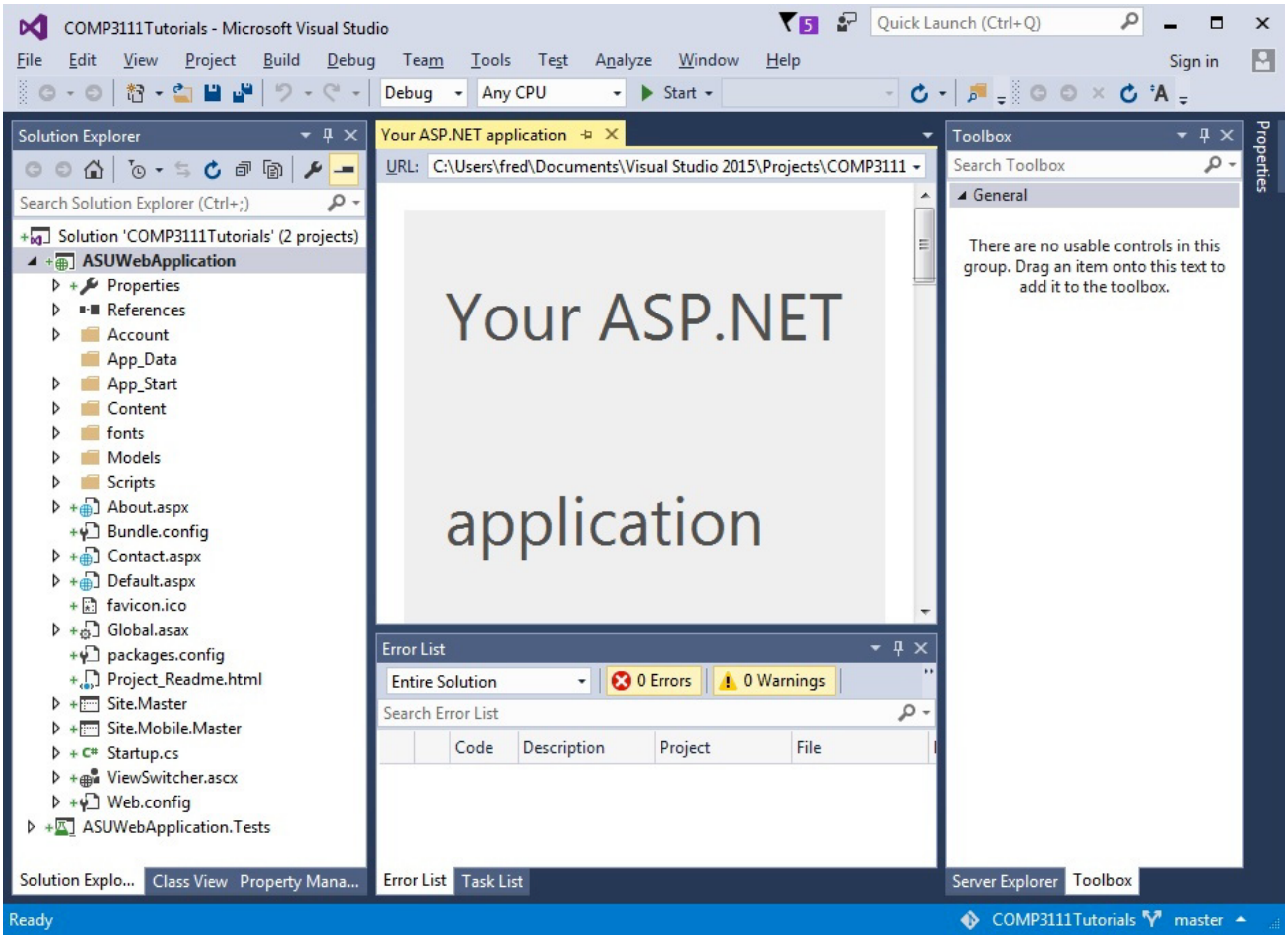


Figure 5: Visual Studio Web Forms Designer interface.

Master Pages

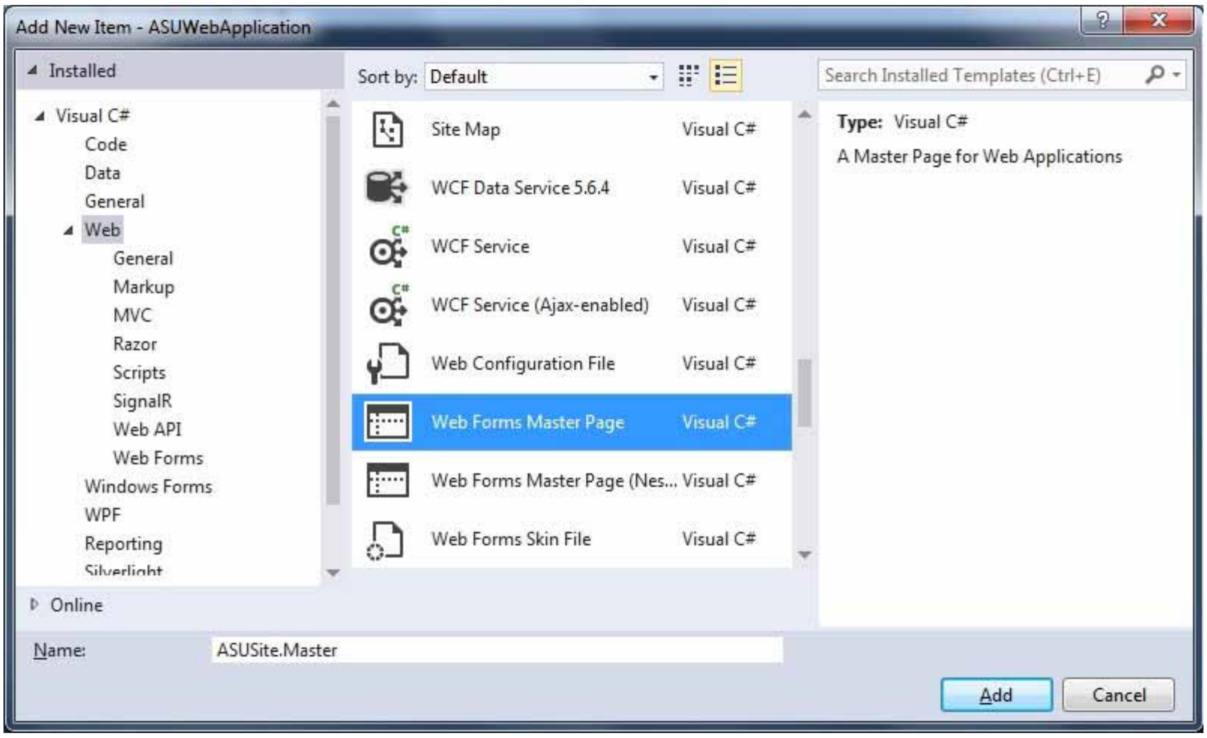
A master page is a template that is used to standardize the layout and behaviour of a web page so that all the web pages in a website have the same look and feel (e.g., website header, navigation controls, etc.). A master page centralizes the common functionality of web pages so that updates can be made in just one place. Moreover, one set of controls and code can be created and the results applied to a set of web pages (e.g., a menu that applies to all web pages).

A master page is an ASP.NET file with the extension `.master`. A master page's predefined layout can include static text, HTML tags and server controls. The master page also contains all of the top-level HTML tags for a page, such as `<html>`, `<head>` and `<form>` (see Figure 15).

A master page includes one or more ContentPlaceHolder controls that define areas on a web page where user supplied content will appear. The content for the master page's ContentPlaceHolder controls is defined by creating individual ASP.NET web pages that are bound to a specific master page. When users request a web page, it is merged with the master page to produce output that combines the layout of the master page with the content from the web page.

Creating Master Pages

When you create a new Web Forms web application, a default master page named `Site.master` is added to your project as shown in the `Solution Explorer` window in Figure 5. However, a website can contain more than one master page providing different web page layouts and a master page (nested master page) can depend on another master page. To add additional master pages to a website do the following.


Add New Item - ASUWebApplication

Installed
- Visual C#
  - Code
  - Data
  - General
  - Web
    - General
    - Markup
    - MVC
    - Razor
    - Scripts
    - SignalR
    - Web API
    - Web Forms
  - Windows Forms
  - WPF
  - Reporting
  - Silverlight
- Online

Sort by: Default

Site Map	Visual C#
WCF Data Service 5.6.4	Visual C#
WCF Service	Visual C#
WCF Service (Ajax-enabled)	Visual C#
Web Configuration File	Visual C#
Web Forms Master Page	Visual C#
Web Forms Master Page (Nes...	Visual C#
Web Forms Skin File	Visual C#

Search Installed Templates (Ctrl+E)

Type: Visual C#

A Master Page for Web Applications

Name: ASUSite.Master

Add Cancel

Figure 6: `Add New Item` window for adding a master page.

1. Right-click the website name in the Solution Explorer and select “Add New Item…” from the pop up menu.
   or
   Click “Website” on the Visual Studio menu bar and select “Add New Item…”.
   or
   Press Crtl+Shift+A.
2. In the `Add New Item` window shown in Figure 6, make the following selections:
   a. Select “Visual C#→Web” in the left pane, under “Installed”.
   b. Select “Web Forms Master Page” *or* Web Forms Master Page (Nested)” in the center pane.
   c. Enter a meaningful name for the master page in the “Name” textbox (e.g., ASUWebSite).
3. Click the Add button.

Visual Studio creates the master page, adds it to the website and opens it in its own tab in the `Document` window.

Editing a Master Page

A master page is edited in the `Document` window. You open a master page by double-clicking it in the `Solution Explorer`. Thereafter, text, HTML tags and server controls can be added to the page in a way similar to how they are added to a web form as discussed in the next section.

For our example website do the following.
1. Open the default master page called Site.master.
2. Change its layout as shown in Figure 7 by doing the following.
   a. Replace the text “Application name” with “ASU Web Site”.
   b. In the footer area under the MainContent area, change the text “My ASP.NET Application” to “ASU”.

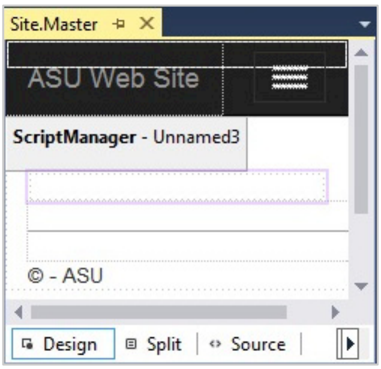


Figure 7: ASU website master page.

Web Forms

As stated previously, a web form is one of the three different programming models used to create ASP.NET web applications. A web form is a page that a user requests through a browser and that forms the user interface (UI) of the web application. These pages are written using a combination of HTML, server controls and server code. When users request a page, it is compiled and executed on the server, which then generates the HTML markup that the browser can render. Visual Studio lets you create web pages by dragging and dropping server controls onto a web form to lay out a web page. Properties, methods and events for server controls or for the page can then be set in order to define the page's behavior, look and feel. Server code to handle the logic for the web page can be written using any .NET language.

Every web form in ASP.NET inherits all of the methods and properties of the Page class, which resides in the `System.Web.UI` namespace. The Page class exposes many properties and methods, which you can make use of in your applications, such as access to server functionality or information about the controls used on a web form. Many of these properties and methods are inherited from the Control class, from which the Page class is derived, or the Object class, from which every class is derived.

A web form is developed in the `Document` window of Visual Studio by placing text and controls onto the design surface and then writing code to handle user interactions, events, etc.[14] Thus, creating a web form in Visual Studio consists of the following three steps.

1. Layout the user interface by adding text and controls to a web form.
2. Set the properties of the controls.
3. Write the event handler code (i.e., procedure) for each control (if necessary).

Creating a Web Form

When you create a new web application project, a default web form named Default.aspx is added to your web application as shown in Figure 5. However, a website can contain many web forms providing different user interfaces.

Add a new web form to our example website by doing the following.

1. Add a new item to the project either from the `Solution Explorer`, the Visual Studio menu bar or by pressing Crtl+Shift+A.
2. In the `Add New Item` window shown in Figure 6, do the following.
   a. Select “Visual C#→Web” in the left pane, under “Installed”.
   b. Select “Web Form with Master Page” in the center pane.
   c. Enter “StudentRegistrationPage.aspx” in the “Name” field.
   d. Click the Add button.
3. In the `Select a Master Page` window, select “Site.master” in the right pane.
4. Click the OK button.

Visual Studio creates the web form, adds it to the website and opens it in its own tab in the `Document` window as shown in Figure 8.

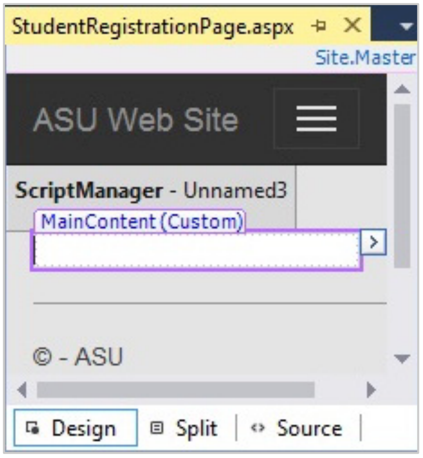


Figure 8: StudentRegistrationPage.aspx web form.

Editing the Name of a Web Form

The name of a web form can be set at the time that it is created (see Figure 6) or it can be changed after it is created. To change the name of a web form highlight it in the Solution Explorer and type a new name for it or right-click it in the Solution Explorer and select “Rename” from the pop-up menu and type a new name for it.

ASP.NET CONTROLS

Controls, which are the building blocks of ASP.NET applications, provide pre-specified functionality, such as graphical user interface (GUI) and validation components. ASP.NET provides many built-in server controls[15] that you can use to create websites. In addition, you can define your own user controls. Almost all built-in server controls have properties that can be set to control things like appearance, validation, etc.[16]

Web Server Controls

The Web Server controls provide the GUI components for an ASP.NET application and are available from the Standard palette of the Toolbox[17]. The following properties apply to all Web Server controls that inherit from the WebControl class.

- *AccessKey* – The control's keyboard shortcut key.

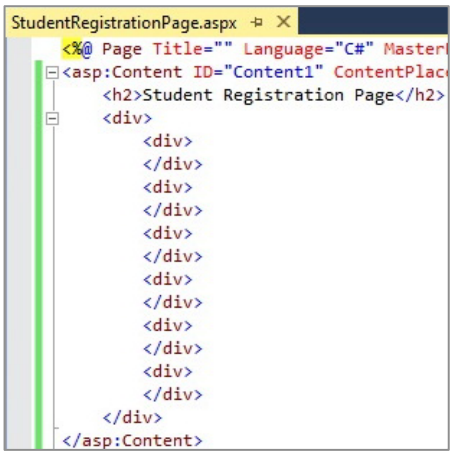
[14] A web page can be developed either visually in the Web Forms Designer or textually using the source code editor in Visual Studio.
[15] Server controls are controls that are executed at the server rather than at the client.
[16] Detailed information about each control and its properties can be obtained from https://msdn.microsoft.com/en-us/library/zsyt68f1%28v=vs.100%29.aspx.
[17] See Appendix A. Other Toolbox controls will be discussed in subsequent tutorials and labs.

- *Attributes* – The collection of additional attributes that can only be set programmatically.
- *BackColor* – The background colour of the control.
- *BorderColor* – The border colour of the control.
- *BorderStyle* – The control's border style, if any.
- *BorderWidth* – The width of the control's border (if any) in pixels.
- *ClientIDMode* – Indicates how the control’s ClientID should be generated.
- *CssClass* – The control’s cascading style sheets (CSS) class.
- *Enabled* – Makes the control functional when set to true (default) or dimmed and inactive when false.
- *EnableTheming* – Indicates whether themes apply to the control.
- *EnableViewState* – Indicates whether the control persists its view state.
- *Font* – The font information for the control.
- *ForeColor* – The foreground colour of the control.
- *Height* – The height of the control.
- *SkinID* – The SkinId of the control skin that provides the skin of the control.
- *TabIndex* – The control's position in the tab order.
- *ToolTip* – The text that appears when the user rests the mouse pointer over the control.
- *ViewStateMode* – Determines whether the control has viewstate enabled or not.
- *Visible* – Indicates whether the control is visible and rendered or hidden.
- *Width* – The width of the control.

Adding Text and Web Server Controls to a Web Form

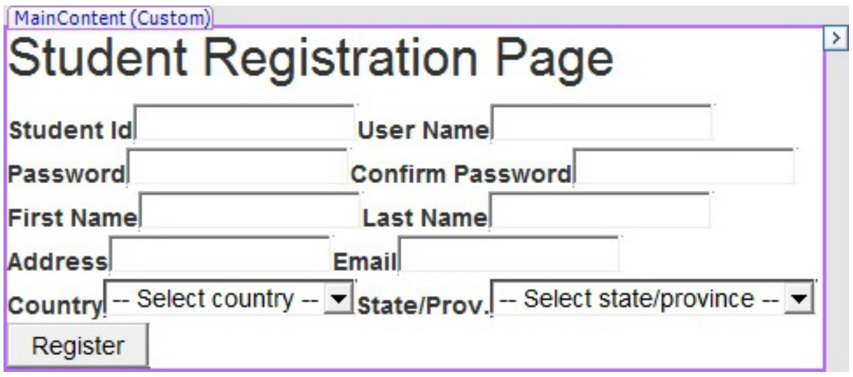
We will place some text and Web Server controls on the StudentRegistrationPage.aspx web form and add some code to programmatically change the items that are displayed in a DropDownList control. As shown in Figure 8, the web form contains a dashed rectangle labeled MainContent(Custom), which represents a ContentPlaceHolder control that is inherited from the master page. Click inside the ContentPlaceHolder control and add the following text and controls.

1. Enter the text “Student Registration Page” on the first line, select the text and then select “Heading 2 <h2>” from the dropdown list in the Formatting toolbar.
2. Add a `<div>` tag from the HTML palette of the Toolbar.
3. Click inside the `<div>` tag and add six more `<div>` tags inside the first one as shown in Figure 9(a).
4. Inside each of the six `<div>` elements add the controls from the Standard palette[18] as given in Table 1 and shown in Figure 9(b). To add a control to the web form, double-click it or drag and drop it onto the web form. The control is placed at the cursor location.


StudentRegistrationPage.aspx

```
<%@ Page Title="" Language="C#" Master
<asp:Content ID="Content1" ContentPlac
    <h2>Student Registration Page</h2>
    <div>
        <div>
        </div>
        <div>
        </div>
        <div>
        </div>
        <div>
        </div>
        <div>
        </div>
        <div>
        </div>
    </div>
</asp:Content>
```

(a) HTML code.



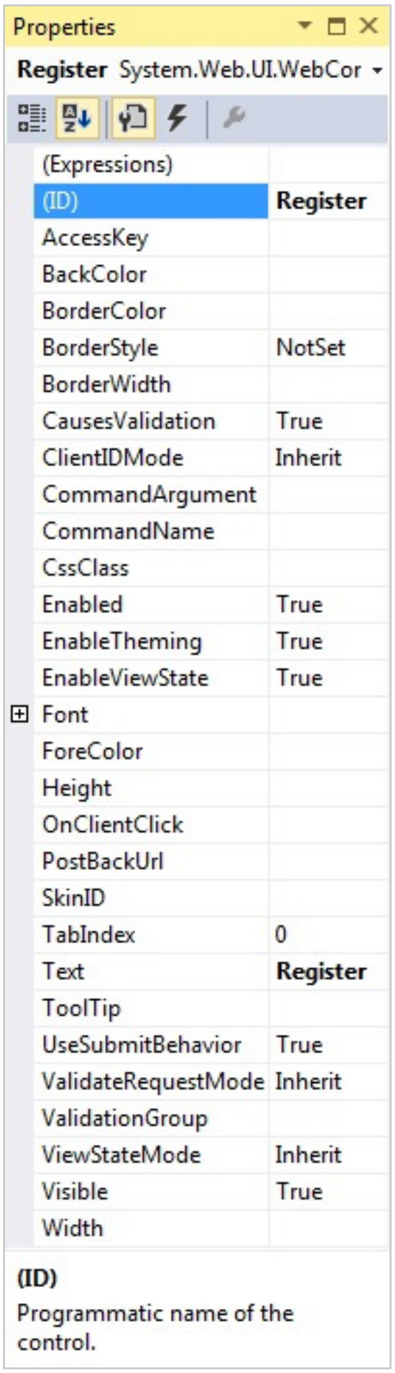
(b) Design layout.

Figure 9: StudentRegistrationPage.aspx web form.

[18] If the `Toolbox` is not visible, it can be made visible by selecting “View→Toolbox” in the Visual Studio menu bar or pressing Ctrl+Alt+X. It can be kept open by selecting its pushpin in the upper right corner of the `Toolbox` window as shown in Figure 5.

Table 1: Properties of the controls in the StudentRegistrationPage.aspx web form.

<div> element	**Control**	**Property**	**Value**[19]
1	Label	ID	
		AssociatedControlID	StudentId
		Text	Student Id
	TextBox	ID	StudentId
	Label	ID	
		AssociatedControlID	UserName
		Text	User Name
	TextBox	ID	UserName
2	Label	ID	
		AssociatedControlID	Password
		Text	Password
	TextBox	ID	Password
		TextMode	Password
	Label	ID	
		AssociatedControlID	ConfirmPassword
		Text	Confirm Password
	Textbox	ID	ConfirmPassword
		TextMode	Password
3	Label	ID	
		AssociatedControlID	FirstName
		Text	First Name
	TextBox	ID	FirstName
	Label	ID	
		AssociatedControlID	LastName
		Text	Last Name
	TextBox	ID	LastName
4	Label	ID	
		AssociatedControlID	Address
		Text	Address
	TextBox	ID	Address
	Label	ID	
		AssociatedControlID	Email
		Text	Email
	TextBox	ID	Email
		TextMode	Email
5	Label	ID	
		AssociatedControlID	Country
		Text	Country
	DropDownList	ID	Country
	Label	ID	
		AssociatedControlID	State
		Text	State/Prov.
	DropDownList	ID	State
6	Button	ID	Register
		Text	Register



(Expressions)	
(ID)	**Register**
AccessKey	
BackColor	
BorderColor	
BorderStyle	NotSet
BorderWidth	
CausesValidation	True
ClientIDMode	Inherit
CommandArgument	
CommandName	
CssClass	
Enabled	True
EnableTheming	True
EnableViewState	True
⊞ Font	
ForeColor	
Height	
OnClientClick	
PostBackUrl	
SkinID	
TabIndex	0
Text	**Register**
ToolTip	
UseSubmitBehavior	True
ValidateRequestMode	Inherit
ValidationGroup	
ViewStateMode	Inherit
Visible	True
Width	

(ID)
Programmatic name of the control.

Figure 10: The Button control's `Properties` window.

[19] Enter the ID property value of the controls *exactly* as given in the table; otherwise errors will arise in future labs. The ID property of all Label controls should be empty.

Setting the Properties of a Control

Controls have various properties that you can set. To set the properties of a control do the following.

1. Select the control on the web form.
2. Set or modify the desired properties in the `Properties` window[20]. A description of the selected property appears in the bottom pane of the `Properties` window.

Use the `Properties` window to set the properties of the added controls as shown in Table 1[21]. Figure 10 shows the `Properties` window for the Button control in our example web form.

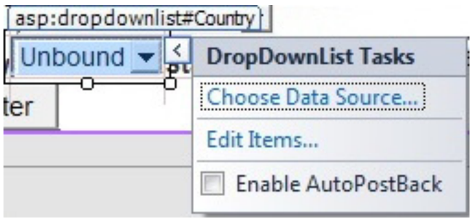
Adding Items to a DropDownList

To add items to a DropDownList control do the following.

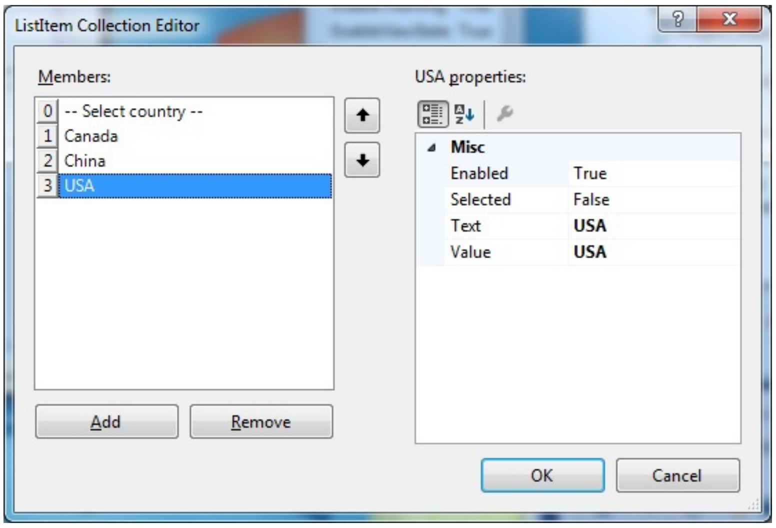
1. Click the control's smart tag[22].
2. Select "Edit Items…" from the drop down menu shown in Figure 11(a).

This opens the `ListItem Collection Editor` window in which you can use the Add button to add items to the DropDownList control.

Add four items to the ListItem Collection of the Country DropDownList control as shown in Figure 11(b). Set their Text properties to "-- Select country --", "Canada", "China" and "USA". Their Value property will automatically be set to the same value as the Text property by default. When you are done, click OK. Similarly, add one item "-- Select state/province --" to the ListItem Collection of the State DropDownList control. Items can also be added to/changed in a DropDownList control programmatically as discussed next.



(a) Select "Edit Items…" in the drop down menu.



Members	
0	-- Select country --
1	Canada
2	China
3	USA

Misc	
Enabled	True
Selected	False
Text	USA
Value	USA

(b) Add items in the `ListItem Collection Editor` window.

Figure 11: Adding items to a DropDownList control.

Adding Event Handler Code for a Control

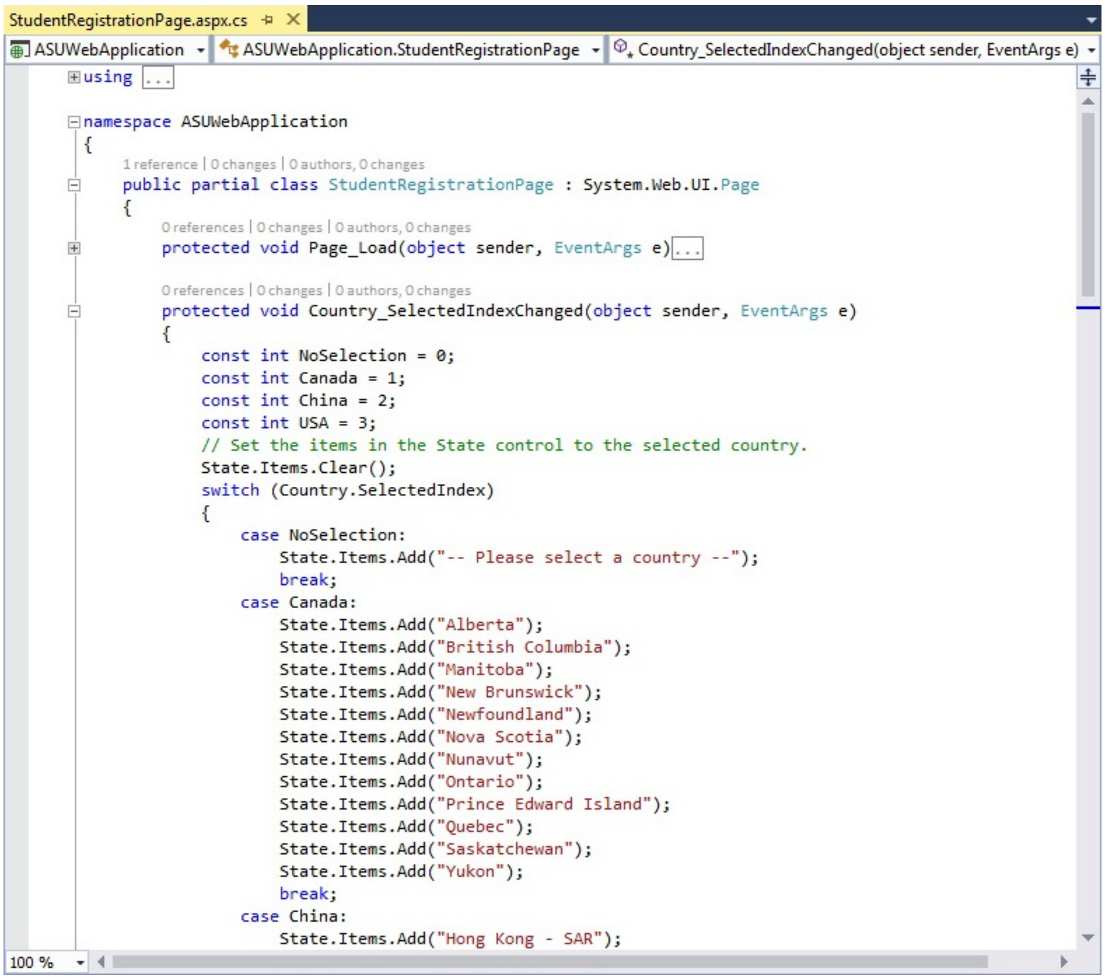
To add event handler code for a control do the following.

1. Double-click the control. This will open the code-behind file[23] for the web form in the `Document` window *and* create an event handler (i.e., procedure) for the control.
2. Type the code in the appropriate event handler associated with the control.

For our example web form, the values of the states/provinces in the State DropDownList control should change whenever a country is selected in the Country DropDownList control.

Double-click the Country DropDownList control in the `Document` window. Add the C# code in the file ASPNETTutorialCode to the Country_SelectedIndexChanged event handler[24]. This code changes the items in the State DropDownList control to match the country selected in the Country DropDownList control.[25]

[20] If the `Properties` window is not visible, right-click the control and select "Properties" from the pop-up menu or press alt+Enter.
[21] Meaningful names should be assigned to all controls on a web form that are programmatically accessed.
[22] A smart tag is the symbol ⊡ that appears in the top right corner of a control.
[23] Visual Studio supports two coding models: *inline* in which both the code and HTML for a web page are stored in a single .aspx file and *code-behind* in which each web page is separated into a .aspx file with the HTML and control tags and a code file with the source code for the page. It is required that you use the code-behind model in this course.



StudentRegistrationPage.aspx.cs

ASUWebApplication | ASUWebApplication.StudentRegistrationPage | Country_SelectedIndexChanged(object sender, EventArgs e)

```
using ...

namespace ASUWebApplication
{
    1 reference | 0 changes | 0 authors, 0 changes
    public partial class StudentRegistrationPage : System.Web.UI.Page
    {
        0 references | 0 changes | 0 authors, 0 changes
        protected void Page_Load(object sender, EventArgs e)...

        0 references | 0 changes | 0 authors, 0 changes
        protected void Country_SelectedIndexChanged(object sender, EventArgs e)
        {
            const int NoSelection = 0;
            const int Canada = 1;
            const int China = 2;
            const int USA = 3;
            // Set the items in the State control to the selected country.
            State.Items.Clear();
            switch (Country.SelectedIndex)
            {
                case NoSelection:
                    State.Items.Add("-- Please select a country --");
                    break;
                case Canada:
                    State.Items.Add("Alberta");
                    State.Items.Add("British Columbia");
                    State.Items.Add("Manitoba");
                    State.Items.Add("New Brunswick");
                    State.Items.Add("Newfoundland");
                    State.Items.Add("Nova Scotia");
                    State.Items.Add("Nunavut");
                    State.Items.Add("Ontario");
                    State.Items.Add("Prince Edward Island");
                    State.Items.Add("Quebec");
                    State.Items.Add("Saskatchewan");
                    State.Items.Add("Yukon");
                    break;
                case China:
                    State.Items.Add("Hong Kong - SAR");
```

100 %

Figure 12: Country_SelectedIndexChanged event handler code.

Testing a Web Form

To view our example web form in a web browser, first save your work and then either select "Debug→Start Debugging" in the Visual Studio menu bar, press the F5 key or right-click the web form in the `Solution Explorer` and select "View In Browser" from the dropdown list.

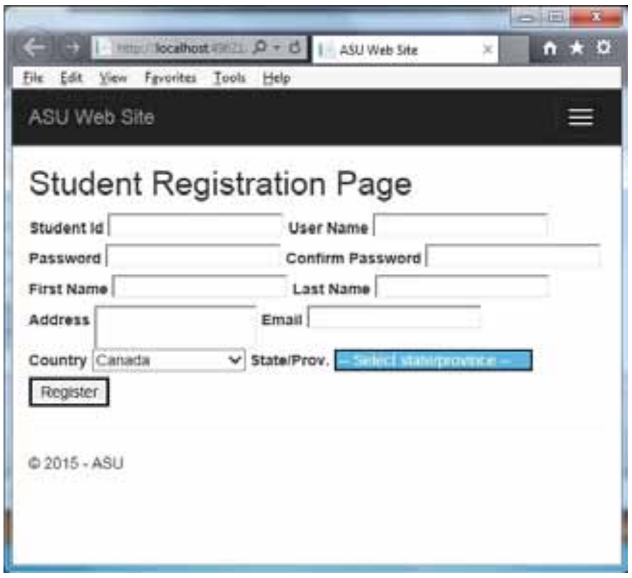
What happens when you change the country value in the country dropdown list? To execute the code associated with a DropDownList control, the web page needs to be posted back (i.e., run on the server and the results posted back (returned) to the same page)[26]. However, you *do not* need to write any code to perform a postback. Instead, you simply set the AutoPostBack property of the Country DropDownList control to "True" to do the postback. This property can be set either in the `Properties` window of the DropDownList control or by selecting the control's smart tag and checking "Enable AutoPostBack" (see Figure 11(a)). After enabling postback, when you select a country, the correct states/provinces for that country are available for selection in the State DropDownList control as shown Figure 13(b).

Close the web browser, select the StudentRegistrationPage.aspx tab and either click on the Source tab at the bottom left of the `Document` window, select "View→Markup" in the Visual Studio menu bar or press Shift+F7 to view the HTML code for the StudentRegistrationPage.aspx file as shown in Figure 14.

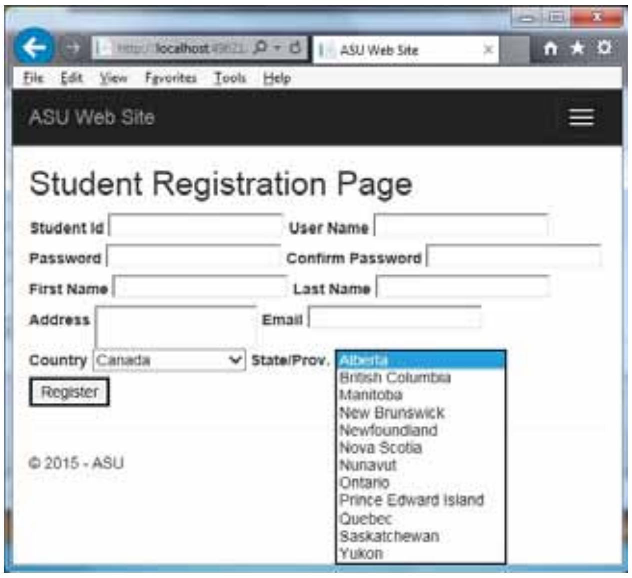
Figure 15 shows the HTML code of the Site.master master page. Notice that the master page contains the HTML tags <head> and <body>, which are lacking on the StudentRegistrationPage.aspx web form. At runtime, a web form page is merged with its master page and so will inherit the HTML tags required to display the page correctly in a browser.

---

[24] The first item in a DropDownList has index 0.
[25] Note that the StudentRegistrationPage class is a partial class and that the web form inherits from the `System.Web.UI.Page` class. A partial class is a class whose source code is split into two or more source code files. This allows you to hide messy details you do not want to see. Visual Studio uses partial classes to sometimes hide automatically generated code.
[26] For more information about post backs, see the article How postback works in ASP.NET.

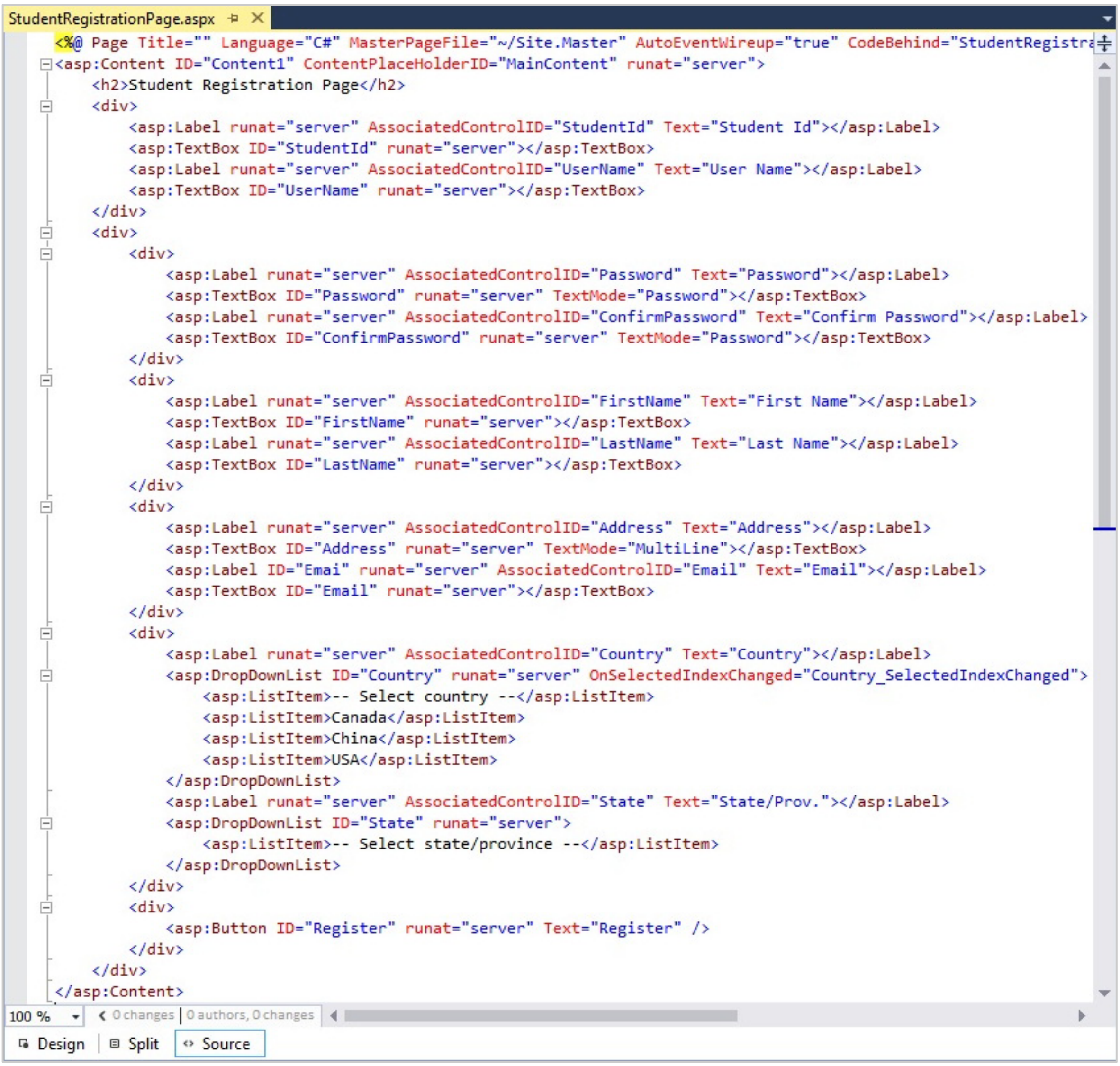


(a) Before enabling Postback.



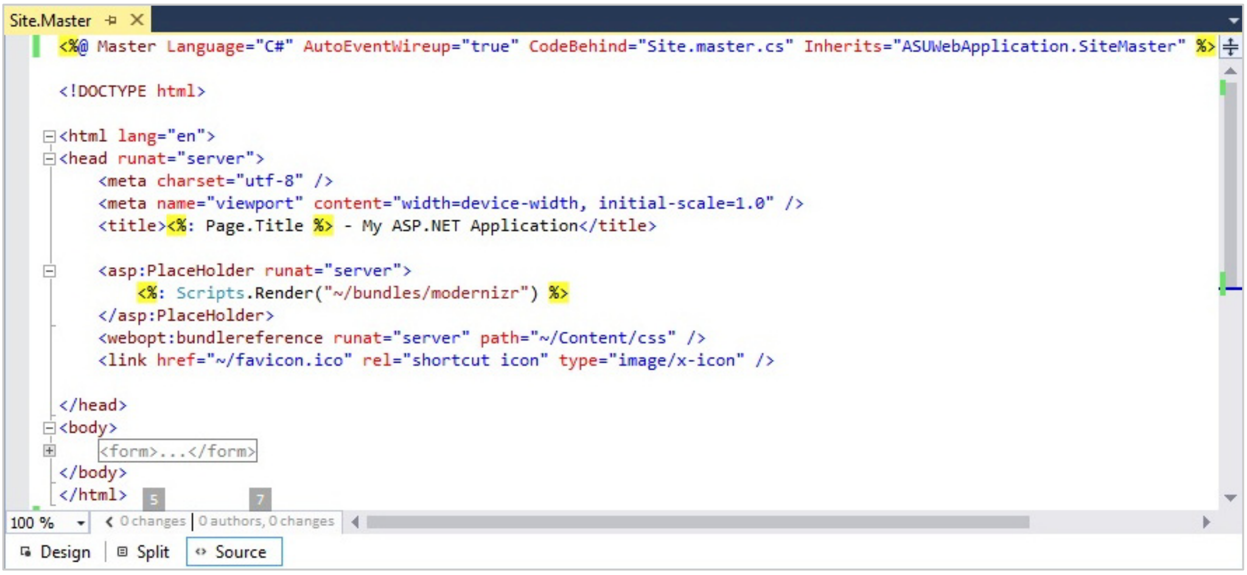
(b) After enabling Postback.

Figure 13: The web form in a browser.



```
<%@ Page Title="" Language="C#" MasterPageFile="~/Site.Master" AutoEventWireup="true" CodeBehind="StudentRegistra
<asp:Content ID="Content1" ContentPlaceHolderID="MainContent" runat="server">
    <h2>Student Registration Page</h2>
    <div>
        <asp:Label runat="server" AssociatedControlID="StudentId" Text="Student Id"></asp:Label>
        <asp:TextBox ID="StudentId" runat="server"></asp:TextBox>
        <asp:Label runat="server" AssociatedControlID="UserName" Text="User Name"></asp:Label>
        <asp:TextBox ID="UserName" runat="server"></asp:TextBox>
    </div>
    <div>
        <div>
            <asp:Label runat="server" AssociatedControlID="Password" Text="Password"></asp:Label>
            <asp:TextBox ID="Password" runat="server" TextMode="Password"></asp:TextBox>
            <asp:Label runat="server" AssociatedControlID="ConfirmPassword" Text="Confirm Password"></asp:Label>
            <asp:TextBox ID="ConfirmPassword" runat="server" TextMode="Password"></asp:TextBox>
        </div>
        <div>
            <asp:Label runat="server" AssociatedControlID="FirstName" Text="First Name"></asp:Label>
            <asp:TextBox ID="FirstName" runat="server"></asp:TextBox>
            <asp:Label runat="server" AssociatedControlID="LastName" Text="Last Name"></asp:Label>
            <asp:TextBox ID="LastName" runat="server"></asp:TextBox>
        </div>
        <div>
            <asp:Label runat="server" AssociatedControlID="Address" Text="Address"></asp:Label>
            <asp:TextBox ID="Address" runat="server" TextMode="MultiLine"></asp:TextBox>
            <asp:Label ID="Emai" runat="server" AssociatedControlID="Email" Text="Email"></asp:Label>
            <asp:TextBox ID="Email" runat="server"></asp:TextBox>
        </div>
        <div>
            <asp:Label runat="server" AssociatedControlID="Country" Text="Country"></asp:Label>
            <asp:DropDownList ID="Country" runat="server" OnSelectedIndexChanged="Country_SelectedIndexChanged">
                <asp:ListItem>-- Select country --</asp:ListItem>
                <asp:ListItem>Canada</asp:ListItem>
                <asp:ListItem>China</asp:ListItem>
                <asp:ListItem>USA</asp:ListItem>
            </asp:DropDownList>
            <asp:Label runat="server" AssociatedControlID="State" Text="State/Prov."></asp:Label>
            <asp:DropDownList ID="State" runat="server">
                <asp:ListItem>-- Select state/province --</asp:ListItem>
            </asp:DropDownList>
        </div>
        <div>
            <asp:Button ID="Register" runat="server" Text="Register" />
        </div>
    </div>
</asp:Content>
```

Figure 14: HTML of the StudentRegistrationPage.aspx web form.



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

<!DOCTYPE html>

<html lang="en">
<head runat="server">
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title><%: Page.Title %> - My ASP.NET Application</title>

    <asp:PlaceHolder runat="server">
        <%: Scripts.Render("~/bundles/modernizr") %>
    </asp:PlaceHolder>
    <webopt:bundlereference runat="server" path="~/Content/css" />
    <link href="~/favicon.ico" rel="shortcut icon" type="image/x-icon" />

</head>
<body>
    <form>...</form>
</body>
</html>
```

Figure 15: HTML of the Site.master master page.

STYLING AN ASP.NET WEBSITE USING BOOTSTRAP

While our web page contains the required information, its appearance, particularly the alignment and spacing of the controls, could be improved. You can try to improve the page's appearance yourself using HTML. A better solution, however, is to use Bootstrap[27], a free, open-source framework for styling a website, which is included in Visual Studio. Bootstrap consists of CSS (cascading style sheets) and JavaScript components and implements a host of best practices and modern design guidelines. It supports *responsive design*, which refers to the ability of a website to adapt to the size of the browser window—whether it's viewed on a large, high-resolution desktop or a small screen phone. Besides making text wrap appropriately, the *way* objects appear on the page changes depending on the available screen size. Moreover, once a web page has been marked up using Bootstrap, it is easy to change its look simply by replacing a CSS file. To enable Bootstrap to style a web page, CSS markup needs to be added to the web form. This can be done either entirely in the Source view of a web form or partly in the Design view and partly in the Source view[28].

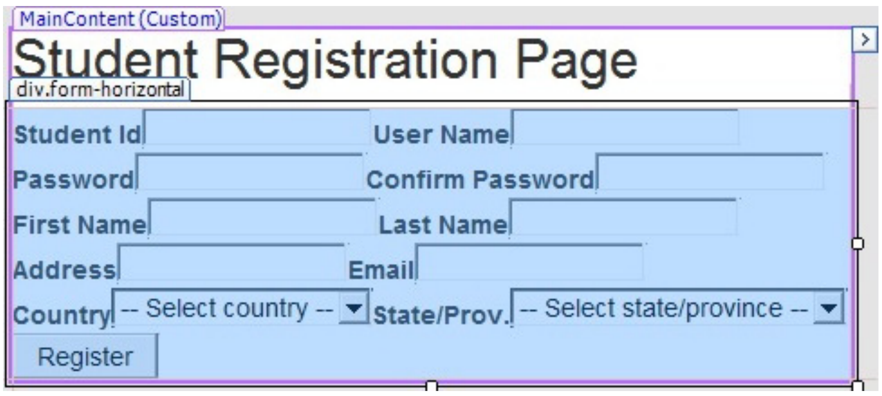
We will demonstrate how to use the Design view to markup a web form to use Bootstrap's horizontal form layout and show what needs to be completed using the Source view. Of Bootstrap's 12-column grid, two columns will be used for the Label controls and four columns for the Textbox and DropDownList controls.

1. Place the cursor inside any of the `<div>` tags in the web form.
2. At the bottom right of the `Document` window, select the left-most `<div>` tag so that all the ASP controls in the page are highlighted as shown in Figure 16(a).
3. In the `Properties` window, select "form-horizontal" from the class property dropdown list as shown in Figure 16(b)[29]. The `<div>` tag on the web form should now appear as shown in Figure 16(a).
4. For each of the `<div>` tags contained within the top-level `<div>` tag, do the following.
   a. Select the `<div>` tag as shown in Figure 16(c).
   b. In the `Properties` window, select "form-group" from the class property dropdown list as shown in Figure 16(d) so that the `<div>` tag now appears as shown in Figure 16(c).
5. For each Label control in the web form, do the following.
   a. Select the Label control to highlight it.
   b. In the `Properties` window, select "control-label" from the CssClass property dropdown list.
   c. With the cursor in the CssClass property field, enter a space and then "col-md-2"[30].
6. For each Textbox and DropDownList control in the web form, do the following.
   a. Select the control to highlight it.
   b. In the `Properties` window, select "form-control" from the CssClass property dropdown list.
   c. In Source view, wrap each control in `<div class="col-md-4">` … `</div>` tags[31].

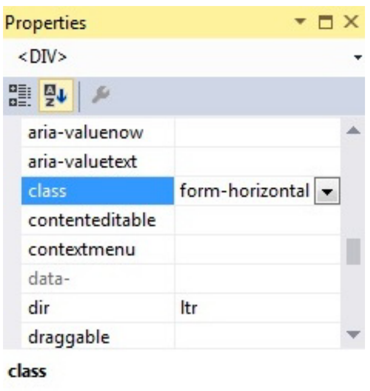
[27] See Appendix B for a brief overview of Bootstrap.
[28] Visual Studio's Web Form designer does not support CSS3, so it does not accurately show all the effects of Bootstrap markup. However, a web form will display correctly when viewed in a browser.
[29] If the dropdown list is empty, the class property value can be entered directly.
[30] The CssClass property dropdown list allows only one value to be selected. Additional values can be entered directly for the property.
[31] ASP.NET input controls, such as Textbox and DropDownList, should be wrapped in `<div>` tags, while Label controls should not.

7. For the button control, do the following.
   a. In the `Properties` window, select “btn” from the CssClass property dropdown list.
   b. With the cursor in the CssClass property field, enter a space and then “button-default”.
   c. In Source view, wrap the Button control in a `<div class=”col-md-offset-2 col-md-10”>` … `</div>` tag

The web page should now look as shown in Figure 17 when viewed in a browser.



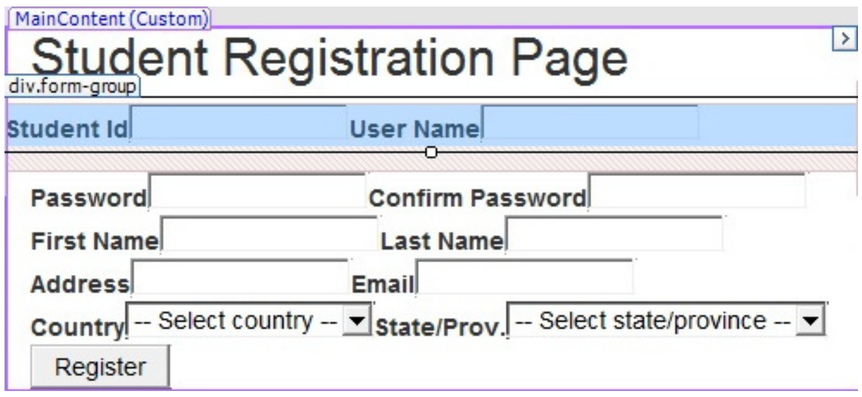
(a) CSS class for enclosing `<div>` tag.



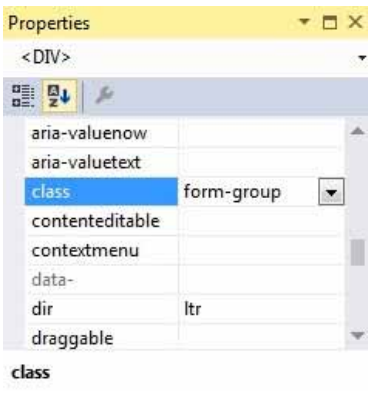
aria-valuenow
aria-valuetext
class: form-horizontal
contenteditable
contextmenu
data-
dir: ltr
draggable

class

(b) `Properties` window.



(c) CSS class for each line of controls.

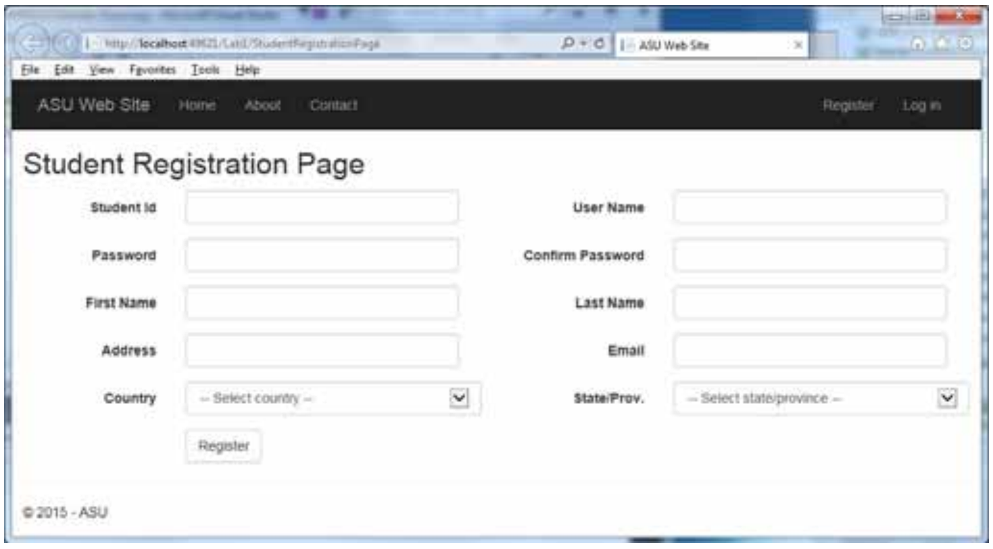


aria-valuenow
aria-valuetext
class: form-group
contenteditable
contextmenu
data-
dir: ltr
draggable

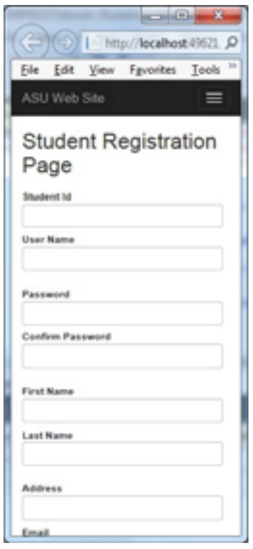
class

(d) `Properties` window.

Figure 16: Adding Bootstrap markup to the StudentRegistrationPage.aspx web form.

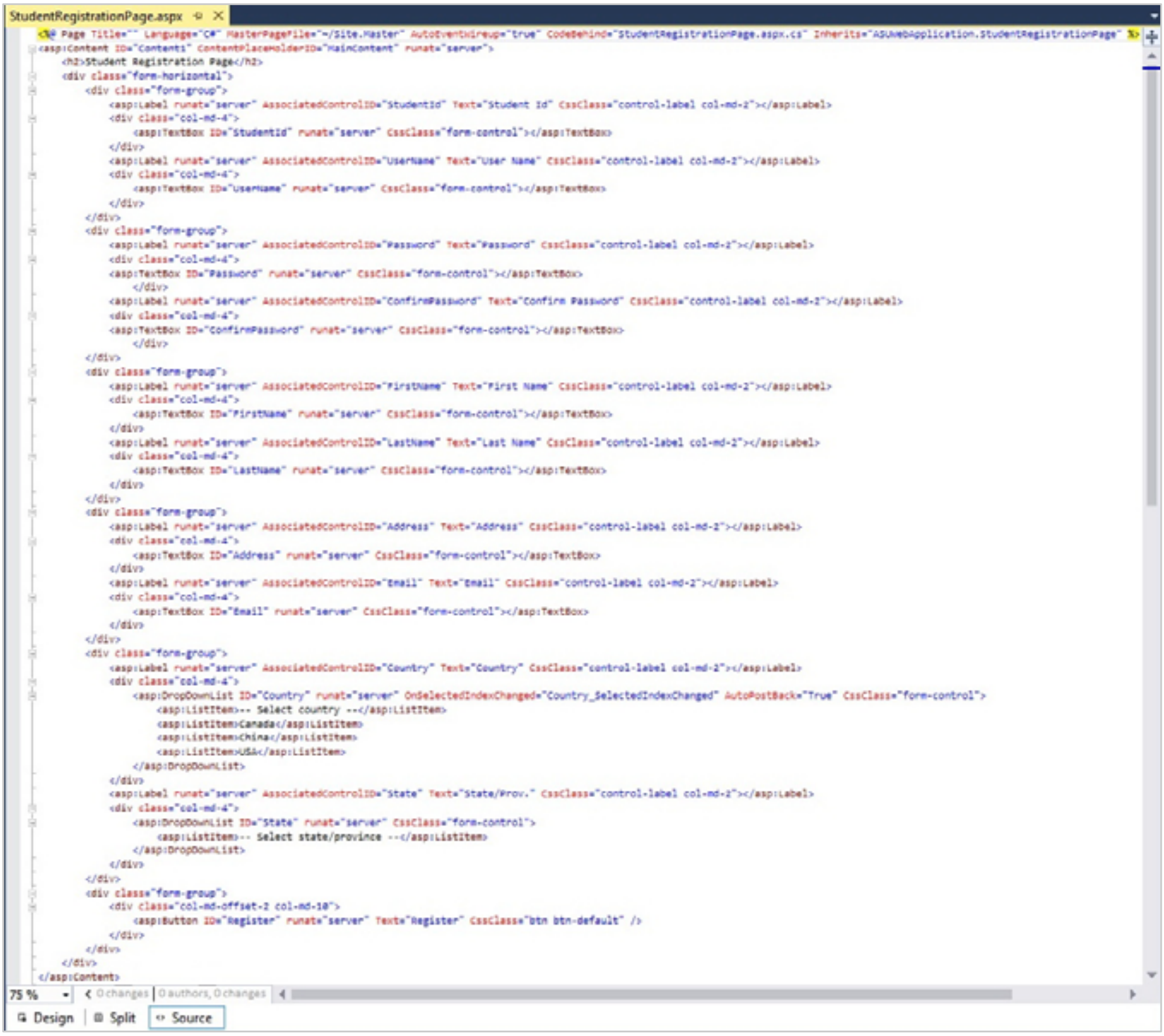


(a) Normal view.



(b) Resized view.

Figure 17: StudentRegistrationPage.aspx web form viewed with Bootstrap markup.



```
<%@ Page Title="" Language="C#" MasterPageFile="~/Site.Master" AutoEventWireup="true" CodeBehind="StudentRegistrationPage.aspx.cs" Inherits="ASUWebApplication.StudentRegistrationPage" %>
<asp:Content ID="Content1" ContentPlaceHolderID="MainContent" runat="server">
    <h2>Student Registration Page</h2>
    <div class="form-horizontal">
        <div class="form-group">
            <asp:Label runat="server" AssociatedControlID="StudentId" Text="Student Id" CssClass="control-label col-md-2"></asp:Label>
            <div class="col-md-4">
                <asp:TextBox ID="StudentId" runat="server" CssClass="form-control"></asp:TextBox>
            </div>
            <asp:Label runat="server" AssociatedControlID="UserName" Text="User Name" CssClass="control-label col-md-2"></asp:Label>
            <div class="col-md-4">
                <asp:TextBox ID="UserName" runat="server" CssClass="form-control"></asp:TextBox>
            </div>
        </div>
        <div class="form-group">
            <asp:Label runat="server" AssociatedControlID="Password" Text="Password" CssClass="control-label col-md-2"></asp:Label>
            <div class="col-md-4">
            <asp:TextBox ID="Password" runat="server" CssClass="form-control"></asp:TextBox>
                </div>
            <asp:Label runat="server" AssociatedControlID="ConfirmPassword" Text="Confirm Password" CssClass="control-label col-md-2"></asp:Label>
            <div class="col-md-4">
            <asp:TextBox ID="ConfirmPassword" runat="server" CssClass="form-control"></asp:TextBox>
                </div>
        </div>
        <div class="form-group">
            <asp:Label runat="server" AssociatedControlID="FirstName" Text="First Name" CssClass="control-label col-md-2"></asp:Label>
            <div class="col-md-4">
                <asp:TextBox ID="FirstName" runat="server" CssClass="form-control"></asp:TextBox>
            </div>
            <asp:Label runat="server" AssociatedControlID="LastName" Text="Last Name" CssClass="control-label col-md-2"></asp:Label>
            <div class="col-md-4">
                <asp:TextBox ID="LastName" runat="server" CssClass="form-control"></asp:TextBox>
            </div>
        </div>
        <div class="form-group">
            <asp:Label runat="server" AssociatedControlID="Address" Text="Address" CssClass="control-label col-md-2"></asp:Label>
            <div class="col-md-4">
                <asp:TextBox ID="Address" runat="server" CssClass="form-control"></asp:TextBox>
            </div>
            <asp:Label runat="server" AssociatedControlID="Email" Text="Email" CssClass="control-label col-md-2"></asp:Label>
            <div class="col-md-4">
                <asp:TextBox ID="Email" runat="server" CssClass="form-control"></asp:TextBox>
            </div>
        </div>
        <div class="form-group">
            <asp:Label runat="server" AssociatedControlID="Country" Text="Country" CssClass="control-label col-md-2"></asp:Label>
            <div class="col-md-4">
                <asp:DropDownList ID="Country" runat="server" OnSelectedIndexChanged="Country_SelectedIndexChanged" AutoPostBack="True" CssClass="form-control">
                    <asp:ListItem>-- Select country --</asp:ListItem>
                    <asp:ListItem>Canada</asp:ListItem>
                    <asp:ListItem>China</asp:ListItem>
                    <asp:ListItem>USA</asp:ListItem>
                </asp:DropDownList>
            </div>
            <asp:Label runat="server" AssociatedControlID="State" Text="State/Prov." CssClass="control-label col-md-2"></asp:Label>
            <div class="col-md-4">
                <asp:DropDownList ID="State" runat="server" CssClass="form-control">
                    <asp:ListItem>-- Select state/province --</asp:ListItem>
                </asp:DropDownList>
            </div>
        </div>
        <div class="form-group">
            <div class="col-md-offset-2 col-md-10">
                <asp:Button ID="Register" runat="server" Text="Register" CssClass="btn btn-default" />
            </div>
        </div>
    </div>
</asp:Content>
```

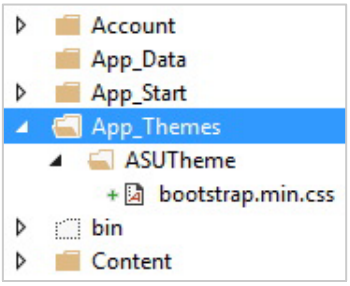
Figure 18: HTML of the StudentRegistrationPage.aspx web form with Bootstrap Markup.

Themes

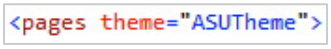
To change the theme of an ASP.NET website, do the following.

1. In the Solution Explorer, right-click the web application node (e.g., "ASUWebApplication") and select "Add→Add ASP.NET Folder→ Theme" from the popup menu. A folder named "App_Themes" and a subfolder named "Theme1" are created.
2. Rename the "Theme1" folder to the name of your theme (e.g., "ASUTheme" as shown in Figure 19(a)).
3. From Windows, drag and drop a downloaded *bootstrap.min.css* file onto the renamed "Theme1" folder in the Solution Explorer (e.g., drag and drop the *bootstrap.min.css* file onto the "ASUTheme" folder shown in Figure 19(a).
4. In the Solution Explorer, open the "Web.config" file and add a theme attribute to the `<pages>` tag and set its value to the name of your theme as shown in Figure 19(b). The theme will now apply to all pages in the entire web site.

View the project in a browser to see the new look.



(a) Add App_Themes folder.

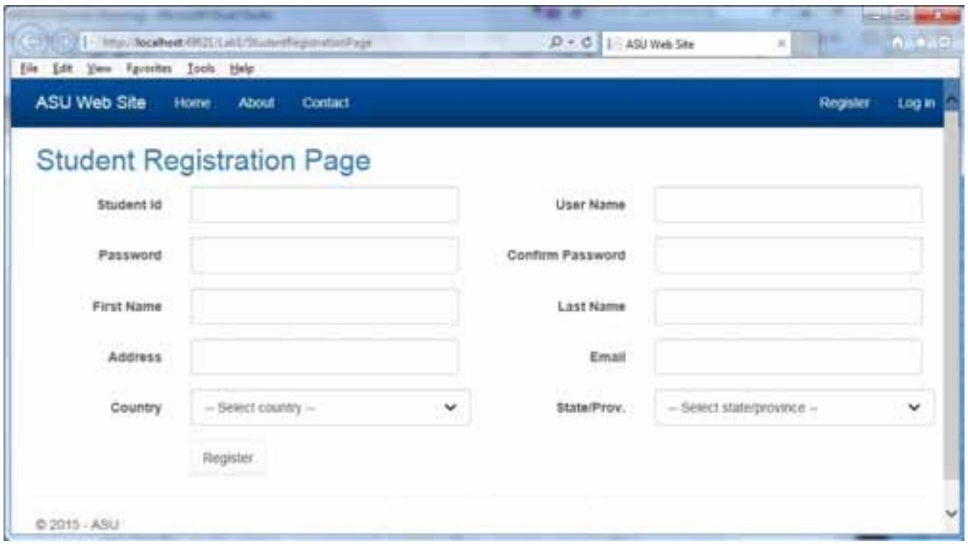


```
<pages theme="ASUTheme">
```

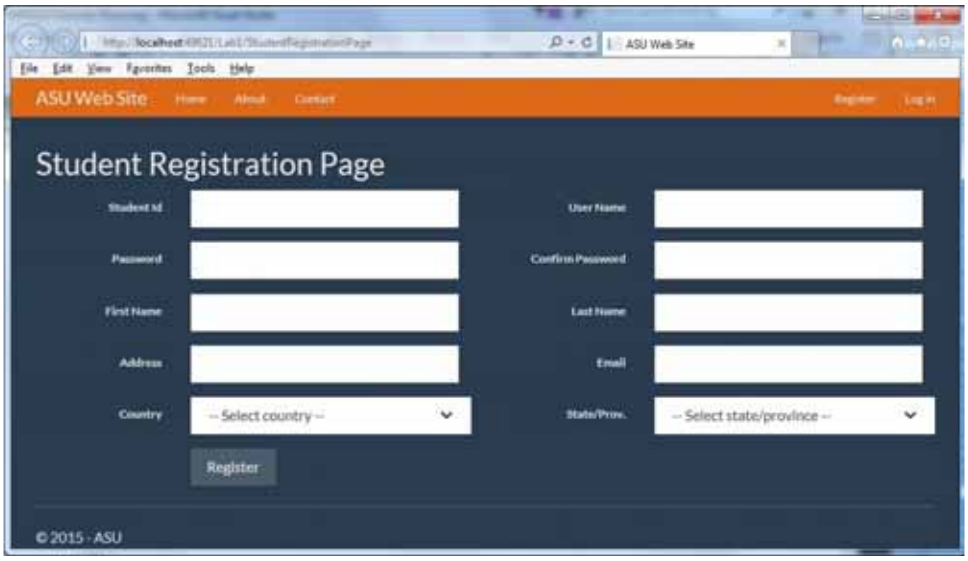
(b) Edit Web.config file.

Figure 19: Adding a Bootstrap stylesheet as a website theme.

Figure 20 shows the effect of two Bootstrap CSS stylesheets used as themes[32].



(a) Cerulean theme.

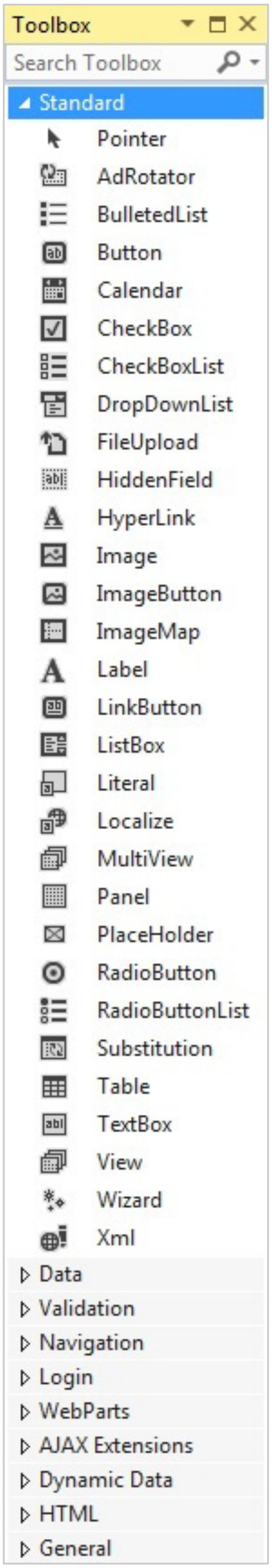


(b) Superhero theme.

Figure 20: Two Bootswatch themes.

[32] See https://msdn.microsoft.com/en-us/library/0yy5hxdk.aspx for more details about how to use ASP.NET themes. Boostrap CSS Stylesheets can be downloaded from http://Bootswatch.com. The stylesheet *bootstrap.min.css* is downloaded by default; if you want to examine the CSS code, download *bootstrap.css* instead of the minified version.

Appendix A: ASP.NET Toolbox Standard Controls[33]



AdRotator

The AdRotator control displays a sequence of images providing a way to display advertisements (ads) on a web page. The ads are selected randomly for display.

BulletedList

The BulletedList control is used to create a list of unordered or ordered (numbered) items. List items can be defined either by creating static items or by binding the control to a data source.

Button

The Button control allows users to post a page to the server and to trigger an event on a page. Three types of button controls are available.

- Button - displays a standard command button.
- LinkButton - displays a hyperlink-style button control having the same appearance as a HyperLink control.
- ImageButton - displays an image and responds to mouse clicks on the image.

Calendar

The Calendar control is used to display a one-month calendar that allows the user to select dates and move to the next and previous months.

CheckBox

The CheckBox control creates a check box that allows the user to switch between a true or false state. It can be bound to a data source.

CheckBoxList

The CheckBoxList control creates a multi-selection check box group. It can be bound to a data source.

DropDownList

The DropDownList control is used to create a single-selection dropdown list that can contain any number of items. It can be bound to a data source.

FileUpload

The FileUpLoad control enables a file to be uploaded to the server by displaying a text box control and a browse button that allow users to select a file to upload.

HiddenField

The HiddenField control is used to store a value in a page, without displaying it, that needs to be persisted across posts to the server.

HyperLink

The HyperLink control creates a link on a web form, displayed as either text or an image, which users can click to move to another page or location on the page.

Image

The Image control displays a Web-compatible image on a web form.

ImageMap

The ImageMap control enables creation of an image that contains defined hotspot

[33] See https://msdn.microsoft.com/en-us/library/x8k61whf%28v=vs.100%29.aspx for more information about the standard controls.

regions. When a user clicks a hot spot region, the control can either generate a post back to the server and run specific code based on the hot spot region that was clicked or navigate to a specified URL.

Label

The Label control displays and allows programmatic manipulation of static text on a web form.

ListBox

The ListBox control creates a single-selection or multi-selection list box that supports data binding.

Literal

The Literal control displays and allows programmatic manipulation of dynamically generated content on a web form. It does not add any HTML tags to the text.

Localize

The Localize control is identical to the Literal control and similar to the Label control. Although the Label control allows a style to be applied to the displayed text, the Localize control does not. The text that is displayed can be programmatically controlled in the Localize control by setting the Text property, which is inherited from the Literal control.

MultiView and View

The MultiView control represents a control that acts as a container for groups of View controls. It allows a group of View controls to be defined, where each View control contains child controls. An application can then render a specific View control to the client based on criteria such as user identity, user preferences, or information passed in a query string parameter.

Panel

The Panel control, which is a container for other controls, can be used for generating controls programmatically and displaying and hiding groups of controls.

PlaceHolder

The PlaceHolder control reserves a location in the page control hierarchy for controls that are added programmatically.

RadioButton and RadioButtonList

The RadioButton control creates an individual radio button on a web form. Multiple radio buttons can be grouped together to provide a mutually exclusive set of choices. The RadioButtonList control creates a single-selection radio button group. Both types of buttons can be bound to a data source.

Substitution

The Substitution control inserts dynamic content into a cached page.

Table

The Table control declares a table and allows it to be manipulated programmatically. Each Table control is made up of rows stored in the Rows collection of the control. Each row is made up of cells stored in the Cells collection of each TableRow.

TextBox

The TextBox control creates either a single-line or multi-line text box to enter text for subsequent processing.

Wizard

The Wizard control provides navigation through a series of steps that collect information from a user.

XML

The Xml control displays an XML document or the results of an XSL transformation.

APPENDIX B: BOOTSTRAP BASICS

Installing Bootstrap

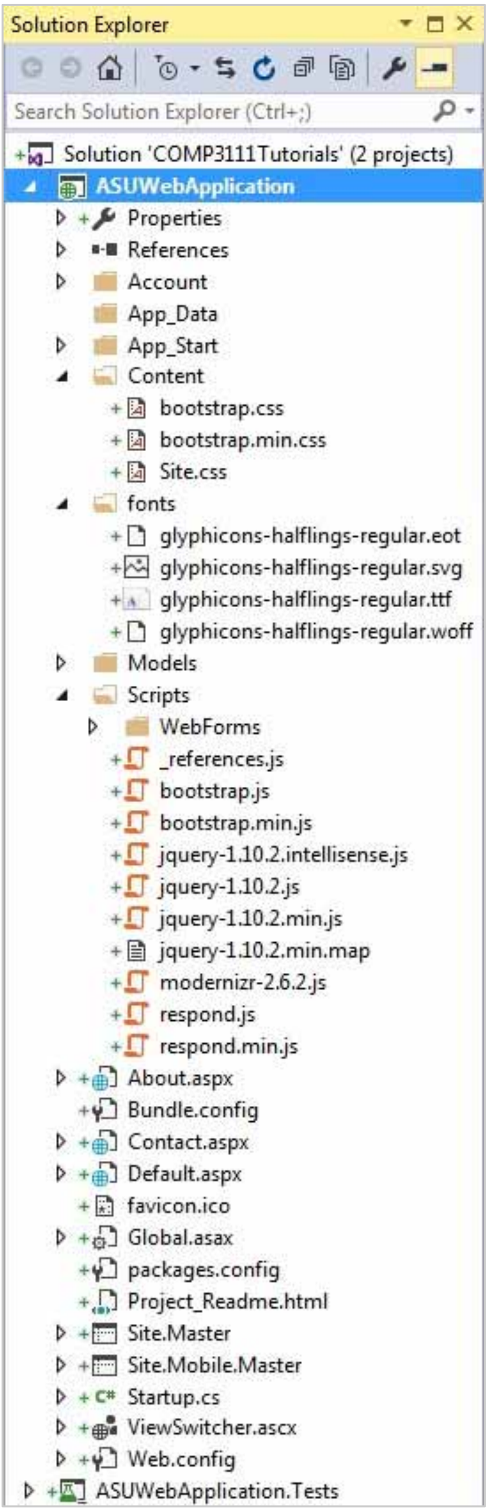
To install Bootstrap in Visual Studio, do the following[34].

1. In the Solution Explorer, right-click on the node “Solution …” (the top-most node) and select “Manage NuGet Packages for Solution…”[35].
4. In the NuGet tab of the `Document` window select the “Browse” tab and enter “bootstrap” into the search box.
5. Select “bootstrap” in the left pane.
6. Click the “Install” button in the right pane.

jQuery is also installed since Bootstrap requires it.

Using Bootstrap in a Project

Bootstrap, which is added to a Visual Studio project by default, adds three folders *Content* (for CSS files), *fonts* and *Scripts* (for javascript files) to a project as shown in Figure 21. The files in these folders are referenced as needed in a web page.



```
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'COMP3111Tutorials' (2 projects)
  ASUWebApplication
    Properties
    References
    Account
    App_Data
    App_Start
    Content
      bootstrap.css
      bootstrap.min.css
      Site.css
    fonts
      glyphicons-halflings-regular.eot
      glyphicons-halflings-regular.svg
      glyphicons-halflings-regular.ttf
      glyphicons-halflings-regular.woff
    Models
    Scripts
      WebForms
      _references.js
      bootstrap.js
      bootstrap.min.js
      jquery-1.10.2.intellisense.js
      jquery-1.10.2.js
      jquery-1.10.2.min.js
      jquery-1.10.2.min.map
      modernizr-2.6.2.js
      respond.js
      respond.min.js
    About.aspx
    Bundle.config
    Contact.aspx
    Default.aspx
    favicon.ico
    Global.asax
    packages.config
    Project_Readme.html
    Site.Master
    Site.Mobile.Master
    Startup.cs
    ViewSwitcher.ascx
    Web.config
  ASUWebApplication.Tests
```

Figure 21: Bootstrap files.

Containers

Bootstrap requires a `<div>` tag, with a container class placed inside it, to wrap an entire page and to house its grid system[36]. A container class is one of:

- *container* – provides a responsive fixed width container;
- *container-fluid* – provides a full width container spanning the entire screen size.

Grid System

Bootstrap uses a responsive, fluid grid system that creates page layouts through a series of rows and 12 columns. It includes predefined classes for easy layout options, as well as powerful mixins for generating more semantic layouts.

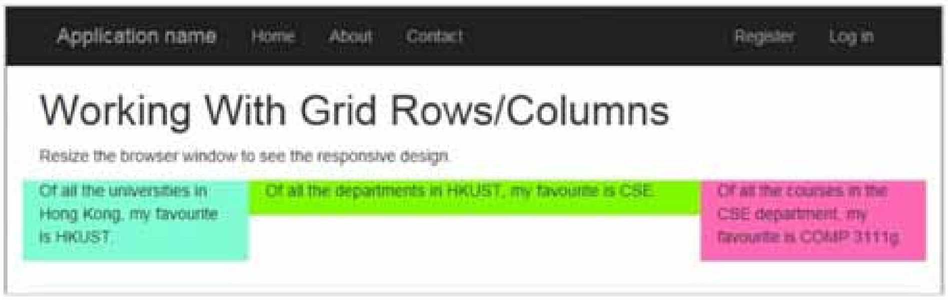
Text or other elements can be placed in any of the 12 columns of the grid, which can be *spanned* in various ways. For example, a three-six-three column layout would be spanned as shown below.

1	2	3	4	5	6	7	8	9	10	11	12
Span-3			Span-6						Span-3		

Figure 22 shows HTML code marked up with Bootstrap classes and the resulting layout[37]. Bootstrap’s responsive design is shown in Figure 23 where the navigation bar becomes a dropdown list and the columns rearrange themselves horizontally as the screen size decreases. Table 2 summarizes some properties of the grid system across different device sizes[38].

[34] Bootstrap is installed by default on the computers in the CSE labs.
[35] NuGet is installed by default on the computers in the CSE labs. NuGet is a free, open-source package management system for Visual Studio that allows third party open-source components to be managed in Visual Studio projects that use the .NET Framework. All installed NuGet packages can be viewed by opening the “packages.config” file in the Solution Explorer.
[36] The Site.master web form contains the tag `<div class="container body-content">`, which wraps the contents of the web forms that inherit from it.
[37] This example is adapted from Hatfield, Bill, “Making Beautiful Style Together: Visual Studio and Bootstrap”.
[38] See http://getbootstrap.com/css/ for complete details of Bootstrap’s grid system, typography and classes.

```
<div class="container-fluid">
   <h1>Working With Grid Rows/Columns</h1>
   <p>Resize the browser window to see the responsive design.</p>
   <div class="row">
      <div class="col-sm-3" style="background-color:aquamarine;">
         <p>Of all the universities in Hong Kong, my favourite is HKUST.</p>
      </div>
      <div class="col-sm-6" style="background-color:chartreuse;">
         <p>Of all the departments in HKUST, my favourite is CSE.</p>
      </div>
      <div class="col-sm-3" style="background-color:hotpink;">
         <p>Of all the courses in the CSE department, my favourite is COMP
            3111.</p>
      </div>
   </div>
</div>
```



Working With Grid Rows/Columns

Resize the browser window to see the responsive design.

Of all the universities in Hong Kong, my favourite is HKUST.

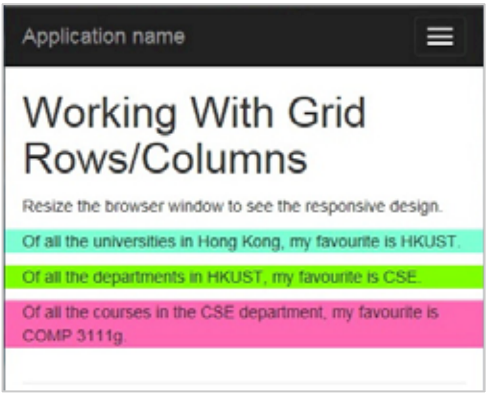
Of all the departments in HKUST, my favourite is CSE.

Of all the courses in the CSE department, my favourite is COMP 3111g.

Figure 22: Bootstrap 3-6-3 column layout.

The basic structure of a Bootstrap grid, as shown below, is to first create a row and then add the desired number of columns where # is replaced by one of the four classes `xs`, `sm`, `md` or `lg`, and all the numbers in each row, indicated by `*`, should always add up to 12.

```
<div class="row">
   <div class="col-#-*"</div>
</div>
<div class="row">
   <div class="col-#-*"</div>
   <div class="col-#-*"</div>
   <div class="col-#-*"</div>
</div>
< div class="row">
   .
   .
   .
</div>
```



Working With Grid Rows/Columns

Resize the browser window to see the responsive design.

Of all the universities in Hong Kong, my favourite is HKUST.

Of all the departments in HKUST, my favourite is CSE.

Of all the courses in the CSE department, my favourite is COMP 3111g.

Figure 23: Bootstrap column layout of Figure 22 resized to a smaller screen.

Table 2: Bootstrap grid properties across different device sizes.

Devices sizes	**Extra small** Phones (<768px)	**Small** Tablets (≥768px)	**Medium** Desktops (≥992px)	**Large** Desktops (≥1200px)
Grid behaviour	Horizontal always	Collapsed to start, horizontal above breakpoints		
Max container width	None (auto)	750px	970px	1170px
Class prefix	col-xs-	col-sm-	col-md-	col-lg-
# of columns	12			
Max column width	Auto	~62px	~81px	~97px
Gutter width	30px (15px on each side of a column)			
Nestable	Yes			
Offsets	Yes			
Column ordering	Yes			

Typography

Bootstrap styles some HTML tags differently than browser defaults[38]. The following contextual classes can be used to color text and backgrounds.

Text Class	Background Class	Meaning
`text-muted`	N/A	Greys out the element.
`text-primary`	`bg-primary`	Indicates a primary or most important element.
`text-success`	`bg-success`	Indicates a successful or positive action.
`text-info`	`bg-info`	Indicates a neutral informative change or action.
`text-warning`	`bg-warning`	Indicates a warning that might need attention.
`text-danger`	`bg-danger`	Indicates a dangerous or potentially negative action.

Bootstrap Classes[39]

Alert

The `alert` class creates a predefined alert message. An alert message is closed by adding `class=”close”` and `data-dismiss=”alert”` to a link or button element. The `fade` and `in` classes are used to add a fading effect when closing the alert message.

The following contextual classes are used to define different alert types.
- `alert-success` – a successful or positive action message.
- `alert-info` – a neutral informative change or action message.
- `alert-warning` – a warning message that might need attention.
- `alert-danger` – a dangerous or potentially negative action message.

Badge

The `badge` class indicates how many numerical items are associated with a link or button.

Buttons

The class `<btn>` is used to create a button, which is styled using the following classes (which also can be used on an `<a>` or `<input>` HMTL element).
- `btn-default` – a standard button.
- `btn-primary` – provides extra visual weight and identifies the primary action in a set of buttons.
- `btn-success` – indicates a successful or positive action.
- `btn-info` – used for informational alert messages.
- `btn-warning` – indicates caution should be taken with this action.
- `btn-danger` – indicates a dangerous or potentially negative action.
- `btn-link` – makes a button appear as a link while maintaining button behaviour.

The classes `btn-xs` (extra small), `btn-sm` (small), `btn-md` (medium) and `btn-lg` (large) define the size of a button.

The class `btn-block` causes a button to span the entire width of its parent element.

The class `active` makes a button appear pressed, while the class `disabled` makes it look unclickable.

Button Group

The `btn-group` class allows a series of buttons to be grouped together on a single line and sized together using the `btn-group-*` class, where `*` is replaced by `xs`, `sm`, `md` or `lg`.

The `btn-group-vertical` class creates a vertical button group.

[39] The description of the classes provided here is for non-ASP.NET web pages. For ASP.NET web pages, the classes are most often specified as part of an `<asp: … />` tag as described in the course’s Tutorial and Lab Notes. A one page “cheat sheet” containing a summary of all class names as well as other useful information can be downloaded from http://creativealive.com/master-cheat-sheet-twitter-bootstrap-3-pdf-download/.

The `btn-group-justified` class causes the buttons to span the entire width of the screen. Button groups can be nested to create dropdown lists.

Collapse

The `collapse` class is used to hide and show a large amount of content. To show/hide content, the `data-toggle=”collapse”` and `data-target=”#id”` attributes are added to a `<a>` or `<button>` tag where `#id` is the id of the collapsible content.

Dropdown

The `dropdown` class is used to create a menu that expands downward while the `dropup` class is used to create a menu that expands upward. To open a dropdown (or dropup) menu, the class `dropdown-toggle` with attribute `data-toggle=”dropdown”` is used within a `<button>` or `<link>` HTML tag and, to indicate that the button or link is a dropdown list, the `caret` class is used to create a caret icon (▼). The `dropdown-menu` class is added to a `<ul>` tag to actually build the dropdown list and the `dropdown-menu-right` class is used to right-align rather than left-align the menu. Within the elements of a dropdown list, the `dropdown-header` class is used to add a header, the `divider` class is used to separate items and the `disabled` class is used to disable an item.

Forms

Bootstrap provides three classes to layout a form.

- `vertical` (the default) – each form element is stacked one on top of the other.
- `form-horizontal` – some form elements are side-by-side and some are stacked on top of each other. The class `control-label` should be added to all `<label>` elements. Bootstrap’s predefined grid can be used to align labels and groups of form controls in a horizontal layout.
- `inline` – all the form elements are left-aligned with the labels alongside.

The following rules apply to all three form layouts.

- Always add labels for every input so screen readers can work correctly (labels can be hidden on other devices by using the class `sr-only`).
- Always place `role=”form”` within the `<form>` HTML tag.
- Wrap labels and form controls in `<div class=”form-group”>` for optimal spacing. Since for a horizontally laid out form the class `form-group` behaves as grid rows, there is no need to use the `row` class.
- Add the class `form-control` to all textual `<input>`, `<textarea>` and `<select>` elements.

Bootstrap supports the following form controls.

- input – for HTML types text, password, datetime, datetime-local, date, month, time, week, email, url, search, tel and color.
- textarea – for mutliple lines of text.
- checkbox – for selecting one or several options in a list via a checkbox.
- radio – for selecting one option from many via a radio button.
- select – for selecting one or several options from a textual list.

The class `form-control-static` allows plain text to be placed next to a form label.

The boolean attribute `readonly` prevents modification of the input’s value.

The classes `has-warning`, `has-error` and `has-success`, placed within the parent tag of a form control, are used to color the form control according to its validation state.

The class `has-feedback`, placed withn the parent tag of a textual `<input>` form control, is used to add feedback icons to the `<input>` form control.

Glyphicons[40]

Bootstrap provides 260 glyphicons from the Glyphicons Halflings set. Glyphicons can be used in text, buttons, toolbars, navigation, forms, etc.

[40] Glyphicons is a library of precisely prepared monochromatic icons and symbols, created with an emphasis on simplicity and easy orientation.

Images[41]

The following classes are used to style the `<img>` tag.

- `image-rounded` – adds rounded corners to an image.
- `img-circle` – shapes the image to a circle.
- `img-thumbnail` – shapes the image to a thumbnail.

The *img-responsive* class creates responsive images that will scale to the parent element.

The *thumbnail* class can be used in conjunction with Bootstrap's grid system to create an image gallery.

Jumbotron

The `jumbotron` class creates a big box for calling extra attention to some special content or information. The information is displayed as a grey box with rounded corners and enlarged text. The `jumbotron` class placed outside a `<div class=”container”>` element extends the box to the screen edges.

Labels

The label class provides additional information about something. The contextual classes `label-default`, `label-primary`, `label-success`, `label-info`, `label-warning` and `label-danger` can be used to colour labels.

List Groups

The class `list-group` along with the class `list-group-item` creates a basic list group of unordered list items. Using `<div>` instead of `<ul>` and `<a>` instead of `<li>` creates a list group with linked items. The class `disabled` placed within a list item disables the item. Contextual text classes can be used to colour list items. Using the classes `list-group-item-heading` and `list-group-item-text` within a list item results in a list item consisting of a heading line followed by a text line.

Navbar

The class `navbar` creates a navigation bar, which is a navigation header that is placed at the top of the page. Different types of navigation bars can be created and the navigation bar positioned using the following classes in conjunction with the `navbar` class.

- `navbar-default` – creates a standard navigation bar with a white background.
- `navbar-inverse` – creates a navigation bar with a black background.
- `navbar-fixed-top` – creates a navigation bar that stays visible in a fixed position at the top of the page.
- `navbar-fixed-bottom` – creates a navigation bar that stays visible in a fixed position at the bottom of the page.

The class `navbar-header` creates the header item in the navigation bar (e.g., the website name). The other items in the navigation bar are created as `<li>` elements within a `<ul class=”nav navbar-nav”>` element. the `navbar-right` class specified for an `<li>` element right-aligns the item in the navigation bar. The elements of a navigation bar can be dropdown lists.

A navigation bar can extend or collapse, depending on the screen size. A `<button>` element placed within the header item having class `nav-bar-toggle`, `data-toggle=”collapse”` and `data-target=”#`*`navigation-bar-id`*`>` can be used to replace the navigation bar on small screens.

Pager

The `pager` class is used to create previous/next buttons providing a form of pagination. The classes `previous` and `next` align the previous/next buttons to the sides of the page.

Pagination

The `pagination` class creates basic pagination. The size of the pagination blocks can be made larger using the class `pagination-lg` or smaller using the class `pagination-sm` in conjunction with the `pagination` class. The class `active` is used to indicate which is the current page. The class `disabled` is used to disable the link for a page.

[41] See http://www.w3schools.com/bootstrap/bootstrap_ref_css_images.asp for a complete reference of all image classes.

Panels

The `panel` class creates a bordered box with some padding around its content. The `panel-body` class, placed within an element nested within the element of the `panel` class, contains the content of the panel. The `panel-heading` class adds a heading to the panel. The `panel-footer` class adds a footer to the panel. The `panel-group` class groups several panels together. The contextual classes `panel-default,` `panel-primary`, `panel-success`, `panel-info`, `panel-warning` and `panel-danger` are used to colour a panel.

Progress Bars

The class `progress` along with the class `progress-bar` creates a default progress bar. The contextual classes `progress-bar-success`, `progress-bar-info`, `progress-bar-warning` and `progress-bar-danger` are used to colour a progress bar, the contextual class `progress-bar-striped` is used to add stripes to a progress bar and the contextual class `active` is used to animate a progress bar.

Tables

Tables are styled using the following classes.

- `table` – Styles a table with light padding and only horizontal dividers.
- `table-striped` – Adds zebra-striping to table rows.
- `table-bordered` – Places borders on all sides of the table and cells.
- `table-hover` – Enables a hover state on table rows.
- `table-condensed` – Makes a table more compact by cutting cell padding in half.
- `table-responsive` – creates a table that will scroll horizontally on small devices (under 768px).

The contextual classes `active`, `success`, `info`, `warning` and `danger` can be used to color table rows or table individual cells.

Tabs/Pills

The classes `nav` and `nav-tabs` create navigation tabs on a page while the classes `nav` and `nav-pills` create navigation pills. The class `nav-justified` centers and justifies the tabs/pills. The class `nav-stacked` displays the pills vertically.

The tabs/pills are created by the `<li>` elements of an unordered list. A tab/pill can hold a dropdown list. The active tab/pill is indicated by placing `class=”active”` with an `<li>` element of the tabs/pills. The `data-toggle=”tab”` and `data-toggle=”pill”` attributes makes the tab/pill toggable. Every toggable tab/pill requires a unique ID and a `<div>` element identified with the tab/pill’s unique ID and with `class=”tab-pane”`. Moreover, all tab-panes need to be wrapped inside a `<div class=”tab-content”>` element.

Wells

The class `well` adds a rounded border around an element with a gray background color and some padding. By default, wells are medium in size. The size is made larger by using the class `well-lg` or smaller using the class `well-sm` instead of `well`.

Tutorials

http://www.tutorialrepublic.com/twitter-bootstrap-tutorial/

http://www.w3schools.com/bootstrap/default.asp

http://www.tutorialspoint.com/bootstrap/