



# **Unit 1**

## **Service Standards and Service Development**

---

### **Lecture 1-6**

## **Advanced Web App Architecture**


Dr. Yinong Chen

<https://myasucourses.asu.edu/>

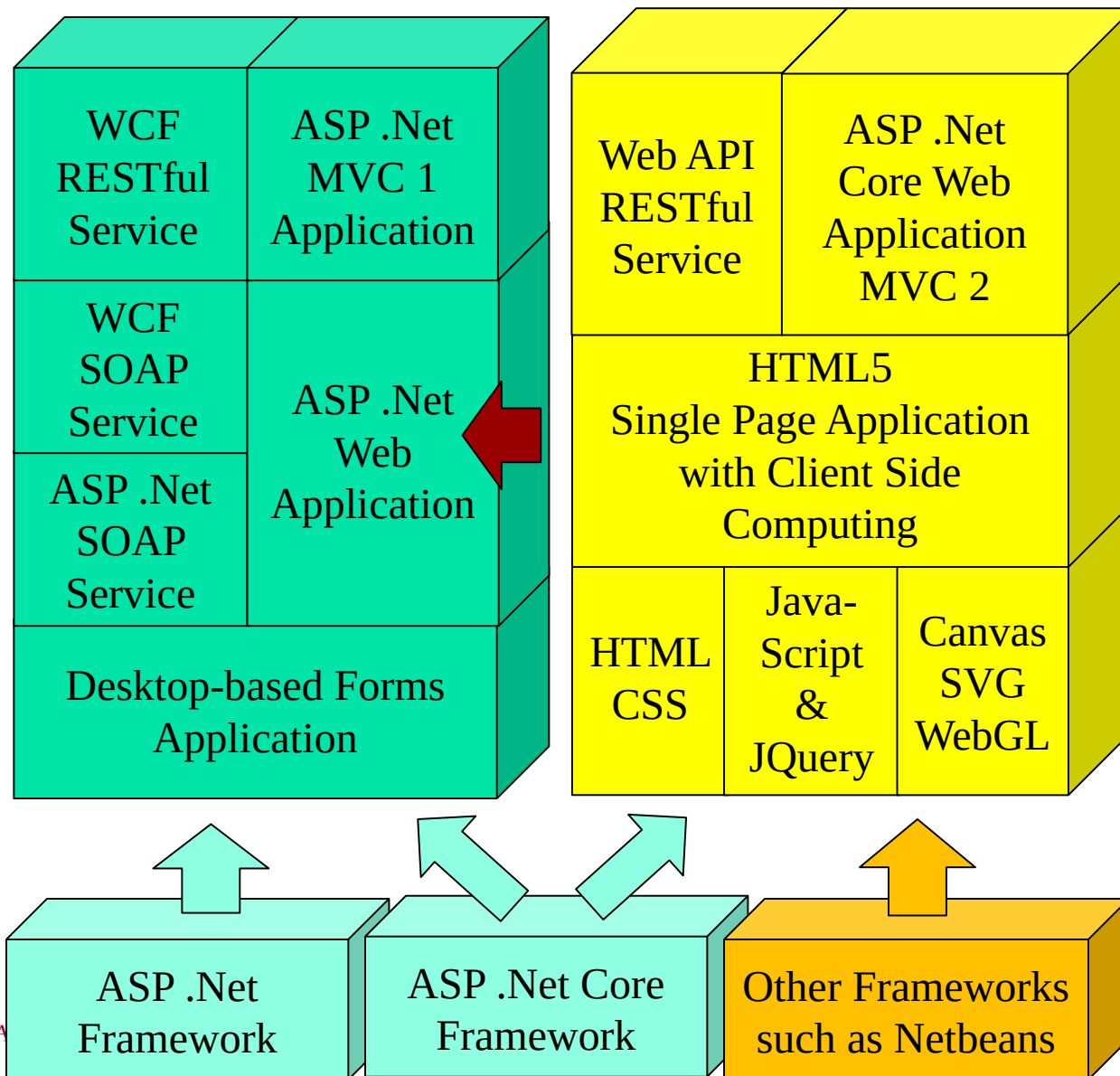


# Lecture Outline

---

- 
- **ASP .Net Web Application Architecture**
  - HTML5 Application
    - Case Study
  - MVC 2 Web Application Architecture
    - Case Study

# The Evaluation Web Technologies

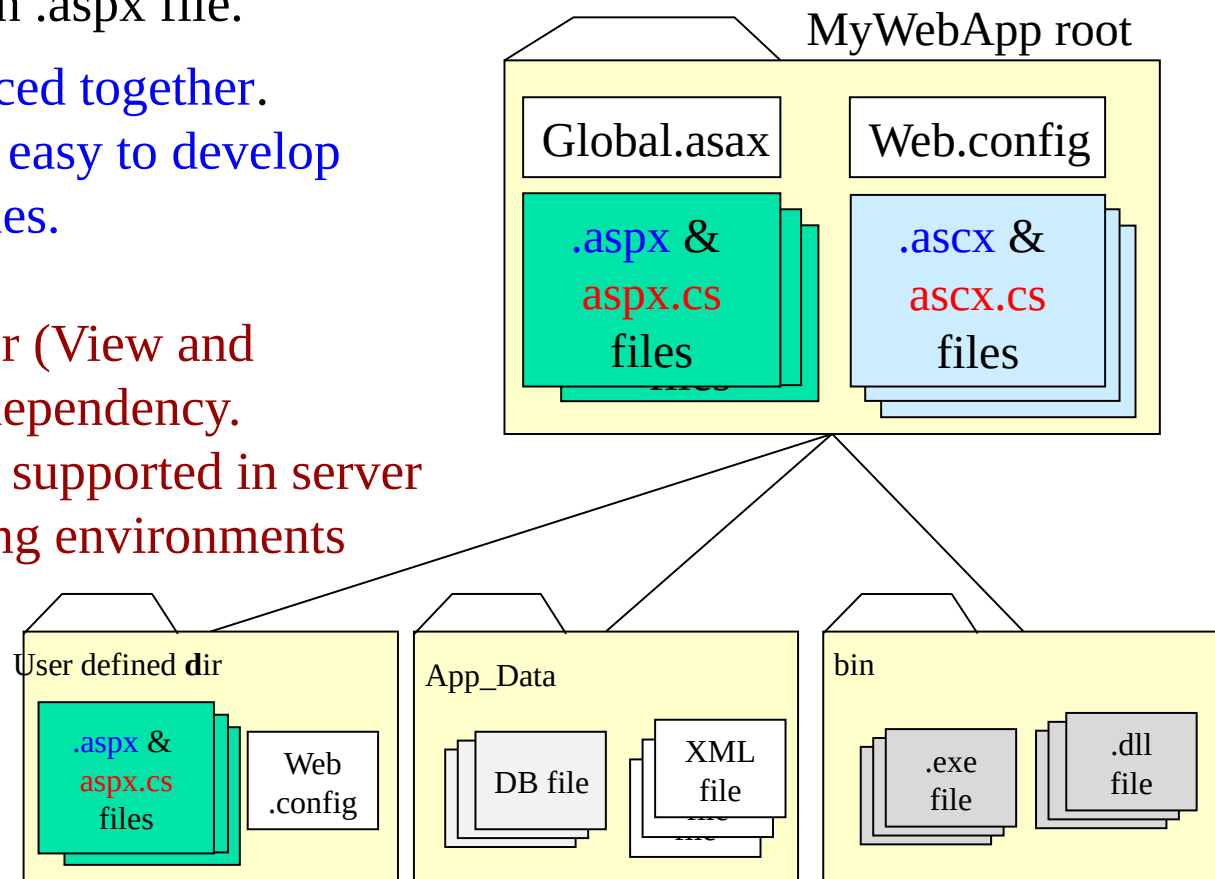
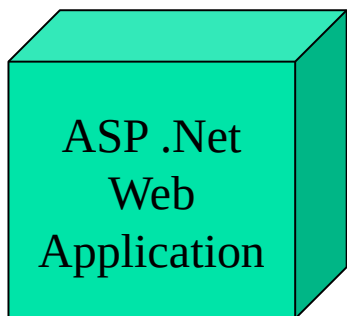


# Review: ASP .Net Web Application Architecture

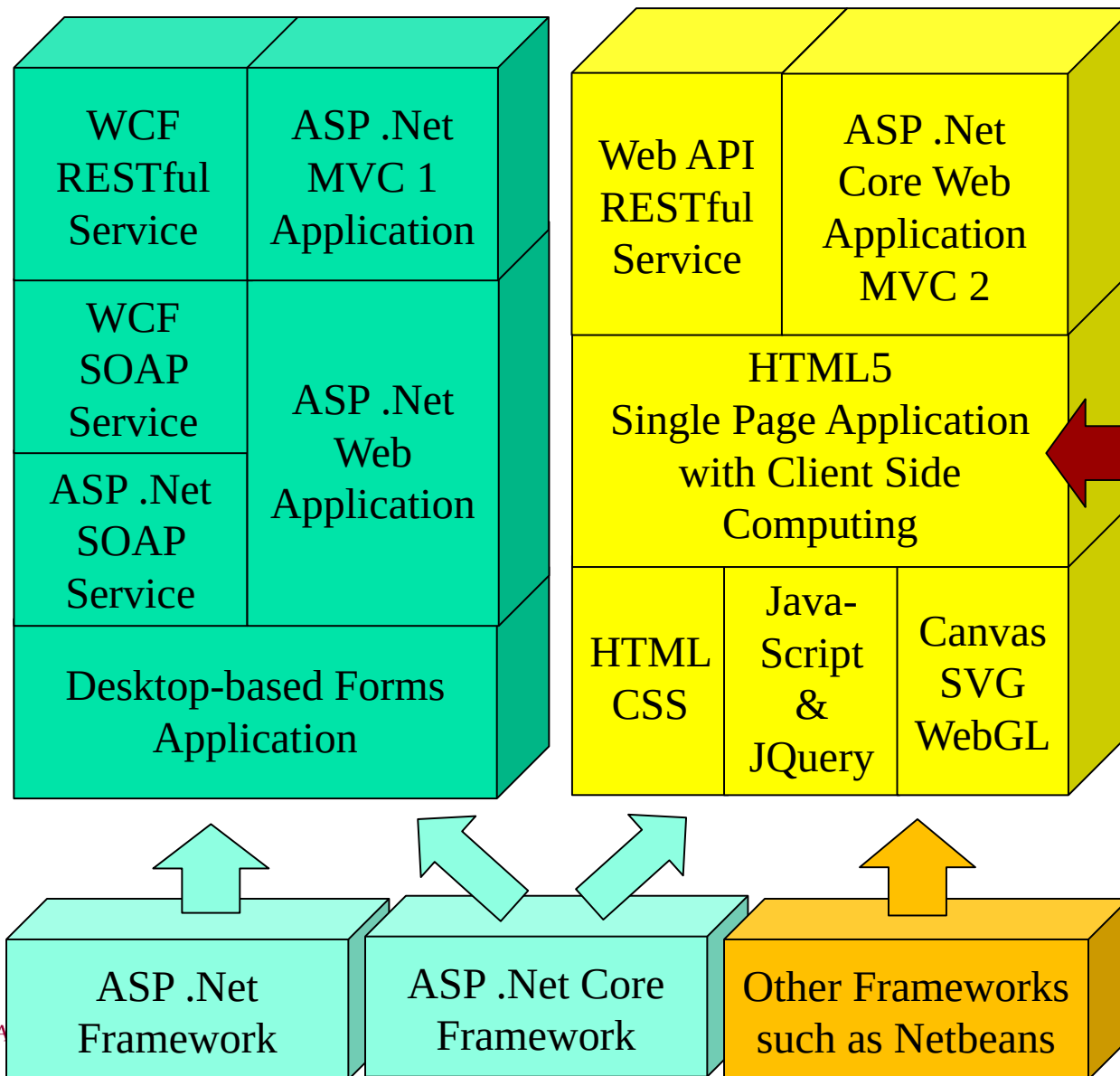
ASP .Net comes from object-oriented programming environment. It makes it easier for developers with OO development experience. It places the presentation layer files (.aspx) and the controller file (.aspx.cs) together.

It generates html file based on .aspx file.

- The advantages
  - Related items are placed together.
  - It matches OO and is easy to develop based on OO principles.
- The disadvantages:
  - Mix two tiers together (View and Controller) and add dependency.
  - ViewState is not well supported in server farms/cloud computing environments



# The Evaluation Web Technologies



# From HTML to HTML5

Reference: [http://www.w3schools.com/html/html5\\_intro.asp](http://www.w3schools.com/html/html5_intro.asp)

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title> *Doc Title* </title>

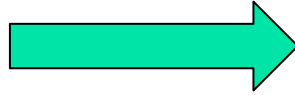
</head>

<body>

*Content of the document.....*

</body>

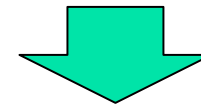
</html>



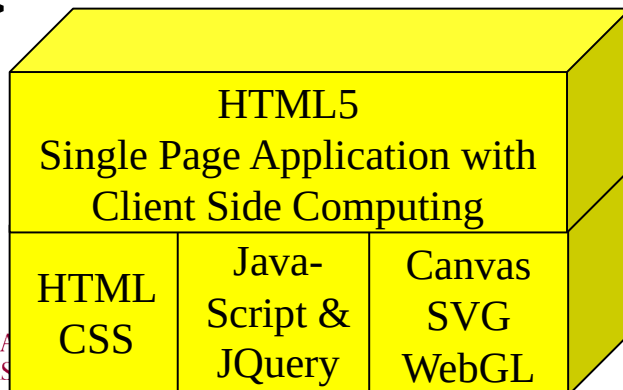
- New semantic elements like <header>, <footer>, <article>, and <section>.
- New form control attributes like number, date, time, calendar, and range.
- New graphic elements like: <svg> and <canvas>.
- New multimedia elements like: <audio> and <video>.



JavaScript and JQuery API Calls



Become a Web Application  
Development Language



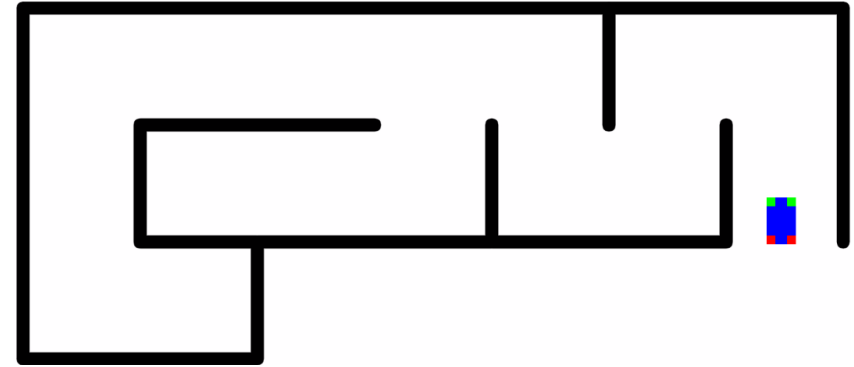
# HTML 5 JavaScript API Library

Reference: <http://html5index.org/>

- Web communication protocols
  - Web Sockets (Connection-oriented)
  - Messaging, and
  - WebRTC  
(Web Real-Time Communication)
- Drag and Drop, Fullscreen
- Canvas, SVG, WebGL
- Animation Timing, Media, Pointer Lock, Web Audio
- File API, File System API, Indexed DB, Offline, Web Storage
- Browser, Shadow DOM, Typed Arrays, Web Workers
- DOM and JQuery
- CSS Object Model, Selectors

Example:

<http://neptune.fulton.ad.asu.edu/VIPLE/Websimulator/>



# Canvas, SVG, or WebGL? Choose the right API

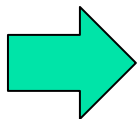
<https://msdn.microsoft.com/en-us/library/dn265058%28v=vs.85%29.aspx?f=255&MSPPErr=-2147217396>

- **Canvas** graphic library is the typical choice for most HTML5 applications. It's simple and speedy, particularly for games with many objects.
- **SVG**: While canvas provides simplicity and speed, SVG provides flexibility. For example: Each graphic object is part of the DOM tree for flexible access, and each graphic object in the application can have one or more associated event handlers for processing.
- **WebGL**: If you are already familiar with OpenGL, you *can* use WebGL for a simple 2D application. If you are not, this API is more complex than you need for a simple graphic application.



# Creating HTML5 Applications

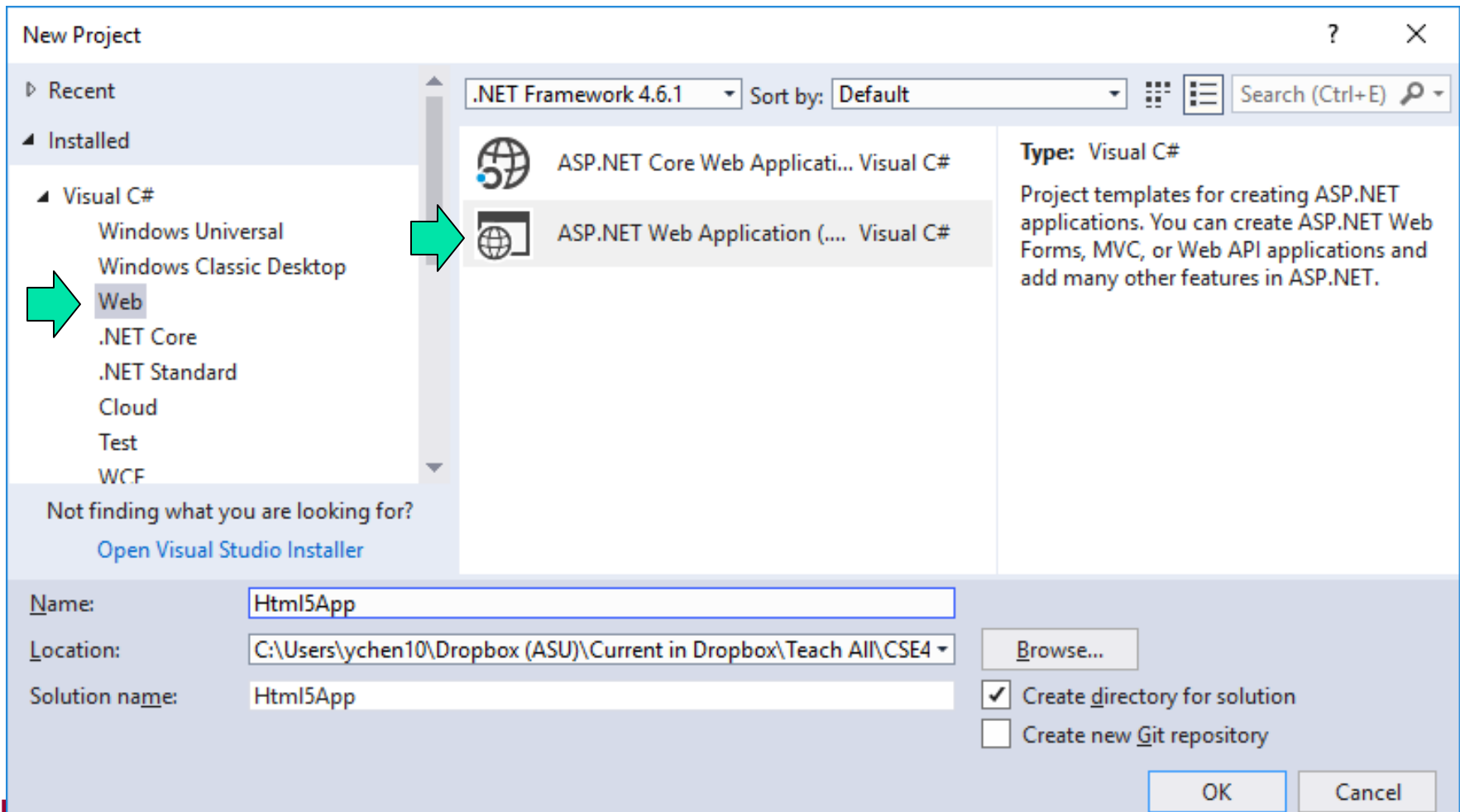
---



- **Using Visual Studio**
  - For HTML5 application development
- Using NetBeans
  - For HTML5 application development

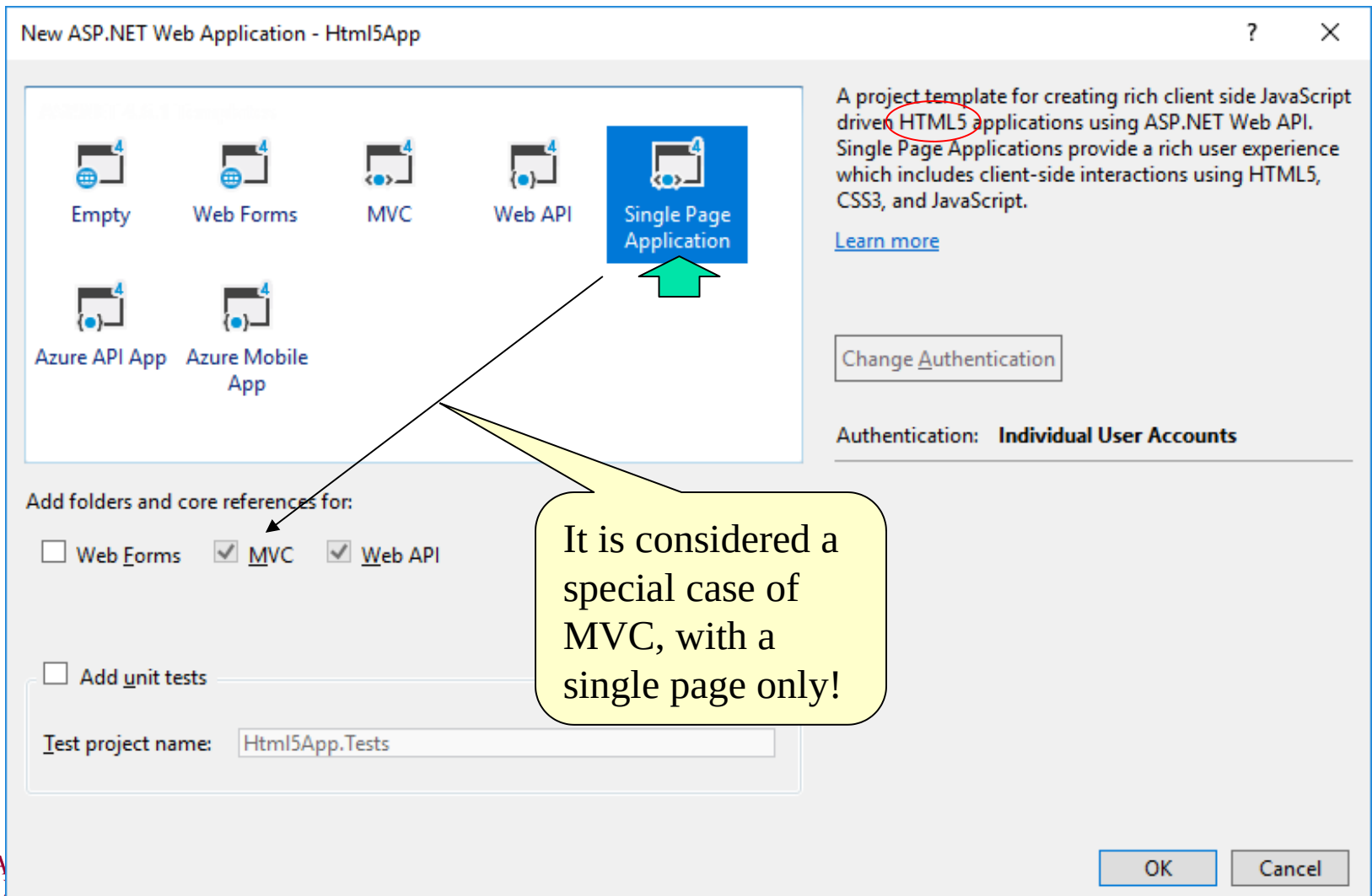
# HTML5 Applications Using Visual Studio

- File □ New □ Project ...



# HTML5 Applications Using Visual Studio

## ■ Choose Single Page Application



# Adding an HTML Page as the Index Page

first application and extend it to the cloud.



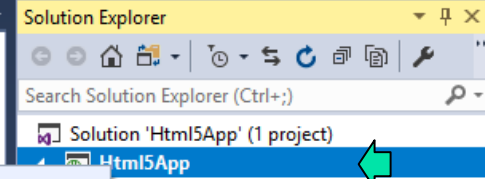
Add a service

Telemetry with Application Insights

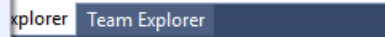
More connected services

- Area...
- New Item... Ctrl+Shift+A
- Existing Item... Shift+Alt+A
- New Scaffolded Item...
- New Folder
- Add ASP.NET Folder
- From Cookiecutter...

- Build
- Rebuild
- Clean
- View
- Analyze
- Convert
- Publish...
- Overview
- Scope to This
- New Solution Explorer View
- Show on Code Map
- Add
- Manage NuGet Packages...
- Manage Bower Packages...
- Set as StartUp Project
- Debug
- Initialize Interactive with Project
- Cut Ctrl+X
- Paste Ctrl+V
- Remove Del
- Rename
- Unload Project
- Open Folder in File Explorer
- Properties Alt+Enter



- Connected Services
- Properties
- References
- App\_Data
- App\_Start
- Areas
- Content
- Controllers
- fonts
- Models
- Providers
- Results
- Scripts
- Views
- ApplicationInsights.config
- favicon.ico
- Global.asax
- packages.config
- Startup.cs
- Web.config



- Project Properties

- Deployment Server
- Start When Deb True
- Anonymous Authentica Enabled

- Managed Pipeline Mod Integrated
- SSL Enabled False
- SSL URL
- URL http://localhost:35181/
- Windows Authentication Disabled
- Misc

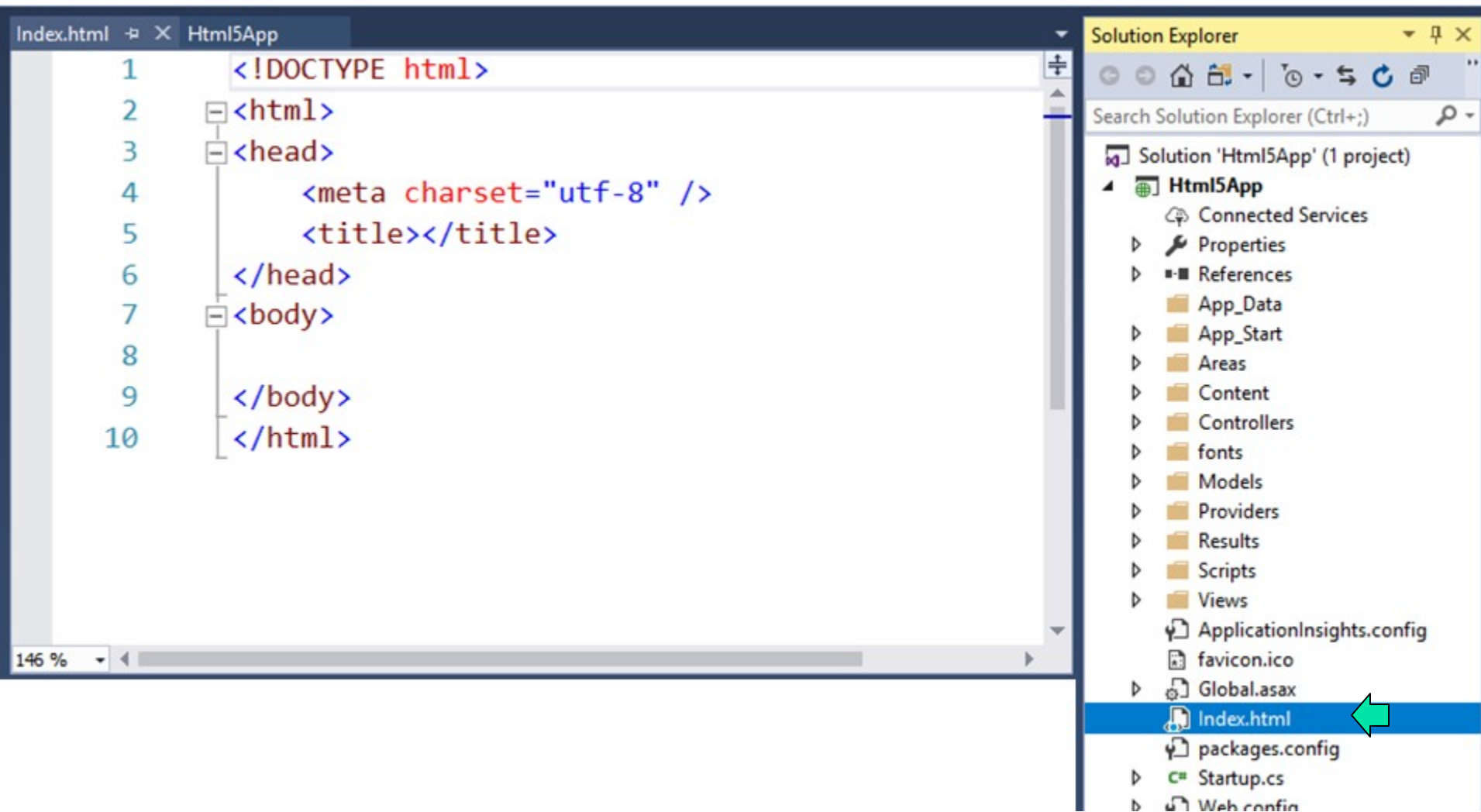
- Connected Service
- Analyzer...
- HTML Page
- JavaScript File
- Style Sheet
- Web Form

Specify Name for Item

Item name: Index

OK Cancel

# Index.html Page Created



# Hello HTML5

- Right Click and choose View in Browser

The screenshot displays the Visual Studio IDE with the following components:

- Code Editor:** Shows the content of `Index.html`. The code is as follows:

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4     <meta charset="utf-8" />
5     <title>Hello</title>
6 </head>
7 <body>
8     Hello HTML5
9 </body>
10</html>
```

A green arrow points to the text "Hello HTML5" on line 8.
- Solution Explorer:** Located on the right, it shows a project structure with folders like `App_Start`, `Areas`, `Content`, `Controllers`, `fonts`, `Models`, `Providers`, `Results`, `Scripts`, and `Views`. Files include `ApplicationInsights.config`, `favicon.ico`, `Global.asax`, `Index.html` (which is selected and highlighted in blue), `packages.config`, `Startup.cs`, and `Web.config`.
- Callout:** A green speech bubble with the text "Right Click, choose View in Browser" has an arrow pointing to the `Index.html` file in the Solution Explorer.
- Browser:** A separate window at the bottom shows the rendered output of the HTML file. The address bar displays `localhost:35181/Index.html`, and the page content is "Hello HTML5".

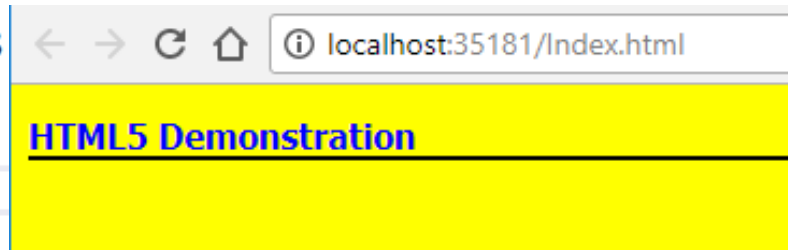
# Adding CSS Stylesheet

- Right Click Project □ Add □ Style Sheet
- Name the page StyleSheet1, and link it into Index.html page

```

StyleSheet1.css  X Index.html
1  body {
2      font-family: Tahoma, Arial, sans-serif;
3      font-size: 14px;
4      color: blue;
5      background: yellow;
6      margin: 8px;
7  }

```



View in Web Browser

```

StyleSheet1.css  Index.html  X
9  h1 {
10     font-
11     margi
12     margi
13     borde
14 }

1  <!DOCTYPE html>
2  <html>
3  <head>
4      <meta charset="utf-8" />
5      <link rel="stylesheet" href="StyleSheet1.css">
6      <title>Hello</title>
7  </head>
8  <body>
9      <h1>HTML5 Demonstration</h1>
10 </body>
11 </html>

```

# GUI Design

The screenshot shows a web development IDE with three main panels:

- Toolbox:** A list of HTML widgets on the left. Arrows point from the 'Input (Submit)', 'Input (Text)', and 'Textarea' widgets to their corresponding code elements in the HTML file.
- Code Editor:** The central pane shows `Index.html` with the following code:

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <meta charset="utf-8" />
5   <link rel="stylesheet" href="StyleSheet1.css">
6   <title>Hello</title>
7 </head>
8 <body>
9   <h1>HTML5 Demonstration</h1>
10  <textarea id="TextArea1" rows="2" cols="40">
11    This demo shows the GUI Design and Code behind the GUI
12  </textarea><br />
13  <input id="Text1" type="text" />
14  <input id="Button1" type="button" value="button" /><br />
15  <input id="Submit1" type="submit" value="submit" />
16 </body>
17 </html>
```
- Solution Explorer:** The right panel shows the project structure, including folders like 'Areas', 'Content', and 'Views', and files like 'Index.html'.

The screenshot shows a web browser window at `localhost:35181/Index.html`. The page has a yellow background and displays the rendered HTML5 demonstration:

- Header:** "HTML5 Demonstration" in blue text.
- Text Area:** A text area containing the text "This demo shows the GUI Design and Code behind the GUI".
- Form Elements:** A text input field, a "button" button, and a "submit" button.



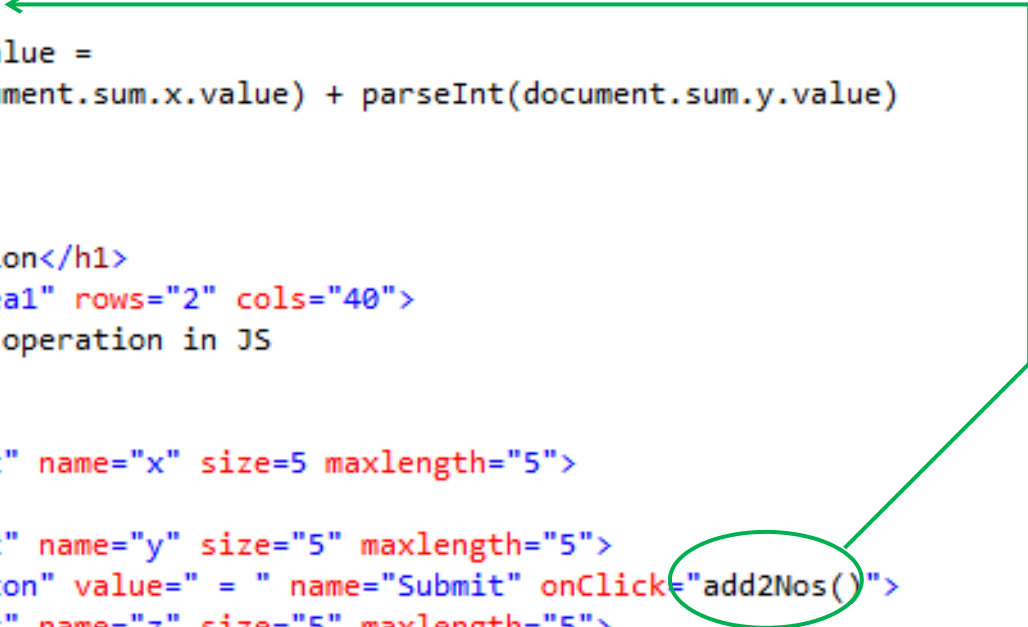
# Adding JavaScript Code in HTML

```

Index.html  X
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8" />
  <link rel="stylesheet" href="StyleSheet1.css">
  <title>Hello</title>
</head>
<body>
  <script type="text/JavaScript">
    function add2Nos() {
      document.sum.z.value =
        parseInt(document.sum.x.value) + parseInt(document.sum.y.value)
    }
  </script>

  <h1>HTML5 Demonstration</h1>
  <textarea id="TextArea1" rows="2" cols="40">
This demo shows addition operation in JS
  </textarea><br />
  <form name="sum">
    <input type="text" name="x" size=5 maxlength="5">
    +
    <input type="text" name="y" size="5" maxlength="5">
    <input type="button" value=" = " name="Submit" onClick="add2Nos()">
    <input type="text" name="z" size="5" maxlength="5">
    <span id="result"></span>
  </form>
</body>
</html>

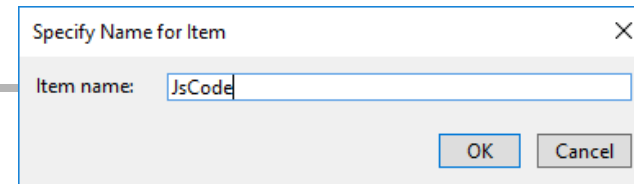
```



# Adding a JavaScript File

Right Click Project □ Add □ JavaScript File

- Name the file JsCode, and link it into Index.html page



**Index.html\***

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8" />
  <link rel="stylesheet" href="StyleSheet1.css">
  <script src="JsCode.js"></script>
  <title>Hello</title>
</head>
<body>
  <h1>HTML5 Demonstration</h1>
  <textarea id="TextArea1" rows="2" cols="40">
This demo shows JavaScript in a external file
  </textarea><br />
  <form name="sum">
    <input type="text" name="x" size=5 maxlength="5">
    and
    <input type="text" name="y" size="5" maxlength="5">
    <input type="button" value=" add " name="Submit" onClick="add2Nos()">
    <input type="button" value=" multiply " name="Submit" onClick="multiply2Nos()">
    <br />
    =
    <br />
    <input type="text" name="z">
    <span id="result"></span>
  </form>
</body>
</html>
```

**HTML5 Demonstration**

This demo shows addition operation in JS

5 and 20 add multiply

= 100

View in Web Browser

**JavaScript Code:**

```
function add2Nos() {
  document.sum.z.value =
    parseInt(document.sum.x.value) + parseInt(document.sum.y.value)
}

function multiply2Nos() {
  document.sum.z.value =
    parseInt(document.sum.x.value) * parseInt(document.sum.y.value)
}
```

**Solution Explorer**

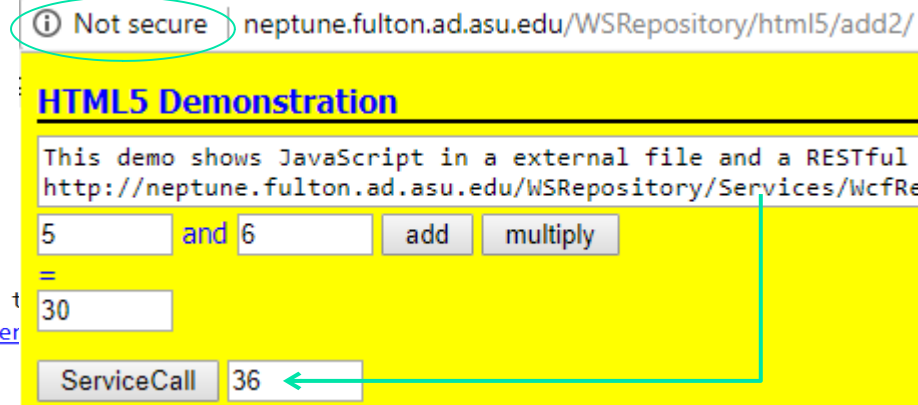
- App\_Start
- Areas
- Content
- Controllers
- fonts
- Models
- Providers
- Results
- Scripts
- Views
- ApplicationIns
- favicon.ico
- Global.asax
- Index.html
- JsCode.js
- packages.config
- Startup.cs
- StyleSheet1.css
- Web.config

# Calling A RESTful Service in JS

19

```
<html>
<head>
  <meta charset="utf-8" />
  <link rel="stylesheet" href="StyleSheet1.css">
  <script src="JsCode.js"></script>
  <title>HTML5 Demo</title>
</head>
<body>
  <h1>HTML5 Demonstration</h1>
  <textarea id="TextArea1" rows="2" cols="100">
    This demo shows JavaScript in a external file and a RESTful service call t
    http://neptune.fulton.ad.asu.edu/WSRepository/Services/WcfRestService4/Ser
  </textarea><br />
  <form name="sum">
    <input id="Text1" type="text" name="x" size=5 maxlength="5" />
    and
    <input id="Text2" type="text" name="y" size="5" maxlength="5" />
    <input type="button" value=" add " name="Submit" onClick="add2Nos()" />
    <input type="button" value=" multiply " name="Submit" onClick="multiply2Nos()" />
    <br />
    =
    <br />
    <input type="text" name="z" size="5" maxlength="5">
    <span id="result"></span><p />
    <input type="button" value=" ServiceCall " name="Submit" onClick="ServiceCall()" />
    <input type="text" name="s" size="5" maxlength="5">
    <span id="result"></span>
  </form>
</body>
</html>
```

<http://neptune.fulton.ad.asu.edu/WSRepository/html5/add2/>



The ServiceCall is asynchronous:

- (1) The generated result will be sent to the html page;
- (2) Call the service and send the result to the response

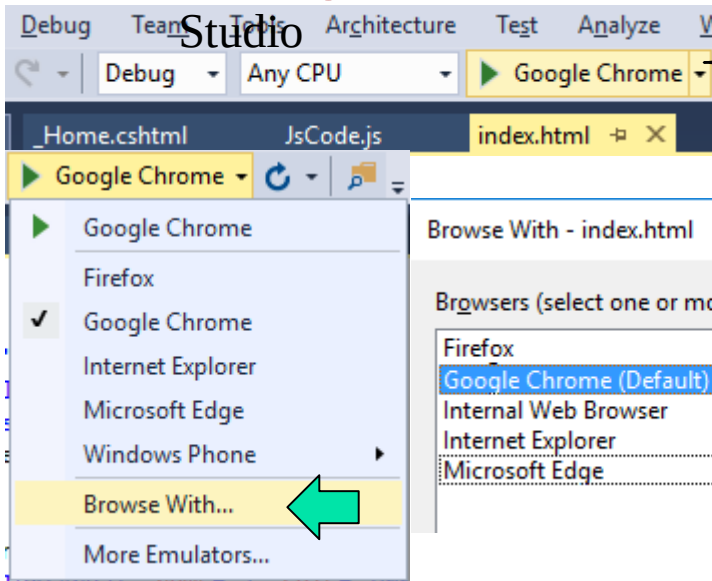
```
function ServiceCall() {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function () {
    if (xhttp.readyState == 4 && xhttp.status == 200)
      document.sum.s.value = xhttp.responseText;
  }
  xhttp.open("GET",
    "http://neptune.fulton.ad.asu.edu/WSRepository/Services/WcfRestService4/Service1/add2?x=11&y=25", true);
  xhttp.setRequestHeader("Content-type", "application/json");
  xhttp.send();
}
```

# W3C: Cross-Origin Resource Sharing (CORS)

**Testing:** Disable Web browser security for testing XMLHttpRequest in Visual

Studio

Click



Browse With - index.html

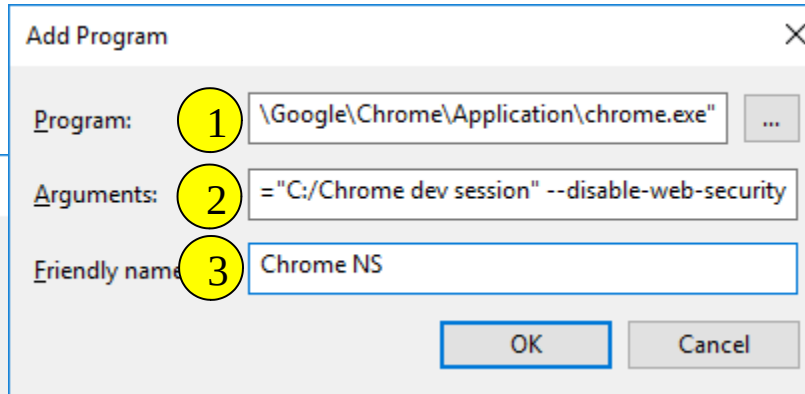
Browsers (select one or more):

Firefox  
Google Chrome (Default)  
Internet Web Browser  
Internet Explorer  
Microsoft Edge

Add...

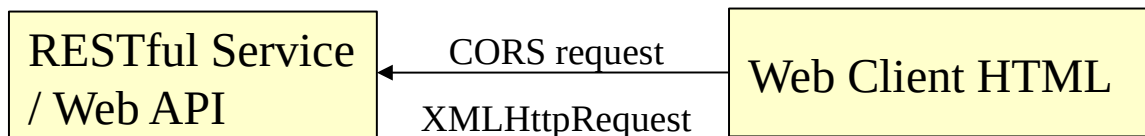
Remove

Set as Default



--user-data-dir="C:/Chrome dev session" --disable-web-security

- Browser security policy prevents a web page from making AJAX requests to another domain.



- CORS is a W3C standard that allows a server to relax the same-origin policy (<https://www.w3.org/TR/cors/>).
- Deployment**, follow:

<https://docs.microsoft.com/en-us/aspnet/web-api/overview/security/enabling-cross-origin-requests-in-web-api>

# Calling A RESTful Service in jQuery

Source: <https://spring.io/guides/gs/consuming-rest-jquery/>

```
<!DOCTYPE html>
<html>
<head>
  <title>Hello jQuery</title>
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.10.2/jquery.min.js"></script>
  <script src="hello.js"></script>
</head>
<body>
  <div>
    <p class="greeting-id">The ID is </p>
    <p class="greeting-content">The content is </p>
  </div>
</body>
</html>
```

```
$(document).ready(function () { // hello.js
  $.ajax({
    url: "http://rest-service.guides.spring.io/greeting"
  }).then(function (data) {
    $('.greeting-id').append(data.id);
    $('.greeting-content').append(data.content);
  });
});
```

http://neptune.fulton.ad.asu.edu/WSRepository/html5/hello/

The ID is 37

The content is Hello, World!

# Example with Animation Definition

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8" />
  <title>first website</title>
  <link href="main.css" rel="stylesheet" />
  <script src="main.js"></script>
</head>
<body>
  <div id="domino">
    <h1>enter an animal</h1>
    <input type="text" id="animal">
    <button onclick="createanimal()">create animal</button>
    <button onclick="displayanimal()">display animal</button>
    <p id="show"></p>
  </div>
</body>
</html>
```

```
var animallist = [];
var item;
currentlist = [];
function createanimal()
{
  item = document.getElementById("animal").value;
  animallist.push(item);
  console.info("animal " + animallist);
}

function displayanimal()
{
  for (var i = 0; i < animallist.length; i++)
  {
    var currentitem = animallist[i];
    currentlist[i] = currentitem;
    document.getElementById("show").innerHTML = currentlist;
  }
}
```

Where will the JS code be executed?

(A) They will be executed in the browser.

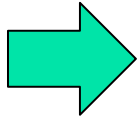
(B) They will be executed in the server.

(C) They will be executed in the browser if they are embedded in the HTML file.

(D) They will be executed in the server if they are embedded in the HTML file.

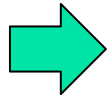
# Creating HTML5 Applications

- Using Visual Studio



- **NetBeans**

- was used for developing object-oriented software
- was used for developing web services
- can be used for developing HTML5 web applications



# Preparation for NetBeans

- Please check if the following recourses have been installed/included:
  - ➡ ■ NetBeans IDE (pick either HTML5/JavaScript package or all package included) <https://netbeans.org/downloads/index.html>
  - ➡ ■ Java Development Kit (JDK) version 7 or 8 (NetBeans 8.2 does not run JDK9!)
  - ➡ ■ Chrome Browser
- Add NetBeans Connector Extensions into Chrome Browser:
  - ➡ ■ Go to the Chrome browser  
([https://chrome.google.com/webstore/detail/netbeansconnector/hafdlehgocfcodbgnpecfajgkeejnaa?utm\\_source=chrome-ntp-icon](https://chrome.google.com/webstore/detail/netbeansconnector/hafdlehgocfcodbgnpecfajgkeejnaa?utm_source=chrome-ntp-icon))  
and click Add the NetBeans Connector page. Click Add when you are prompted to confirm that you want to add the extension



# NetBeans Connector


[+ ADD TO CHROME](#)

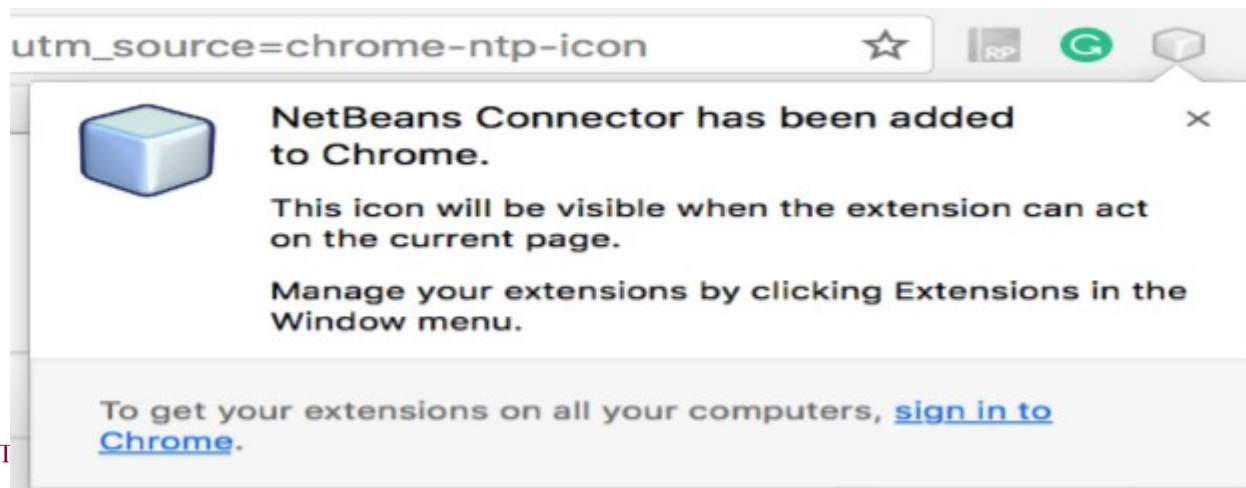
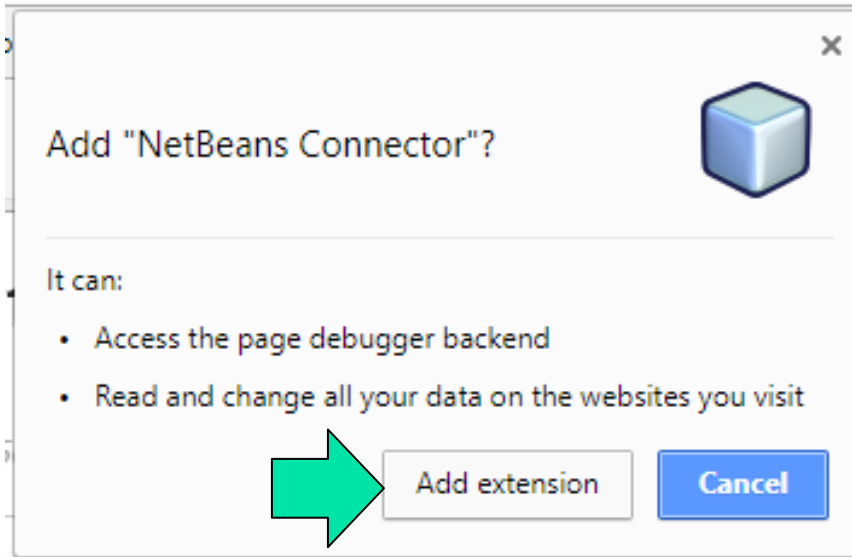

[https://chrome.google.com/webstore/detail/netbeansconnector/hafdlehgocfodbgjnpecfajgkeejnaa?utm\\_source=chrome-ntp-icon](https://chrome.google.com/webstore/detail/netbeansconnector/hafdlehgocfodbgjnpecfajgkeejnaa?utm_source=chrome-ntp-icon)

The screenshot shows the NetBeans IDE 7.3 interface. The main editor displays the HTML code for 'index.html', which includes a Bootstrap navbar and a container with a row of images. A 'NetBeans Connector' debugging window is open, showing a message: '"NetBeans Connector" is debugging this tab.' Below the message, there is a 'View details >' button and a preview of a rabbit image. The 'img - Navigator' panel on the left shows the DOM tree, highlighting the 'img' element with class 'thumb'. The 'Index.html - CSS Styles' panel on the right shows the 'img.thumb' properties, including height (100px), width (100px), and color (#333333).

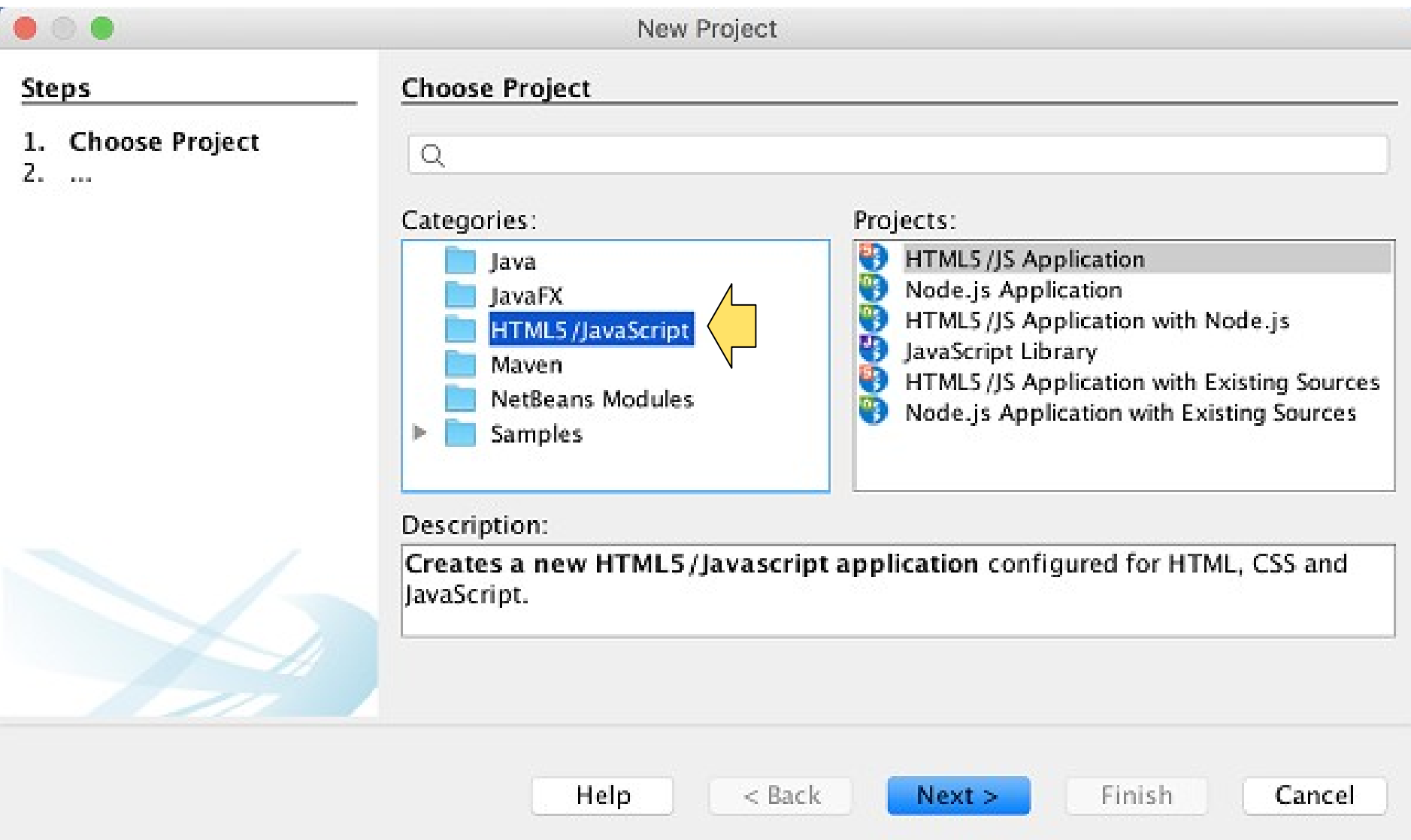
# Add to Chrome



+ ADD TO CHROME



# Now, Start NetBeans IDE



# HTML5 Case Study: ASU IOT Maze Simulator

<http://neptune.fulton.ad.asu.edu/VIPLE/Web2DSimulator/>

Add a New Line

Remove a Line

Default: 

Forward

1. If 

Right

sensor 

>

100

pixels

Then: 

Delayed Turn Right

 by 

90

 pixels

2. Else if 

Forward

sensor 

<=

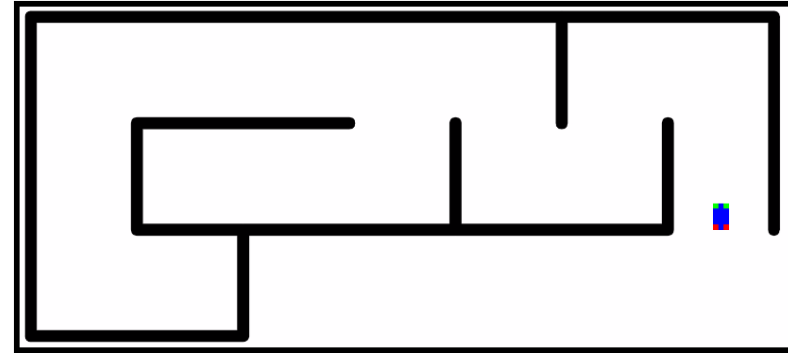
50

pixels

Then: 

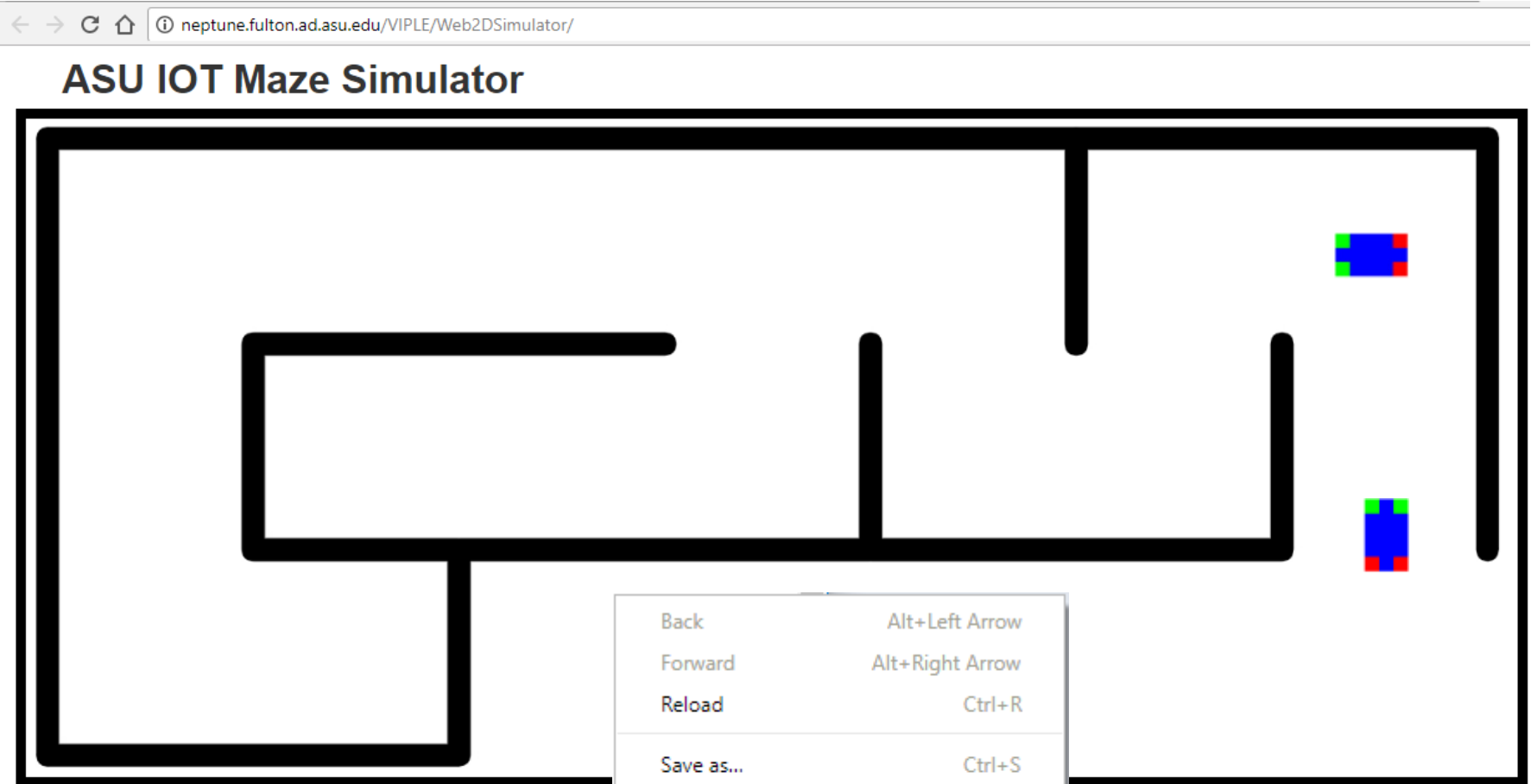
Turn Left

Run



# HTML5 Case Study: ASU IOT Maze Simulator

<http://neptune.fulton.ad.asu.edu/VIPLE/Web2DSimulator/>



Study the source code to  
learn Canvas-based  
animation

Right-click the page

- Creating User Interface Elements with Canvas or SVG

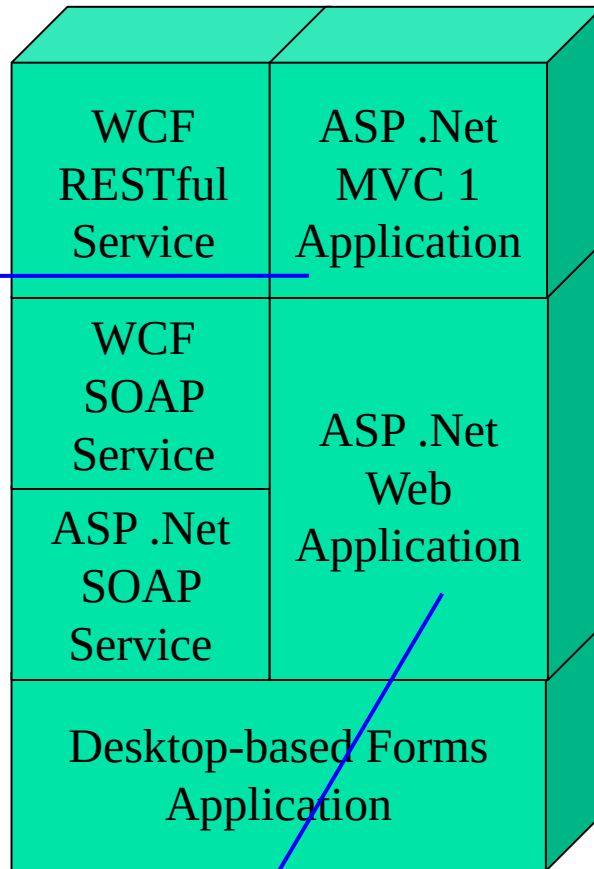
[https://msdn.microsoft.com/en-us/library/gg589494\(v=vs.85\).aspx#creating\\_UI\\_elements\\_drawingUI](https://msdn.microsoft.com/en-us/library/gg589494(v=vs.85).aspx#creating_UI_elements_drawingUI)

- Introducing the HTML5 Canvas

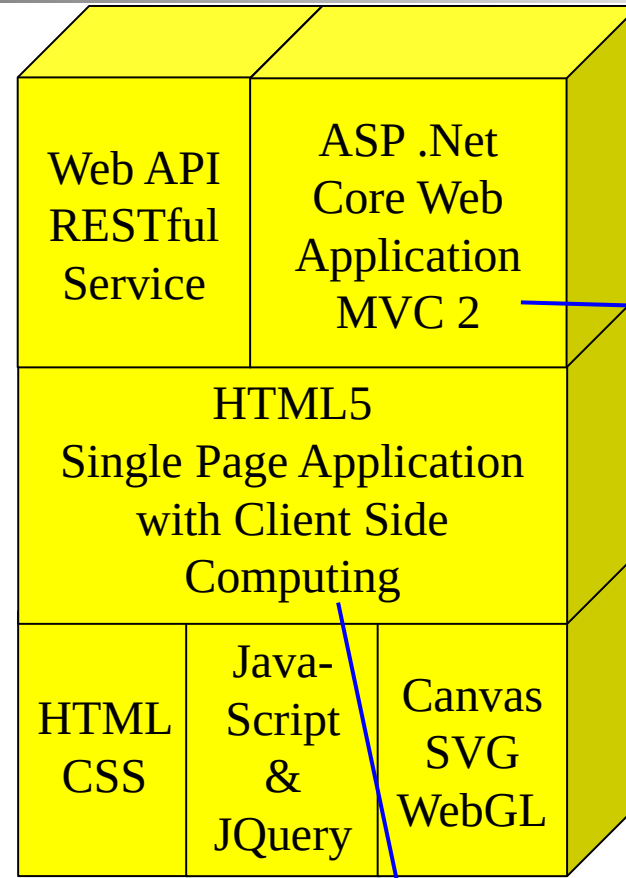
<https://msdn.microsoft.com/en-us/magazine/hh708752.aspx>

# Why MVC Architecture

ASP .Net MVC 1 is a simple re-organization of ASP .Net Web Application.



- **ASP .Net Web Application** mixes two tiers together and add dependency.
- ViewState is not well supported in server farms/cloud computing



**MVC 2** addresses all these issues

- **HTML5** is mainly for one-page application.
- Data are hard to transfer between pages.
- Code viewable: Proprietary and security issues

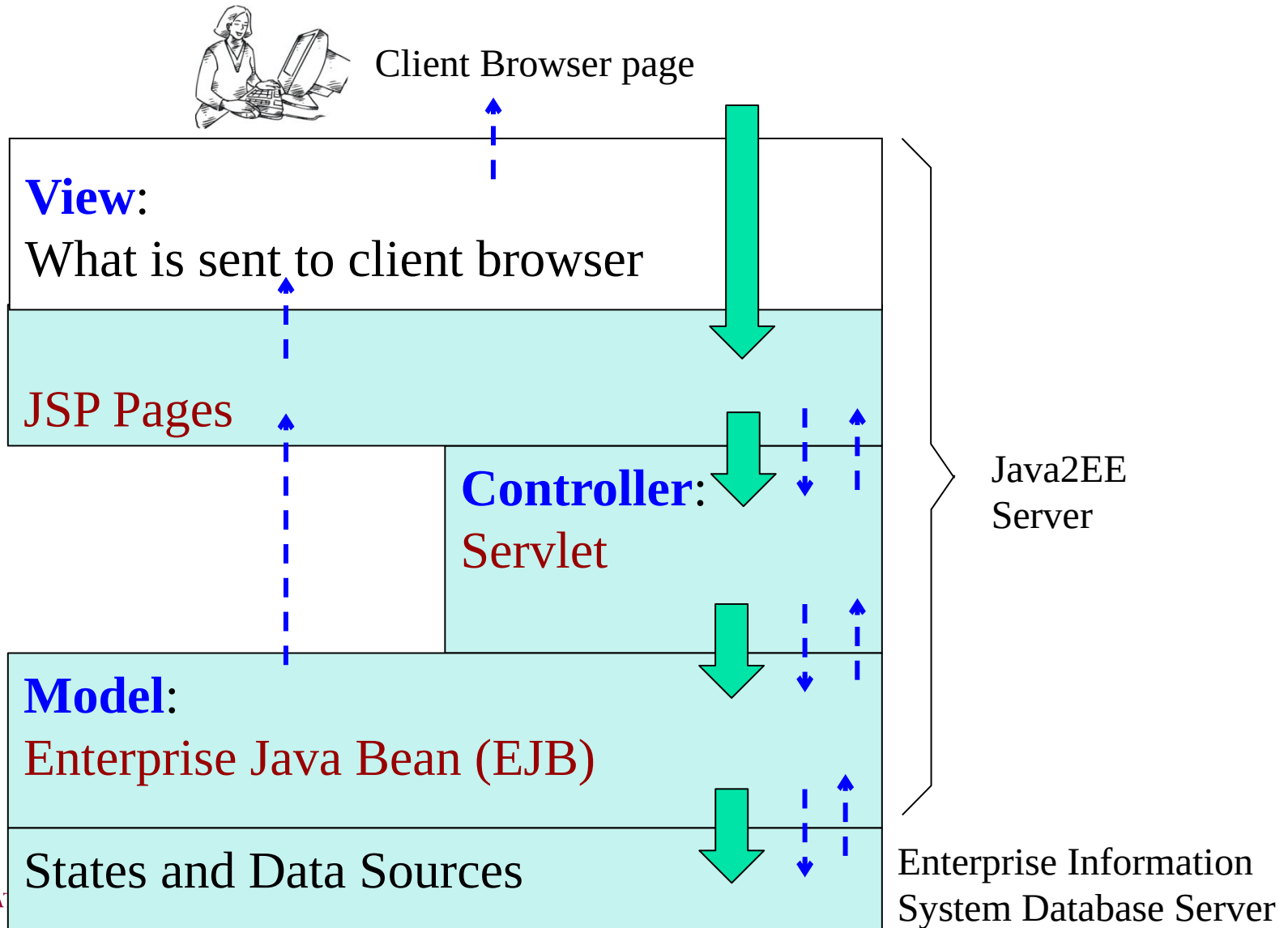
# MVC as an Architecture Pattern



- The **MVC** pattern was originally formulated in the late 1970s by Trygve Reenskaug at Xerox PARC, as part of the Smalltalk.  
[<http://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller>].
- In **MVC**, the system is divided into three groups of components:
  - **M**odel: contains the components representing data and application logic for data processing;
  - **V**iew: Display the results from model;
  - **C**ontroller: It is between the Model and View. It takes input from the user, provides **handlers** of the inputs (simple processing), and let the model to process more complex functions.



# J2EE's MVC Architecture



# Model, View, and Controller in ASP .Net

- **View:** A directory contains the components that will **generate** the application's display (**GUI**) or the presentation layer of the application (output part).
- **Controller:** A directory contains the components that **handle user inputs**, work with the model, and ultimately select a view to render the displays in GUI.
- **Model:** A directory contains objects that that implement the part of the application logic related to the application's data domain. Often, model objects **retrieve and store model state** in a date sets, files, and database.

# Automatically Generate Unit Test

<http://msdn.microsoft.com/en-us/library/dd410597%28v=vs.100%29.aspx>

- In the project, right-click the **Controllers** folder, click **Add**, and then click **Class**.
- In the **Name** text box, type **MapsControllerTest**.
- Click **Add**.

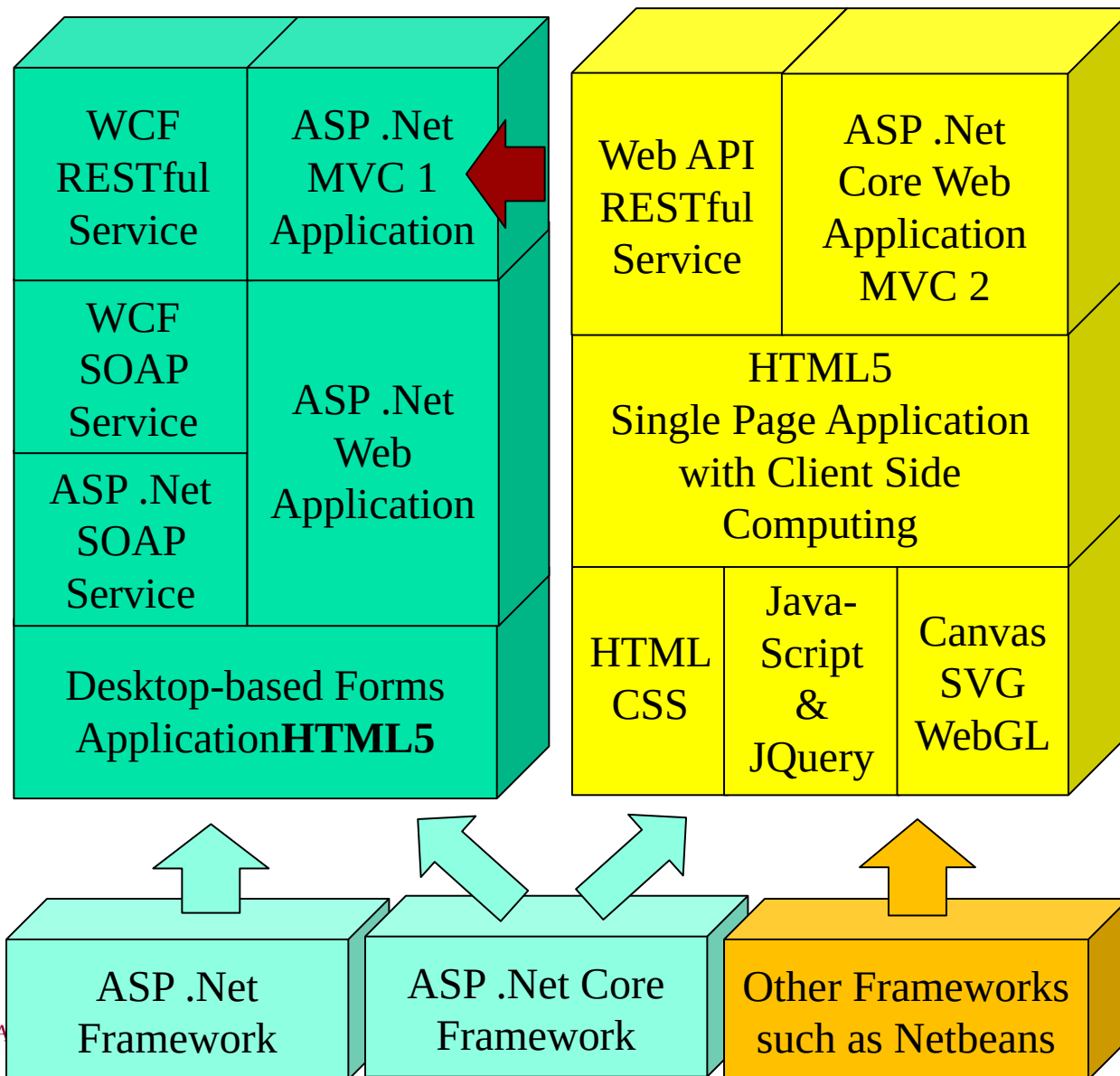
```
namespace MvcBasicWalkthrough.Tests.Controllers {  
    [TestClass]  
    public class MapsControllerTest {  
        [TestMethod] public void ViewMaps() {  
            // Arrange  
            MapsController controller = new MapsController();  
            // Act  
            ViewResult result = controller.ViewMaps() as ViewResult;  
            // Assert  
            Assert.IsNotNull(result);  
        }  
    }  
}
```

# Benefits of MVC Architecture

The loosely coupled feature (compared with ASP .Net architecture) among the tiers in Views, Models, and Controllers:

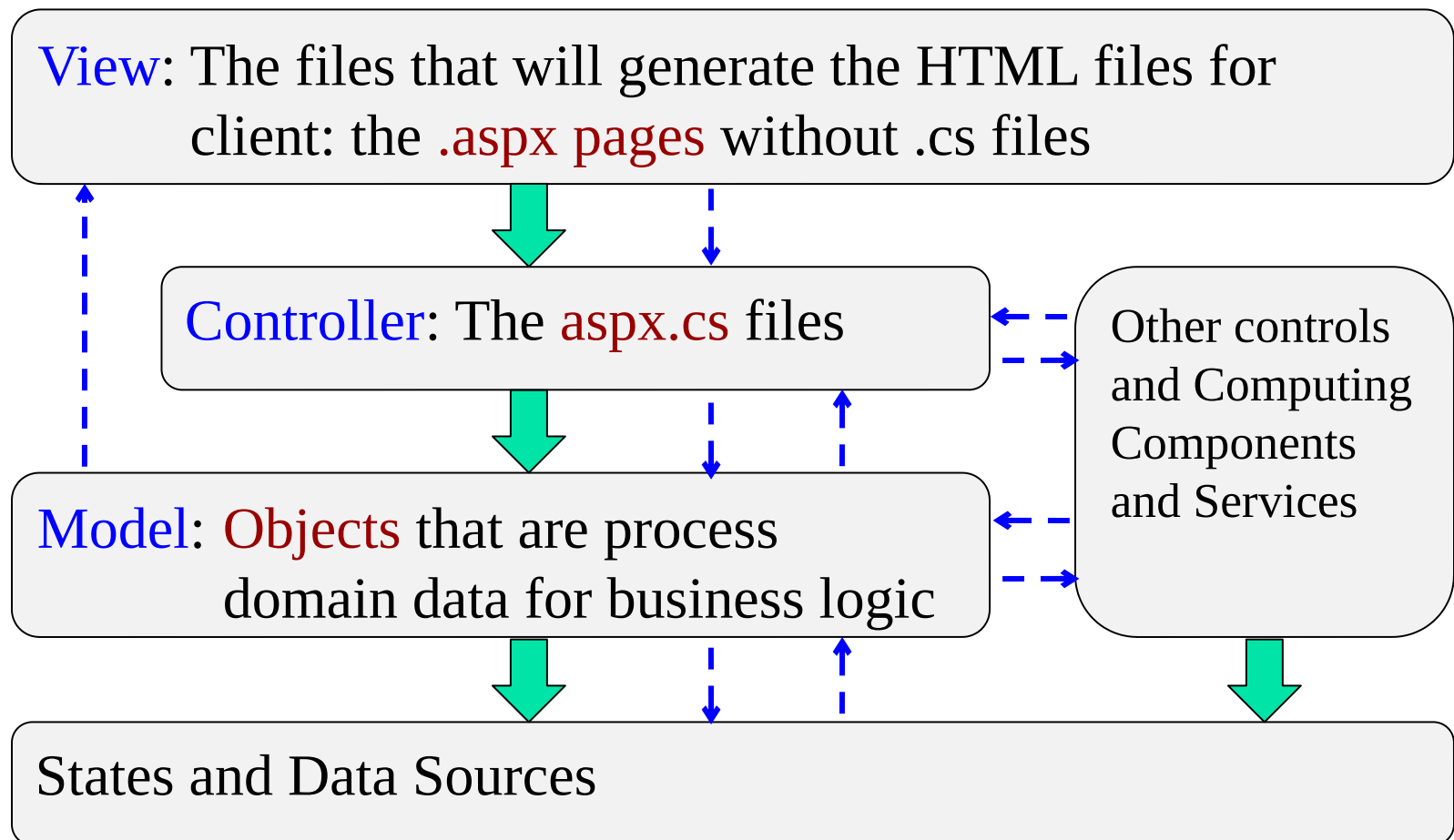
- Allow team members to develop different parts of the application with reduced interactions: **GUI and code are separated**;
- The template can generate unit tests by automatically adding **mock objects** (simulated objects) that mimic the behavior of other objects in controlled ways in the application;
- The restricted use of the **view state**, **page postback**, and **related server controls**, such as **Gridview**, allows the framework to render the XHTML page less dependent on the server features;
- The MVC applications are more friendly with URI-based access used on **RESTful services**, the **feed data formats**, and **cloud computing** environment (CSE446).

# The Evaluation Web Technologies



# MVC 1 in ASP .Net: A Four Tier Architecture

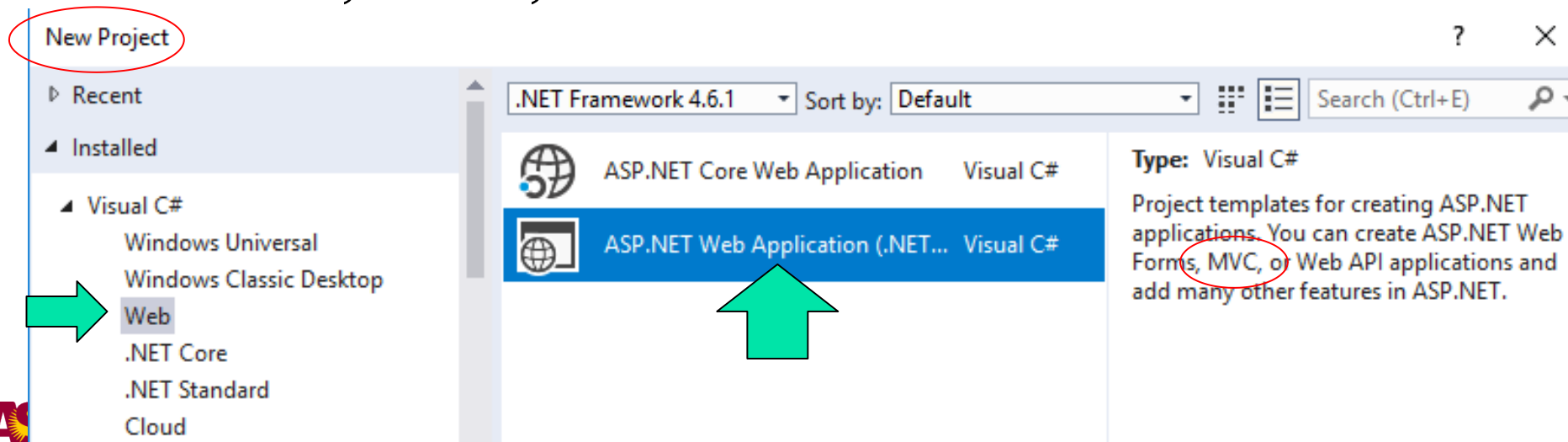
with loosely coupling among the tiers



# ASP .Net MVC Architecture (MVC 1)

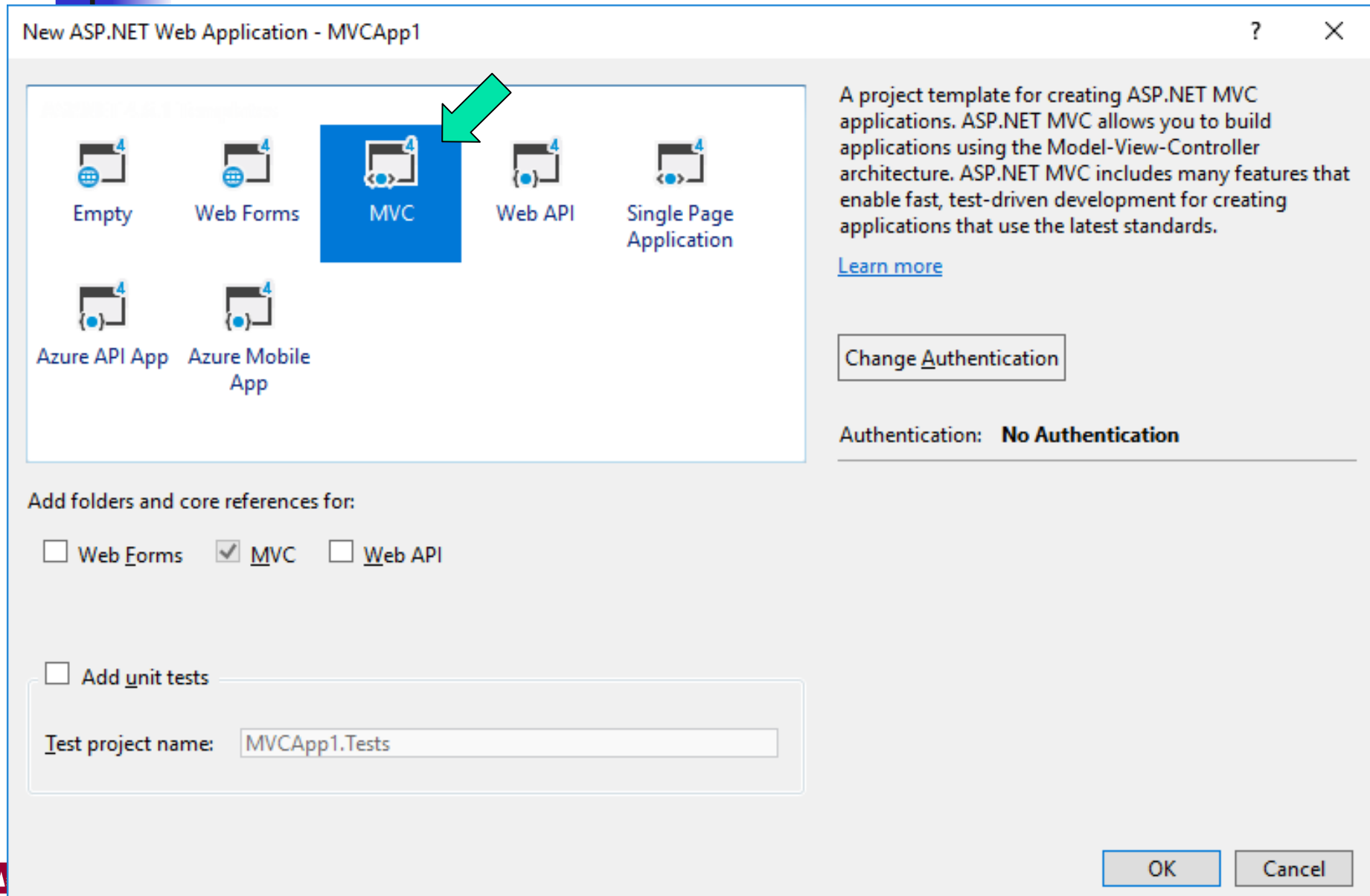
Reading: Text Chapter 5, Section 5.7

- ASP .Net offers MVC as an alternative **project template**;
- It is available for Web Apps, Mobile Apps, and **Cloud Apps**, without using any View State!
- Once selected the MVC template, the main components of the application will be organized in three directories: **Models, Views, and Controllers.**



# MVC can be applied for developing different apps

40





# An MVC 1 Project: Organized Differently

41

The screenshot shows the Visual Studio IDE with the following components:

- Code Editor:** Displays the `HomeController.cs` file. The code includes the following structure:

```
using System.Web;
using System.Web.Mvc;

namespace MvcApplication1.Controllers
{
    public class HomeController : Controller
    {
        public ActionResult Index()
        {
            ViewBag.Message = "Modify this template to jump-start your web application development. MVC patterns."
            return View();
        }

        public ActionResult About()
        {
            ViewBag.Message = "Your app description here."
            return View();
        }

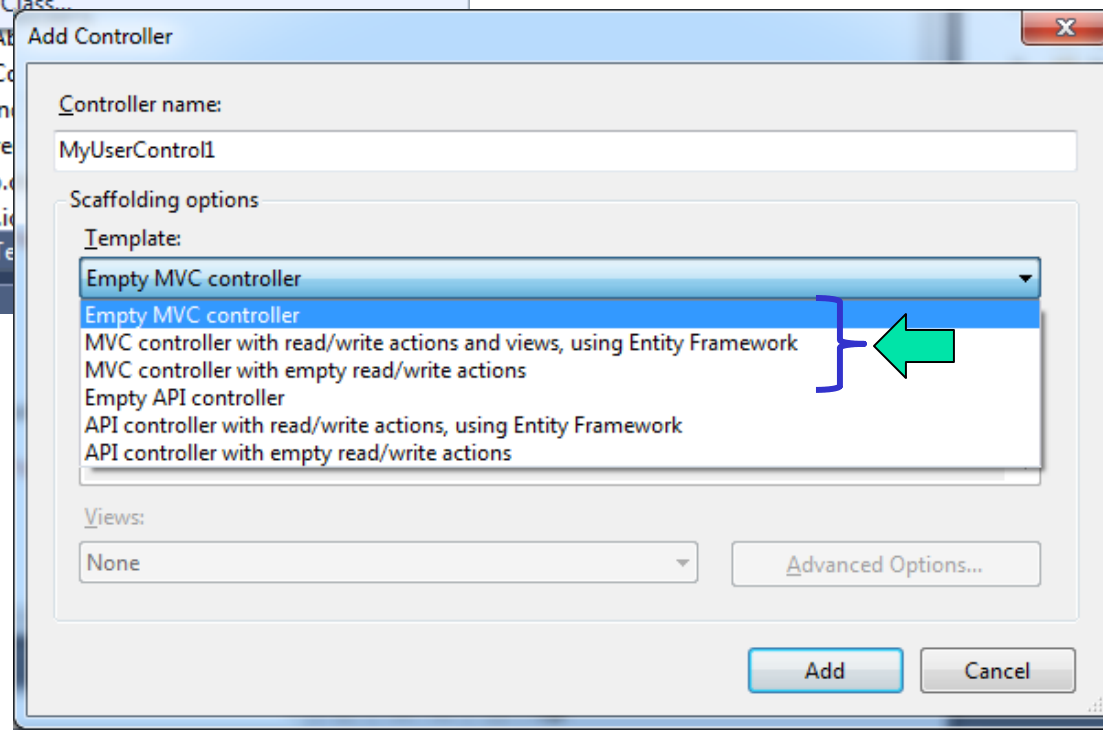
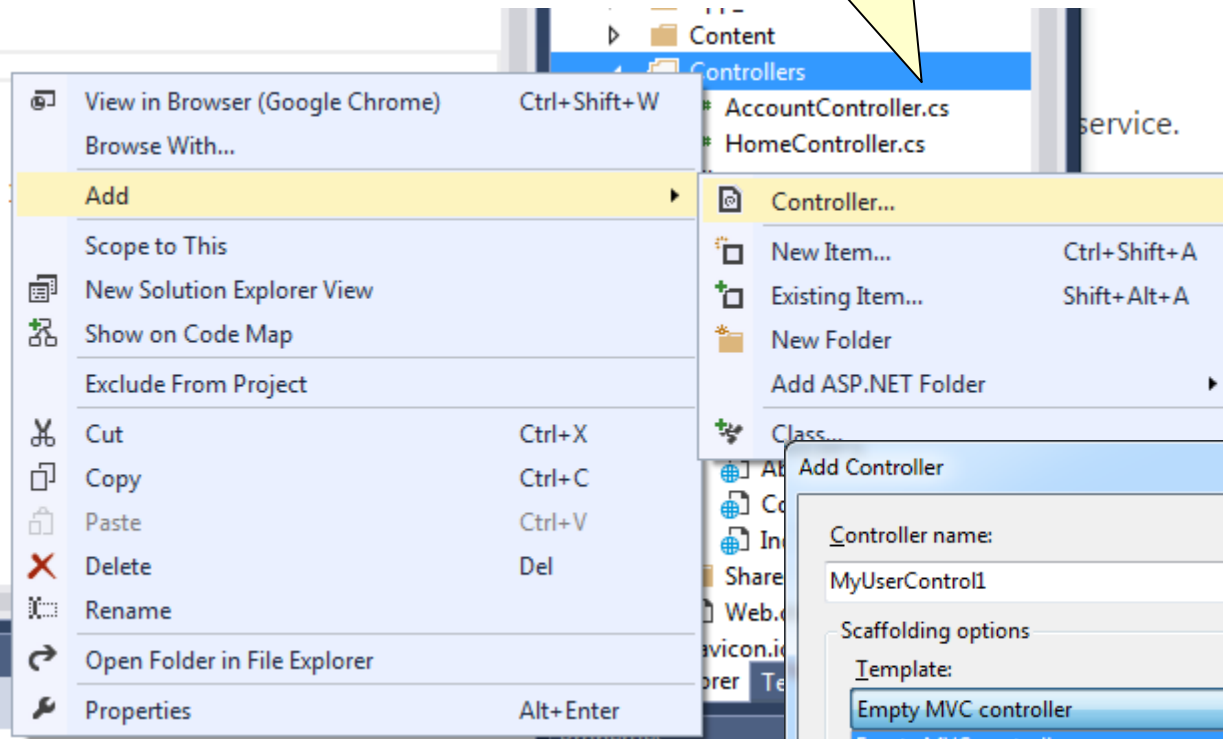
        public ActionResult Contact()
        {
            ViewBag.Message = "Your contact page."
            return View();
        }
    }
}
```
- Solution Explorer:** Shows the project structure for `MvcApplication1`. The folders and files are:
  - Properties
  - References
  - App\_Data
  - App\_Start
  - Content
  - Controllers (containing `AccountController.cs` and `HomeController.cs`)
  - Filters
  - Images
  - Models (containing `AccountModels.cs`)
  - Scripts
  - Views (containing `Account` and `Home` folders)
  - Shared
  - Web.config

Three yellow callout boxes provide additional context:

- Top Callout:** ".aspx.cs files, are moved to here" (points to the `Controllers` folder in the Solution Explorer).
- Middle Callout:** "Put any business logic not related I/O are here" (points to the `HomeController.cs` file in the Solution Explorer).
- Bottom Callout:** ".aspx files, without .aspx.cs files" (points to the `Views` folder in the Solution Explorer).

# Add my Controller

Right click



# New Items are Added to the MVC 1 Project

The screenshot displays the Visual Studio IDE with the 'Solution Explorer' on the right and the 'Server Objects & Events' window on the left. The 'Server Objects & Events' window shows the code for 'MyUserController1.cs', which is a traditional user control with both .ascx and .cs files. The 'Solution Explorer' shows the project structure for 'ASPNetMvcWebApp1' (2 projects). The project structure includes:

- Properties
- References
- App\_Data
- Content
- Controllers
  - AccountController.cs
  - HomeController.cs
  - MyUserController1.cs
- Models
  - AccountModels.cs
- Scripts
- Views
  - Account
    - ChangePassword.aspx
    - ChangePasswordSuccess.aspx
    - LogOn.aspx
    - Register.aspx
  - Home
    - About.aspx
    - Index.aspx
    - ViewPageStaff.aspx
  - Shared
    - MyViewUserController1.ascx
    - MyWebUserController.ascx
    - MyWebUserController.ascx.cs
    - MyWebUserController.ascx.designer.cs

Annotations explain the addition of new items:

- MVC user control's .cs file is added separately here.** (Points to MyUserController1.cs in the Controllers folder)
- A View Page is added here** (Points to ViewPageStaff.aspx in the Home folder)
- A MVC user control. Only .ascx file is added** (Points to MyViewUserController1.ascx in the Shared folder)
- A traditional User control with both .ascx and .cs files** (Points to MyWebUserController.ascx and MyWebUserController.ascx.cs in the Shared folder)

# How are **View** and **Controller** Connected?

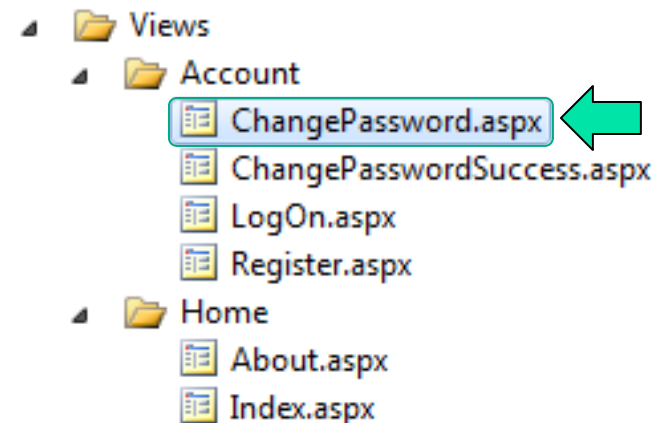
44

- In ASP .Net, the .aspx files and .aspx.cs files are placed together in the same folder. The links are **implicit**. You must click the buttons to create the implicit links. It works because the two pages are tied together.
- **In MVC, the links** are explicit.
  - Each View component (.aspx file) in browser contains a list of **JavaScript** calls.
  - The Controller of the component contains the methods to be called. View page and .cs page can be completely separated.

# ChangePassword in ChangePassword.aspx, making JS calls

45

```
<%@ Page Language="C#" MasterPageFile="~/Views/Shared/Site.Master" Inherits="System.Web.Mvc.ViewPage<MvcApp.Models.ChangePasswordModel>"
    >
<asp:Content ID="changePasswordTitle" ContentPlaceHolderID="TitleContent" runat="server">
    Change Password
</asp:Content>
<asp:Content ID="changePasswordContent" ContentPlaceHolderID="MainContent" runat="server">
    <h2>Change Password</h2>
    <p> Use the form below to change your password. </p>
    <p> New passwords are required to be a minimum of <%: Membership.MinRequiredPasswordLength %> characters in length. </p>
    <script src="<%: Url.Content("~/Scripts/jquery.validate.min.js") %>" type="text/javascript"></script>
    <script src="<%: Url.Content("~/Scripts/jquery.validate.unobtrusive.min.js") %>" type="text/javascript"></script>
    <% using (Html.BeginForm()) { %>
        <%: Html.ValidationSummary(true, "Password change was unsuccessful. Please correct the errors and try again.") %>
        <div>
            <fieldset>
                <legend>Account Information</legend>
                <div class="editor-label">
                    <%: Html.LabelFor(m => m.OldPassword) %>
                </div>
                <div class="editor-field">
                    <%: Html.PasswordFor(m => m.OldPassword) %>
                    <%: Html.ValidationMessageFor(m => m.OldPassword) %>
                </div>
                <div class="editor-label">
                    <%: Html.LabelFor(m => m.NewPassword) %>
                </div>
                <div class="editor-field">
                    <%: Html.PasswordFor(m => m.NewPassword) %>
                    <%: Html.ValidationMessageFor(m => m.NewPassword) %>
                </div>
                <div class="editor-label">
                    <%: Html.LabelFor(m => m.ConfirmPassword) %>
                </div>
                <div class="editor-field">
                    <%: Html.PasswordFor(m => m.ConfirmPassword) %>
                    <%: Html.ValidationMessageFor(m => m.ConfirmPassword) %>
                </div>
            </fieldset>
        </div>
    <% %>
    </div>
</asp:Content>
```



# ChangePassword method in AccountController.cs

// POST: /Account/ChangePassword

[Authorize]

[HttpPost]

public ActionResult **ChangePassword**(ChangePasswordModel model) {

if (ModelState.IsValid) {

// ChangePassword will throw an exception rather than return false in certain failure scenarios.

bool changePasswordSucceeded;

try {

MembershipUser currentUser = Membership.GetUser(User.Identity.Name, true /\* userIsOnline \*/);

changePasswordSucceeded = currentUser.ChangePassword(model.OldPassword, model.NewPassword);

}

catch (Exception){

changePasswordSucceeded = false;

}

if (changePasswordSucceeded) {

return RedirectToAction("ChangePasswordSuccess");

}

else {

ModelState.AddModelError("", "The current password is incorrect or the new password is invalid.");

}

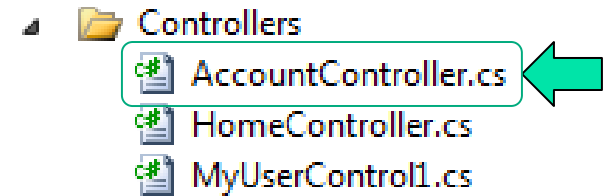
}

// If we got this far, something failed, redisplay form

return View(model);

}


To be called  
from JavaScript




# ChangePassword method in AccountModels.cs

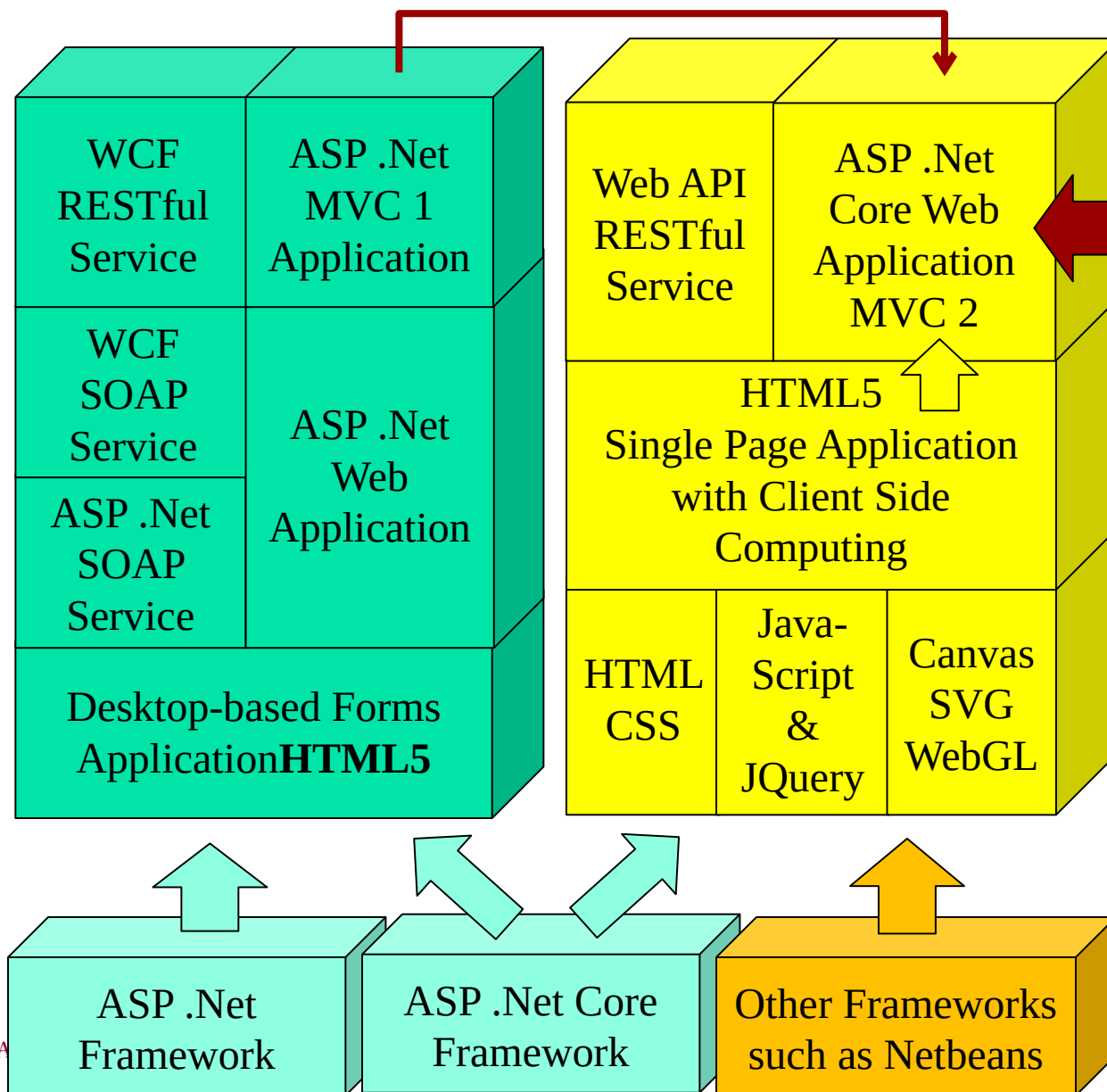
```
public bool ChangePassword(string userName, string oldPassword, string newPassword) {  
    if (String.IsNullOrEmpty(userName)) throw new ArgumentException("Value cannot be null  
or empty.", "userName");  
    if (String.IsNullOrEmpty(oldPassword)) throw new ArgumentException("Value cannot be  
null or empty.", "oldPassword");  
    if (String.IsNullOrEmpty(newPassword)) throw new ArgumentException("Value cannot be  
null or empty.", "newPassword");  
    // The underlying ChangePassword() will throw an exception rather  
    // than return false in certain failure scenarios.  
    try {  
        MembershipUser currentUser = _provider.GetUser(userName, true /* userIsOnline */);  
        return currentUser.ChangePassword(oldPassword, newPassword);  
    }  
    catch (ArgumentException) {  
        return false;  
    }  
    catch (MembershipPasswordException) {  
        return false;  
    }  
}
```

Access  
database

 Models

 AccountModels.cs 

# From MVC 1 to MVC 2

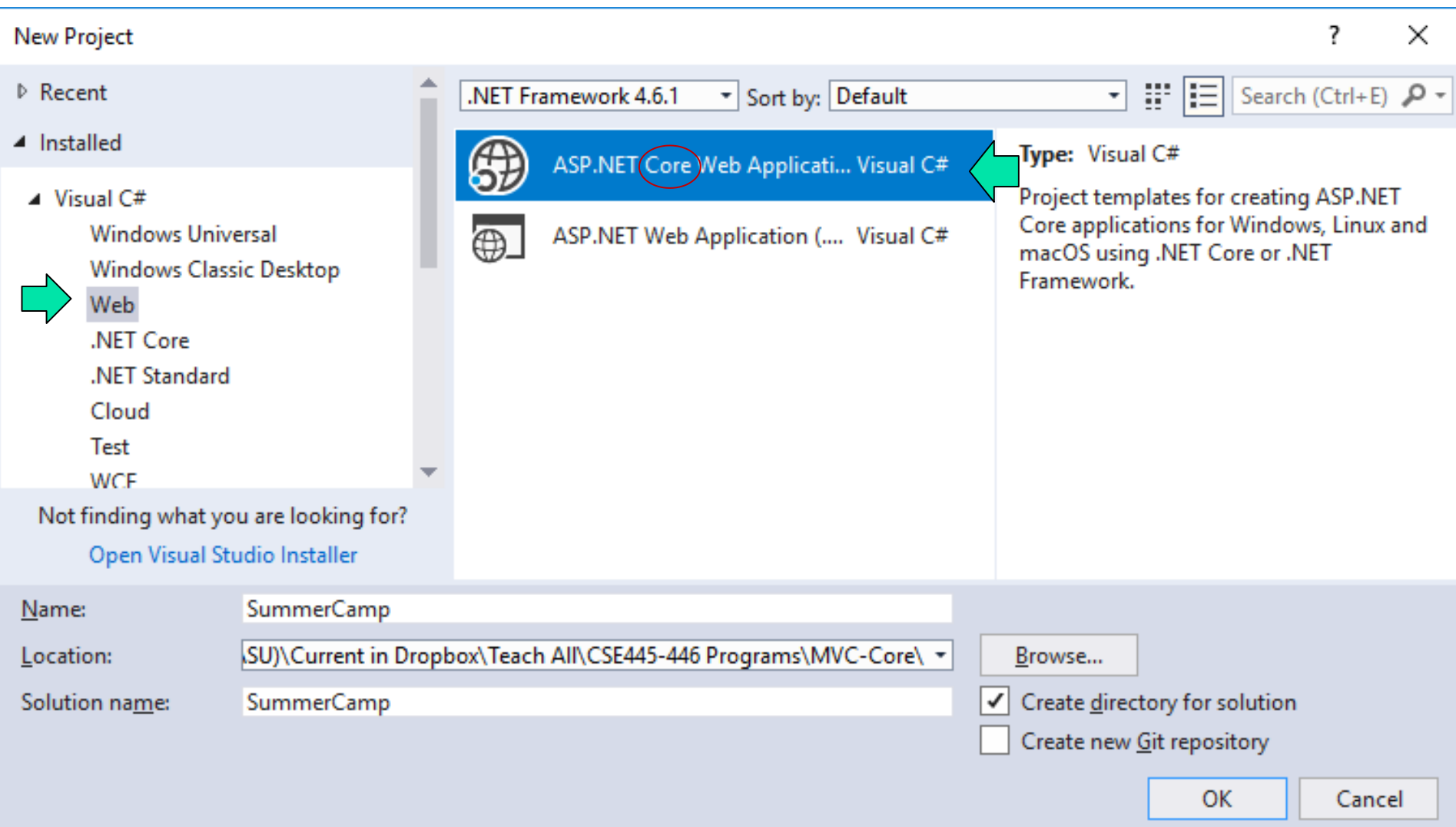




# Case Study: Summer Camp Registration

- Create the main page for information
- Inviting for enrollment
- Signup form
- Store student list in a List structure
- Showing Controllers □ View
- Showing Controllers □ Model

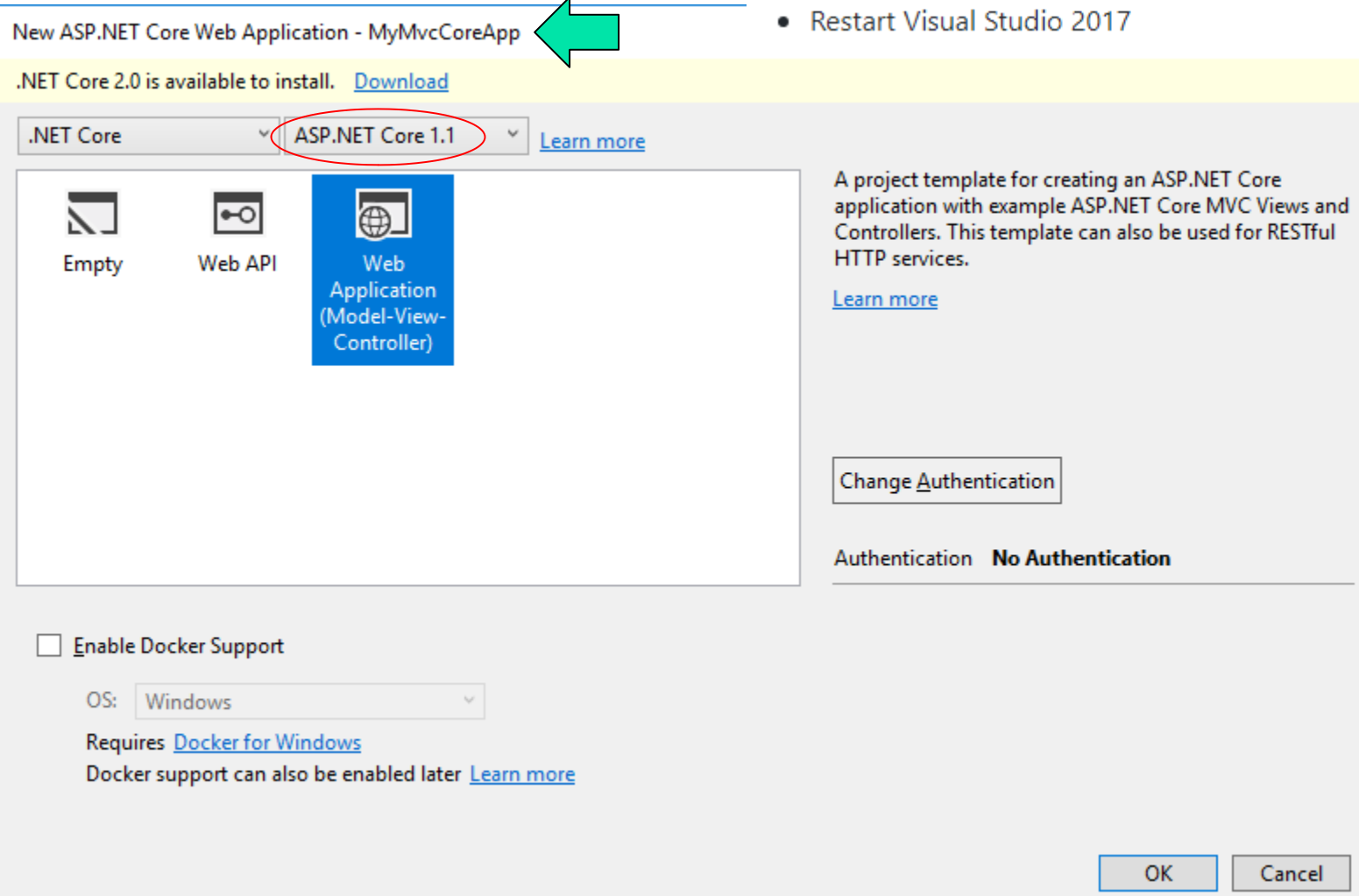
# Getting Started with MVC 2 on ASP .Net Core



# Updating to ASP .Net Core 2.0 from 1.1

To get started with .NET Core 2.0 in Visual Studio 2017 follow the steps

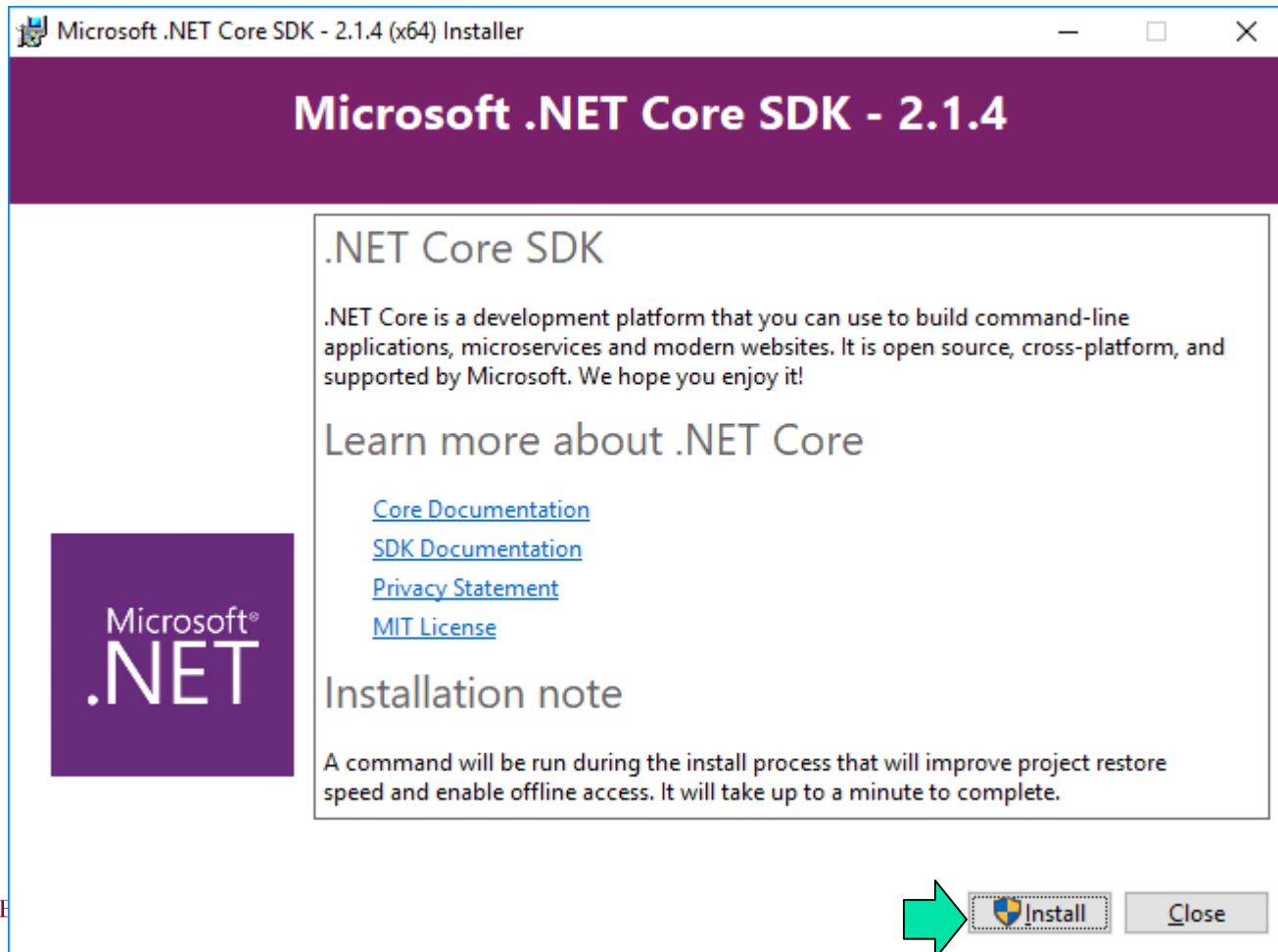
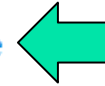
- Install .NET Core 2.0 - <https://www.microsoft.com/net/core>
- Restart Visual Studio 2017



# Updating to ASP .Net Core 2.0

To get started with .NET Core 2.0 in Visual Studio 2017 follow the steps

- Install .NET Core 2.0 - <https://www.microsoft.com/net/core>
- Restart Visual Studio 2017



# Updated to ASP .Net Core 2.0

New ASP.NET Core Web Application - SummerCamp

.NET Core | **ASP.NET Core 2.0** [Learn more](#)

Empty Web API Web Application **Web Application (Model-View-Controller)** Angular

React.js React.js and Redux

A project template for creating an ASP.NET Core application with example ASP.NET Core MVC Views and Controllers. This template can also be used for RESTful HTTP services. [Learn more](#)

**Change Authentication**

☒ No Authentication  
☐ Individual User Accounts  
☐ Work or School Accounts  
☐ Windows Authentication

[Learn more about third-party open source authentication options](#) OK Cancel

Authentication **No Authentication**

☐ Enable Docker Support

OS: Windows

Requires [Docker for Windows](#)

Docker support can also be enabled later [Learn more](#)

OK Cancel

# MVC 2 Project in ASP .Net Core

HomeController.cs SummerCamp

SummerCamp SummerCamp.Controllers.Home Index()

```
3 using System.Diagnostics;
4 using System.Linq;
5 using System.Threading.Tasks;
6 using Microsoft.AspNetCore.Mvc;
7 using SummerCamp.Models;
8
9 namespace SummerCamp.Controllers
10 {
11     References
12     public class HomeController : Controller
13     {
14         References
15         public IActionResult Index()
16         {
17             return View();
18         }
19
20         References
21         public IActionResult About()
22         {
23             ViewData["Message"] = "Your application description page";
24
25             return View();
26         }
27
28         References
29         public IActionResult Contact()
30         {
31             ViewData["Message"] = "Your contact page";
32
33             return View();
34         }
35     }
36 }
```

MVC Controllers with .cs files

Models with .cs files

MVC Views with .cshtml files for html and binding with models:  
@{ ViewData["Title"] = "Home Page"; }

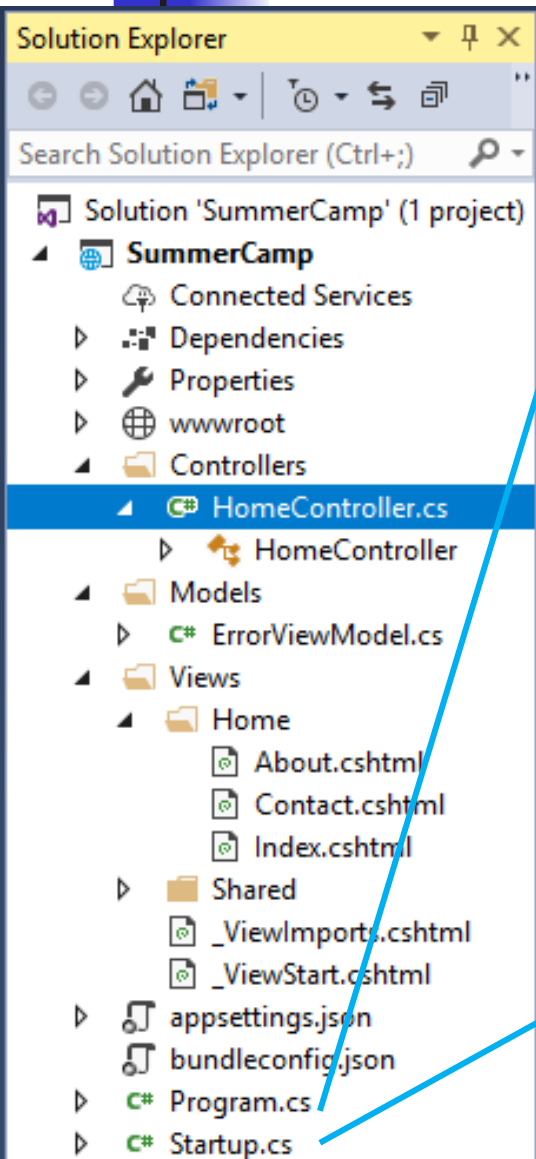
Solution Explorer

Search Solution Explorer (Ctrl+;)

Solution 'SummerCamp' (1 project)

- SummerCamp
  - Connected Services
  - Dependencies
  - Properties
  - wwwroot
  - Controllers
    - C# HomeController.cs
  - Models
    - HomeController
    - Models
    - C# ErrorViewModel.cs
  - Views
    - Home
      - About.cshtml
      - Contact.cshtml
      - Index.cshtml
    - Shared
      - \_ViewImports.cshtml
      - \_ViewStart.cshtml
  - appsettings.json
  - bundleconfig.json
  - Program.cs
  - Startup.cs

# Program Starting Point



```

namespace SummerCamp
{
    0 references
    public class Program
    {
        0 references
        public static void Main(string[] args)
        {
            BuildWebHost(args).Run();
        }

        1 reference
        public static IWebHost BuildWebHost(string[] args) =>
            WebHost.CreateDefaultBuilder(args)
                .UseStartup<Startup>()
                .Build();

        public Startup(IConfiguration configuration)|...|

        1 reference
        public IConfiguration Configuration { get; }

        // This method gets called by the runtime. Use this method to add services to the container.
        0 references
        public void ConfigureServices(IServiceCollection services)|...|

        // This method gets called by the runtime. Use this method to configure the HTTP request pipeline.
        0 references
        public void Configure(IApplicationBuilder app, IHostingEnvironment env)
        {
            if (env.IsDevelopment())
            {
                app.UseBrowserLink();
                app.UseDeveloperExceptionPage();
            }
            else { app.UseExceptionHandler("/Home/Error"); }

            app.UseStaticFiles();

            app.UseMvc(routes =>
            {
                routes.MapRoute(
                    name: "default",
                    template: "{controller=Home}/{action=Index}/{id?}");
            });
        }
    }
}

```

# HomeController: System Generated

```
namespace SummerCamp.Controllers
{
    References
    public class HomeController : Controller
    {
        References
        public IActionResult Index()
        {
            return View();
        }

        References
        public IActionResult About()
        {
            ViewData["Message"] = "Your application description page.";
            return View();
        }

        2
        References
        public IActionResult Contact()
        {
            ViewData["Message"] = "Your contact page.";
            return View();
        }

        References
        public IActionResult Error()
        {
            return View(new ErrorViewModel { RequestId = Activity.Current?.Id ?? HttpContext.Trace
        }
    }
}
```

localhost:6547/Home/About

SummerCamp Home About Contact

## About

Your application description page.

Use this area to provide additional information.

© 2018 - SummerCamp



# Correlation Between Controller and View

**Solution Explorer**

- Solution 'SummerCamp' (1 project)
  - SummerCamp
    - Connected Services
    - Dependencies
    - Properties
    - wwwroot
    - Controllers
      - HomeController.cs**
    - Models
      - ErrorViewModel.cs
    - Views
      - Home
        - About.cshtml
        - Contact.cshtml
        - Index.cshtml
      - Shared
        - \_ViewImports.cshtml
        - \_ViewStart.cshtml
    - appsettings.json
    - bundleconfig.json
    - Program.cs
    - Startup.cs

**HomeController.cs**

```
public IActionResult About()
{
    ViewData["Message"] = "Your application description page.";
    return View();
}
```

**About.cshtml**

```
1 @{}
2     ViewData["Title"] = "About";
3 }
4 <h2>@ViewData["Title"]</h2>
5 <h3>@ViewData["Message"]</h3>
6
7 <p>Use this area to provide additional information.</p>
```

**Define "Title"**

localhost:6547/Home/About

SummerCamp Home About Contact

About

Your application description page.

Use this area to provide additional information.

# Modify the Index.cshtml Page

- Write your code in Index page:

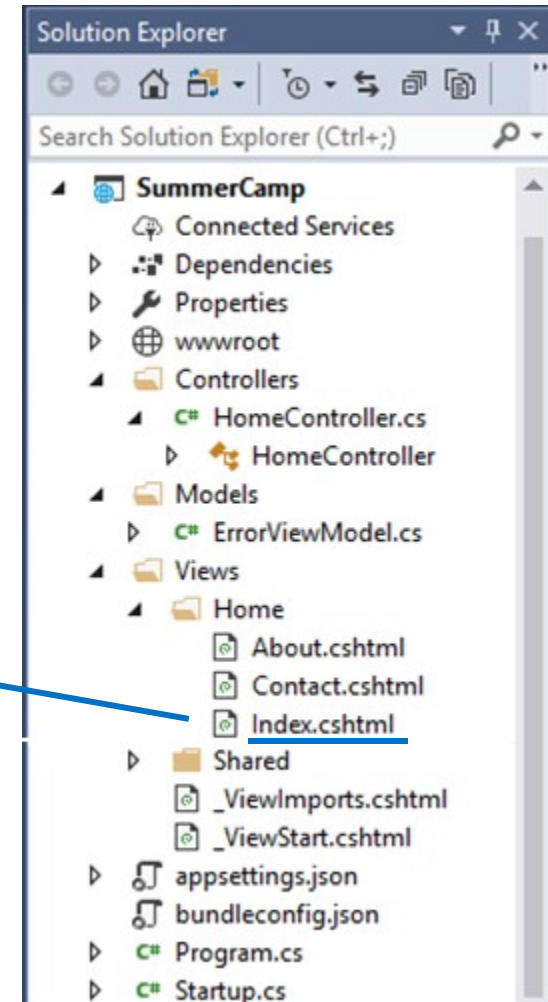
```

Index.cshtml  HomeController.cs  StudentList.cs  Confirmation.cshtml

<code>
Layout = null;
}

<!DOCTYPE html>
<html>
<head>
  <meta name="viewport" content="width=device-width" />
  <title>Semester</title>
  <link rel="stylesheet" href="/lib/bootstrap/dist/css/bootstrap.css" />
</head>
<body>
  <div class="text-center">
    <h3>Hello high school students, it is @ViewBag.Semester.</h3>
    <h3>We are going to offer a summer robotics camp</h3>
    <h4>You can sign up the camp now:</h4>
    <a class="btn btn-primary" asp-action="SignupForm">Sign Up Now</a>
  </div>
</body>
</html>
  </code>

```



A page to be added

# Modify HomeController.cs

localhost:6547

- Link the code to the SemesterView page

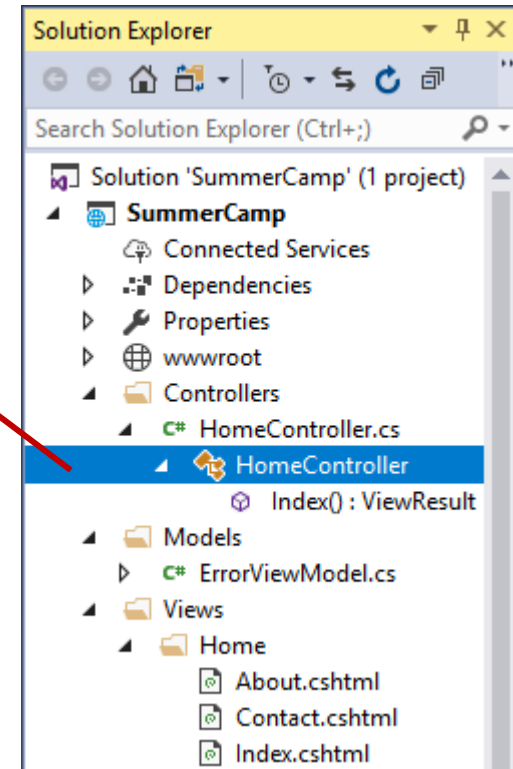
Hello high school students, it is **Spring**!

We are going to offer a summer robotics camp

You can sign up the camp now:

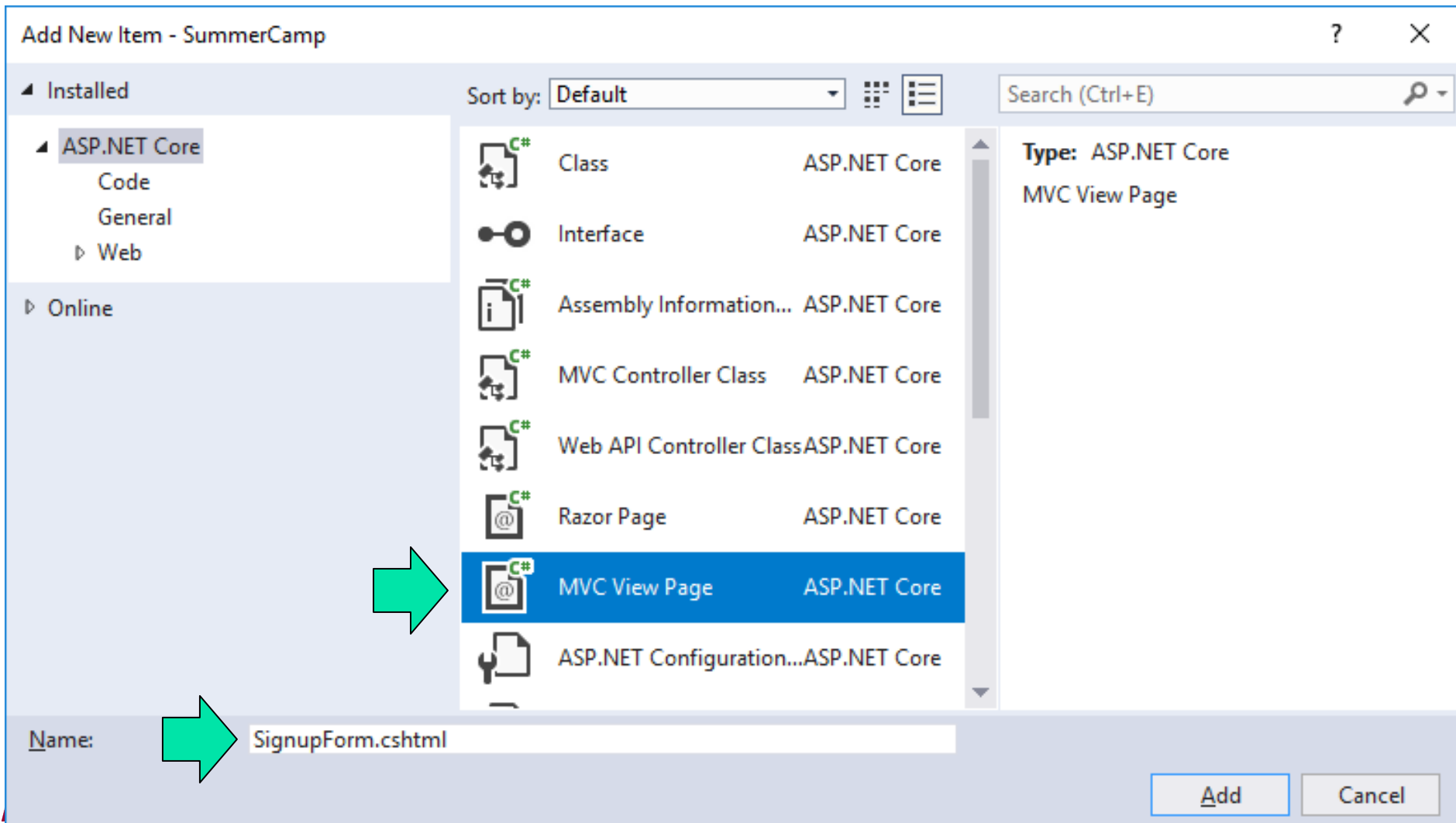
Sign Up Now

```
namespace SummerCamp.Controllers
{
    References
    public class HomeController : Controller
    {
        References
        public ActionResult Index()
        {
            int m = DateTime.Now.Month;
            if (m < 6)
                ViewBag.Semester = "Spring";
            else if (m < 8)
                ViewBag.Semester = "Summer";
            else if (m >= 8)
                ViewBag.Semester = "Fall";
            return View("Index");
        }
    }
}
```



# Add a New MVC View Page

- Right Views-Home and Choose Add ▢ New Item:

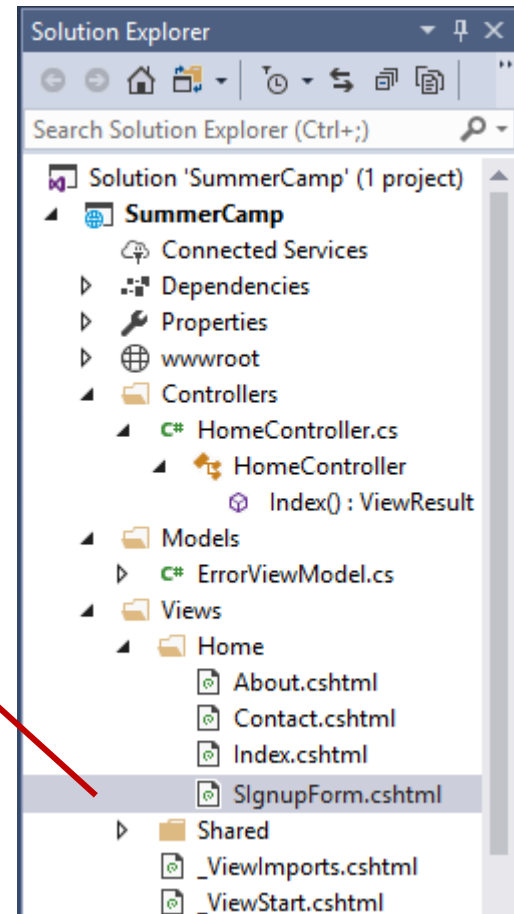


# MVC View Page SignupForm Added

```

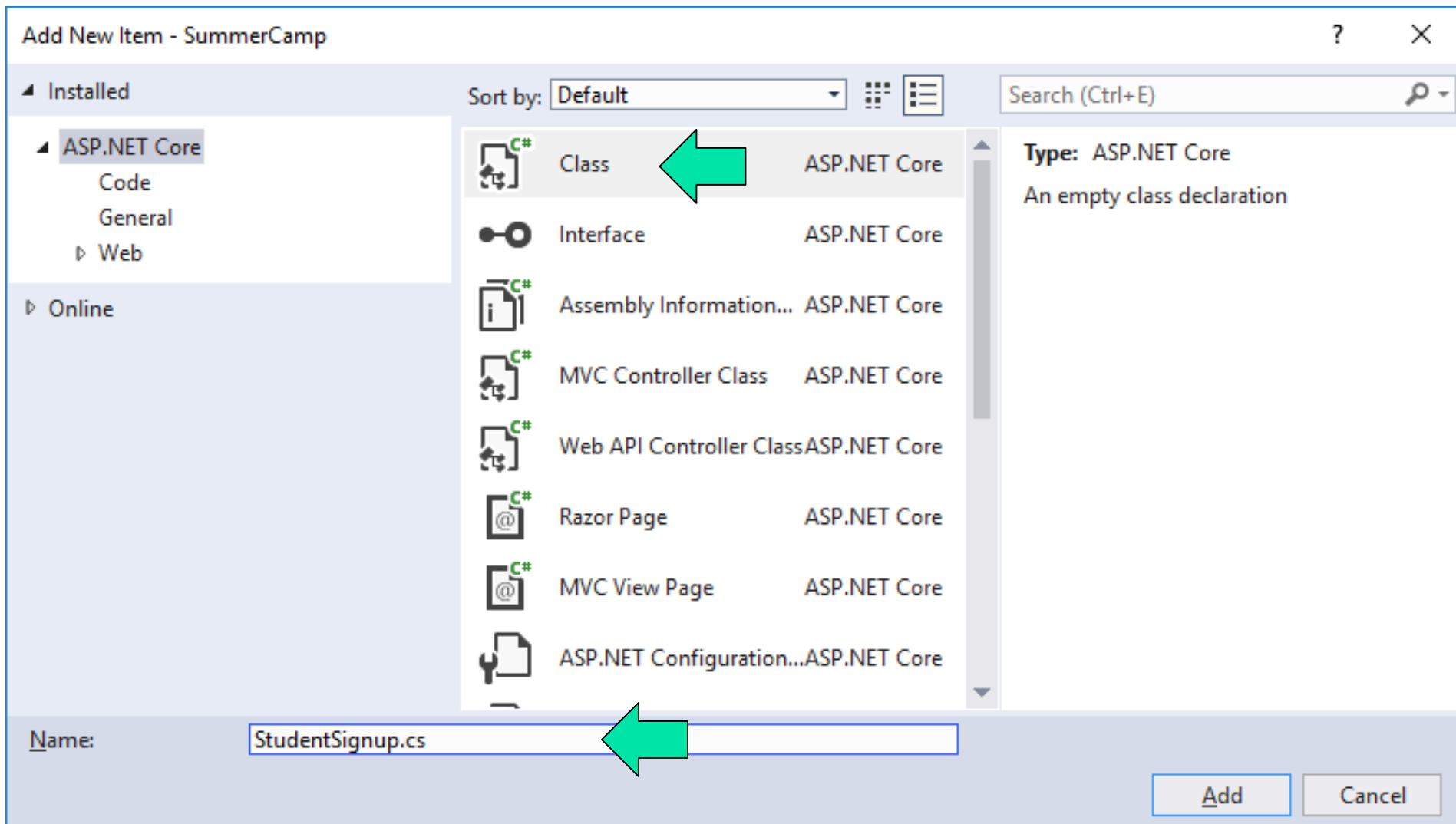
<body>
  <div class="panel panel-success">
    <div class="panel-heading text-center"><h4>RSVP</h4></div>
    <div class="panel-body">
      <form class="p-a-1" asp-action="SignupForm" method="post">
        <div asp-validation-summary="All"></div>
        <div class="form-group">
          <label asp-for="Name">Your name:</label>
          <input class="form-control" asp-for="Name" />
        </div>
        <div class="form-group">
          <label asp-for="Email">Your email:</label>
          <input class="form-control" asp-for="Email" />
        </div>
        <div class="form-group">
          <label>Are you a high school student currently?</label>
          <select class="form-control" asp-for="Signup">
            <option value="">Option</option>
            <option value="true">Yes, I am a high school student</option>
            <option value="false">No, I am not a high school student</option>
          </select>
        </div>
        <div class="text-center">
          <button class="btn btn-primary" type="submit">
            Submit the Form Now
          </button>
        </div>
      </form>
    </div>
  </div>
</body>

```



# Add a Model to deal with Student Signup

- Right-click Project □ Add □ Class



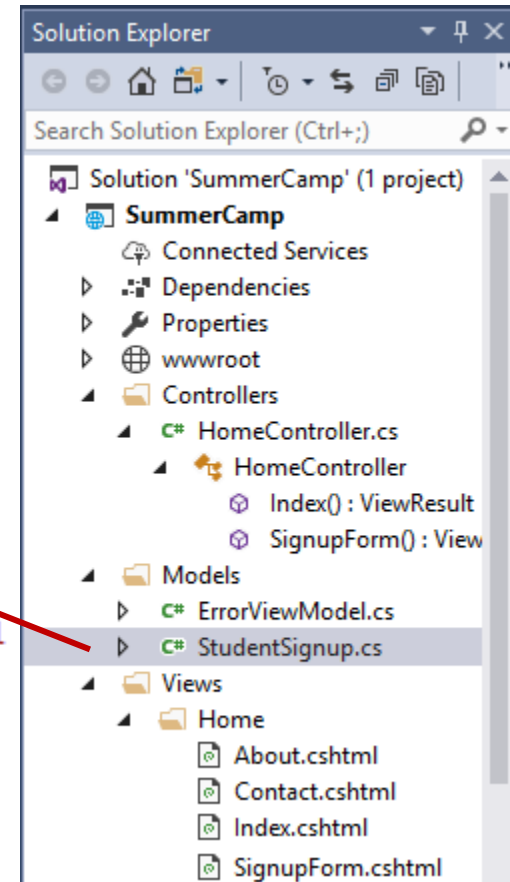
# Define the Signup Information in Model Page

```
using System.ComponentModel.DataAnnotations;

namespace SummerCamp.Models
{
    4 references
    public class StudentSignup
    {
        [Required(ErrorMessage = "Please enter your name")]
        2 references
        public string Name { get; set; }

        [Required(ErrorMessage = "Please enter your email address")]
        [RegularExpression(".+\\@.+\\..+",
            ErrorMessage = "Please enter a valid email address")]
        2 references
        public string Email { get; set; }

        [Required(ErrorMessage = "Please let us know if you are a high school
        1 reference
        public bool? Signup { get; set; }
    }
}
```



Your name:

Your email:

Are you a high school student currently?

Option

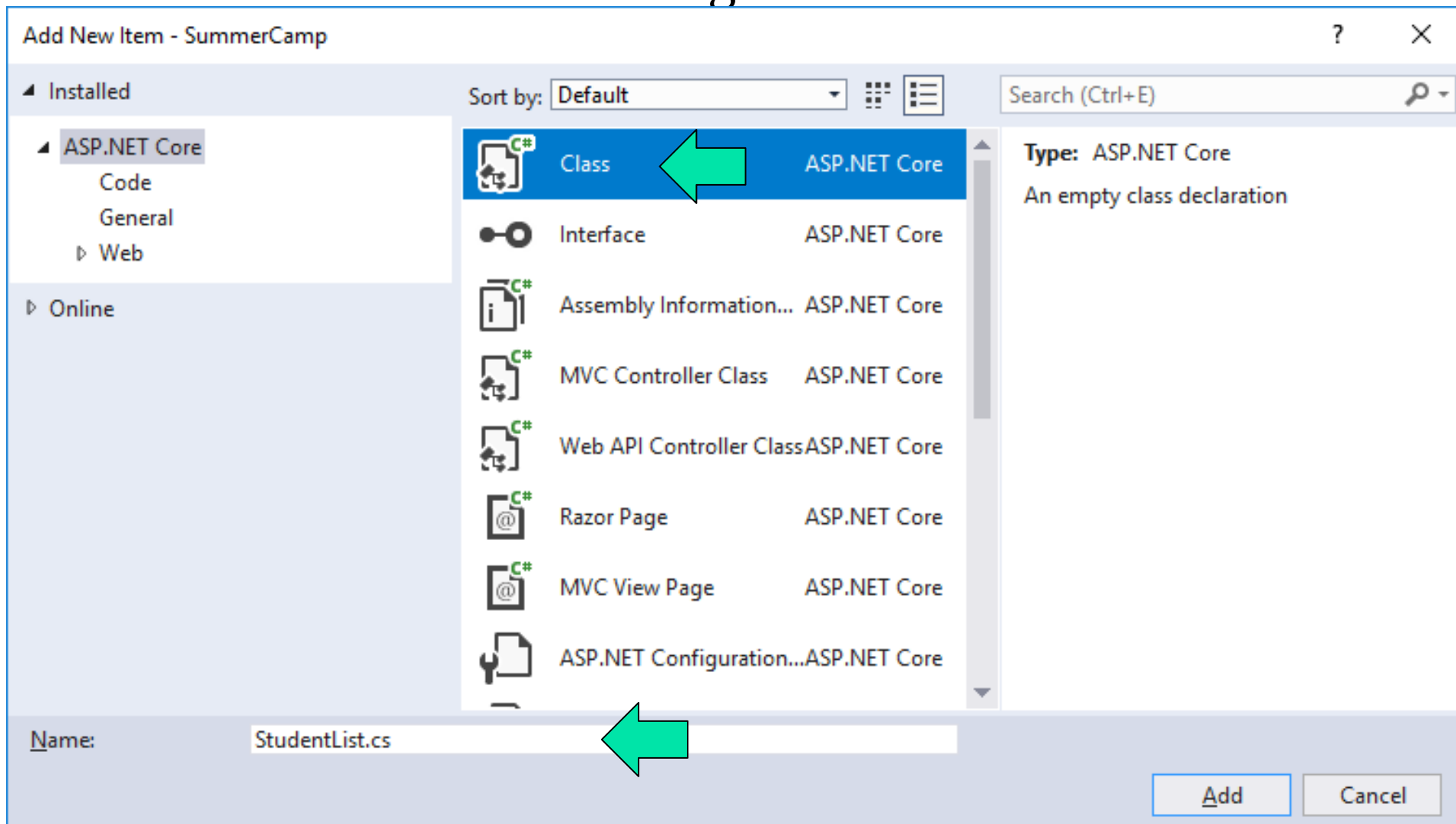
Submit the Form Now

Take to the response form

Start the Program and  
click Signup Now:

# Receive & Store Signed up Student Information

- Add a Model for storing student list:





# Use a List Type to Hold the Student List

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace SummerCamp.Models
{
    2 references
    public class StudentList
    {
        private static List<StudentSignup> mylist = new List<StudentSignup>();

        1 reference
        public static IEnumerable<StudentSignup> Responses
        {
            get
            {
                return mylist;
            }
        }

        1 reference
        public static void AddResponse(StudentSignup student)
        {
            mylist.Add(student);
        }
    }
}
```

Search Solution Explorer (Ctrl+;)

Solution 'SummerCamp' (1 project)

- SummerCamp
  - Connected Services
  - Dependencies
  - Properties
  - wwwroot
  - Controllers
    - HomeController.cs
  - Models
    - ErrorViewModel.cs
    - StudentList.cs**
    - StudentSignup.cs
  - Views
    - Home
      - About.cshtml
      - Confirmation.cshtml
      - Contact.cshtml
      - Index.cshtml
      - RegistrationList.cshtml
      - SignupForm.cshtml
    - Shared
      - \_ViewImports.cshtml
      - \_ViewStart.cshtml

# Demo:

Hello high school students, it is Spring!

We are going to offer a summer robotics camp

You can sign up the camp now:

Sign Up Now

Your name:

Mary Smith

Your email:

mary@asu.edu

Are you a high school student currently?

Yes, I am a high school student

Submit the Form Now

Thank you, Mary Smith!

Thank you for signing up to the summer robotics camp



Click [here](#) to see who have signed up.

Here is the list of students who have signed up the summer robotics camp

Name	Email
Jone Doe	jonh@asu.edu
Mary Smith	mary@asu.edu

# Chapter Summary

