Welcome to Backend programing

Topics

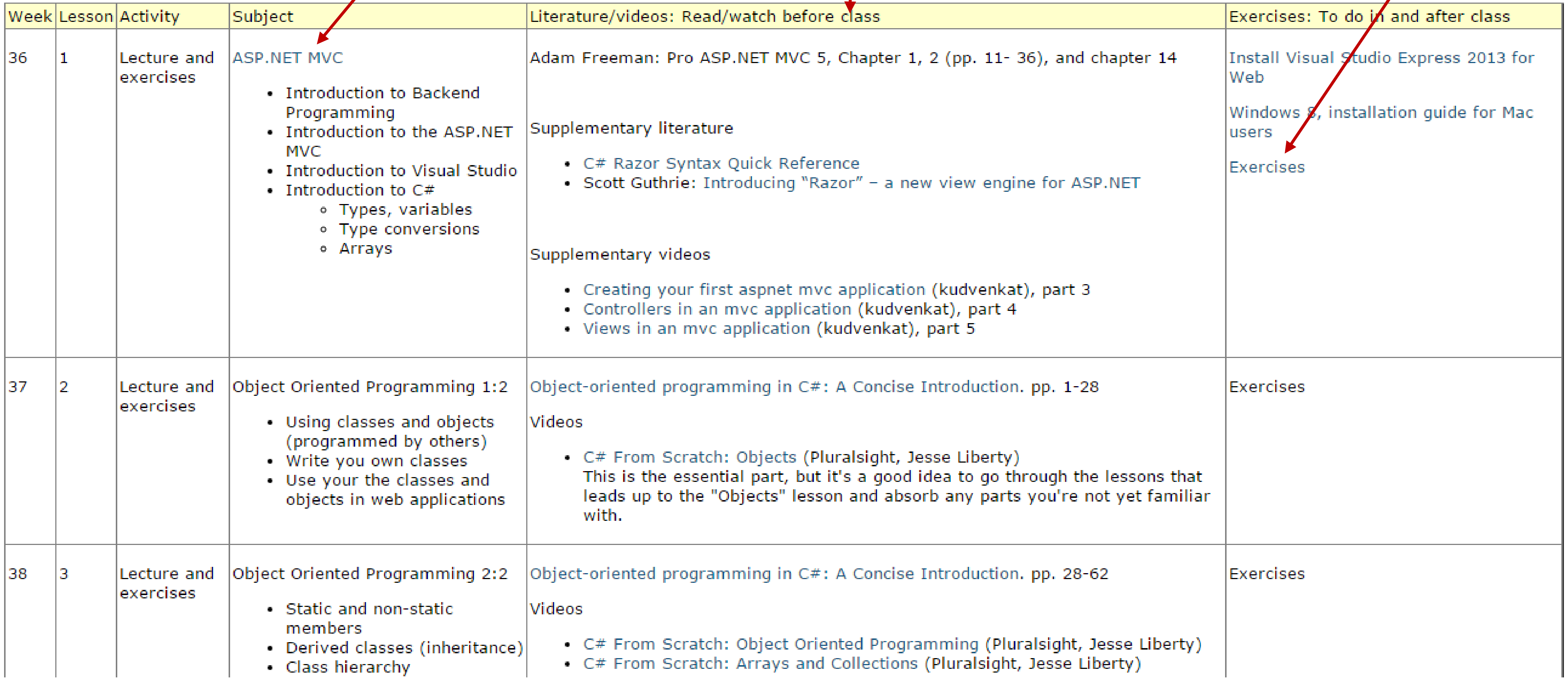
- Introduction to Backend Programming.
  - Lesson plan on Fronter …
- The ASP.NET platform
  - The ASP.NET MVC framework
  - Visual Studio
- Exercises
- Introduction to C#
  - Types, variables
  - Type conversions
  - Arrays
- Exercises

The lesson plan is on Fronter

Presentation

Literature to read, videos to watch

Doing exercises



Week	Lesson	Activity	Subject	Literature/videos: Read/watch before class	Exercises: To do in and after class
36	1	Lecture and exercises	ASP.NET MVC • Introduction to Backend Programming • Introduction to the ASP.NET MVC • Introduction to Visual Studio • Introduction to C# ◦ Types, variables ◦ Type conversions ◦ Arrays	Adam Freeman: Pro ASP.NET MVC 5, Chapter 1, 2 (pp. 11- 36), and chapter 14 Supplementary literature • C# Razor Syntax Quick Reference • Scott Guthrie: Introducing "Razor" – a new view engine for ASP.NET Supplementary videos • Creating your first aspnet mvc application (kudvenkat), part 3 • Controllers in an mvc application (kudvenkat), part 4 • Views in an mvc application (kudvenkat), part 5	Install Visual Studio Express 2013 for Web Windows 8, installation guide for Mac users Exercises
37	2	Lecture and exercises	Object Oriented Programming 1:2 • Using classes and objects (programmed by others) • Write you own classes • Use your the classes and objects in web applications	Object-oriented programming in C#: A Concise Introduction. pp. 1-28 Videos • C# From Scratch: Objects (Pluralsight, Jesse Liberty) This is the essential part, but it's a good idea to go through the lessons that leads up to the "Objects" lesson and absorb any parts you're not yet familiar with.	Exercises
38	3	Lecture and exercises	Object Oriented Programming 2:2 • Static and non-static members • Derived classes (inheritance) • Class hierarchy	Object-oriented programming in C#: A Concise Introduction. pp. 28-62 Videos • C# From Scratch: Object Oriented Programming (Pluralsight, Jesse Liberty) • C# From Scratch: Arrays and Collections (Pluralsight, Jesse Liberty)	Exercises

Materials on Fronter

- Lesson plan with
  - Subjects and literature for each lesson
  - Presentations
  - Exercises
- Code examples for each lesson
- Recommended solutions for the exercises

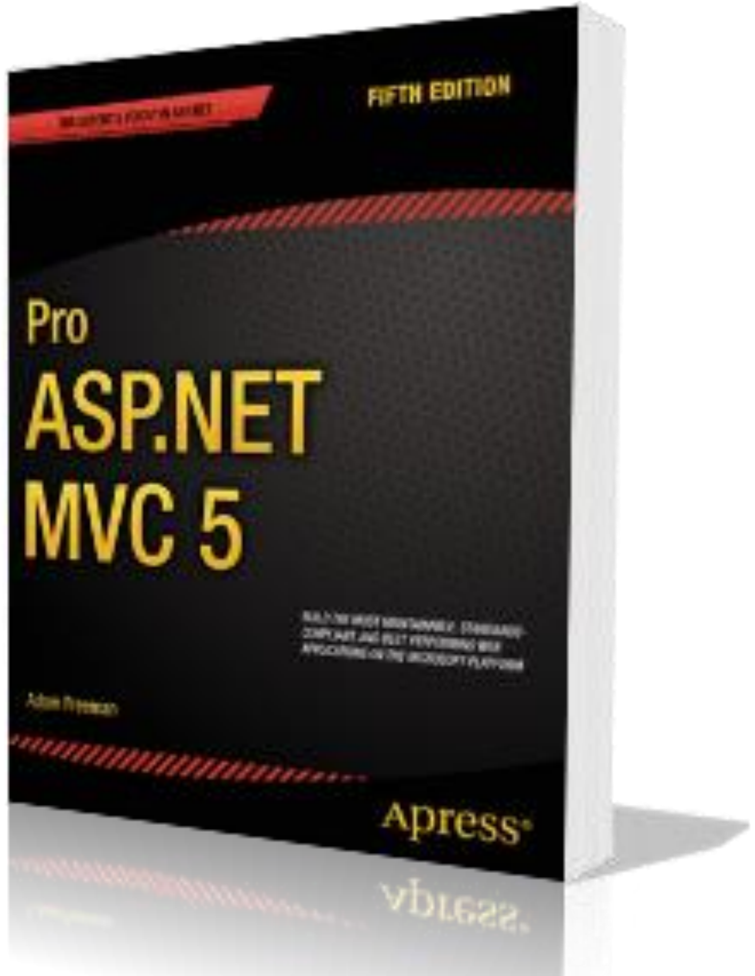
- Mandatory assignments 1-2

Literature

Primary
Adam Freeman:
Pro ASP.NET MVC 5,
Apress 2014

Plus
Selected articles

Plus
Videos

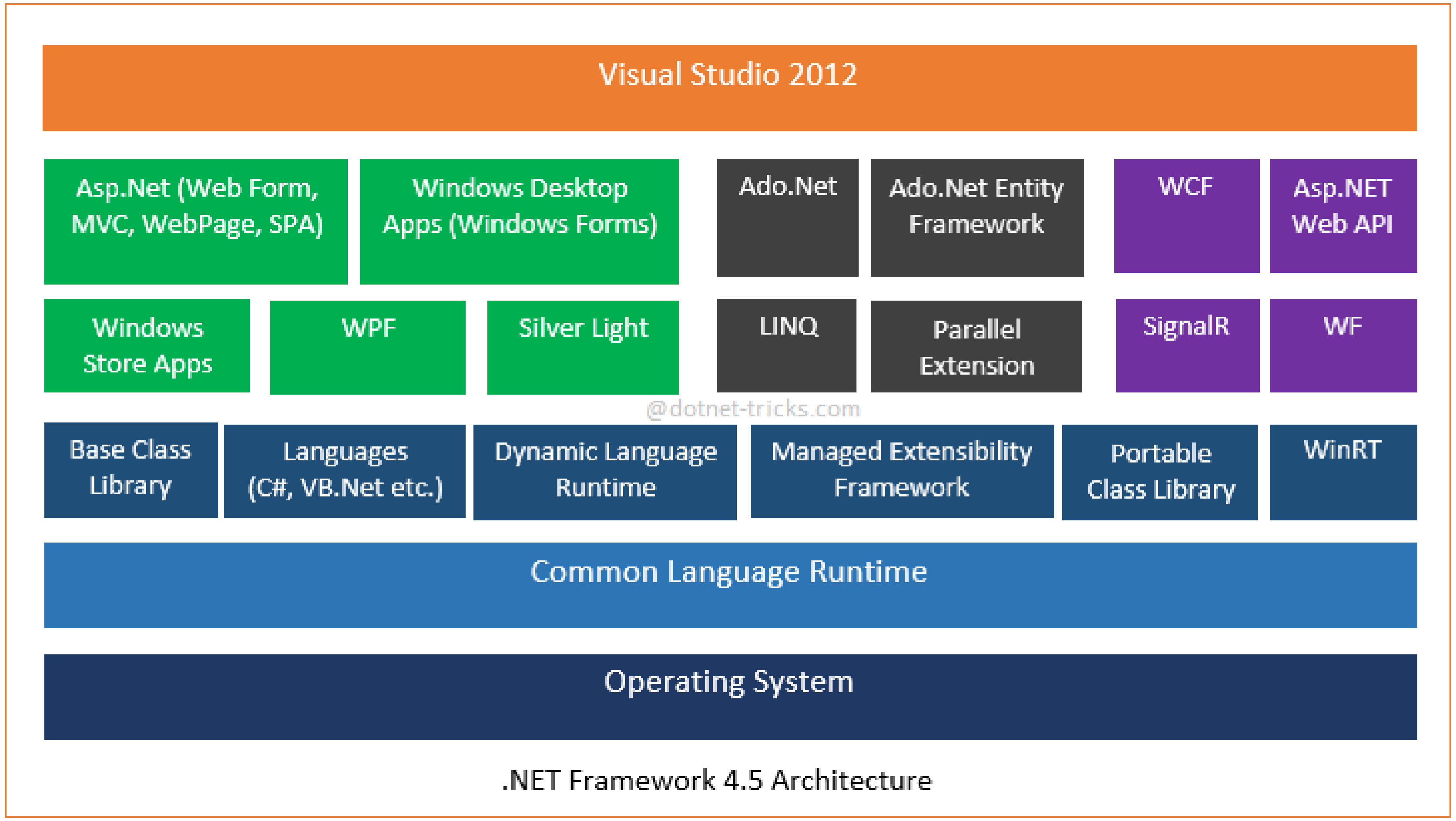


The ASP.NET Platform

ASP.NET and C#

- C# is the official language of the ASP.NET platform; but you can use other languages as well, e.g. Visual Basic and J#
- You can mix different languages in the same application
- All languages in .NET are object oriented
  - .NET gives you access to a comprehensive framework of predefined classes (organized in namespaces)

.NET class library



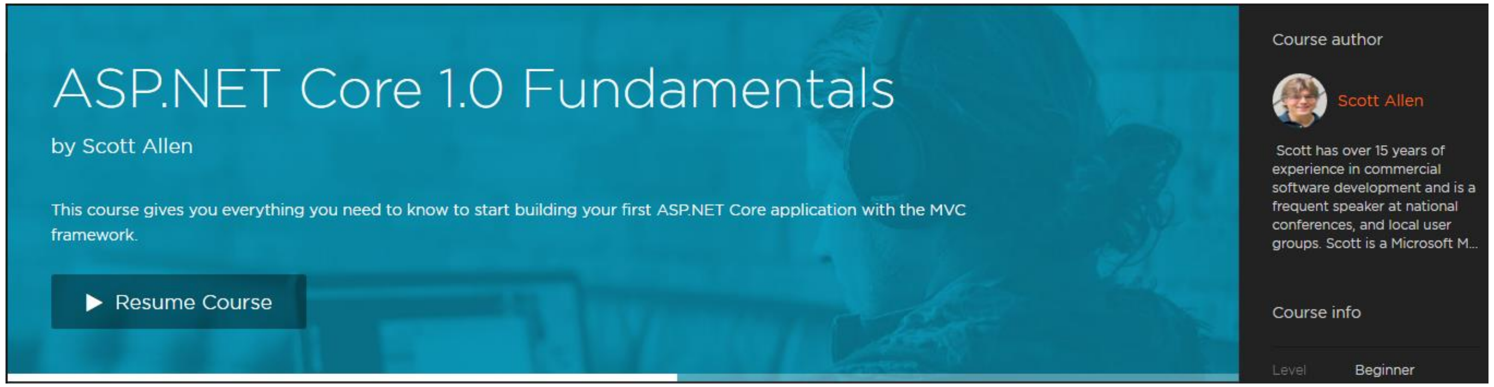
For a full list see, [.NETFramework Class Library]() and [ASP.NET MVC 4 .5.1 Reference]()

ASP.NET Core (still in pre-release)

- Install – Microsoft.AspNet.Mvc.Core 6.0.0-rc1-final

```
PM> Install-Package Microsoft.AspNet.Mvc.Core -Pre
```

- [Introduction to ASP.NET Core](app.pluralsight.com/library/courses/aspdotnet-core-1-0-fundamentals/table-of-contents) (Microsoft ASP.NET)



ASP.NET Core 1.0 Fundamentals

by Scott Allen

This course gives you everything you need to know to start building your first ASP.NET Core application with the MVC framework.

Course author

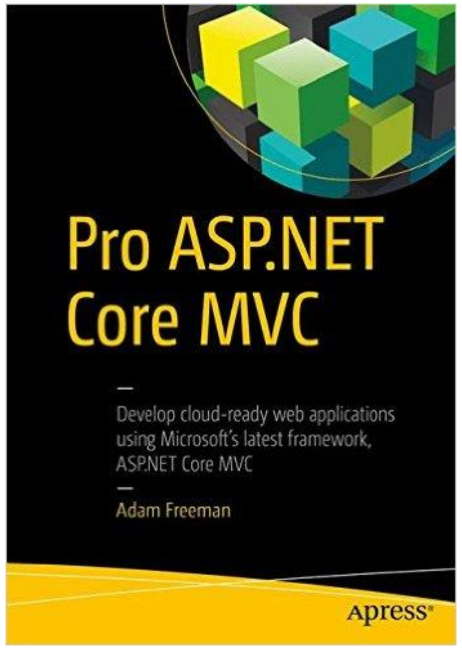
Scott Allen

Scott has over 15 years of experience in commercial software development and is a frequent speaker at national conferences, and local user groups. Scott is a Microsoft M...

Course info

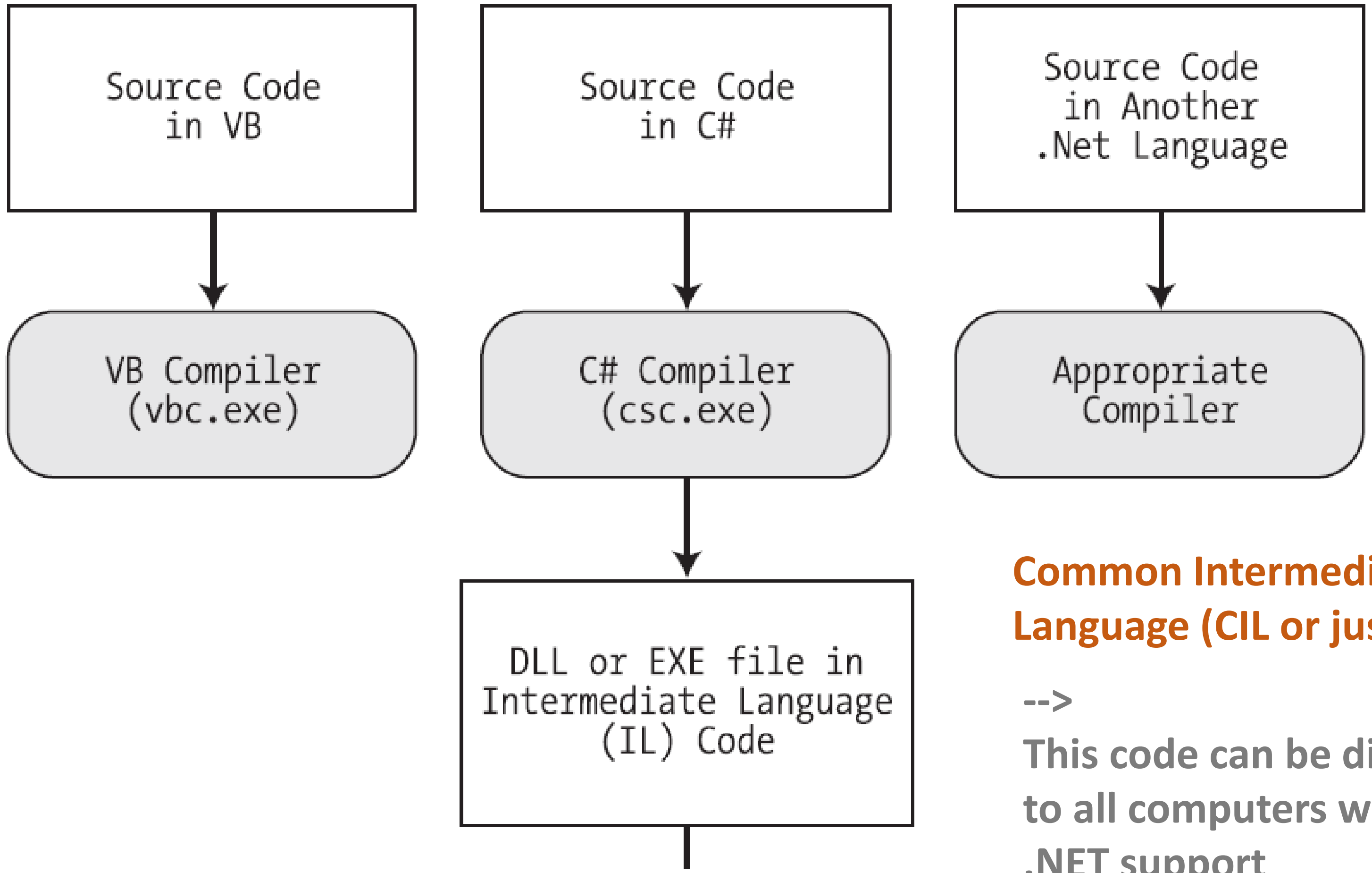
Level Beginner

app.pluralsight.com/library/courses/aspdotnet-core-1-0-fundamentals/table-of-contents


Pro ASP.NET Core MVC
Develop cloud-ready web applications using Microsoft's latest framework, ASP.NET Core MVC
Adam Freeman
Apress

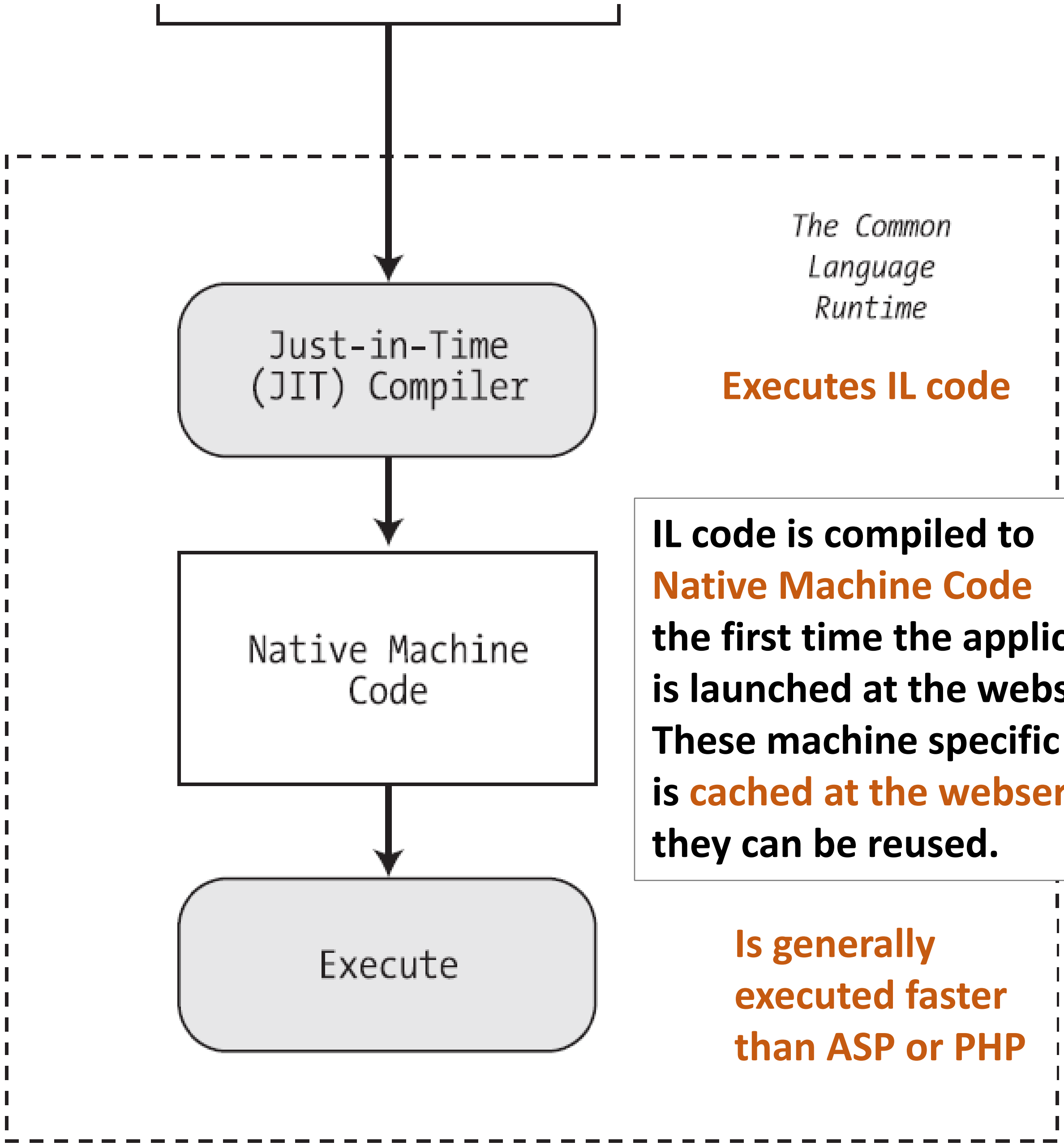
October 18, 2016

C#, VB and .NET languages



Common Intermediate Language (CIL or just IL)

-->
This code can be distributed to all computers with .NET support



Executes IL code

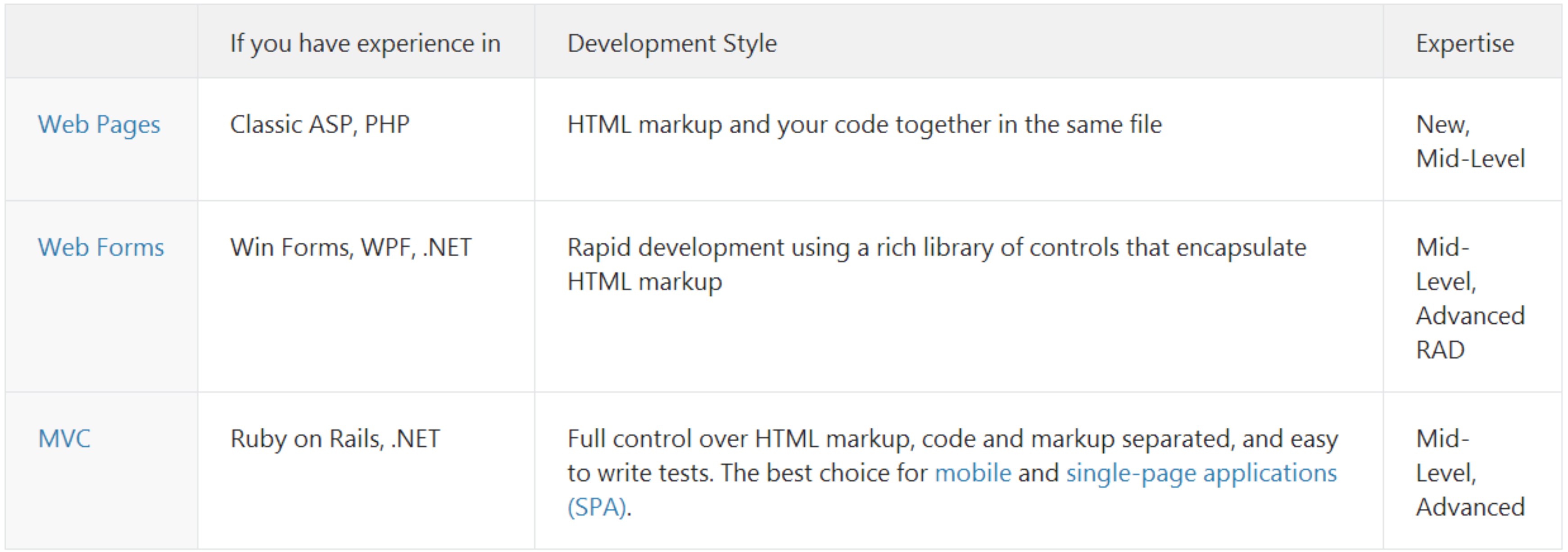
IL code is compiled to Native Machine Code the first time the application is launched at the webserver. These machine specific files is cached at the webserver so they can be reused.

Is generally executed faster than ASP or PHP

Visual Studio IDE

- IDE stands for ”Integrated Development Environment” and is a development tool, with a lot of nice facilities for software developers
- An IDE contains:
  - An editor for programming code with color highlighting, code completion (”IntelliSense”), line numbering, expansion and collapsing of code etc.
  - Tools for code auto generation
  - A compiler
  - A debugger
- We will use Visual Studio Community
- Advanced Professional and Enterprise editions exists – and can be downloaded from Dreamspark (www.dreamspark.com)

ASP.NET offers three frameworks for creating web applications



	If you have experience in	Development Style	Expertise
Web Pages	Classic ASP, PHP	HTML markup and your code together in the same file	New, Mid-Level
Web Forms	Win Forms, WPF, .NET	Rapid development using a rich library of controls that encapsulate HTML markup	Mid-Level, Advanced RAD
MVC	Ruby on Rails, .NET	Full control over HTML markup, code and markup separated, and easy to write tests. The best choice for mobile and single-page applications (SPA).	Mid-Level, Advanced

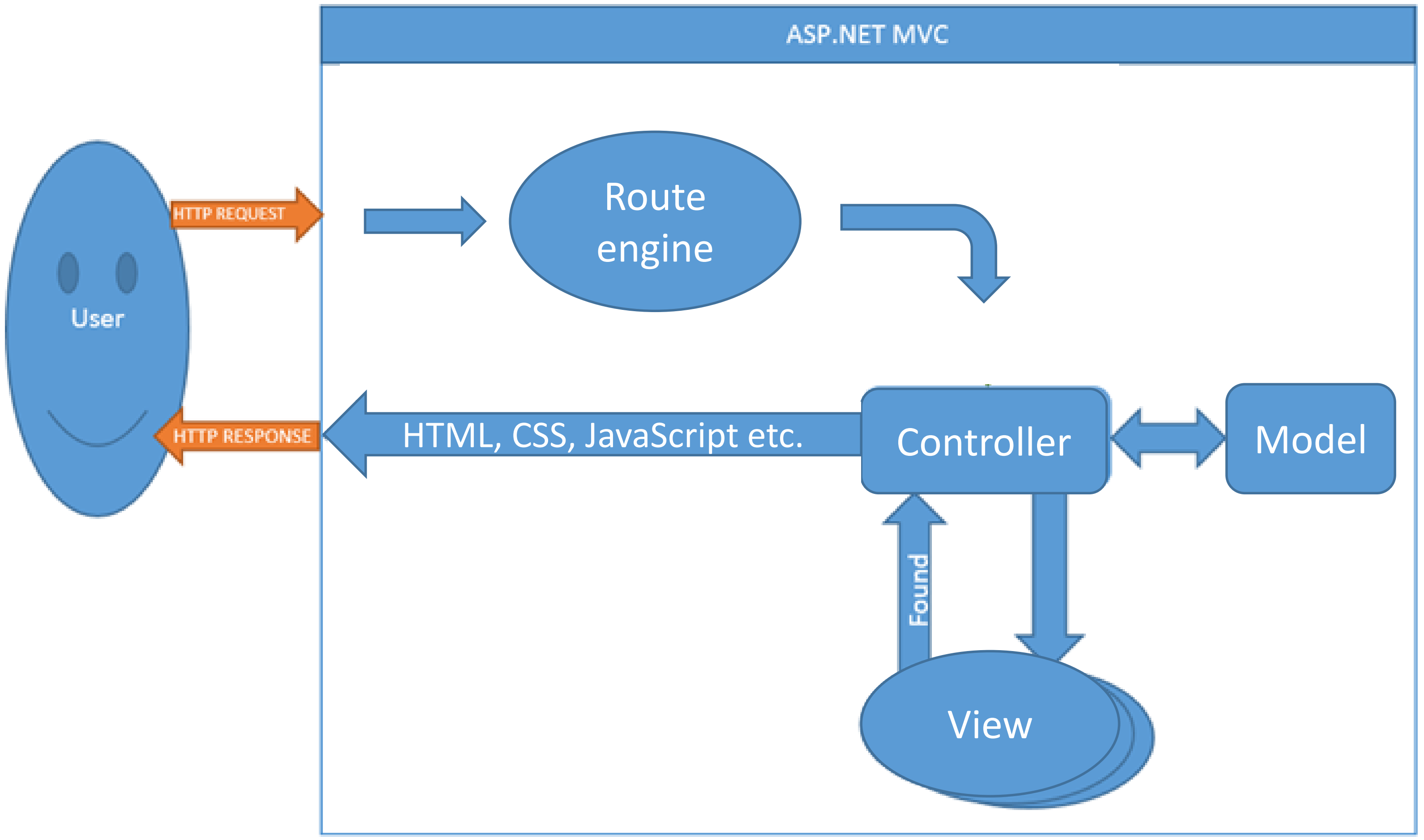
http://www.asp.net/get-started/websites

Why ASP.NET MVC?

- C# is a great language. Easy to grasp and flexible
- Full control over markup
- Enables a clean separation of concerns (Model, View, Controller)
- MVC architecture helps you to structure code and write in accordance with the DRY principle
- Includes features for fast, TDD-friendly development (Test Driven Development)
- Great IDE: Visual Studio offers a great programming environment
- Fast: Compiled language
- Job opportunities

ASP.NET MVC

MVC architectural pattern

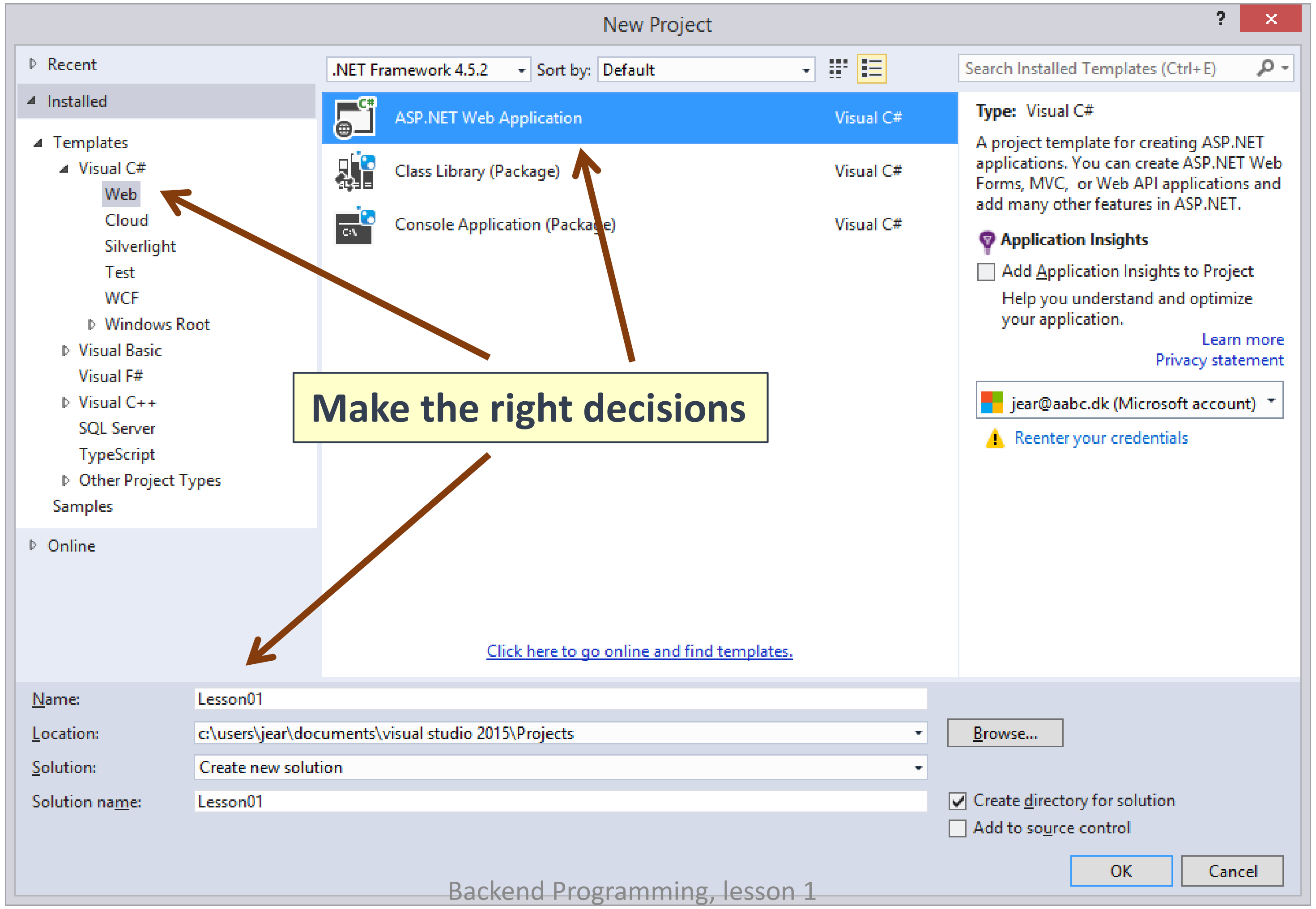


MVC applied to Web Frameworks

- **The Model**: Data Access Layer. A set of classes that describes the data you're working with as well as the business rules for how the data can be changed and manipulated
- **The View**: A template to generate HTML dynamically. Defines how the application's UI will be displayed (HTML, CSS, JavaScript, server side code: C#)
- **The Controller**: Manages the relationship between the View and the Model. It responds to user input, talks to the Model, and decides which view to render (if any).

Let's do some demos

My first ASP.NET MVC Project
File -> New Project (Ctrl+Shift+N)



New Project

- Recent
- Installed
  - Templates
    - Visual C#
      - Web
      - Cloud
      - Silverlight
      - Test
      - WCF
      - Windows Root
    - Visual Basic
    - Visual F#
    - Visual C++
    - SQL Server
    - TypeScript
    - Other Project Types
  - Samples
- Online

.NET Framework 4.5.2 Sort by: Default

Search Installed Templates (Ctrl+E)

ASP.NET Web Application	Visual C#
Class Library (Package)	Visual C#
Console Application (Package)	Visual C#

Click here to go online and find templates.

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

Help you understand and optimize your application.

Learn more
Privacy statement

jear@aabc.dk (Microsoft account)

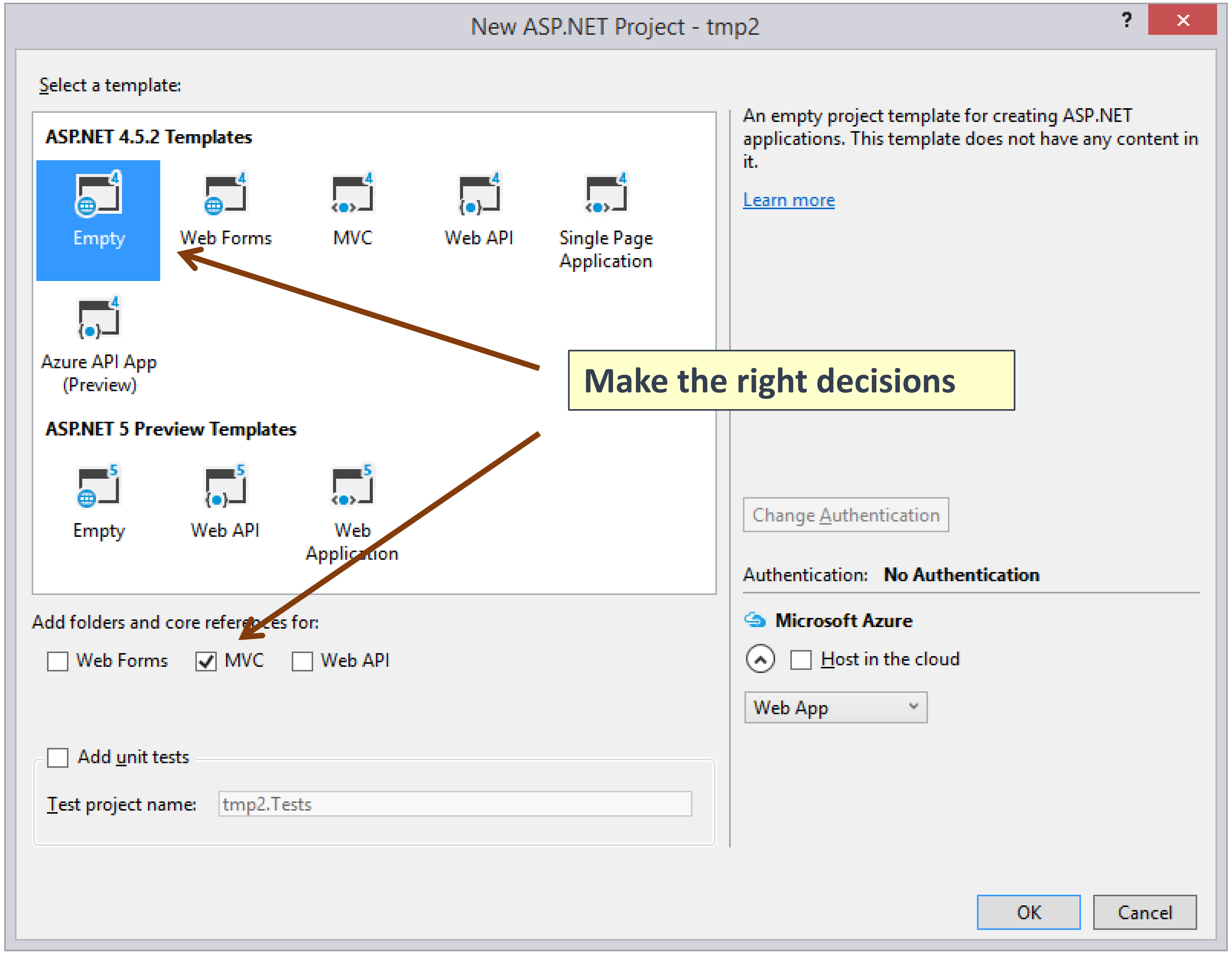
Reenter your credentials

Make the right decisions

Name:	Lesson01
Location:	c:\users\jear\documents\visual studio 2015\Projects
Solution:	Create new solution
Solution name:	Lesson01

Browse...

☑ Create directory for solution
☐ Add to source control


New ASP.NET Project - tmp2

Select a template:

ASP.NET 4.5.2 Templates

ASP.NET 5 Preview Templates

An empty project template for creating ASP.NET applications. This template does not have any content in it.

Learn more

Make the right decisions

Add folders and core references for:

☐ Web Forms ☑ MVC ☐ Web API

☐ Add unit tests

Test project name: tmp2.Tests

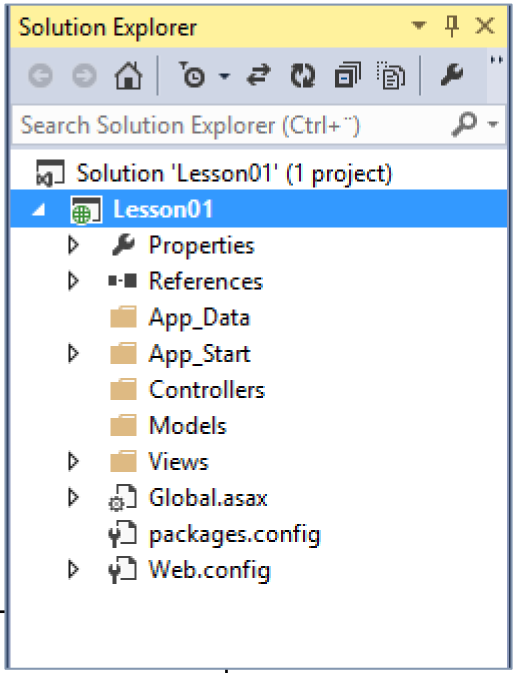
Authentication: **No Authentication**

Microsoft Azure

☐ Host in the cloud

Web App

The MVC
Application Structure



Solution Explorer

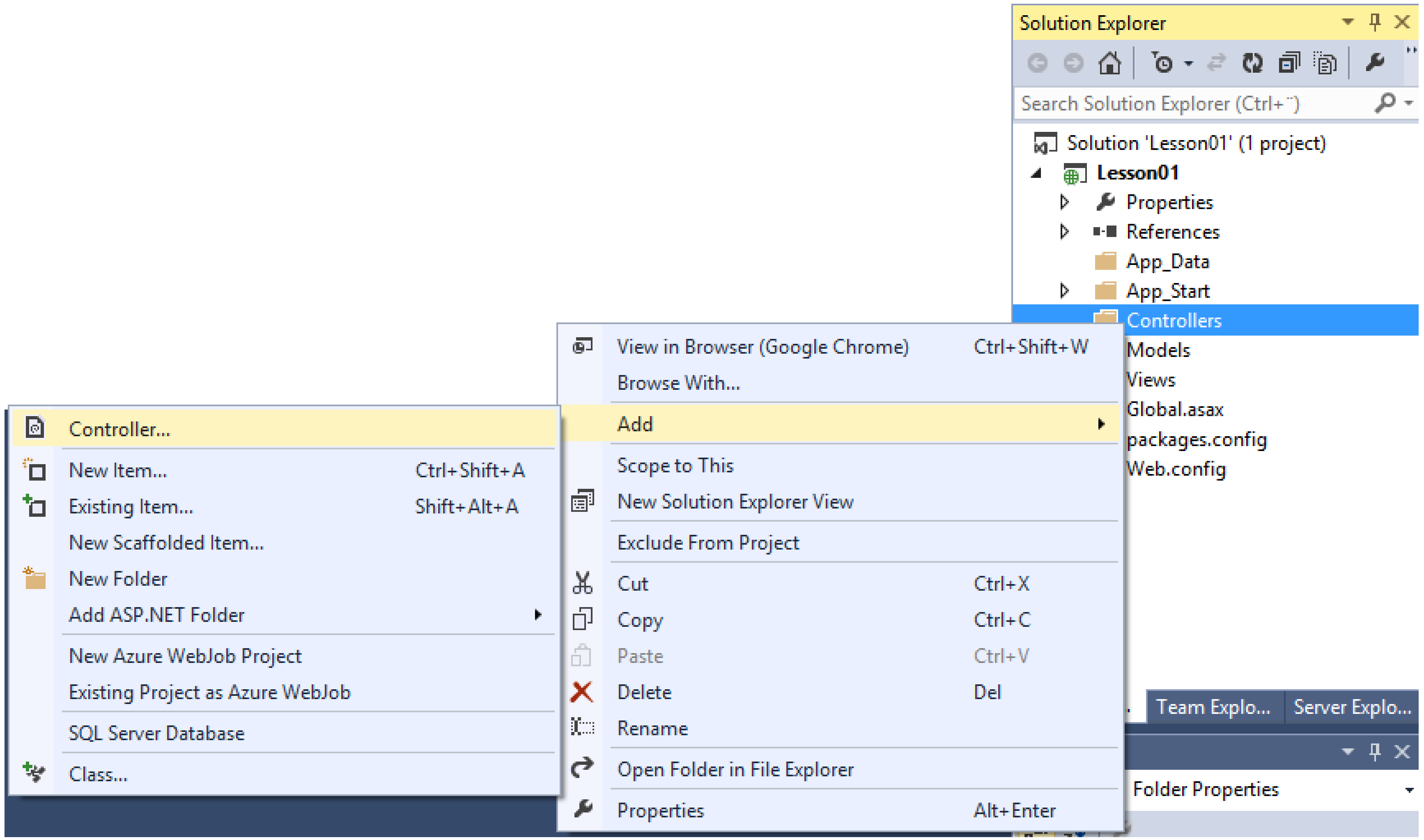
Search Solution Explorer (Ctrl+¨)

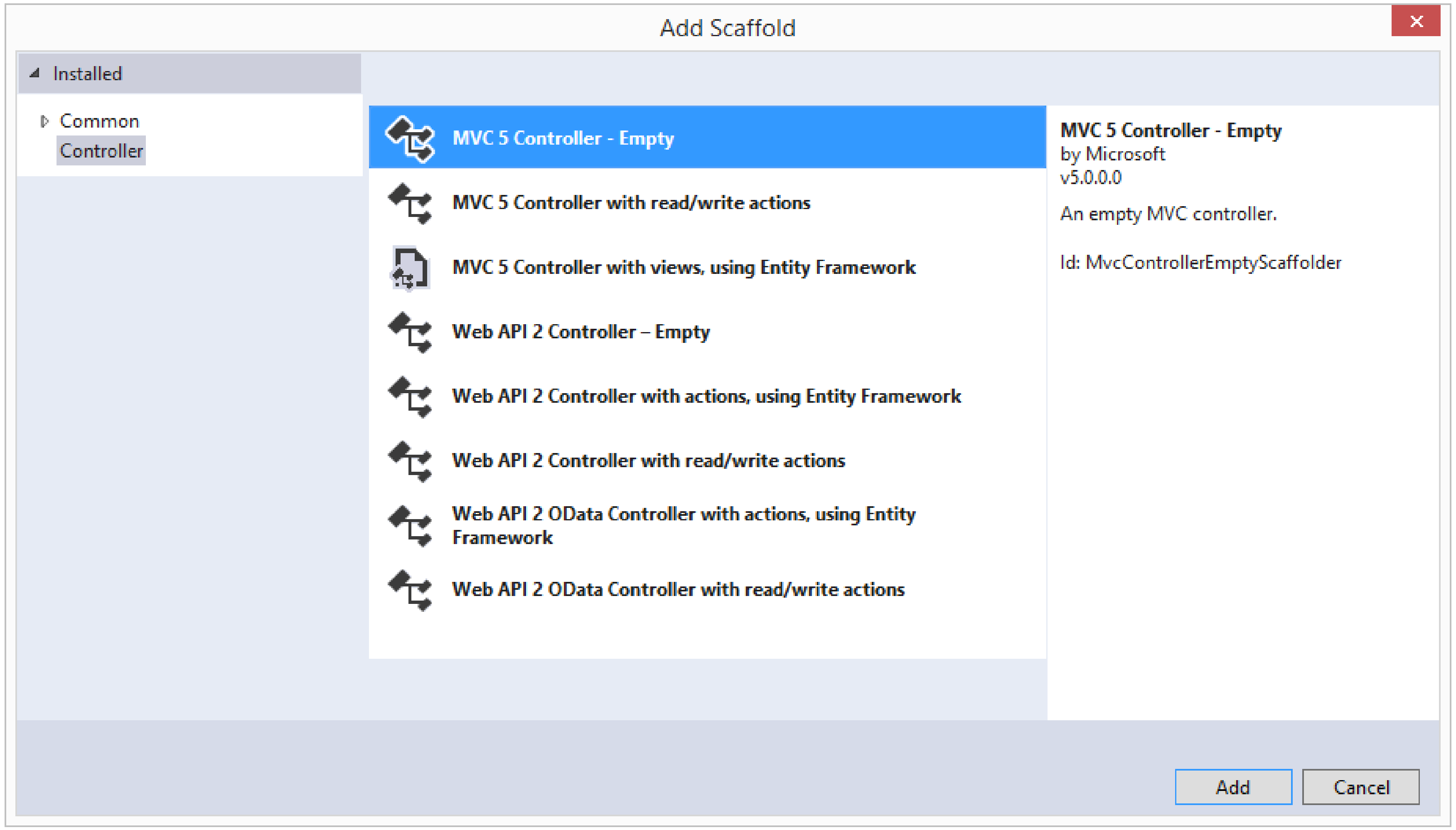
Solution 'Lesson01' (1 project)
- Lesson01
  - Properties
  - References
  - App_Data
  - App_Start
  - Controllers
  - Models
  - Views
  - Global.asax
  - packages.config
  - Web.config

Top-Level Directories

Directory	Purpose
`/Controllers`	Where you put Controller classes that handle URL requests
`/Models`	Where you put classes that represent and manipulate data and business objects
`/Views`	Where you put UI template files that are responsible for rendering output such as HTML
`/App_Data`	Where you store data files you want to read/write
`/App_Start`	Where you put configuration code for features like Routing, bundling, and Web API
`/Scripts`	Where you put Java Script library files and scripts (`.js`)
`/Content`	Where you put CSS, images, and other site content , other than scripts

Add a new Controller




Add Scaffold

- Installed
  - Common
  - Controller

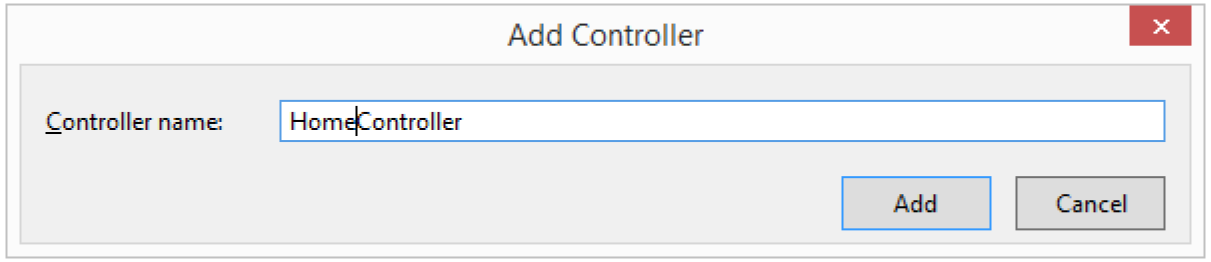
MVC 5 Controller - Empty
MVC 5 Controller with read/write actions
MVC 5 Controller with views, using Entity Framework
Web API 2 Controller – Empty
Web API 2 Controller with actions, using Entity Framework
Web API 2 Controller with read/write actions
Web API 2 OData Controller with actions, using Entity Framework
Web API 2 OData Controller with read/write actions

MVC 5 Controller - Empty
by Microsoft
v5.0.0.0

An empty MVC controller.

Id: MvcControllerEmptyScaffolder

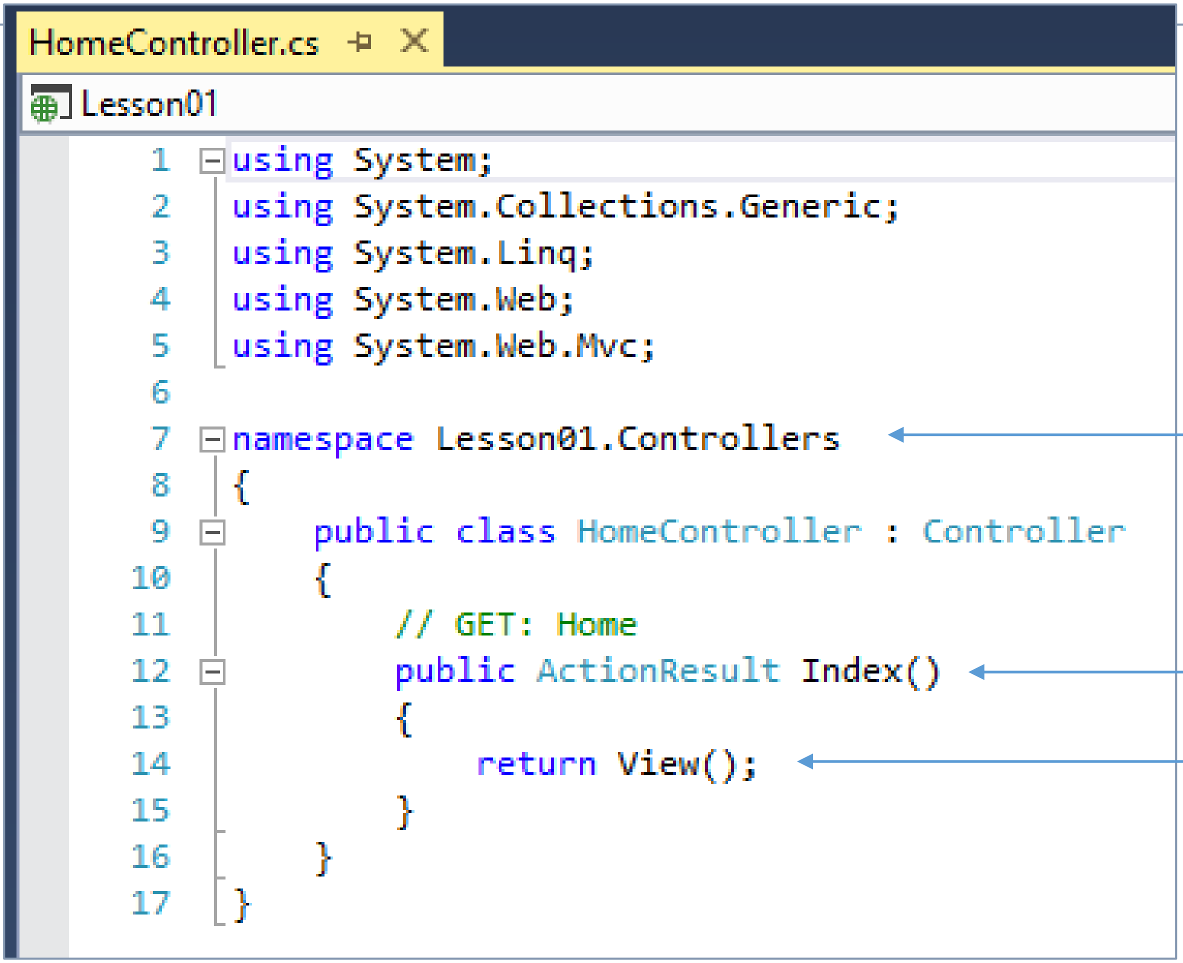
Add Cancel


Add Controller

Controller name: HomeController

Add Cancel

The new Controller



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;

namespace Lesson01.Controllers
{
    public class HomeController : Controller
    {
        // GET: Home
        public ActionResult Index()
        {
            return View();
        }
    }
}
```

Class references

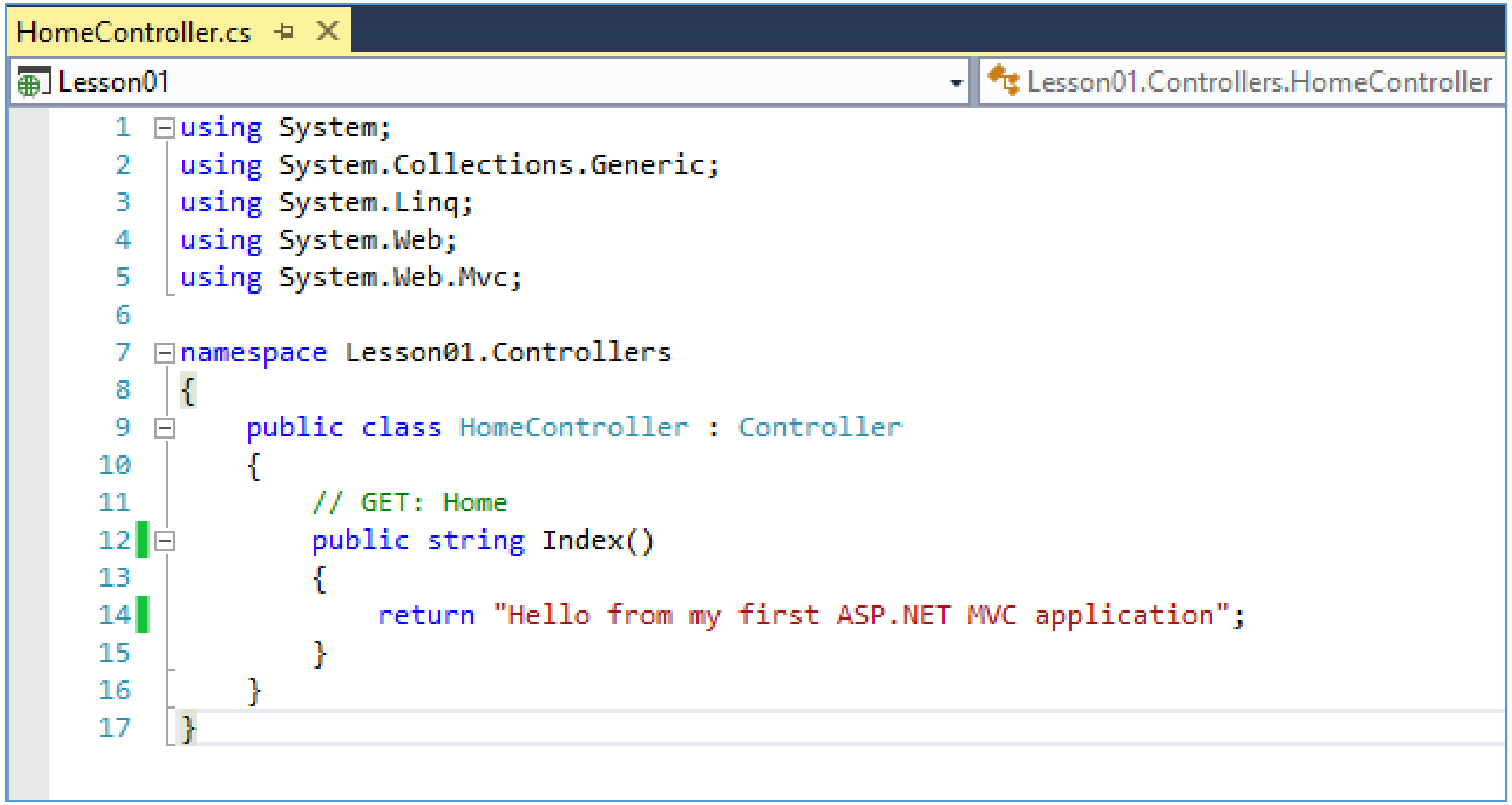
Namespace

Class name

Action method

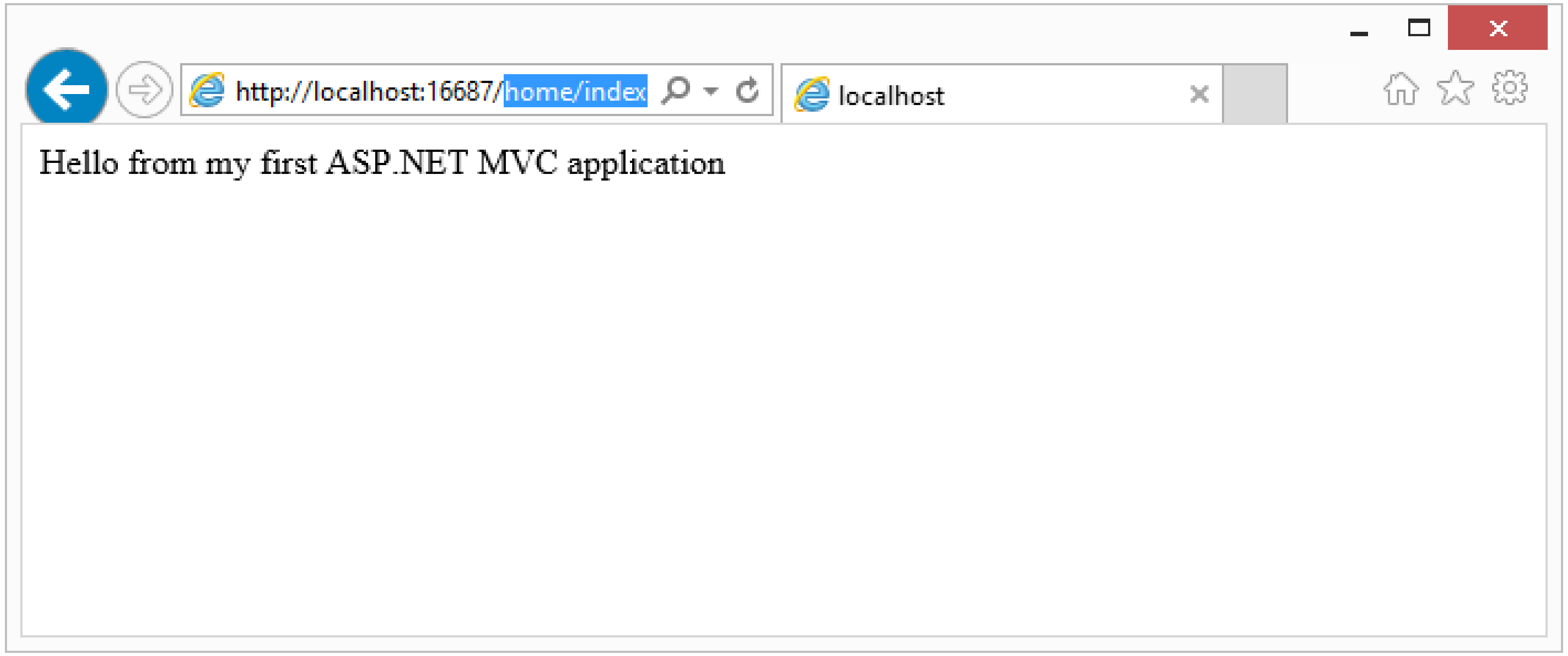
Return value

Modify controller



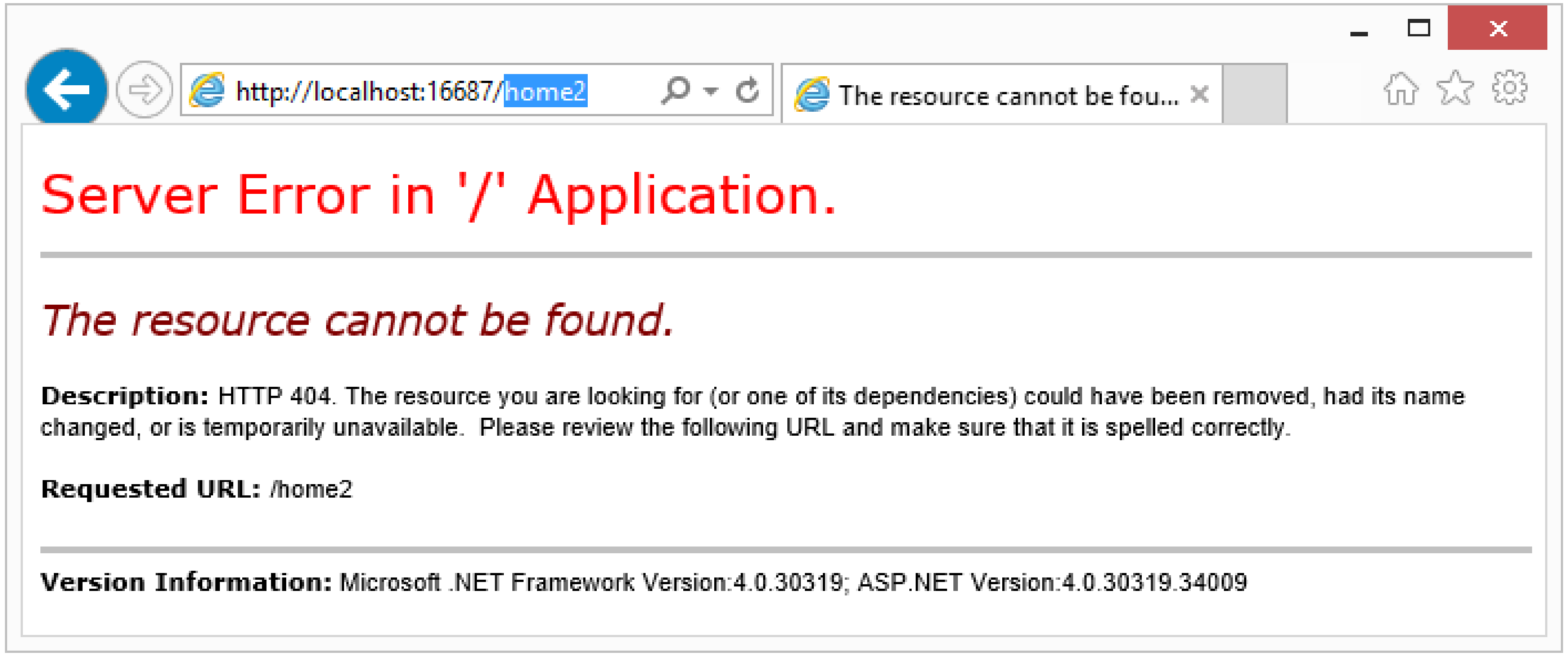
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;

namespace Lesson01.Controllers
{
    public class HomeController : Controller
    {
        // GET: Home
        public string Index()
        {
            return "Hello from my first ASP.NET MVC application";
        }
    }
}
```

Run (Ctrl+F5)



Hello from my first ASP.NET MVC application

Call non-existent Controller



Server Error in '/' Application.

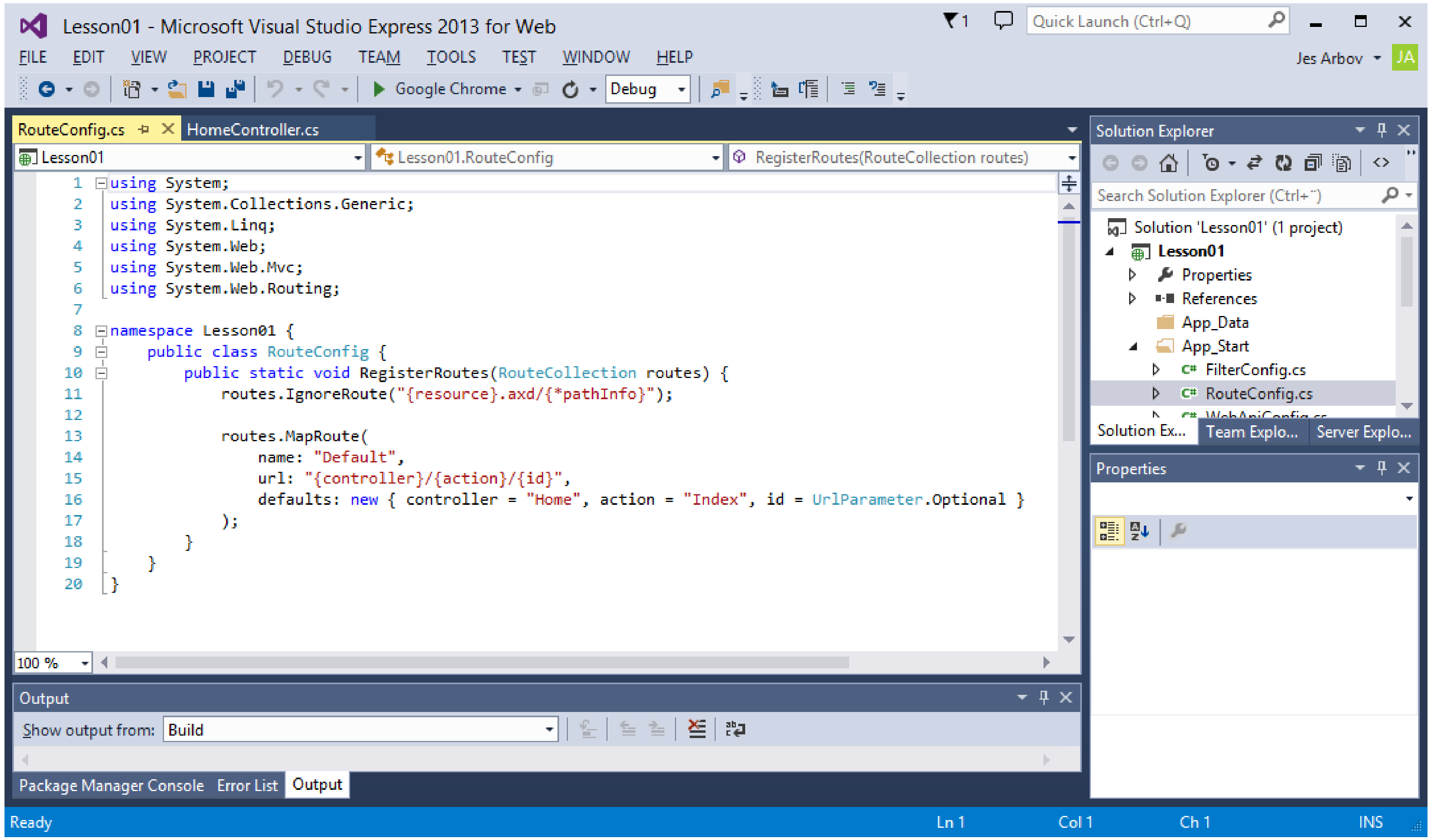
The resource cannot be found.

Description: HTTP 404. The resource you are looking for (or one of its dependencies) could have been removed, had its name changed, or is temporarily unavailable. Please review the following URL and make sure that it is spelled correctly.

Requested URL: /home2

Version Information: Microsoft .NET Framework Version:4.0.30319; ASP.NET Version:4.0.30319.34009

Routes Setting (**`RouteConfig.cs`**)

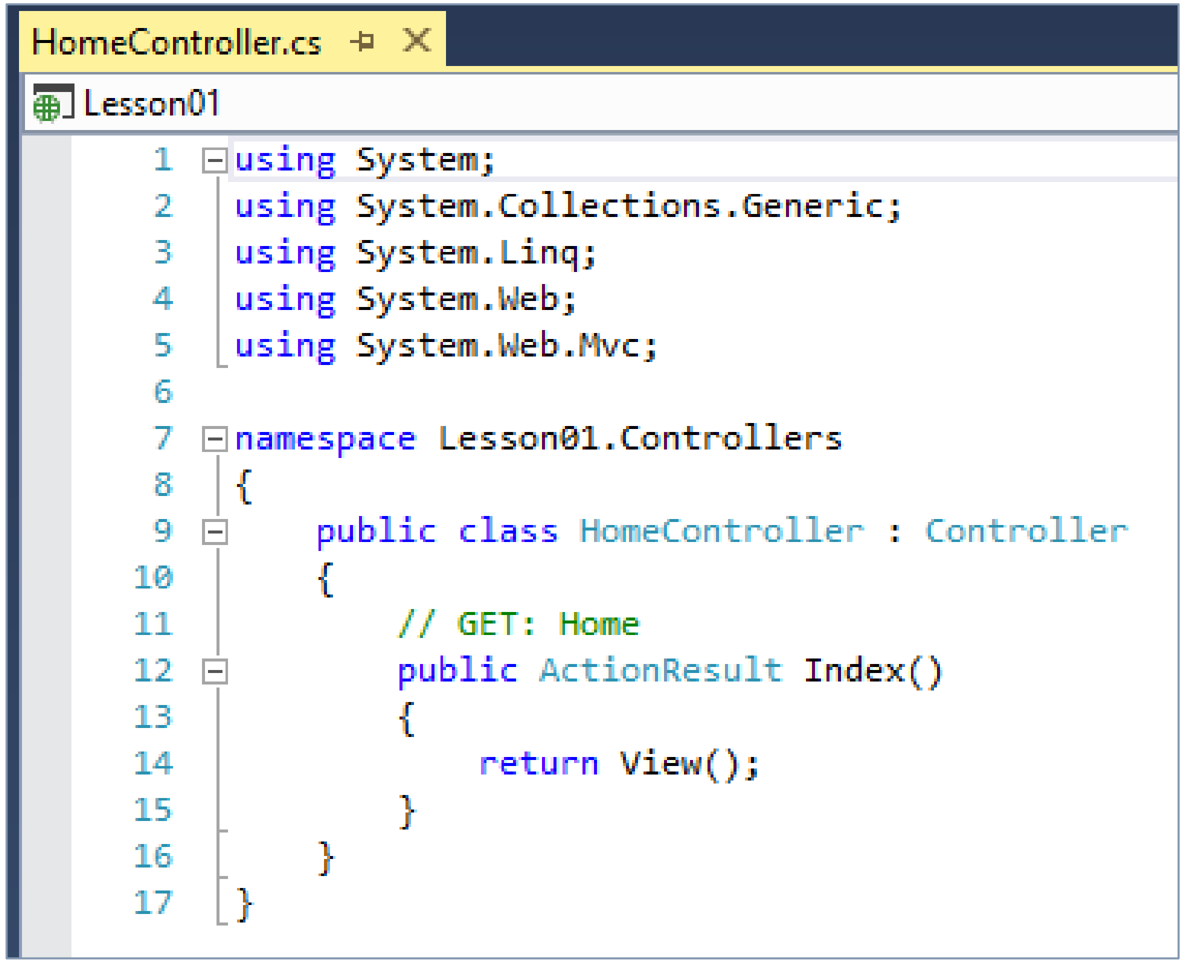


```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using System.Web.Routing;

namespace Lesson01 {
    public class RouteConfig {
        public static void RegisterRoutes(RouteCollection routes) {
            routes.IgnoreRoute("{resource}.axd/{*pathInfo}");

            routes.MapRoute(
                name: "Default",
                url: "{controller}/{action}/{id}",
                defaults: new { controller = "Home", action = "Index", id = UrlParameter.Optional }
            );
        }
    }
}
```

The Controller’s role

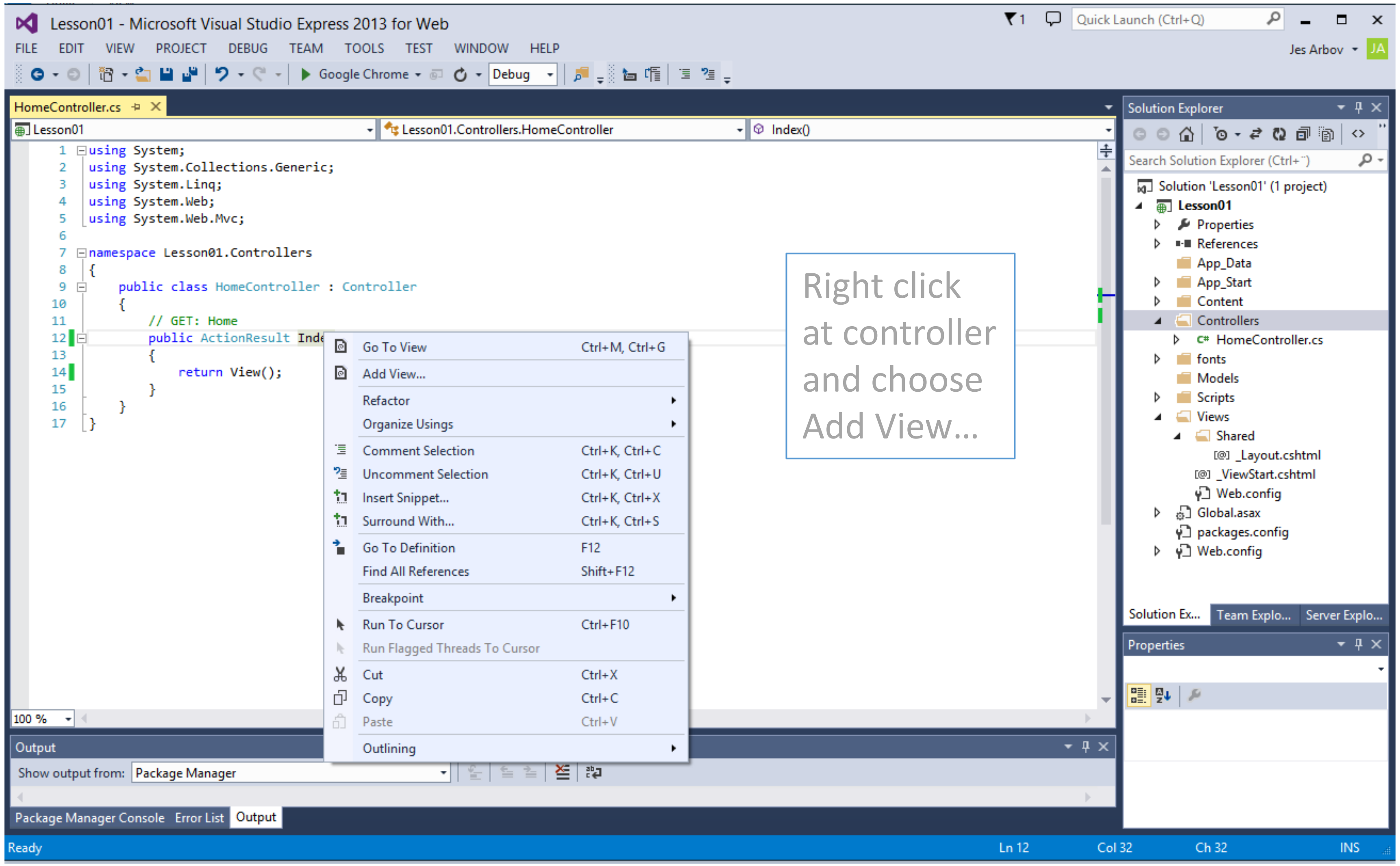
- The is **no direct relationship** between the URL and a file on the web server’s hard drive
- The relationship exists between the URL and a method on a controller class.
- A good way to think about how MVC works in a web scenario is that MVC serves up the results of **method calls**, not dynamically generated pages.

Let’s change the return type of the Action Method back ActionResult and return a View …



```csharp
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;

namespace Lesson01.Controllers
{
    public class HomeController : Controller
    {
        // GET: Home
        public ActionResult Index()
        {
            return View();
        }
    }
}
```

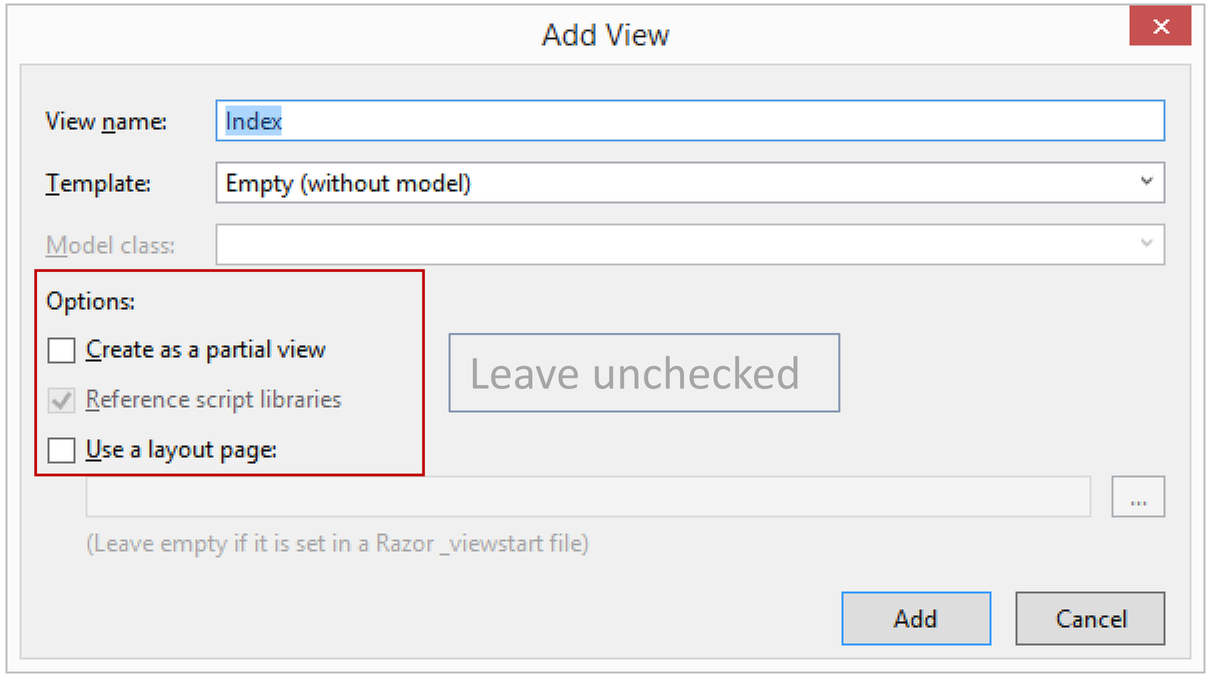
… and Add a View



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;

namespace Lesson01.Controllers
{
    public class HomeController : Controller
    {
        // GET: Home
        public ActionResult Inde
        {
            return View();
        }
    }
}
```

Lesson01 - Microsoft Visual Studio Express 2013 for Web
FILE EDIT VIEW PROJECT DEBUG TEAM TOOLS TEST WINDOW HELP
Google Chrome
Debug
Quick Launch (Ctrl+Q)
Jes Arbov
HomeController.cs
Lesson01
Lesson01.Controllers.HomeController
Index()
Go To View Ctrl+M, Ctrl+G
Add View...
Refactor
Organize Usings
Comment Selection Ctrl+K, Ctrl+C
Uncomment Selection Ctrl+K, Ctrl+U
Insert Snippet... Ctrl+K, Ctrl+X
Surround With... Ctrl+K, Ctrl+S
Go To Definition F12
Find All References Shift+F12
Breakpoint
Run To Cursor Ctrl+F10
Run Flagged Threads To Cursor
Cut Ctrl+X
Copy Ctrl+C
Paste Ctrl+V
Outlining
Right click at controller and choose Add View…
Solution Explorer
Search Solution Explorer (Ctrl+´)
Solution 'Lesson01' (1 project)
Lesson01
Properties
References
App_Data
App_Start
Content
Controllers
HomeController.cs
fonts
Models
Scripts
Views
Shared
_Layout.cshtml
_ViewStart.cshtml
Web.config
Global.asax
packages.config
Web.config
Solution Ex... Team Explo... Server Explo...
Properties
100 %
Output
Show output from: Package Manager
Package Manager Console Error List Output
Ready Ln 12 Col 32 Ch 32 INS

… if you have upgraded to MVC 5 extra Bootstrap files will be added to the project


Add View

View name: Index

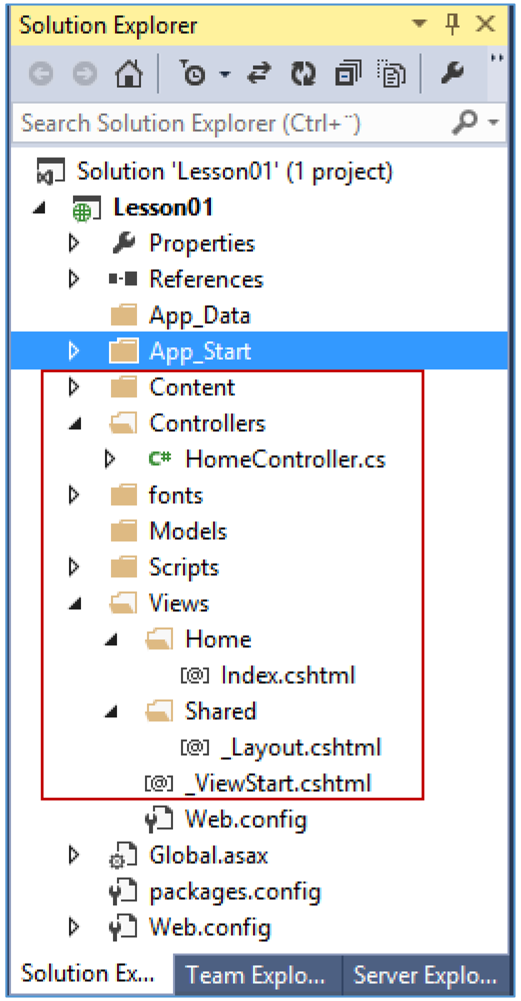
Template: Empty (without model)

Model class:

Options:

- [ ] Create as a partial view
- [✓] Reference script libraries
- [ ] Use a layout page:

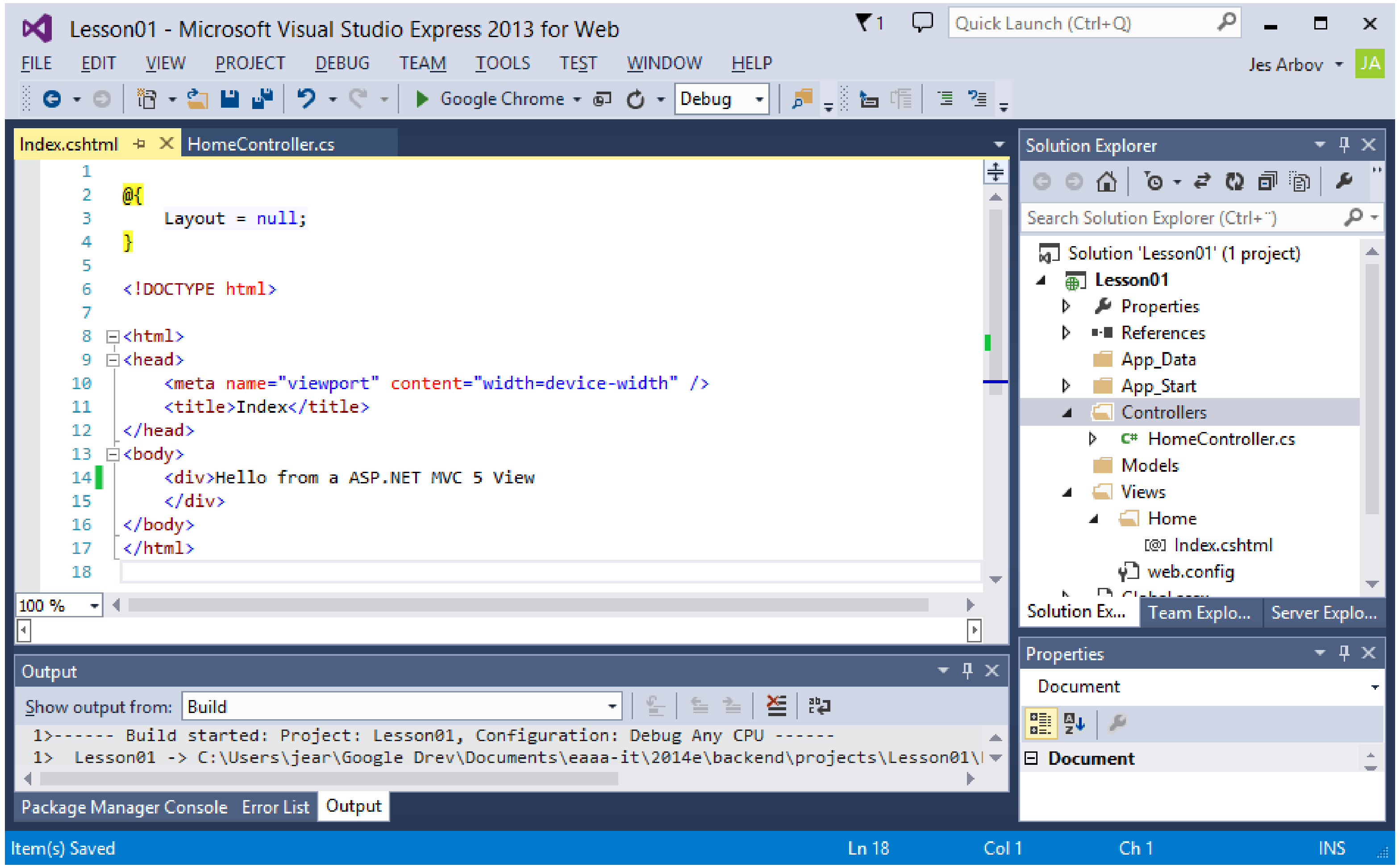
(Leave empty if it is set in a Razor _viewstart file)


Solution Explorer

Search Solution Explorer (Ctrl+¨)

```
Solution 'Lesson01' (1 project)
  Lesson01
    Properties
    References
    App_Data
    App_Start
    Content
    Controllers
      HomeController.cs
    fonts
    Models
    Scripts
    Views
      Home
        Index.cshtml
      Shared
        _Layout.cshtml
      _ViewStart.cshtml
      Web.config
    Global.asax
    packages.config
    Web.config
```

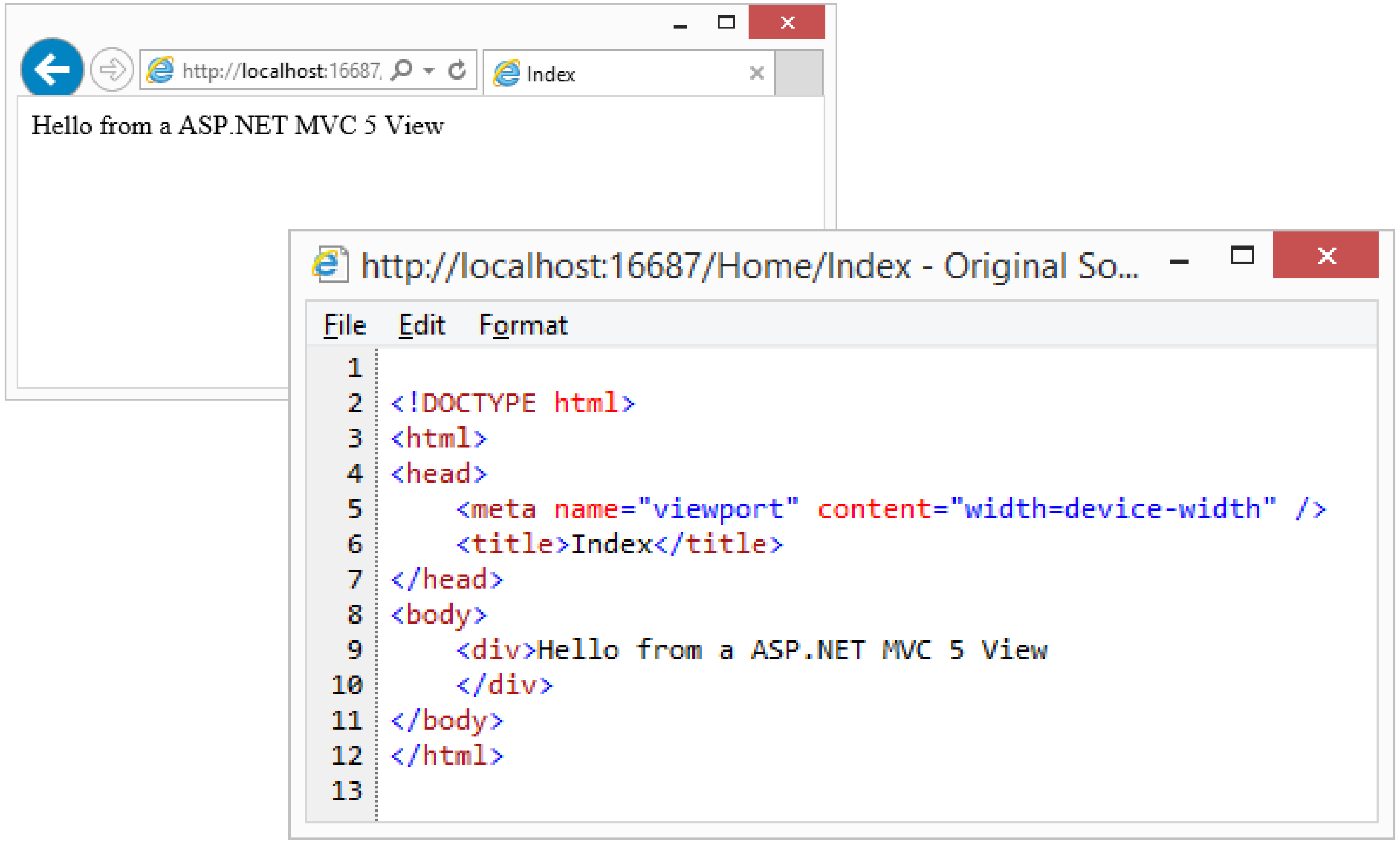
The View



```
@{
    Layout = null;
}

<!DOCTYPE html>

<html>
<head>
    <meta name="viewport" content="width=device-width" />
    <title>Index</title>
</head>
<body>
    <div>Hello from a ASP.NET MVC 5 View
    </div>
</body>
</html>

```

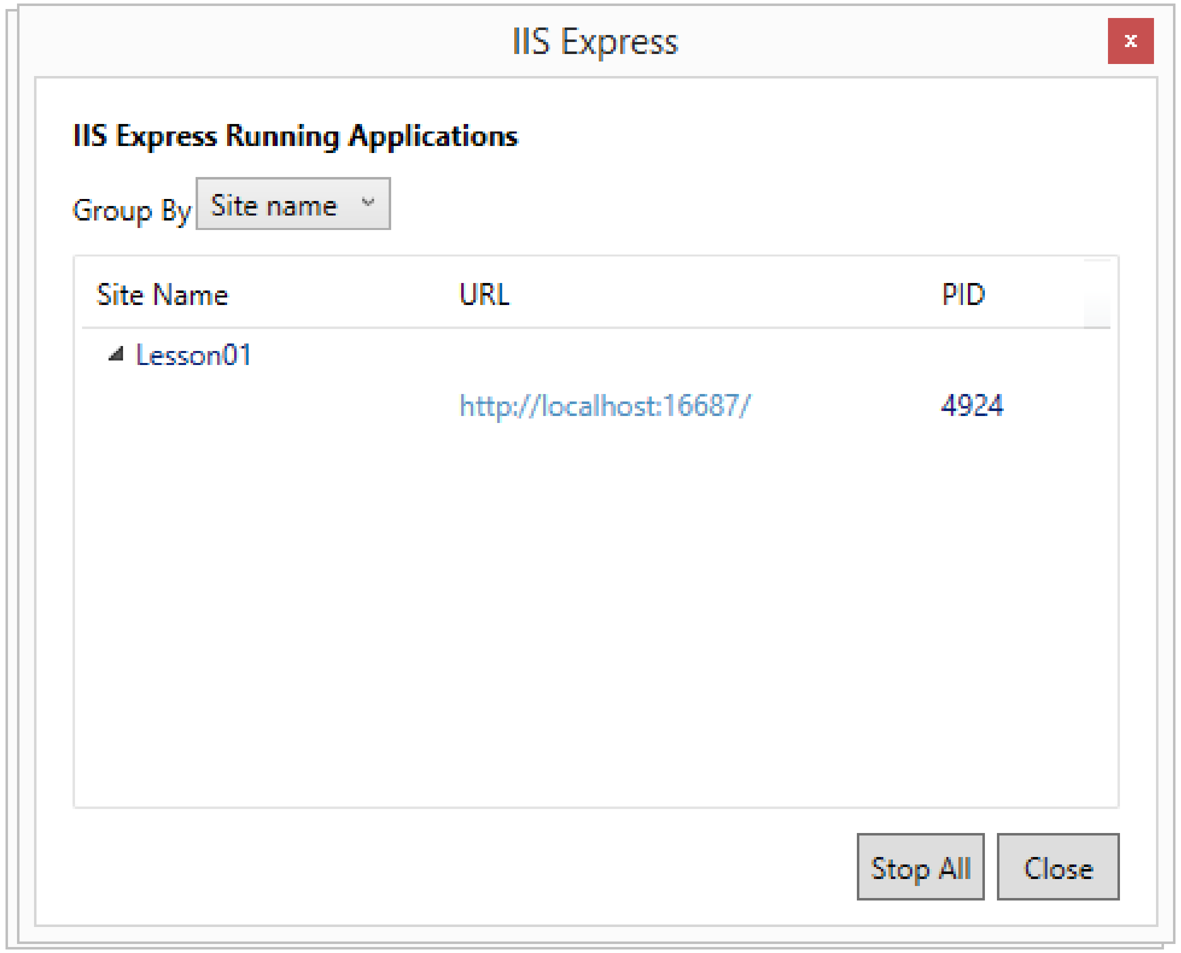
```
1>------ Build started: Project: Lesson01, Configuration: Debug Any CPU ------
1>  Lesson01 -> C:\Users\jear\Google Drev\Documents\eaaa-it\2014e\backend\projects\Lesson01\
```

Executing the web page (Ctrl+F5)


http://localhost:16687
Index
Hello from a ASP.NET MVC 5 View
http://localhost:16687/Home/Index - Original So...
File
Edit
Format

```

<!DOCTYPE html>
<html>
<head>
    <meta name="viewport" content="width=device-width" />
    <title>Index</title>
</head>
<body>
    <div>Hello from a ASP.NET MVC 5 View
    </div>
</body>
</html>

```

An instance of IIS Express runs in the background



IIS Express

IIS Express Running Applications

Group By: Site name

Site Name	URL	PID
Lesson01		
	http://localhost:16687/	4924

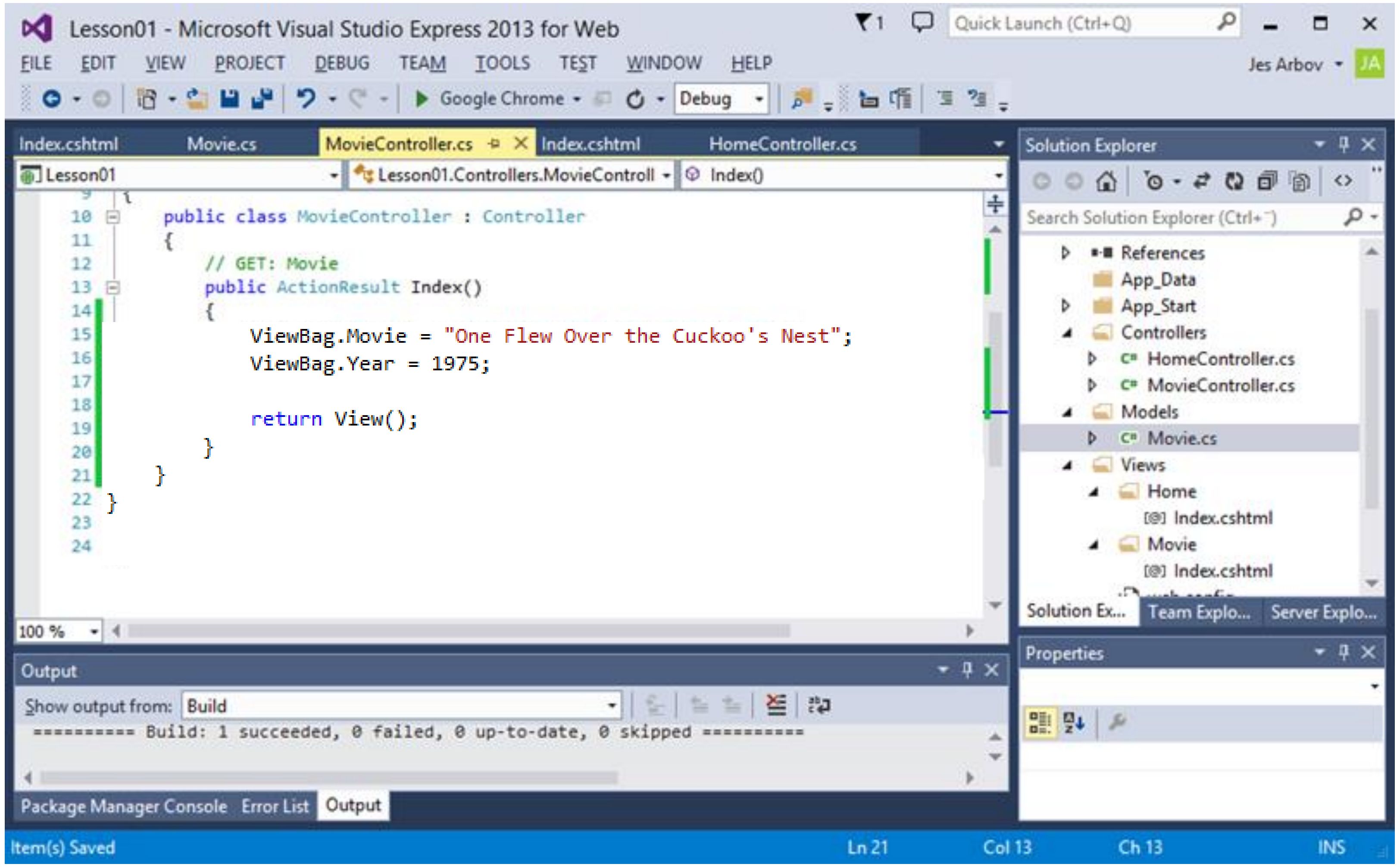
ASP.NET MVC Views

- Unlike file based frameworks, such as PHP and ASP.NET Web Forms, views are not directly accessible. You can't point your browser to a view and have it render.
- Instead, a **view** is **always rendered** by **a controller**, which provides the data the view will render.
- It is the controller that handles incoming and outgoing requests from the client/user
- If a view needs data, that data is sent from the controller

How to send data from the Controller to the View

- ViewData (key/value pairs)
  - `ViewData["movie"] = "One Flew Over the Cuckoo's Nest";`
  - `ViewData["year"] = 1975;`
- ViewBag (properties, same underlying object as ViewData)
  - `ViewBag.Movie = "Forrest Gump";`
  - `ViewBag.Year = 1994;`

Example 1: The Controller



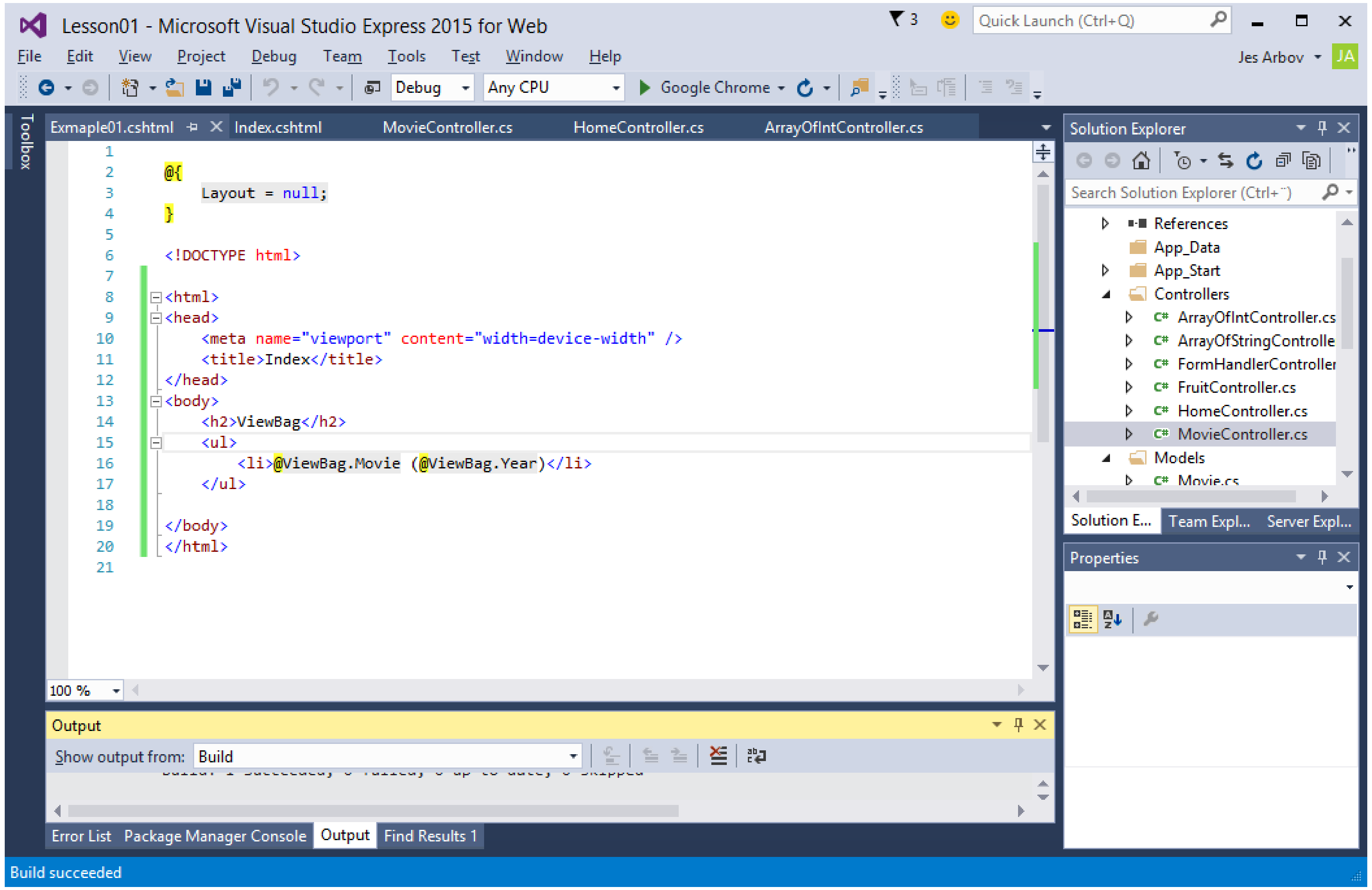
Lesson01 - Microsoft Visual Studio Express 2013 for Web
FILE EDIT VIEW PROJECT DEBUG TEAM TOOLS TEST WINDOW HELP
Google Chrome
Debug
Index.cshtml Movie.cs MovieController.cs Index.cshtml HomeController.cs
Lesson01
Lesson01.Controllers.MovieControll
Index()
Solution Explorer
References
App_Data
App_Start
Controllers
HomeController.cs
MovieController.cs
Models
Movie.cs
Views
Home
Index.cshtml
Movie
Index.cshtml
Properties
Output
Package Manager Console Error List Output
Item(s) Saved Ln 21 Col 13 Ch 13 INS

```
public class MovieController : Controller
{
    // GET: Movie
    public ActionResult Index()
    {
        ViewBag.Movie = "One Flew Over the Cuckoo's Nest";
        ViewBag.Year = 1975;

        return View();
    }
}
}
```

Show output from: Build

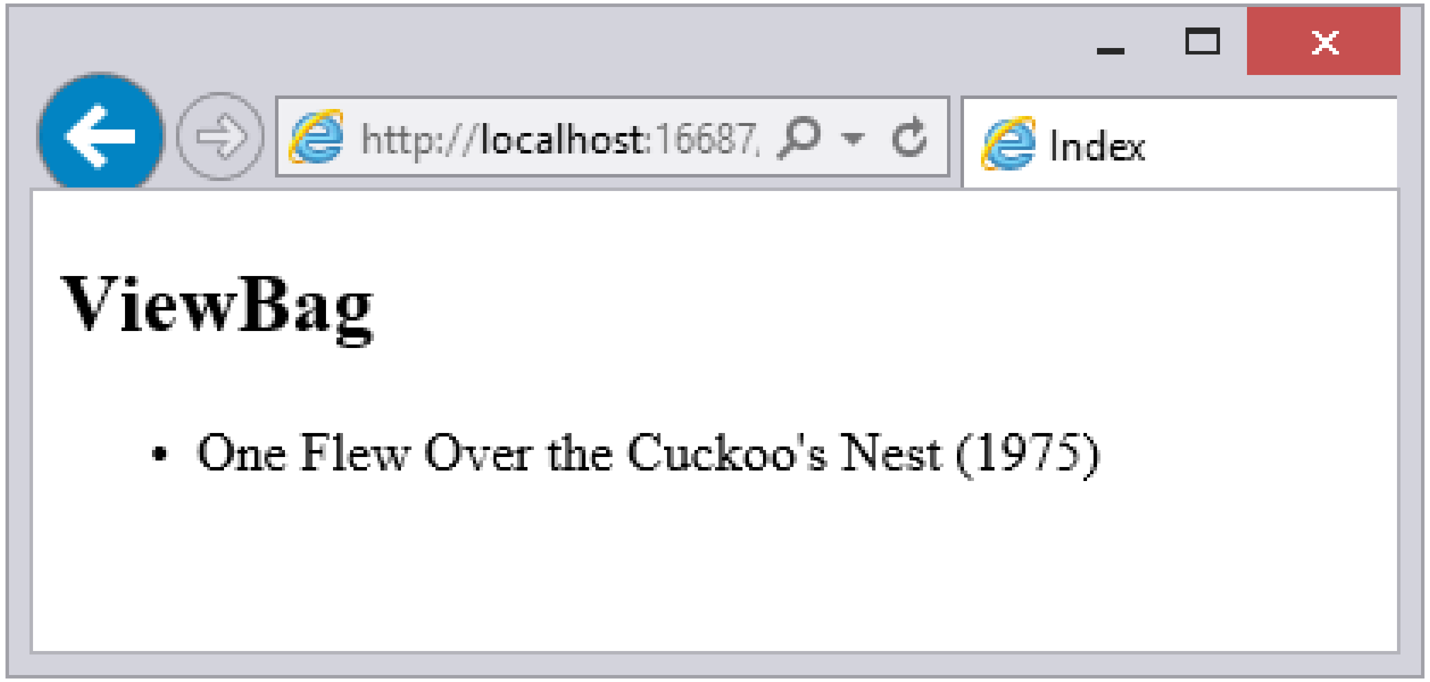
```
========== Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped ==========
```

Example 1: The View



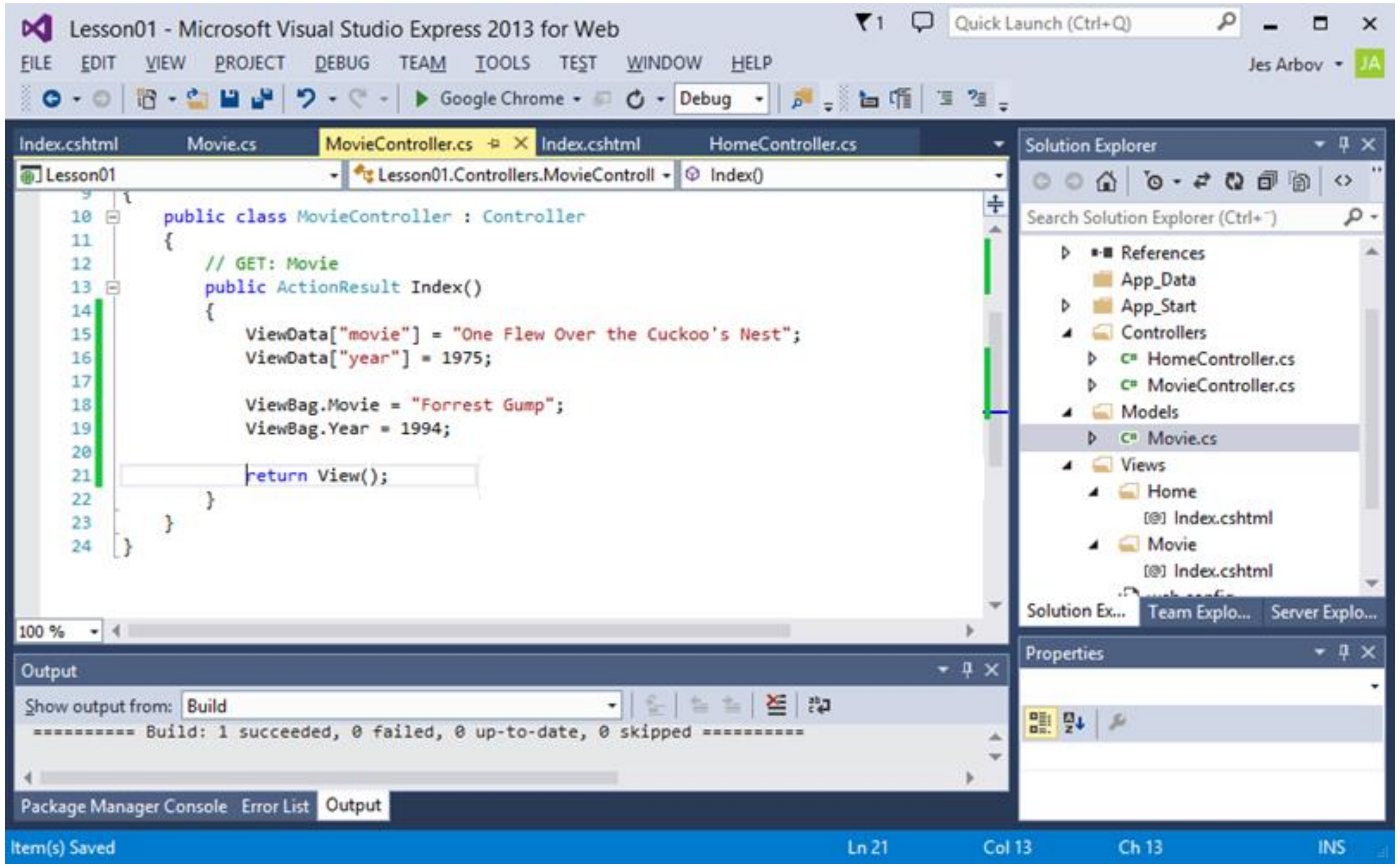
```
@{
    Layout = null;
}

<!DOCTYPE html>

<html>
<head>
    <meta name="viewport" content="width=device-width" />
    <title>Index</title>
</head>
<body>
    <h2>ViewBag</h2>
    <ul>
        <li>@ViewBag.Movie (@ViewBag.Year)</li>
    </ul>

</body>
</html>
```

Example 1: The output



```
ViewBag

• One Flew Over the Cuckoo's Nest (1975)
```

Example 2: The Controller

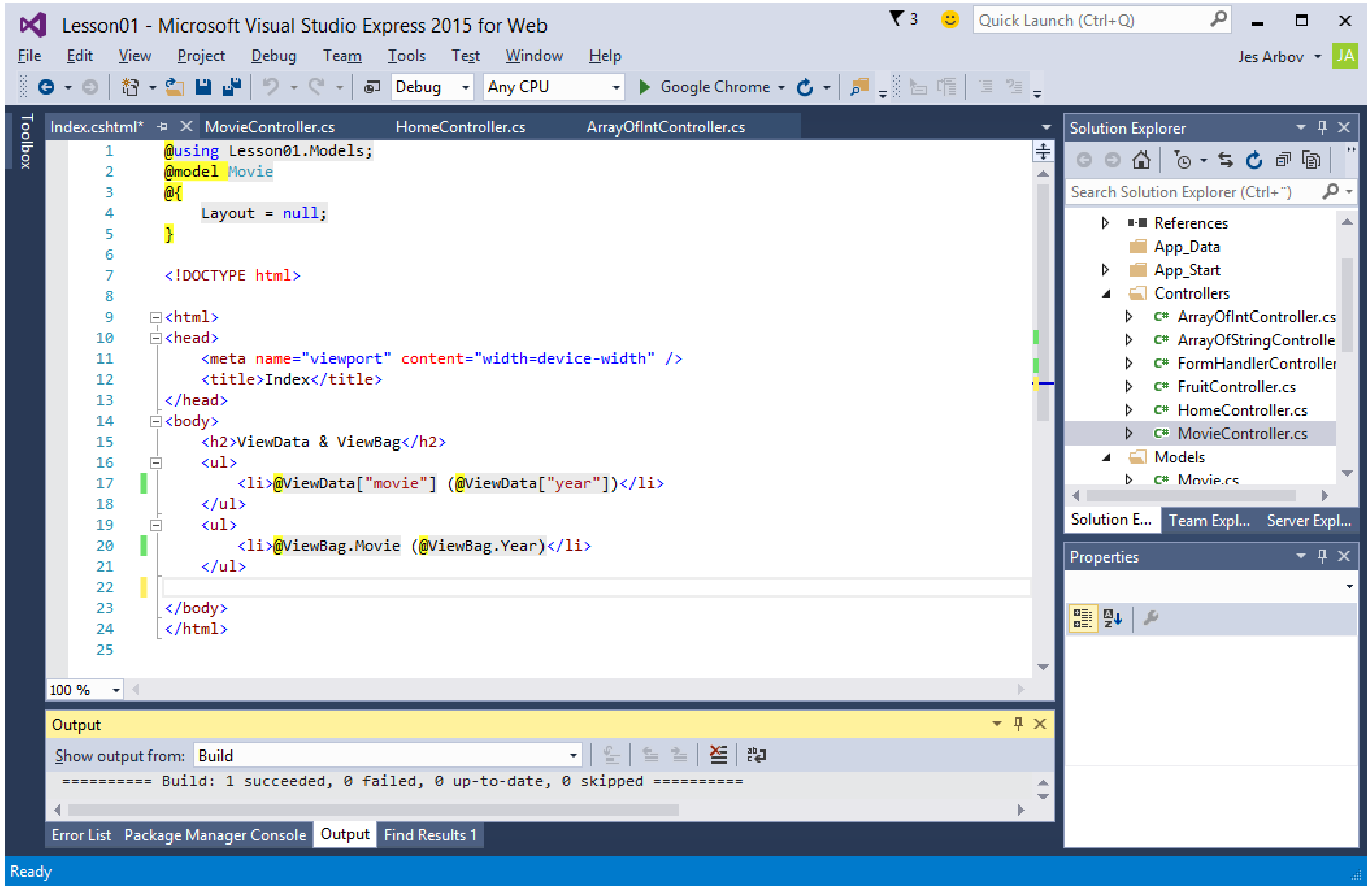


Lesson01 - Microsoft Visual Studio Express 2013 for Web

```
public class MovieController : Controller
{
    // GET: Movie
    public ActionResult Index()
    {
        ViewData["movie"] = "One Flew Over the Cuckoo's Nest";
        ViewData["year"] = 1975;

        ViewBag.Movie = "Forrest Gump";
        ViewBag.Year = 1994;

        return View();
    }
}
}
```

```
========== Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped ==========
```

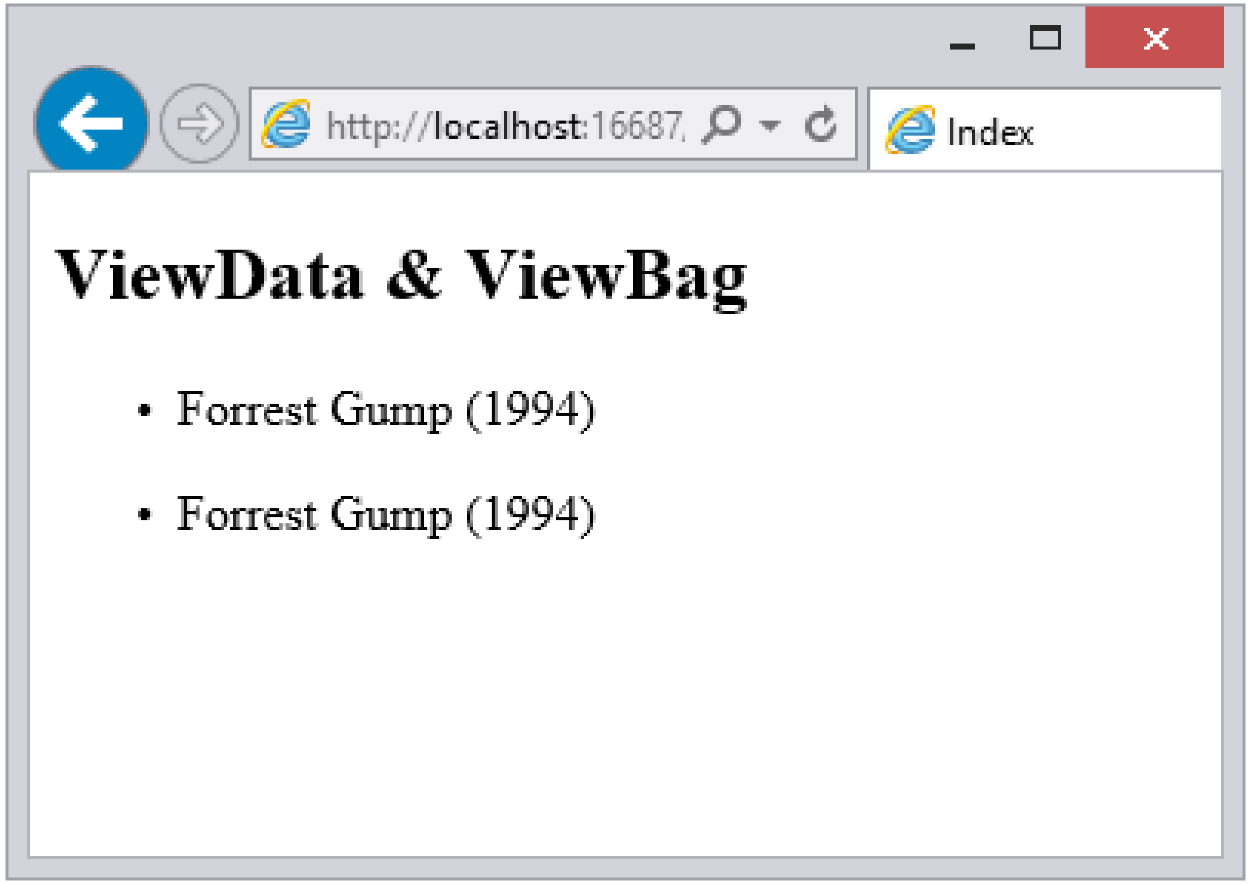
Example 2: The View



```
@using Lesson01.Models;
@model Movie
@{
    Layout = null;
}

<!DOCTYPE html>

<html>
<head>
    <meta name="viewport" content="width=device-width" />
    <title>Index</title>
</head>
<body>
    <h2>ViewData & ViewBag</h2>
    <ul>
        <li>@ViewData["movie"] (@ViewData["year"])</li>
    </ul>
    <ul>
        <li>@ViewBag.Movie (@ViewBag.Year)</li>
    </ul>

</body>
</html>
```

```
========== Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped ==========
```

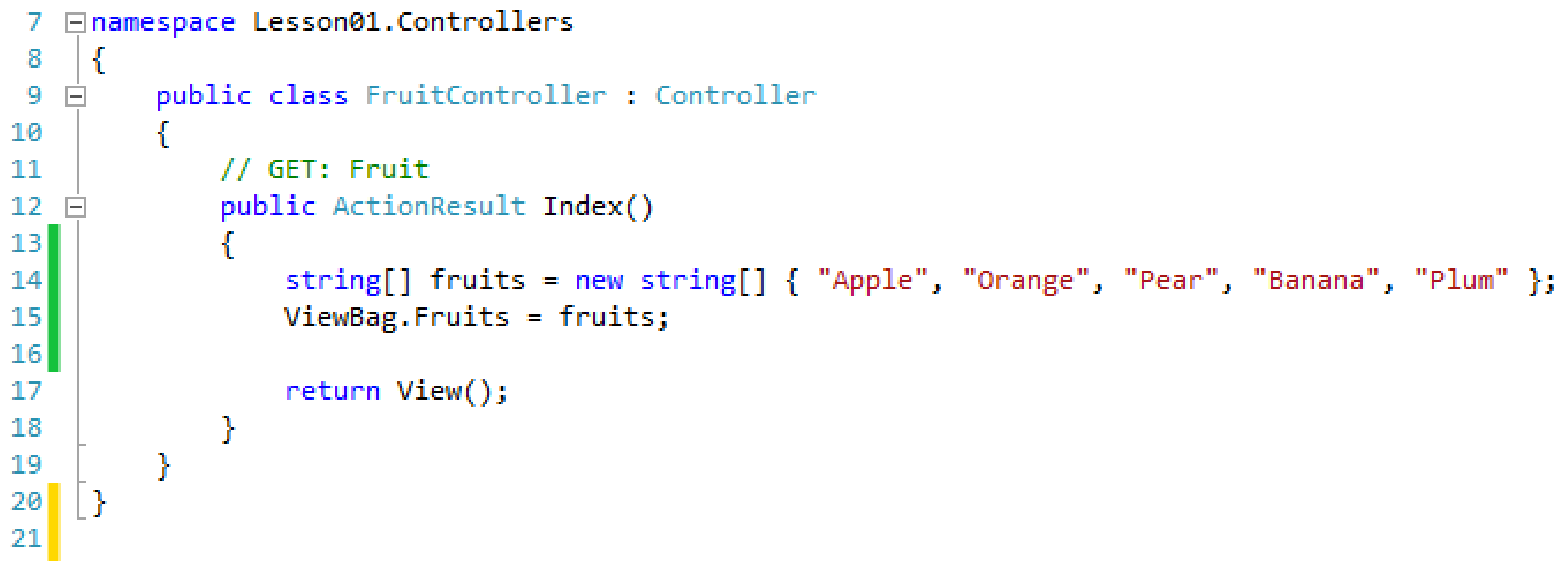
Example 2: The output



ViewData & ViewBag

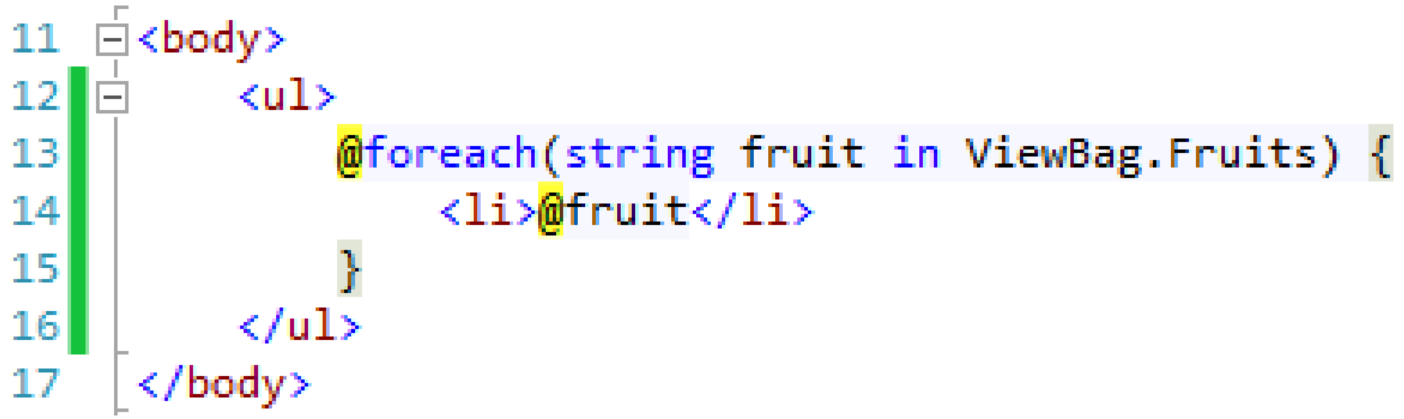
- Forrest Gump (1994)
- Forrest Gump (1994)

FruitController

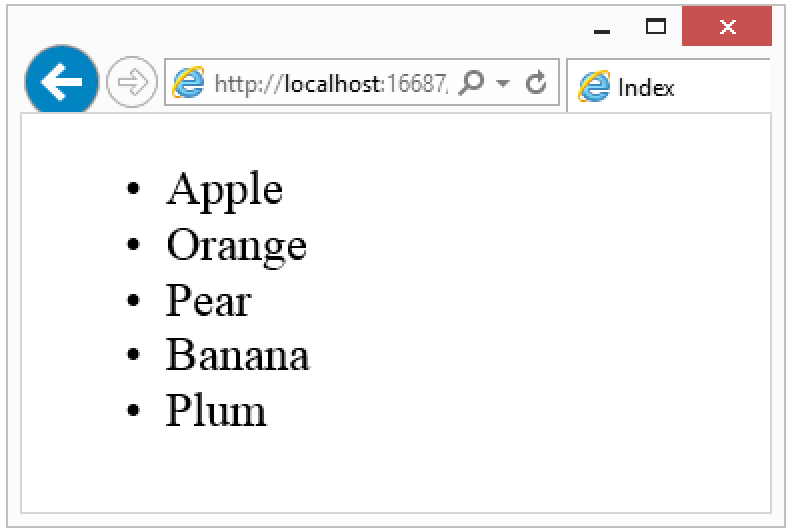


```
namespace Lesson01.Controllers
{
    public class FruitController : Controller
    {
        // GET: Fruit
        public ActionResult Index()
        {
            string[] fruits = new string[] { "Apple", "Orange", "Pear", "Banana", "Plum" };
            ViewBag.Fruits = fruits;

            return View();
        }
    }
}

```

The View



```
<body>
    <ul>
        @foreach(string fruit in ViewBag.Fruits) {
            <li>@fruit</li>
        }
    </ul>
</body>
```



- Apple
- Orange
- Pear
- Banana
- Plum

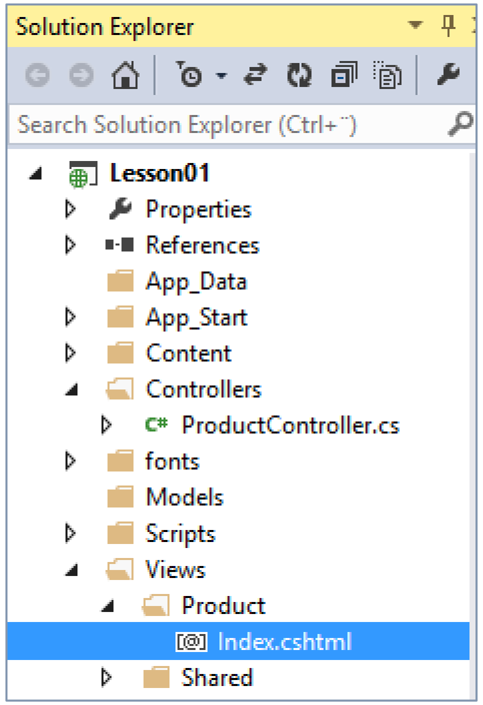
ASP.NET MVC Conventions

Convention over configuration

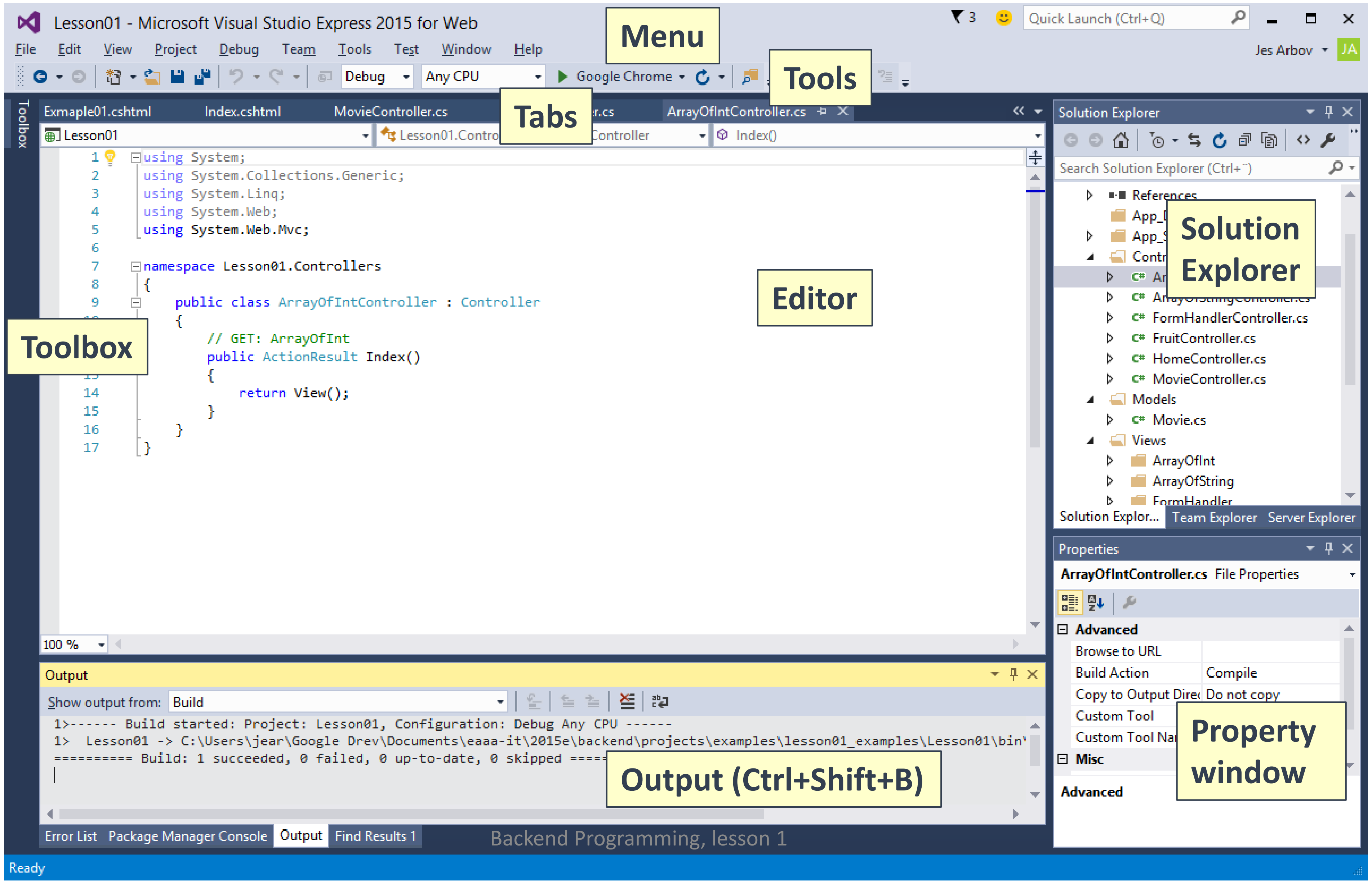
- Directories
  - Controllers
  - Models
  - Views
- Naming conventions for Controllers
  - Each controller's class name ends with *Controller*: `ProductController`
- Naming conventions for Views
  - Views that controllers use is in a **subfolder** named after the controller and **filename** named after the action method. For example: **`/Views/Product/Index.chtml`**

Controller name

Action Method name



Visual Studio



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;

namespace Lesson01.Controllers
{
    public class ArrayOfIntController : Controller
    {
        // GET: ArrayOfInt
        public ActionResult Index()
        {
            return View();
        }
    }
}
```

```
1>------ Build started: Project: Lesson01, Configuration: Debug Any CPU ------
1>  Lesson01 -> C:\Users\jear\Google Drev\Documents\eaaa-it\2015e\backend\projects\examples\lesson01_examples\Lesson01\bin\
========== Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped ====
```

Try it yourself

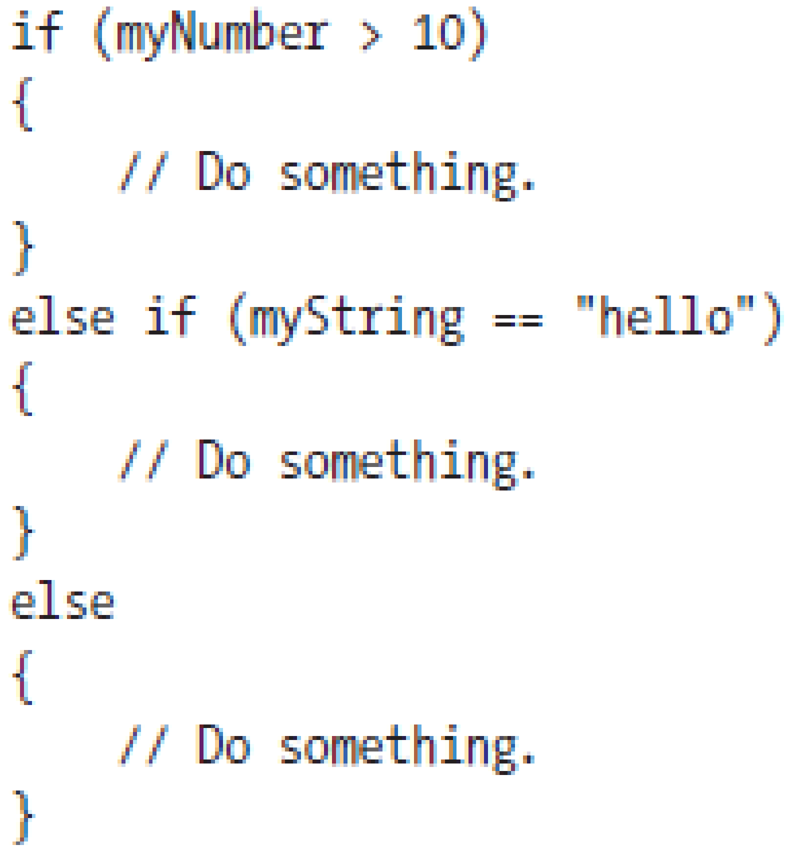
Example 2, slide 19-47

Introduction to C#

Basic C# syntax

- Syntax like C. The language looks a lot “like” JavaScript and PHP
  - Case sensitive
  - All sentences ends with `;`
  - Code blocks in `{}`, conditions in `()`
  - Selection (`if`, `if else`)
  - Loops (`for`, `while`, `do … while`)
  - Operators (assignment `=`; compare `==`, `!=`, `<`, `<=`, `>=`, `>`; logical `!`, `&&` and `||`)

Selection

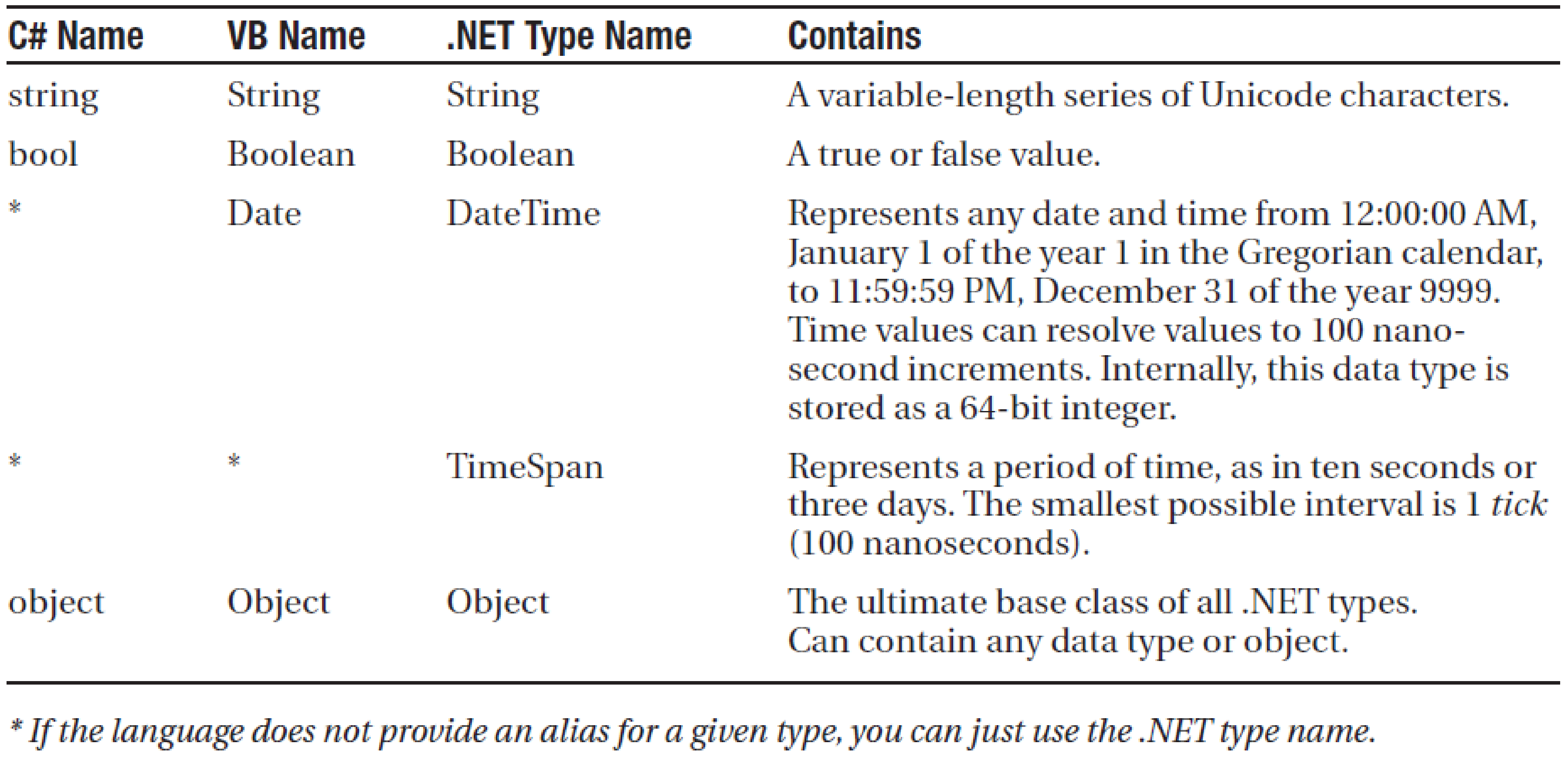


```
if (myNumber > 10)
{
    // Do something.
}
else if (myString == "hello")
{
    // Do something.
}
else
{
    // Do something.
}
```

In C# all variables must be **declared** with a **type**

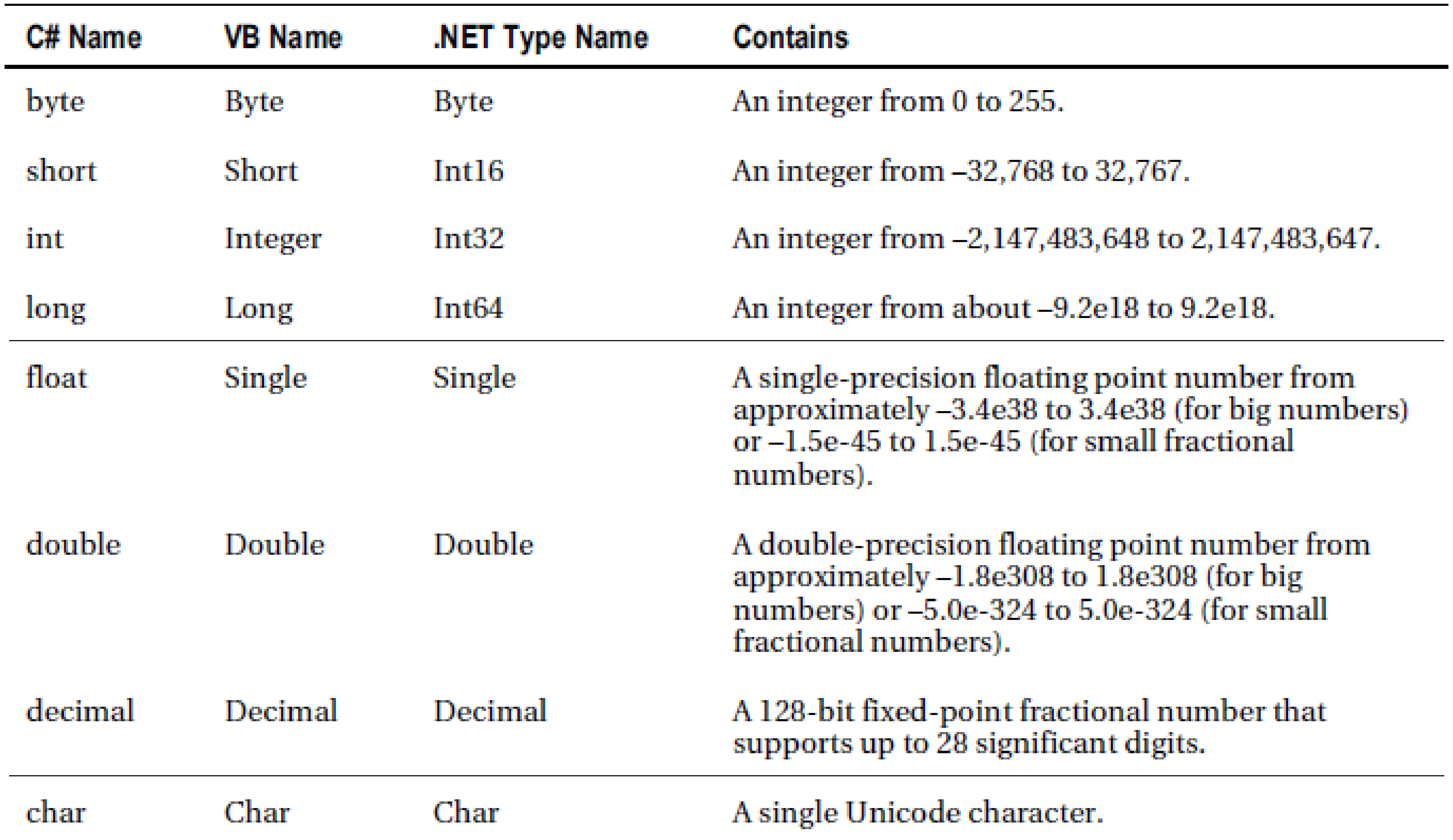
```
Type      Name
 ↓         ↓
string name;           // declaration
name = "Brian Wilson"; // assignment
             ↑
           Value
// Initialization:
// combined declaration and assignment
string name = "Brian Wilson";
               ______________
  ↑        ↑          ↑
 Type     Name    Initializer

name = 200; // What's the problem here …?
// illegal because name is a string and
// 200 is an int
```

Types in C#, VB and .NET



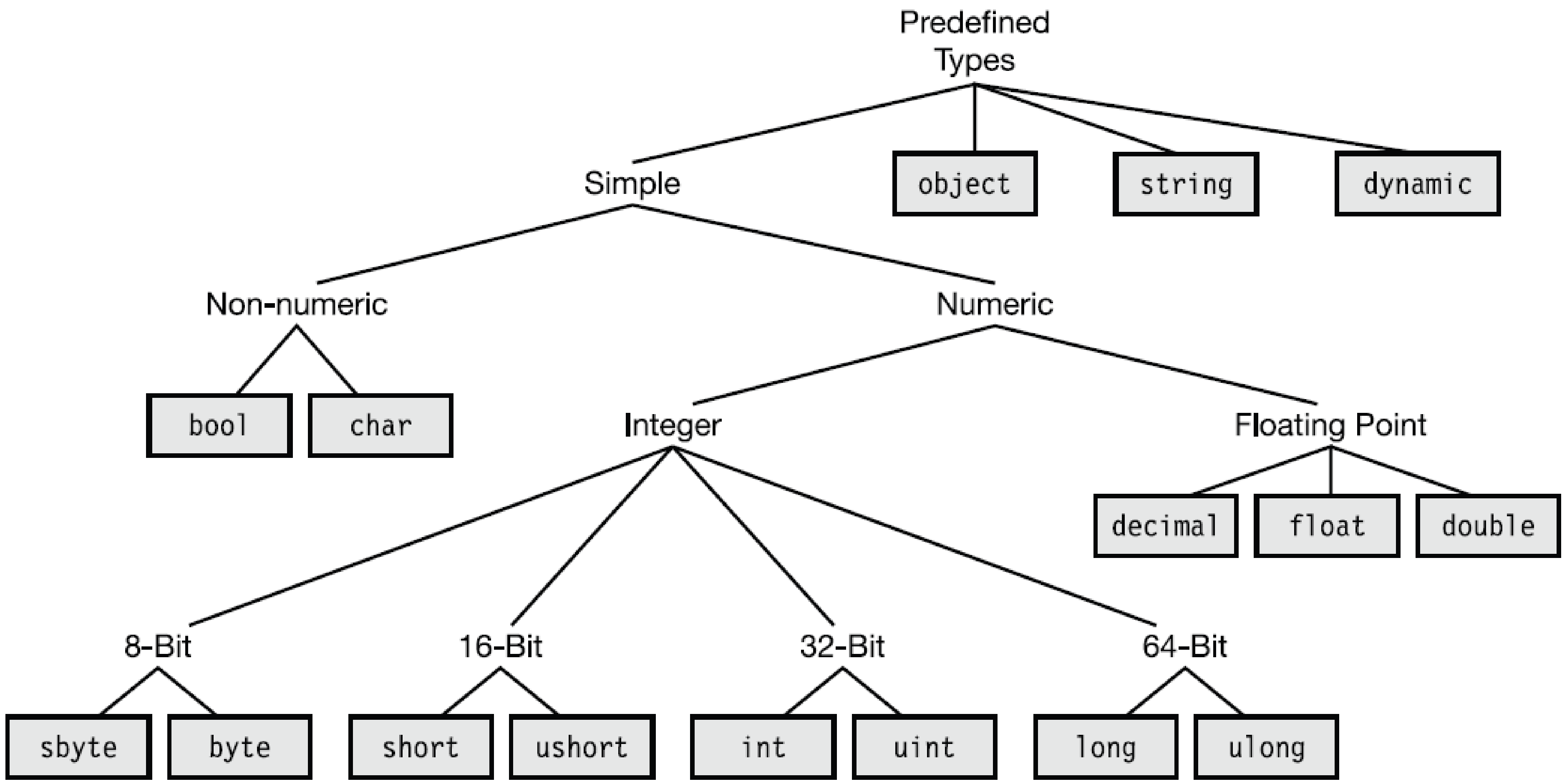
C# Name	VB Name	.NET Type Name	Contains
string	String	String	A variable-length series of Unicode characters.
bool	Boolean	Boolean	A true or false value.
*	Date	DateTime	Represents any date and time from 12:00:00 AM, January 1 of the year 1 in the Gregorian calendar, to 11:59:59 PM, December 31 of the year 9999. Time values can resolve values to 100 nano-second increments. Internally, this data type is stored as a 64-bit integer.
*	*	TimeSpan	Represents a period of time, as in ten seconds or three days. The smallest possible interval is 1 *tick* (100 nanoseconds).
object	Object	Object	The ultimate base class of all .NET types. Can contain any data type or object.

** If the language does not provide an alias for a given type, you can just use the .NET type name.*



C# Name	VB Name	.NET Type Name	Contains
byte	Byte	Byte	An integer from 0 to 255.
short	Short	Int16	An integer from –32,768 to 32,767.
int	Integer	Int32	An integer from –2,147,483,648 to 2,147,483,647.
long	Long	Int64	An integer from about –9.2e18 to 9.2e18.
float	Single	Single	A single-precision floating point number from approximately –3.4e38 to 3.4e38 (for big numbers) or –1.5e-45 to 1.5e-45 (for small fractional numbers).
double	Double	Double	A double-precision floating point number from approximately –1.8e308 to 1.8e308 (for big numbers) or –5.0e-324 to 5.0e-324 (for small fractional numbers).
decimal	Decimal	Decimal	A 128-bit fixed-point fractional number that supports up to 28 significant digits.
char	Char	Char	A single Unicode character.

The predefined types



Scope of variables

- The scope of the declaration is within the nearest block defined by { and }, starting at the place of declaration (scope of n is marked with *):

```
if (...)
{
   ...
   int n=7;                 *
   while (...)              *
   {                        *
   }                        *
                            *
}
```

Converting between numbertypes

```
int i = 12;                // 32bit
long l = 4294967296;       // 64bit
decimal d = 1945.763;      // 128bit

l=i;        // ok more digits
d=i;        // ok more digits
d=l;        // ok more digits
i=l;        // illegal: possible loss of digits
i=(int)l;   // OK explicit typecasting
i=(int)d;   // OK d is truncated to 1945
            // not rounded  to 1946
```

Converting between strings and numbers

```
string intStr = "24";
string doubleStr = "80.349";

// convert string to int
int n = Convert.ToInt32(intStr);
// convert string to double
double d = Convert.ToDouble(doubleStr);

// convert int to string, 3 alternatives
Int m = 562;
string str = Convert.ToString(m);
string str = m.ToString(); // shorter
string str = "" + m; // m is converted to string and
                     // appended to the empty string
```

Manipulating strings (string concatenation)

```
string firstname = "Jes";
string lastname = "Arbov";

// combine to fullname
string fullname = firstname + " " + lastname;
string name = lastname + ", " + firstname;

// fullname = "Jes Arbov"
// names = "Arbov, Jes"
```

DateTime

- `DateTime` is a type which can contain a day and a time.
- `DateTime` is not part of the C# language, but a type defined in the framework.

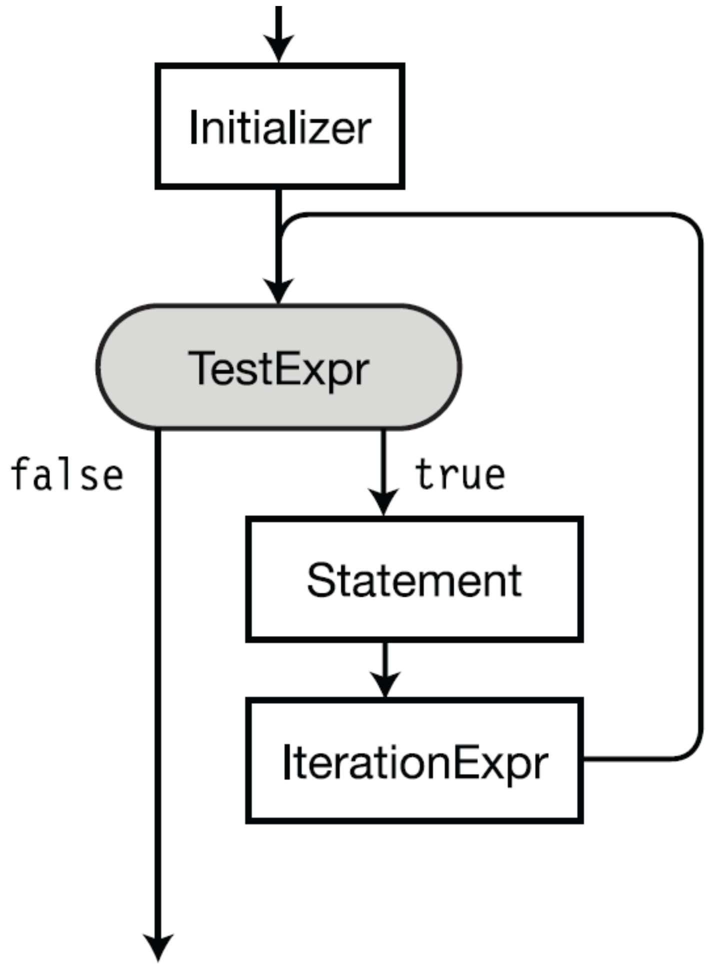
DateTime examples

```
DateTime dt1 = new DateTime(2016, 9, 30);

string s1 = dt1.ToShortDateString();  // 30-09-2016;
string s2 = dt1.ToLongTimeString();   // 00:00:00
string s4 = dt1.ToLongDateString(); // 30. september 2016
string s3 = dt1.ToString(); // 30-09-2016 00:00:00

DateTime dt2 = new DateTime(2016, 9, 1, 9, 20, 40);

string s4 = dt2.ToString(); // 01-09-2016 09:20:40

DateTime dt3 = DateTime.Today; // current day at 00:00:00
DateTime dt4 = DateTime.Now;   // current day and time
```

For documentation, see: http://msdn.microsoft.com/en-us/library/System.DateTime.aspx

Array of int and a for loop

```
@{
    int[] nums = { 1, 7, 9, 20 };
    // add numbers in array
    int sum = 0;
    for (int i = 0; i < nums.Length; i++)
    {
        sum = sum + nums[i];
    }
}
@sum // 37
```



Arrays of type string

```
<ul>
    @{
            string[] colorNames = new string[5];
            colorNames[0] = "Yellow";
            colorNames[1] = "Green";
            colorNames[2] = "Red";
            colorNames[3] = "Blue";
            colorNames[4] = "White";

            for (int i = 0; i < colorNames.Length; i++) {
                <li>@colorNames[i]</li>
            }
    }
     @s
</ul>
```

The foreach loop

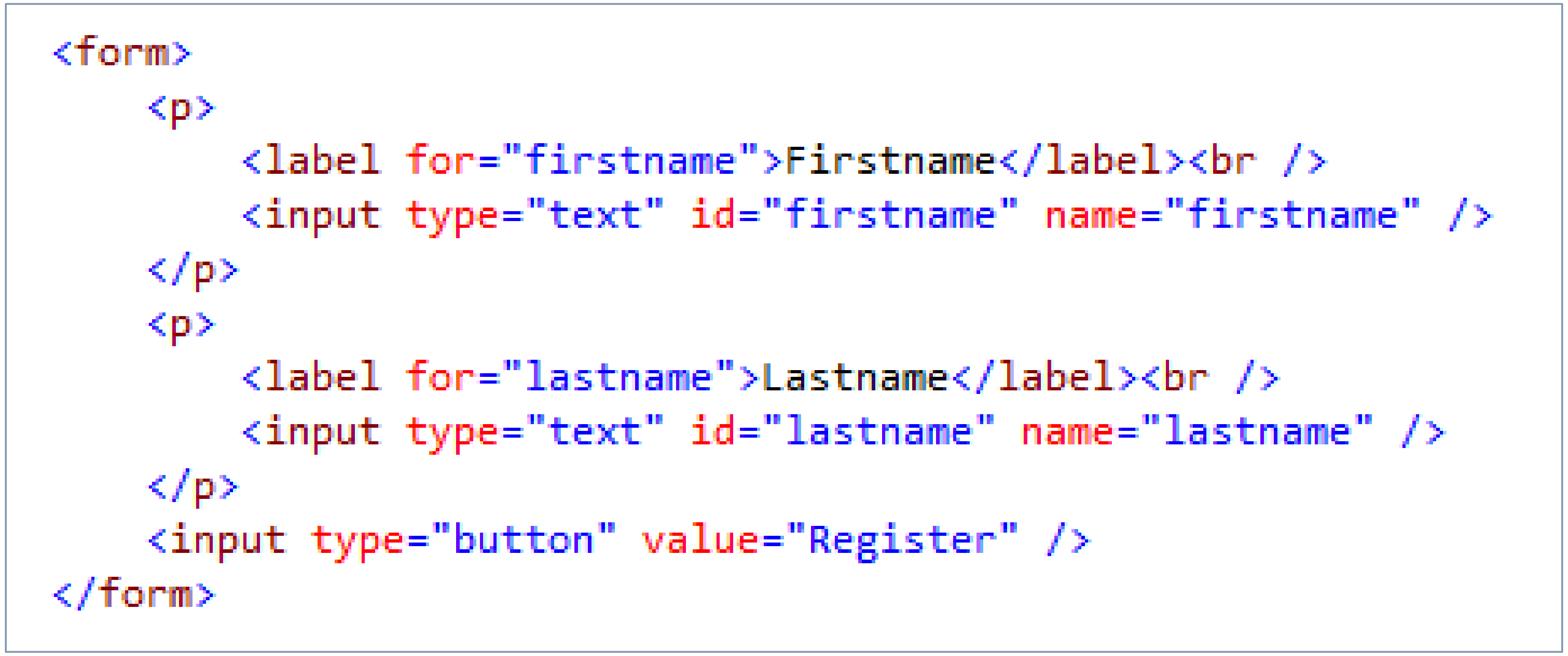
```
<ul>
    @{
        string[] colorNames = new string[5];
        colorNames[0] = "Yellow";
        colorNames[1] = "Green";
        colorNames[2] = "Red";
        colorNames[3] = "Blue";
        colorNames[4] = "White";

        foreach (string color in colorNames) {
            <li>@color</li>
        }
    }
</ul>
```

Exercises 1-2

Handling form data

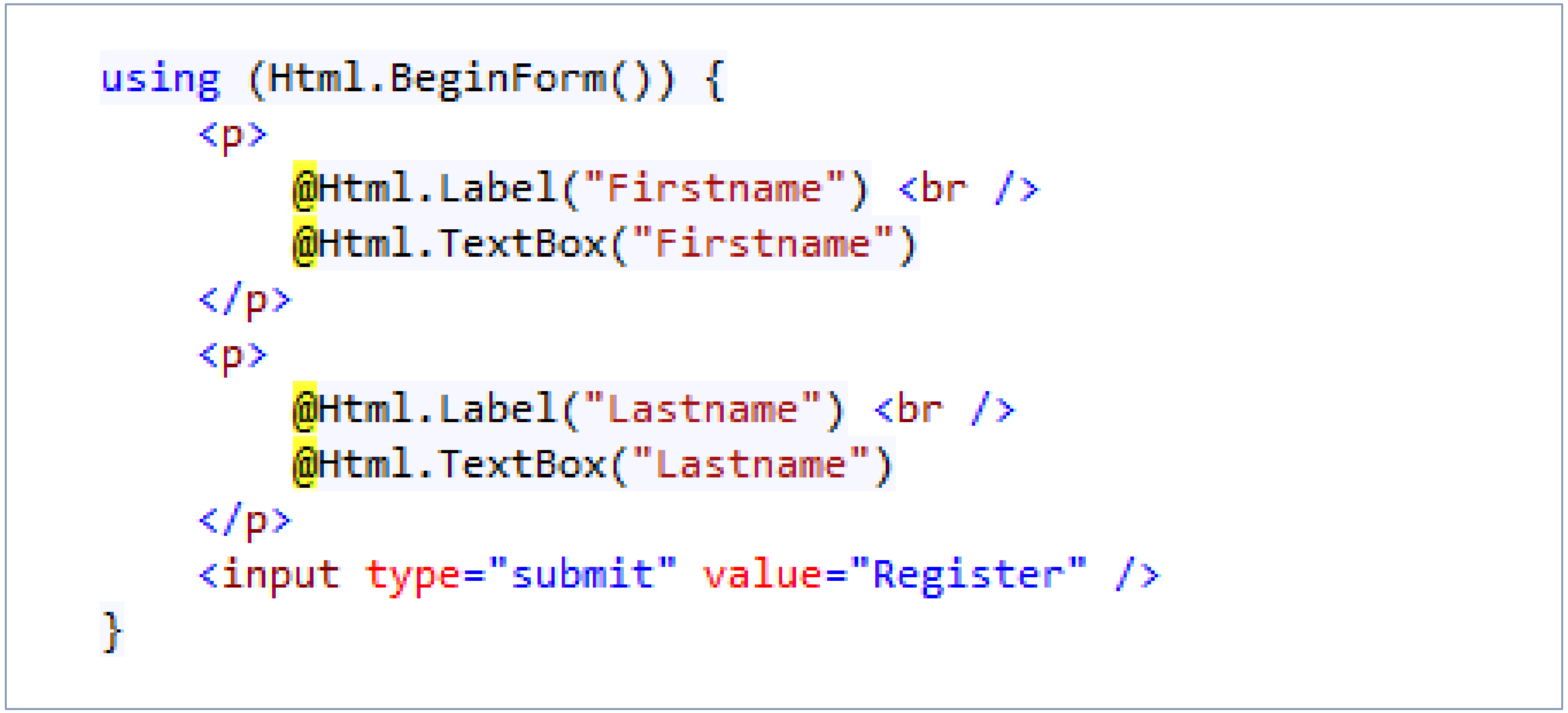
A short introduction for exercise 3-4

Creating Forms in Views



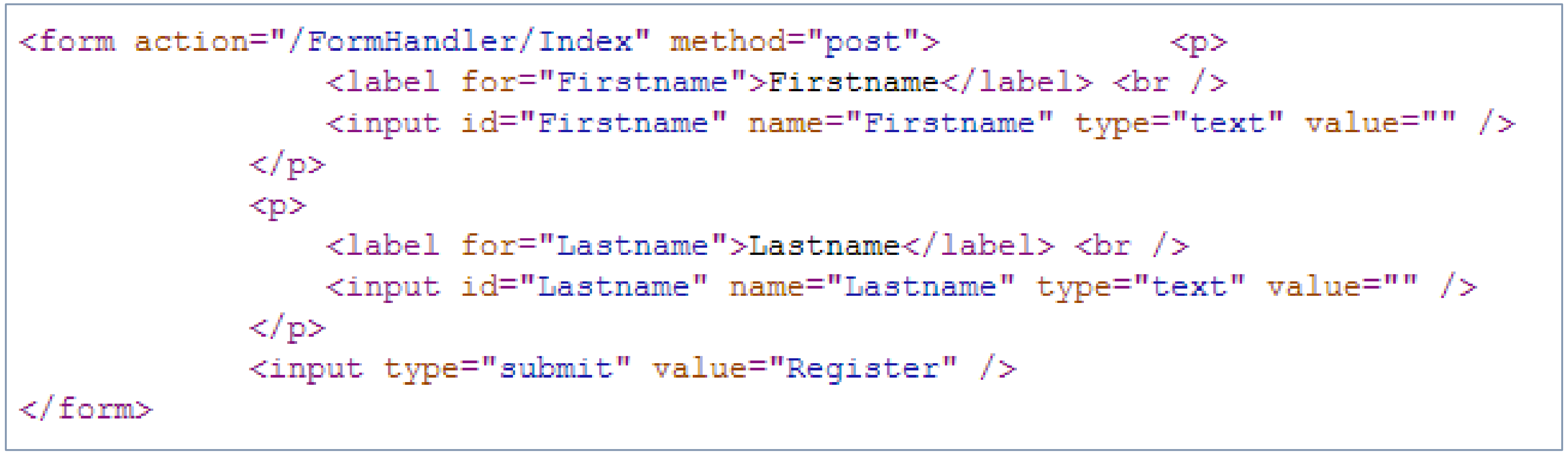
```
<form>
    <p>
        <label for="firstname">Firstname</label><br />
        <input type="text" id="firstname" name="firstname" />
    </p>
    <p>
        <label for="lastname">Lastname</label><br />
        <input type="text" id="lastname" name="lastname" />
    </p>
    <input type="button" value="Register" />
</form>
```

Creating Forms in Views with Html Helpers



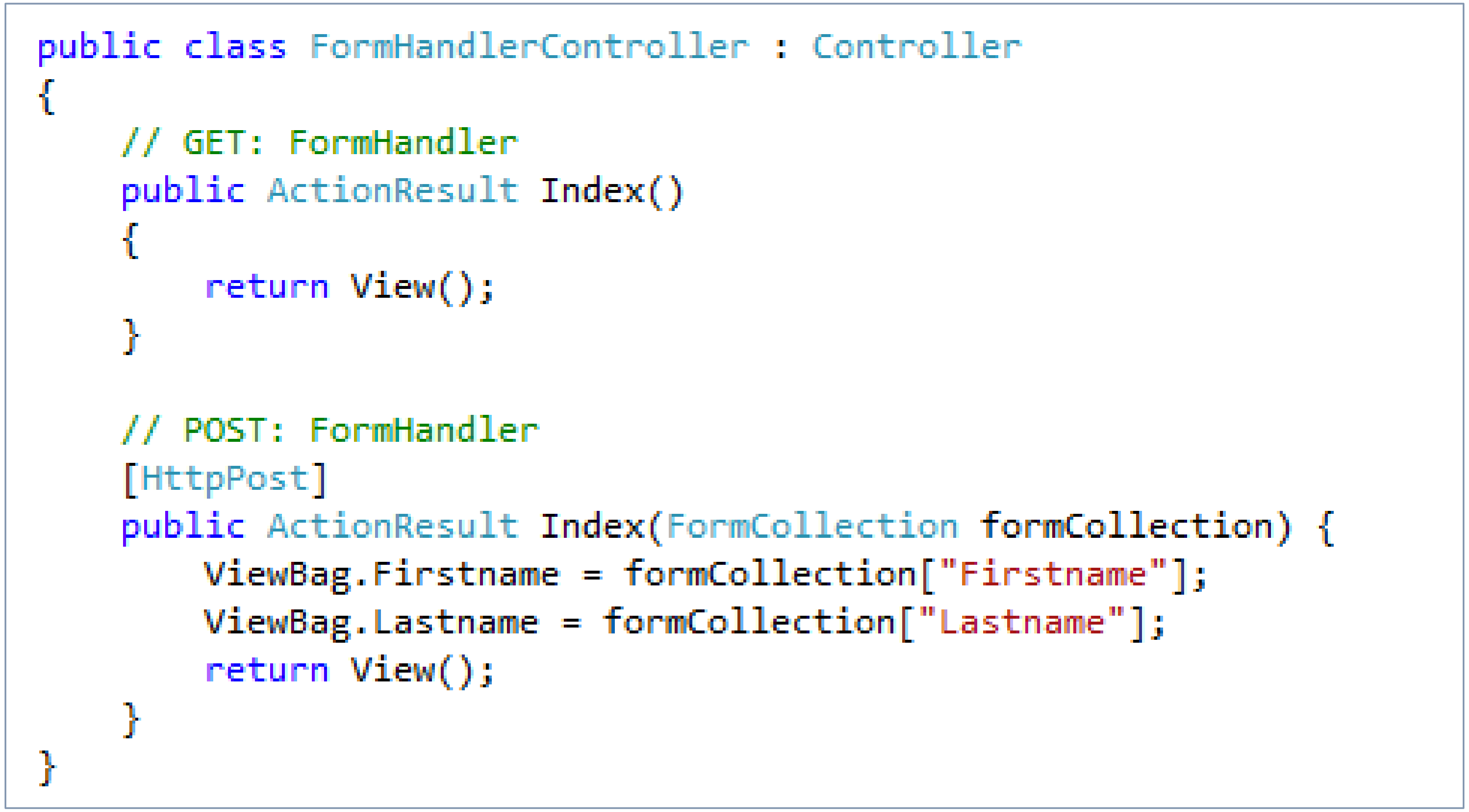
```
using (Html.BeginForm()) {
    <p>
        @Html.Label("Firstname") <br />
        @Html.TextBox("Firstname")
    </p>
    <p>
        @Html.Label("Lastname") <br />
        @Html.TextBox("Lastname")
    </p>
    <input type="submit" value="Register" />
}
```

The HTML Output



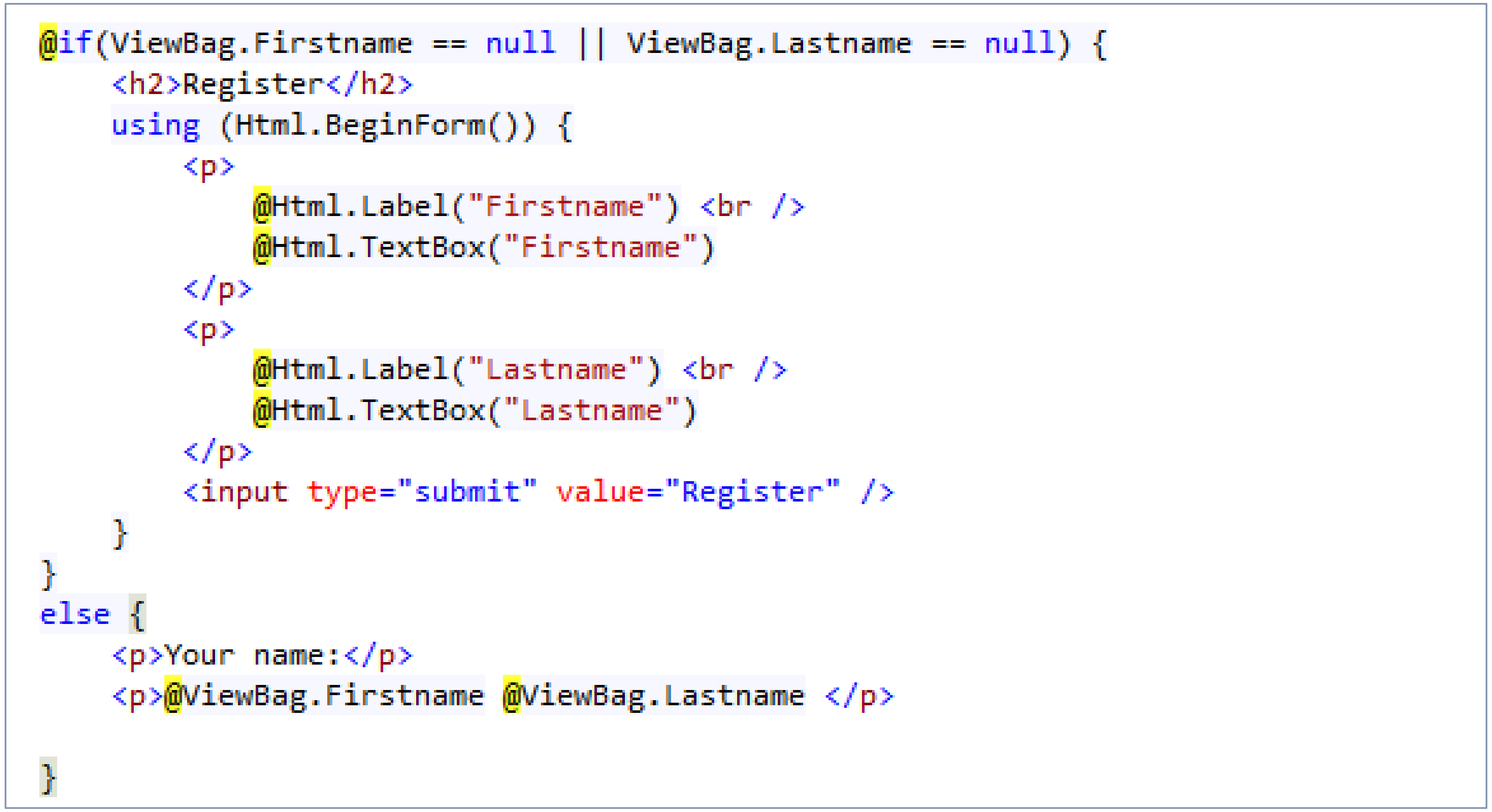
```
<form action="/FormHandler/Index" method="post">            <p>
                <label for="Firstname">Firstname</label> <br />
                <input id="Firstname" name="Firstname" type="text" value="" />
            </p>
            <p>
                <label for="Lastname">Lastname</label> <br />
                <input id="Lastname" name="Lastname" type="text" value="" />
            </p>
            <input type="submit" value="Register" />
</form>
```

Handling form data with ActionMethodSelectorAttributes



```
public class FormHandlerController : Controller
{
    // GET: FormHandler
    public ActionResult Index()
    {
        return View();
    }

    // POST: FormHandler
    [HttpPost]
    public ActionResult Index(FormCollection formCollection) {
        ViewBag.Firstname = formCollection["Firstname"];
        ViewBag.Lastname = formCollection["Lastname"];
        return View();
    }
}
```

The View: An example



```
@if(ViewBag.Firstname == null || ViewBag.Lastname == null) {
    <h2>Register</h2>
    using (Html.BeginForm()) {
        <p>
            @Html.Label("Firstname") <br />
            @Html.TextBox("Firstname")
        </p>
        <p>
            @Html.Label("Lastname") <br />
            @Html.TextBox("Lastname")
        </p>
        <input type="submit" value="Register" />
    }
}
else {
    <p>Your name:</p>
    <p>@ViewBag.Firstname @ViewBag.Lastname </p>

}
```

Exercise 3-4

Next week: OOP 1:2

- Object-oriented programming in C#: A Concise Introduction, pp. 1-28

- C# From Scratch: Objects (Pluralsight, Jesse Liberty)
  This is the essential part, but it's a good idea to go through the lessons that leads up to the "Objects" lesson and absorb any parts you're not yet familiar with.