Introduction to ASP .Net Framework

Course Code: CSC 4164 Course Title: ADVANCED PROGRAMMING WITH .NET

Dept. of Computer Science
Faculty of Science and Technology

Lecture No:	**02**	**Week No:**	**01**	**Semester:**	**Summer 2020-21**
Lecturer:	*Tanvir Ahmed, tanvir.ahmed@aiub.edu*				

Lecture Outline

- Overview of .NET Architecture
- Introduction to ASP.Net Framework
- Introduction to Visual Studio

Overview of .NET

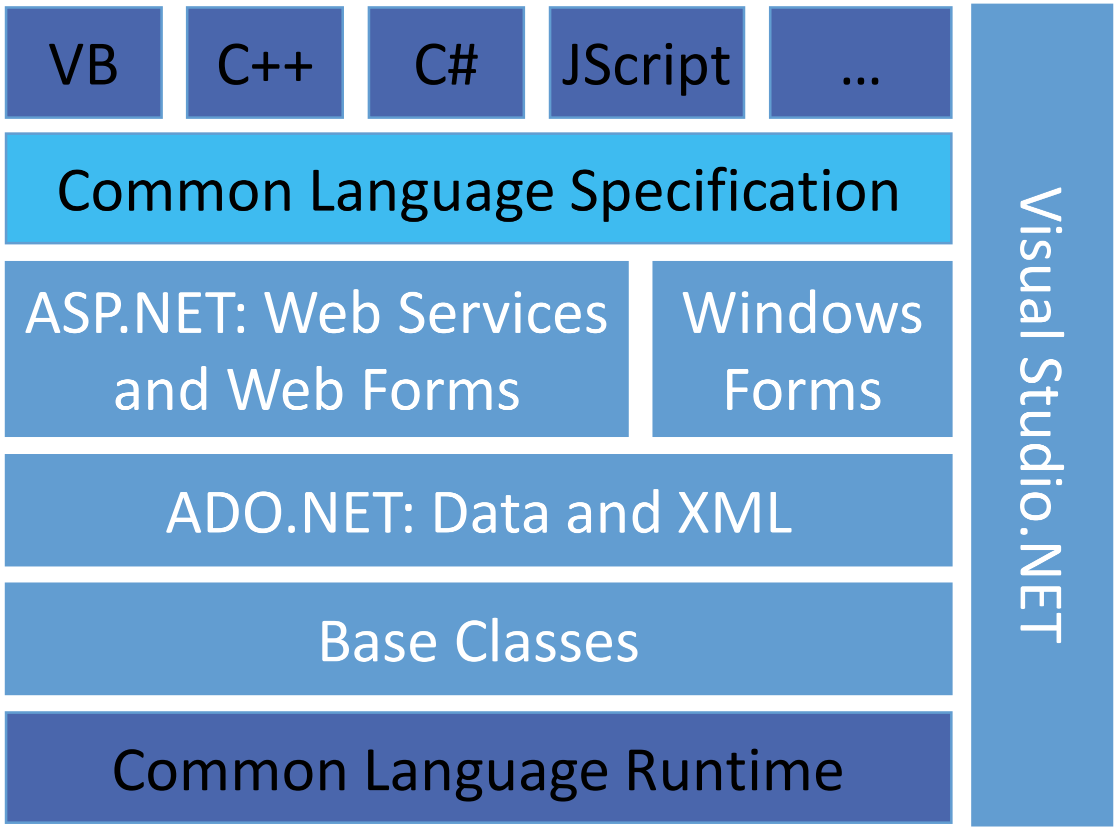
.NET Architecture

What is .NET Framework

- The .Net framework is a software development platform developed by Microsoft.
- The framework was meant to create applications, which would run primarily on the Windows Platform.
-
- The first version of the .Net framework was released in the year 2002.
- The .Net framework can be used to create any kind of applications - Form based, Web based, Mobile applications

Overview of .NET

.NET Architecture

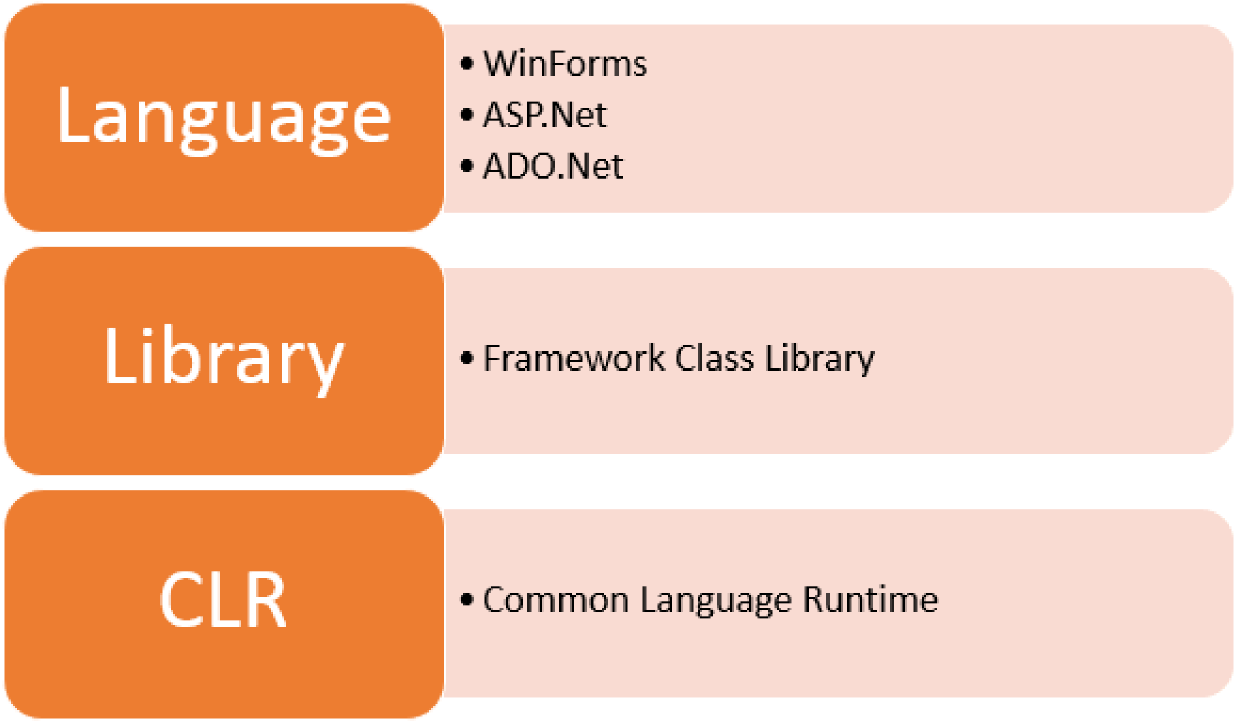


- Enormous platform
- Over 10000++ types
- Well-defined sub-systems and partitioning of responsibility

Overview of .NET

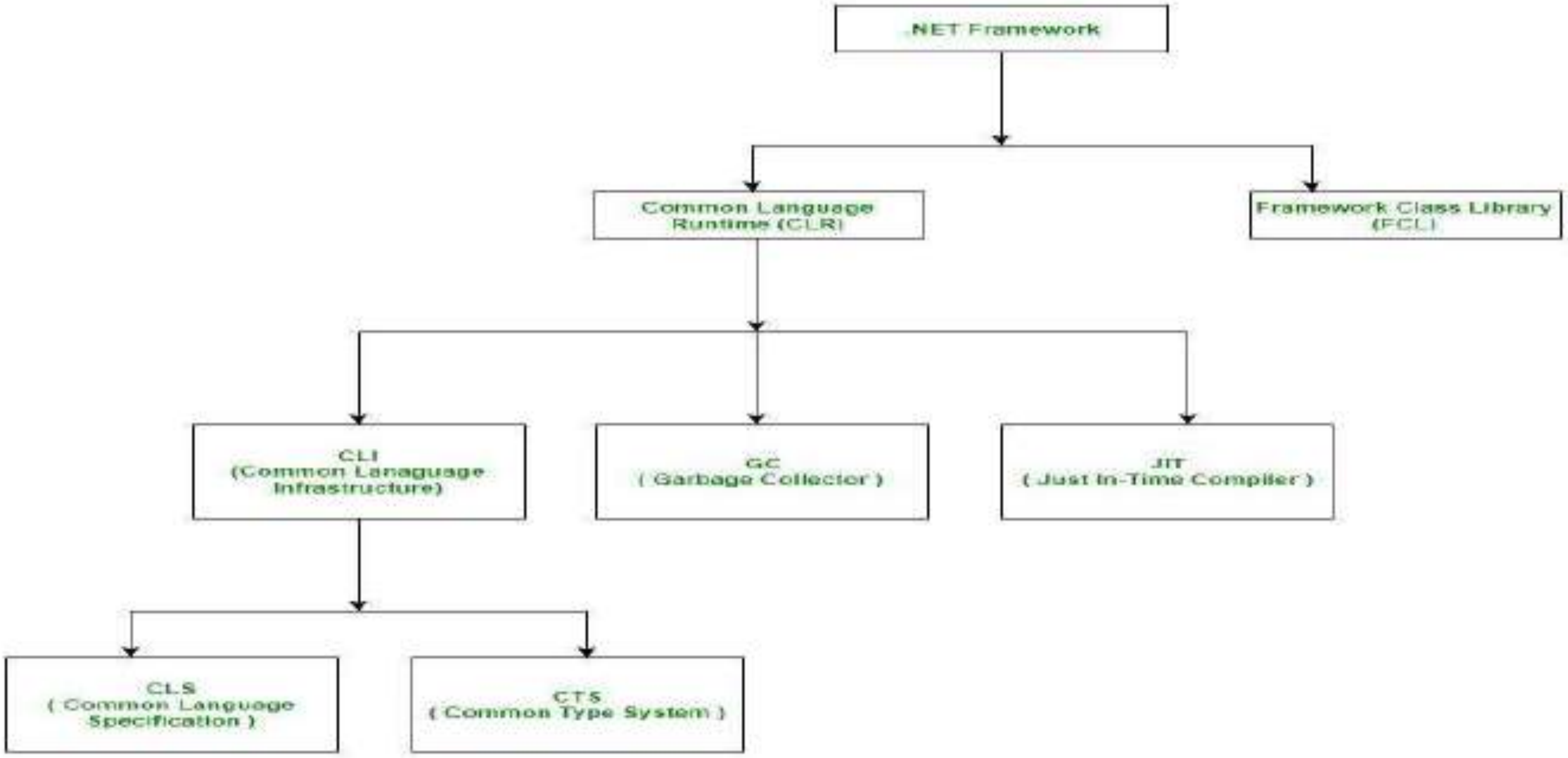
.NET Architecture

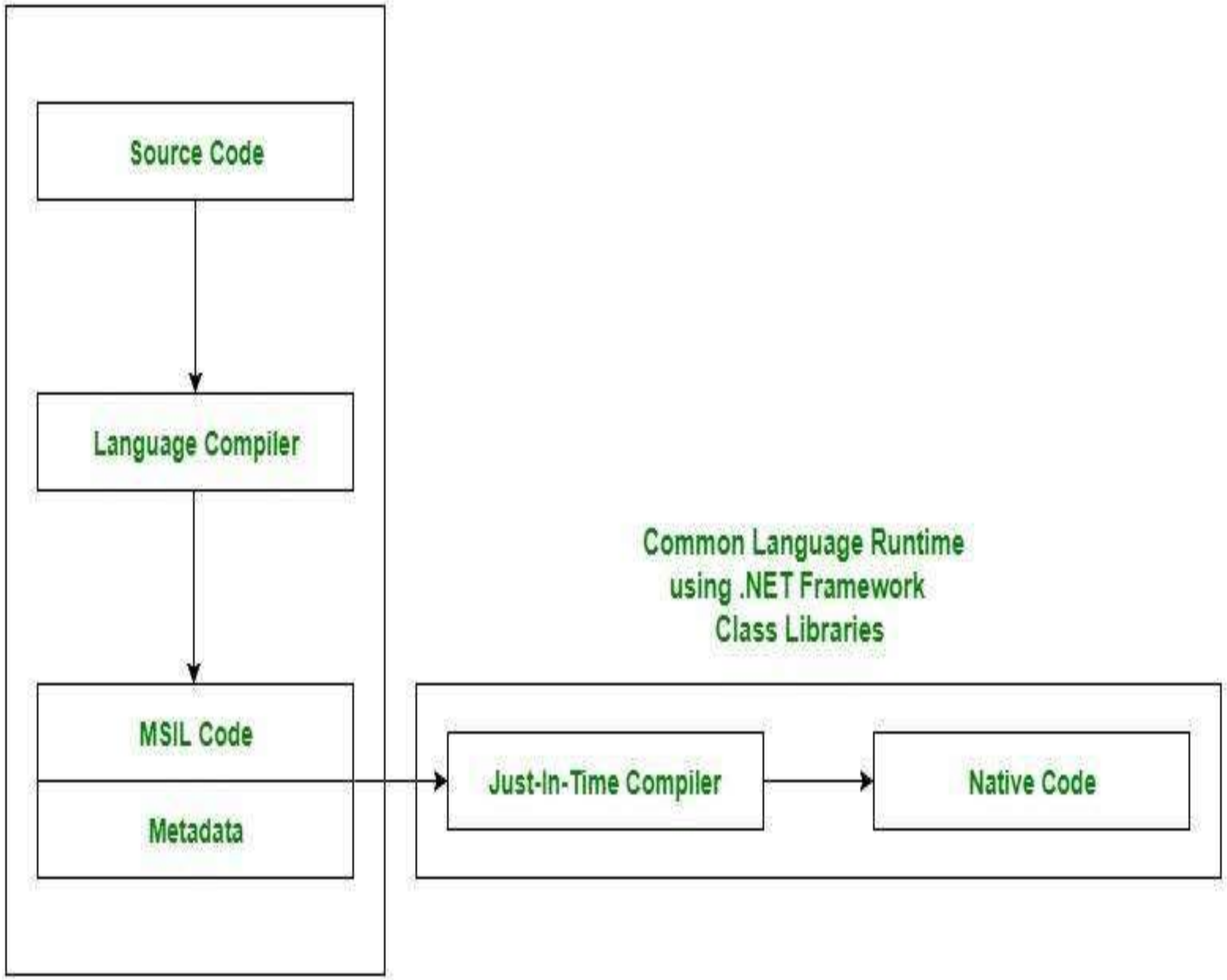
.Net Framework Architecture



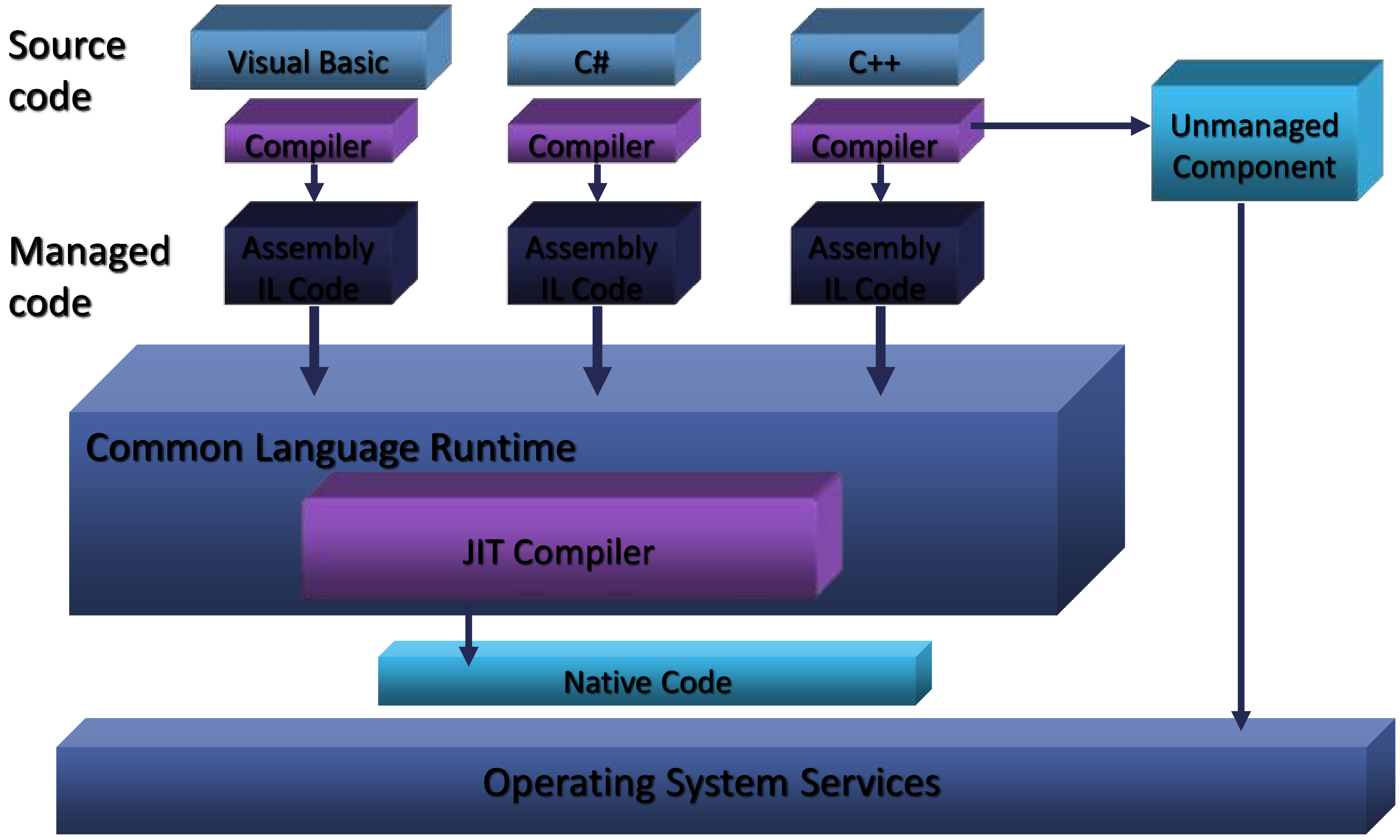
Overview of .NET

.NET Architecture


.NET Framework
Common Language Runtime (CLR)
Framework Class Library (FCL)
CLI (Common Lanaguage Infrastructure)
GC (Garbage Collector)
JIT (Just In-Time Compiler)
CLS (Common Language Specification)
CTS (Common Type System)


Source Code
Language Compiler
MSIL Code
Metadata
Common Language Runtime using .NET Framework Class Libraries
Just-In-Time Compiler
Native Code

.NET Execution Model



MSIL, IL & JIT

Microsoft Intermediate Language (MSIL), the intermediate code produced by the compiler after compiling the source code. This intermediate code is known as MSIL.

Intermediate Language (IL) is also known as MSIL (Microsoft Intermediate Language) or CIL (Common Intermediate Language).

Just In-Time Compiler (JIT), responsible for converting the CIL(Common Intermediate Language) into machine code using the Common Language Runtime environment.

Common Language Runtime (CLR) provides an environment to execute .NET applications on target machines. The responsibilities of CLR are listed as follows**:**

- Automatic memory management
- Garbage Collection
- Code Access Security
- Code verification
- JIT compilation of .NET code

According to Wiki…

In computer programming, a **software framework** is an abstraction in which software providing generic functionality can be selectively changed by additional user-written code, thus providing application-specific software. It provides a standard way to build and deploy applications and is a universal, reusable software environment that provides particular functionality as part of a larger software platform to facilitate development of software applications, products and solutions.

Introduction to ASP.Net Framework

What is .NET Framework?

.NET is a platform for the developers made up of tools, programming languages and libraries. The framework is a complete environment that allows developers to develop, run and deploy applications such as:

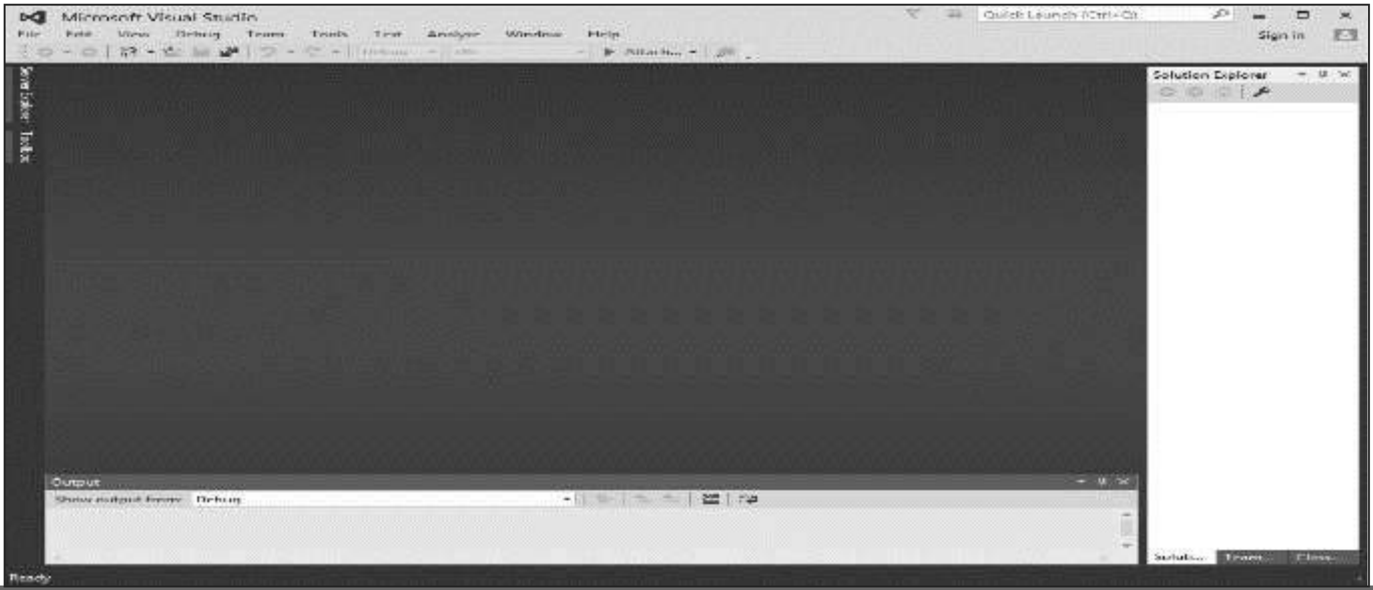
- Console applications.
- Windows Forms applications.
- Windows Presentation Foundation (WPF) applications.
- Web applications (ASP.NET applications).
- Web services.
- Windows services and some more.

The main components of .NET Framework

- .NET Framework Class Library
- Common Language Runtime
- Dynamic Language Runtimes (DLR)
- Application Domains
- Runtime Host
- Common Type System
- Metadata and Self-Describing Components
- Cross-Language Interoperability
- .NET Framework Security
- Profiling
- Side-by-Side Execution

Introduction to Visual Studio

Visual Studio is an IDE (integrated development environment) for building apps. Similar to using Microsoft Word to write documents, you'll use Visual Studio to create web apps.

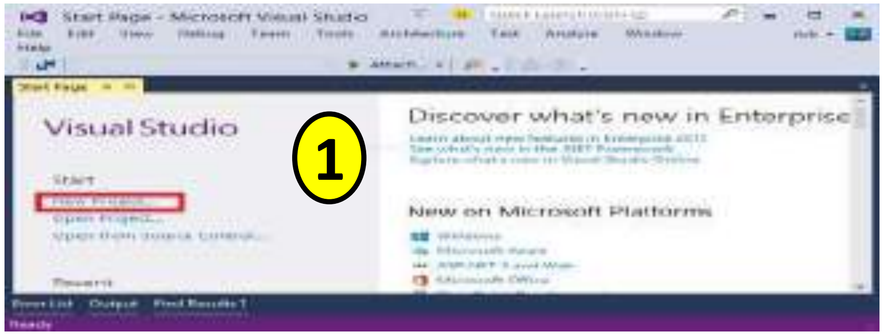


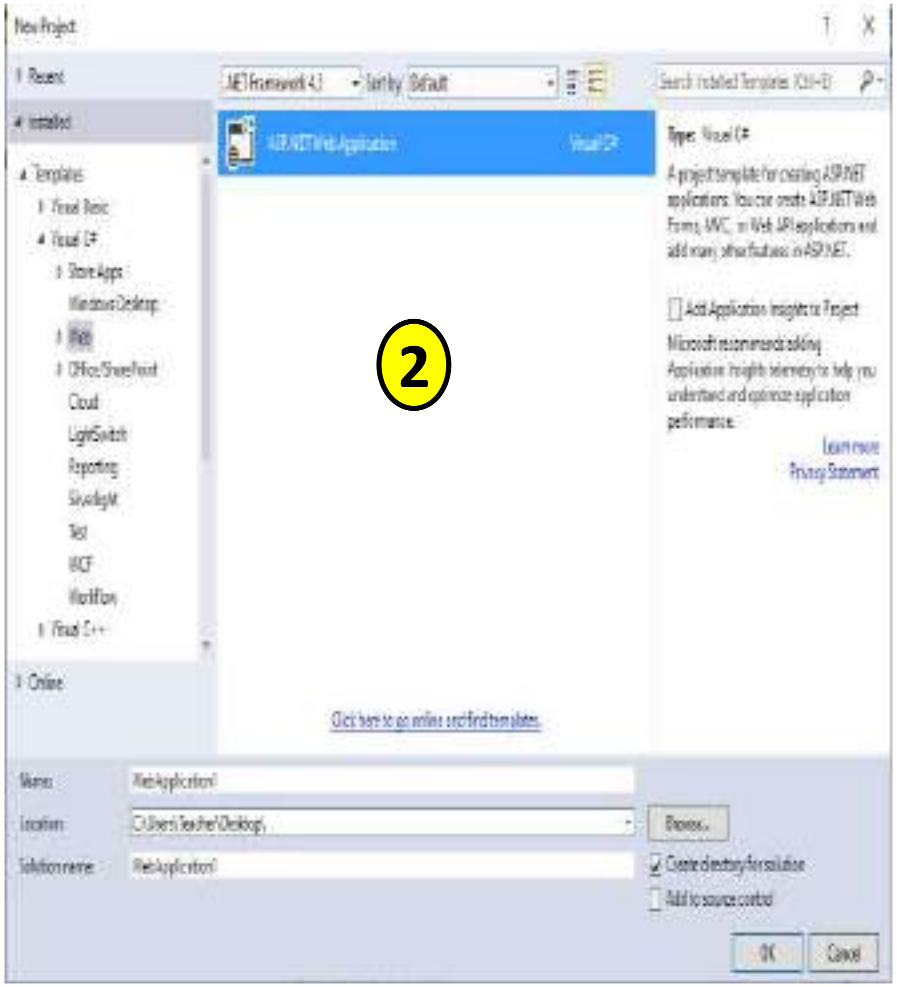
Visual Studio

- VS supports multiple languages (C#, C++, Visual Basic, J#) in one IDE.
- VS manages features and content in a convenient way.
- All Visual Studio .NET languages are object-oriented.
- All programs have a similar structure.
- All programs compiled into Common Intermediate Language (CIL)

- For this course at least Visual Studio 2013

Creating a web application


Start Page - Microsoft Visual Studio
File Edit View Debug Team Tools Architecture Test Analyze Window Help
Start Page
Visual Studio
Start
New Project...
Open Project...
Open from Source Control...
Recent
Discover what's new in Enterprise
New on Microsoft Platforms
Windows
Microsoft Azure
ASP.NET 5 and Web
Microsoft Office
Error List Output Find Results 1
Ready
1



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.

☐ Add Application Insights to Project

Microsoft recommends adding Application Insights telemetry to help you understand and optimize application performance.

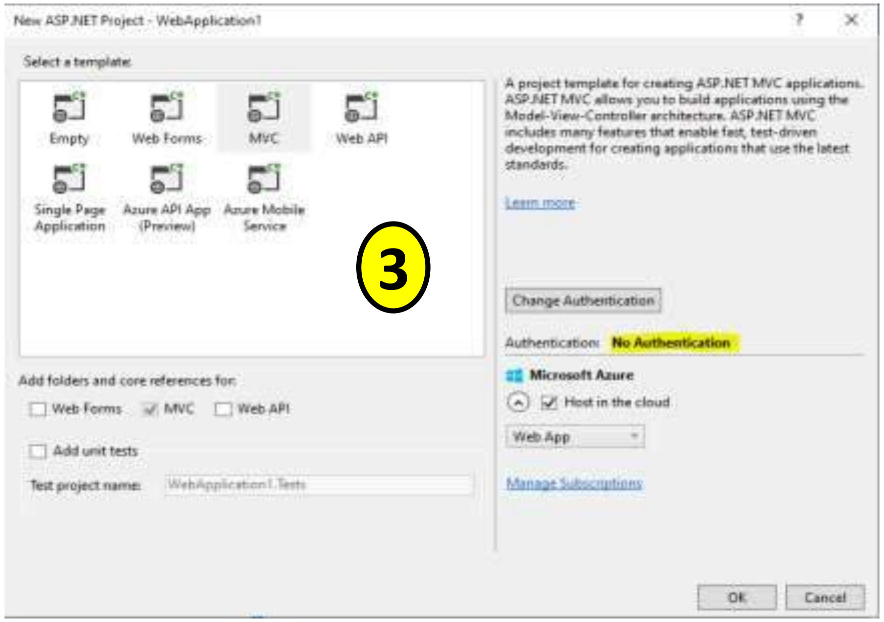
Name: WebApplication1

Location: C:\Users\Teacher\Desktop\

Solution name: WebApplication1

☑ Create directory for solution

☐ Add to source control



A project template for creating ASP.NET MVC applications. ASP.NET MVC allows you to build applications using the Model-View-Controller architecture. ASP.NET MVC includes many features that enable fast, test-driven development for creating applications that use the latest standards.

Authentication: No Authentication

☑ Host in the cloud

Web App

Add folders and core references for:

☐ Web Forms ☑ MVC ☐ Web API

☐ Add unit tests

Test project name: WebApplication1.Tests

Books

1. Visual Studio from the Microsoft Developer Network (MSDN) https://msdn.microsoft.com/enus/library/dd831853(v=vs.120).aspx (particularly note the Visual Studio IDE User Guide and the Application Development in Visual Studio links).
2. C# 4.0 The Complete Reference; Herbert Schildt; McGraw-Hill Osborne Media; 2010
3. Beginning ASP.NET 4: in C# and VB; Imar Spaanjaars,2010

References

1. ASP.NET; URL: https://dotnet.microsoft.com/apps/aspnet
2. .NET Architecture; URL: https://www.geeksforgeeks.org
3. URL: https://www. tutorialspoint.com/index.htm

Thank you!

Introduction to ASP .Net MVC

Course Code: CSC 4164 Course Title: ADVANCED PROGRAMMING WITH .NET

Dept. of Computer Science
Faculty of Science and Technology

Lecture No:	**03**	**Week No:**	**02**	**Semester:**	**Fall 2020-2021**
Lecturer:	*Victor Stany Rozario, stany@aiub.edu*				

Lecture Outline

- Traditional web application VS MVC application
- MVC application lifecycle
- Creating MVC application from scratch
- Routing configuration
- Controllers and Action Methods
- Different Response Types
- Views and different View engines

.NET

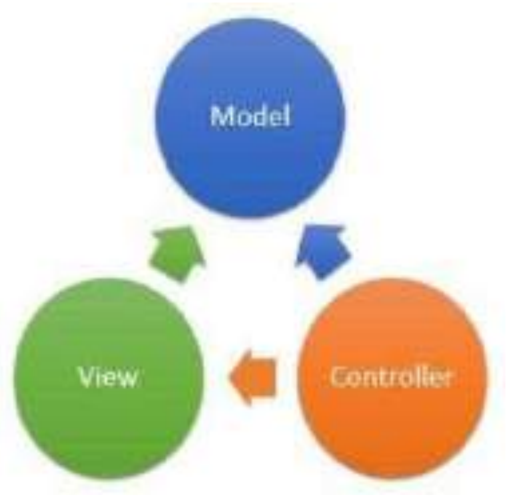
- **NET Framework** – A technology introduced in 2002 which includes the ability to create executables, web applications, and services using C#, Visual Basic, and F#.
- **ASP.NET** – An open-source server-side web application framework which is a subset of Microsoft's .NET framework. Their first iteration of ASP.NET included a technology called Web Forms.
- **ASP.NET WebForms** – (2002 – current) A proprietary technique developed by Microsoft to manage state and form data across multiple pages.
- **ASP.NET MVC** - Microsoft's framework for developing fast web applications using their .NET platform with either the C# or VB.NET language.

MVC application

What is MVC?

MVC is an architectural pattern that separates an application into three major parts called the Model, the View and the Controller.

•**Models**: Classes representing data of the application and that use validation logic to enforce business rules for that data.
•**Views**: Views are the components that display the application's user interface (UI).
•**Controllers**: Classes that handle browser requests, retrieve model data, and specify view to return response to the browser.



MVC

Characteristics:

- An alternative to ASP .NET Web Forms
- Presentation framework
  - Lightweight
  - Highly testable
- Integrated with the existing ASP .NET features:
  - Master pages
  - Membership-Based Authentication

Advantages:

- Easier to manage complexity (divide and conquer)
- It does not use server forms and view state
- Front Controller pattern (rich routing)
- Better support for test-driven development
- Ideal for distributed and large teams
- High degree of control over the application behavior

Features:

- Separation of application tasks
- Support for test-driven development
- Extensible and pluggable framework

Advantages of MVC

ASP.NET MVC has a separation of concerns: Separation of concerns means that your business logic is not contained in a View or controller. The business logic should be found in the models of your application. This makes web development even easier because it allows you to focus on integrating your business rules into reusable models.

ASP.NET MVC provides testability out of the box: Another selling point is that ASP.NET MVC allows you to test every single one of your components, thereby making your code almost bulletproof. The more unit tests you provide for your application, the more durable your application will become.

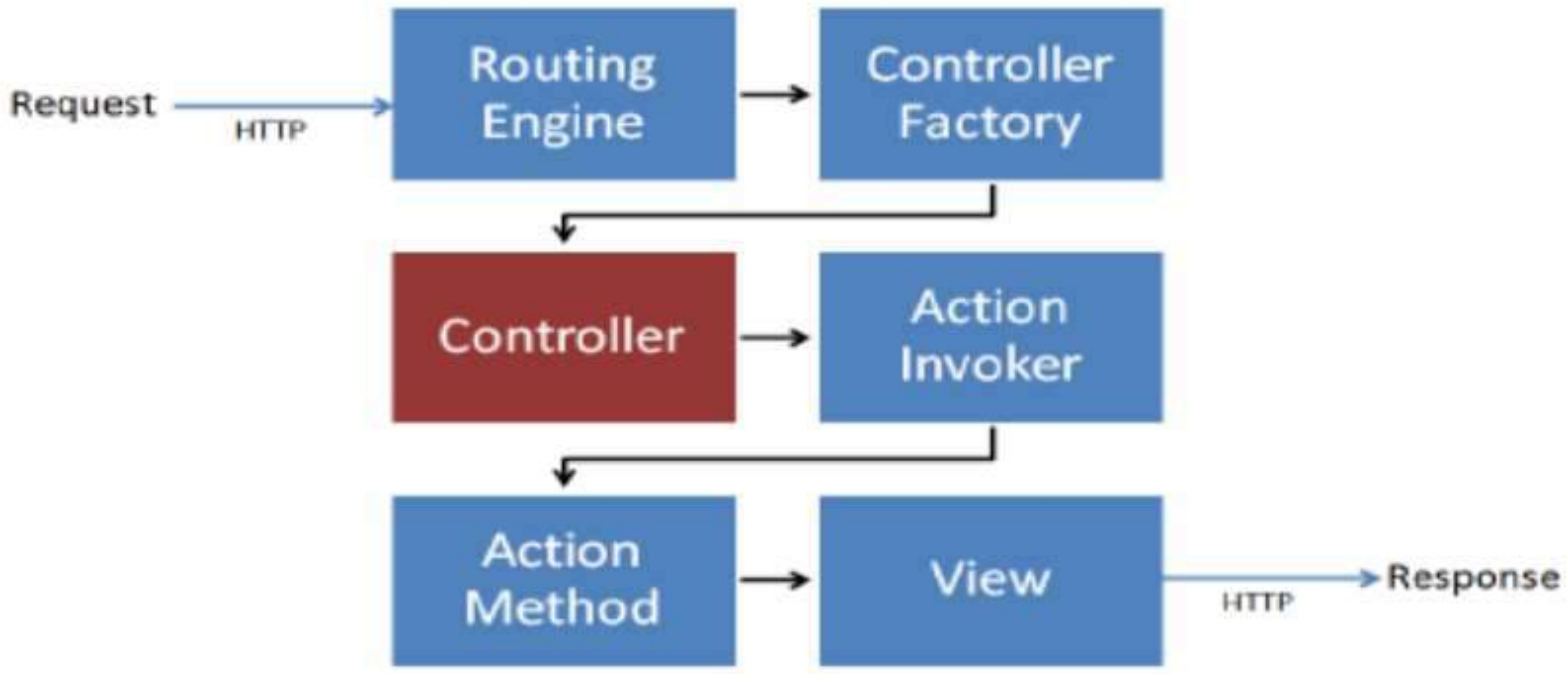
ASP.NET MVC has a smaller "View" footprint: With WebForms, there is a server variable called ViewState that tracks all of the controls on a page. If you have a ton of controls on your WebForm, the ViewState can grow to become an issue. ASP.NET MVC doesn't have a ViewState, thus making the View lean and mean.

ASP.NET MVC has more control over HTML: Since server-side controls aren't used, the View can be as small as you want it to be. It provides a better granularity and control over how you want your pages rendered in the browser.

Differences between Traditional Web Application and MVC Application

- MVC applications are testable, maintainable and easier to update rather than traditional ones.
- Complexities are easily manageable in MVC applications due to separation of concerns.
- Enables full control over the rendered HTML.
- Better accessibility for implementing compliance with Web standards.
- Facilitates adding more interactivity and responsiveness.

MVC application lifecycle



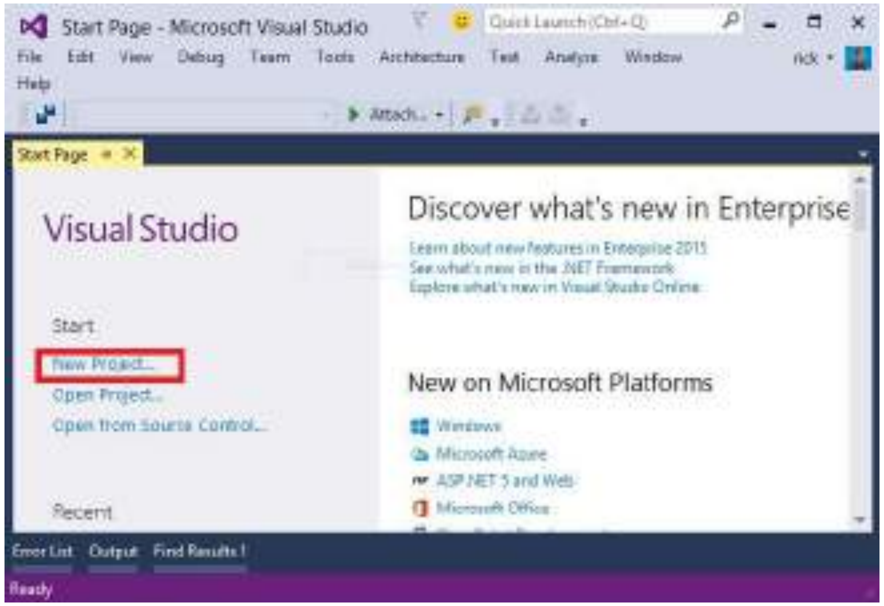
The entry point for every MVC application begins with routing. After the ASP.NET platform has received a request, it figures out how it should be handled through the URL Routing Module. Modules are .NET components that can hook into the application life cycle and add functionality. The routing module is responsible for matching the incoming URL to routes that we define in our application.

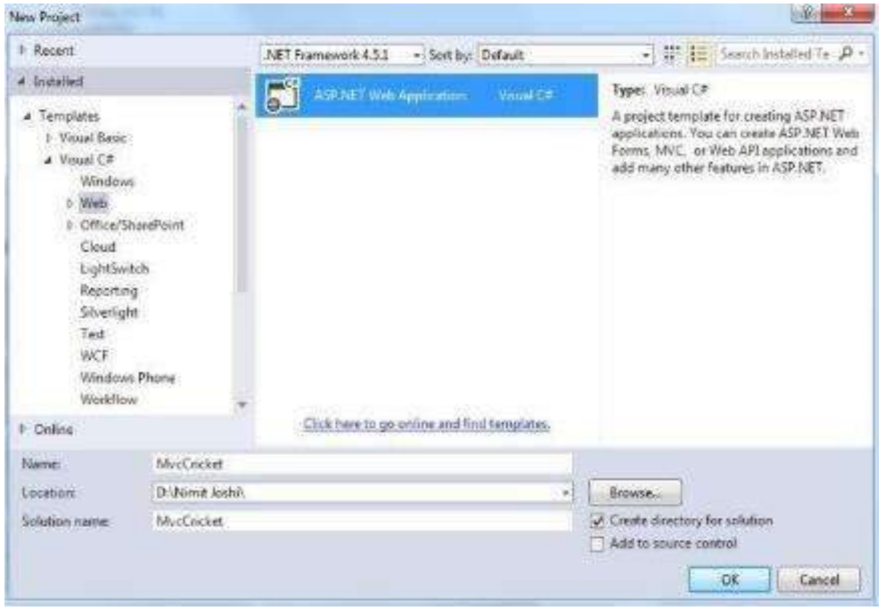
All routes have an associated route handler with them, and this is the entry point to the MVC framework.

Creating MVC application

Step 1: Open Visual Studio

Step 2: Click on New Project to create a web application


Start Page - Microsoft Visual Studio
Quick Launch (Ctrl+Q)
File Edit View Debug Team Tools Architecture Test Analyze Window
Help
Attach...
Start Page
Visual Studio
Discover what's new in Enterprise
Learn about new features in Enterprise 2015
See what's new in the .NET Framework
Explore what's new in Visual Studio Online
Start
New Project...
Open Project...
Open from Source Control...
New on Microsoft Platforms
Windows
Microsoft Azure
ASP.NET 5 and Web
Microsoft Office
Recent
Error List Output Find Results 1
Ready



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.

Name: MvcCricket

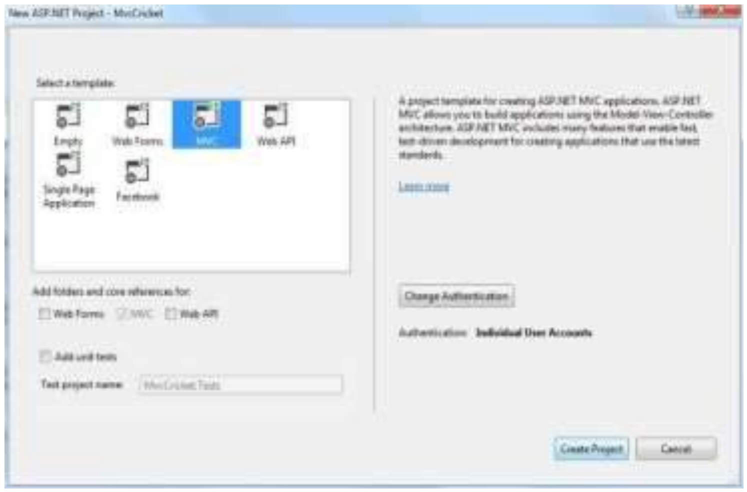
Location: D:\Nimit Joshi\

Solution name: MvcCricket

[✓] Create directory for solution

[ ] Add to source control

Step 3: Select the MVC Template


New ASP.NET Project - MvcCricket

Select a template:

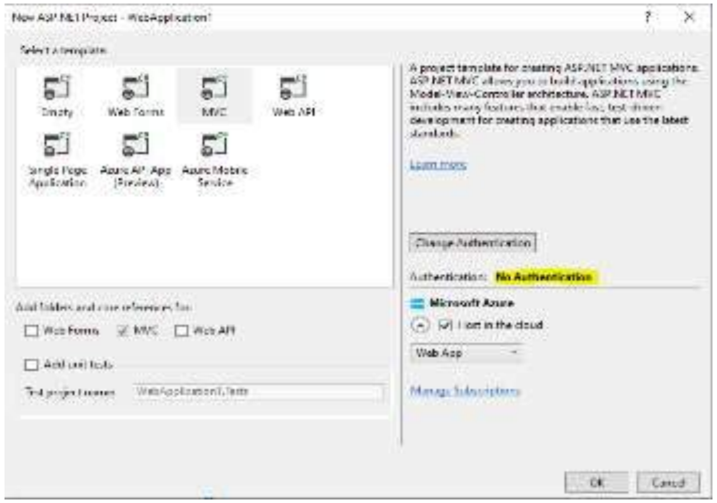
A project template for creating ASP.NET MVC applications. ASP.NET MVC allows you to build applications using the Model-View-Controller architecture. ASP.NET MVC includes many features that enable fast, test-driven development for creating applications that use the latest standards.

Add folders and core references for: Web Forms, MVC (✓), Web API

Authentication: **Individual User Accounts**

Test project name: MvcCricket.Tests

Step 4: Change authentication


New ASP.NET Project - WebApplication1

Select a template:

A project template for creating ASP.NET MVC applications. ASP.NET MVC allows you to build applications using the Model-View-Controller architecture. ASP.NET MVC includes many features that enable fast, test-driven development for creating applications that use the latest standards.

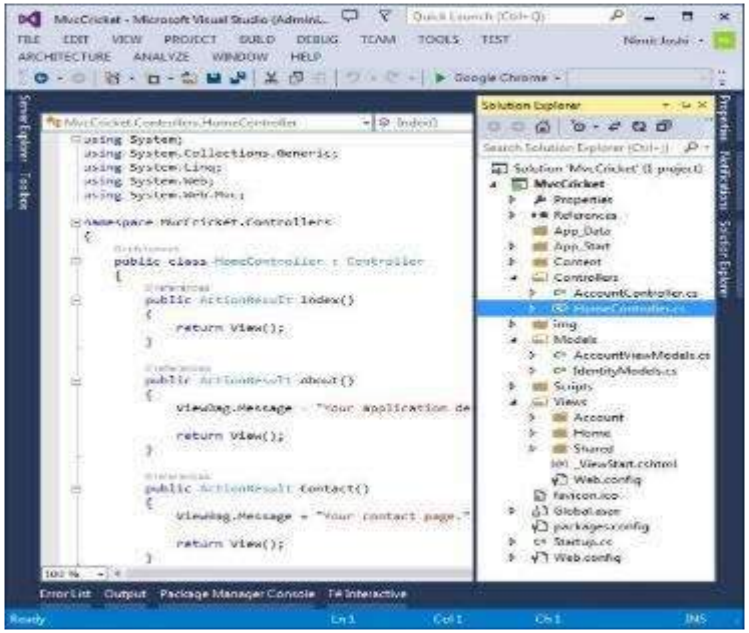
Authentication: **No Authentication**

Add folders and core references for: Web Forms, MVC (✓), Web API

Microsoft Azure: Host in the cloud (✓), Web App

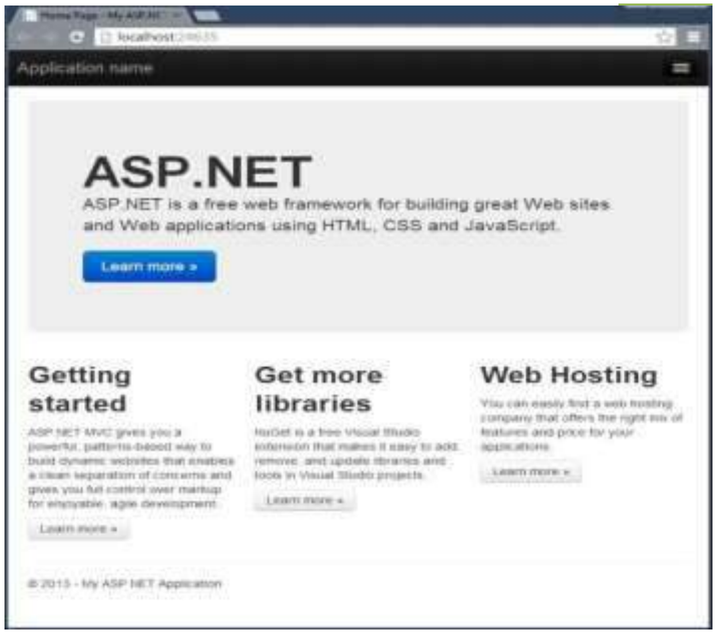
Test project name: WebApplication1.Tests

You are ready to start after clicking onto **OK**


MvcCricket - Microsoft Visual Studio (Administrator)

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

namespace MvcCricket.Controllers
{
    public class HomeController : Controller
    {
        public ActionResult Index()
        {
            return View();
        }

        public ActionResult About()
        {
            ViewBag.Message = "Your application de

            return View();
        }

        public ActionResult Contact()
        {
            ViewBag.Message = "Your contact page."

            return View();
        }
```

Solution 'MvcCricket' (1 project)
- MvcCricket
  - Properties
  - References
  - App_Data
  - App_Start
  - Content
  - Controllers
    - AccountController.cs
    - HomeController.cs
  - img
  - Models
    - AccountViewModels.cs
    - IdentityModels.cs
  - Scripts
  - Views
    - Account
    - Home
    - Shared
    - _ViewStart.cshtml
    - Web.config
  - favicon.ico
  - Global.asax
  - packages.config
  - Startup.cs
  - Web.config


Home Page - My ASP.NET

localhost

Application name

ASP.NET

ASP.NET is a free web framework for building great Web sites and Web applications using HTML, CSS and JavaScript.

Getting started

ASP.NET MVC gives you a powerful, patterns-based way to build dynamic websites that enables a clean separation of concerns and gives you full control over markup for enjoyable, agile development.

Get more libraries

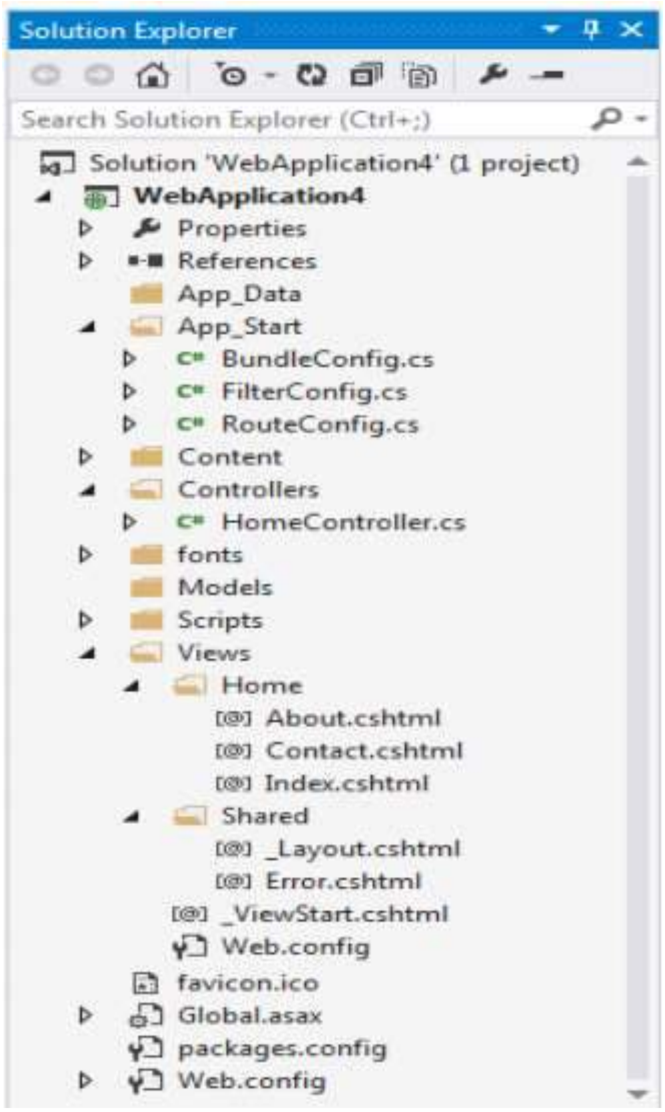
NuGet is a free Visual Studio extension that makes it easy to add, remove, and update libraries and tools in Visual Studio projects.

Web Hosting

You can easily find a web hosting company that offers the right mix of features and price for your applications.

© 2015 - My ASP.NET Application

Layout of MVC structure


```
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'WebApplication4' (1 project)
  WebApplication4
    Properties
    References
    App_Data
    App_Start
      BundleConfig.cs
      FilterConfig.cs
      RouteConfig.cs
    Content
    Controllers
      HomeController.cs
    fonts
    Models
    Scripts
    Views
      Home
        About.cshtml
        Contact.cshtml
        Index.cshtml
      Shared
        _Layout.cshtml
        Error.cshtml
      _ViewStart.cshtml
      Web.config
    favicon.ico
    Global.asax
    packages.config
    Web.config
```

• **App_Data** – While I don't use this folder often, it's meant to hold data for your application (just as the name says). A couple of examples would include a portable database (like SQL Server Compact Edition) or any kind of data files (XML, JSON, etc.). I prefer to use SQL Server.

• **App_Start** – The App_Start folder contains the initialization and configuration of different features of your application.

- **BundleConfig.cs** – This contains all of the configuration for minifying and compressing your JavaScript and CSS files into one file.
- **FilterConfig.cs** – Registers Global Filters.
- **RouteConfig.cs** – Configuration of your routes.

There are other xxxxConfig.cs files that are added when you apply other MVCrelated technologies (for example, WebAPI adds WebApiConfig.cs).

Layout of MVC structure

- **Controllers** – The controllers folder is where we place the controllers.
- **Models** – This folder contains your business models. It's better when you have these models in another project, but for demo purposes, we'll place them in here.
- **Scripts** – This is where your JavaScript scripts reside.
- **Views** – This parent folder contains all of your HTML "Views" with each controller name as a folder. Each folder will contain a number of cshtml files relating to the methods in that folder's controller.
- **Views/Shared** – The Shared folder is meant for any shared cshtml files you need across the website.
- **Global.asax** – The Global.asax is meant for the initialization of the web application. If you look inside the Global.asax, you'll notice that this is where the RouteConfig.cs, BundleConfig.cs, and FilterConfig.cs are called when the application runs.
- **Web.Config** – The web.config is where you place configuration settings for your application. For new MVC developers, it's good to know that you can place settings inside the <appsettings> tag and place connection strings inside the <connectionstring> tag.

Routing Configuration

Routing is the process of directing an HTTP request to a controller and the functionality of this processing is implemented in **System.Web.Routing**. The **Global.asax** file is where you will define the route for your application.

```
namespace  MVCFirstApp
{
        public  class  MvcApplication  :  System.Web.HttpApplication
        {
                protected  void  Application_Start()
                {
                AreaRegistration.RegisterAllAreas();
                RouteConfig.RegisterRoutes(RouteTable.Routes);
                }
        }
}
```

Following is the implementation of **RouteConfig** class, which contains one method **RegisterRoutes**. Define routes to map URLs to a specific controller action. ASP.NET MVC application uses routing rules defined in **Global.asax** to find out the appropriate Controller and pass the request.

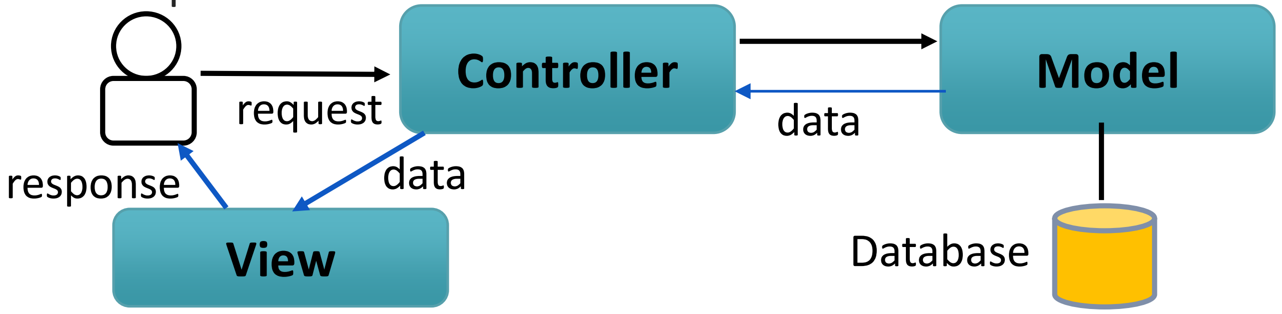
```
namespace MVCFirstApp
{
    public class RouteConfig
    {
        public static void RegisterRoutes(RouteCollection routes)
        {
            routes.IgnoreRoute("{resource}.axd/{*pathInfo}");
            routes.MapRoute(
            name: "Default",
            url: "{controller}/{action}/{id}",
            defaults: new { controller = "Home", action = "Index", id =
            UrlParameter.Optional }
            );
        }
    }
}
```

Controllers and Action Methods

Controllers

In ASP.NET MVC, a Controller defines and groups a set of actions. The activities performed by the controller are given below:

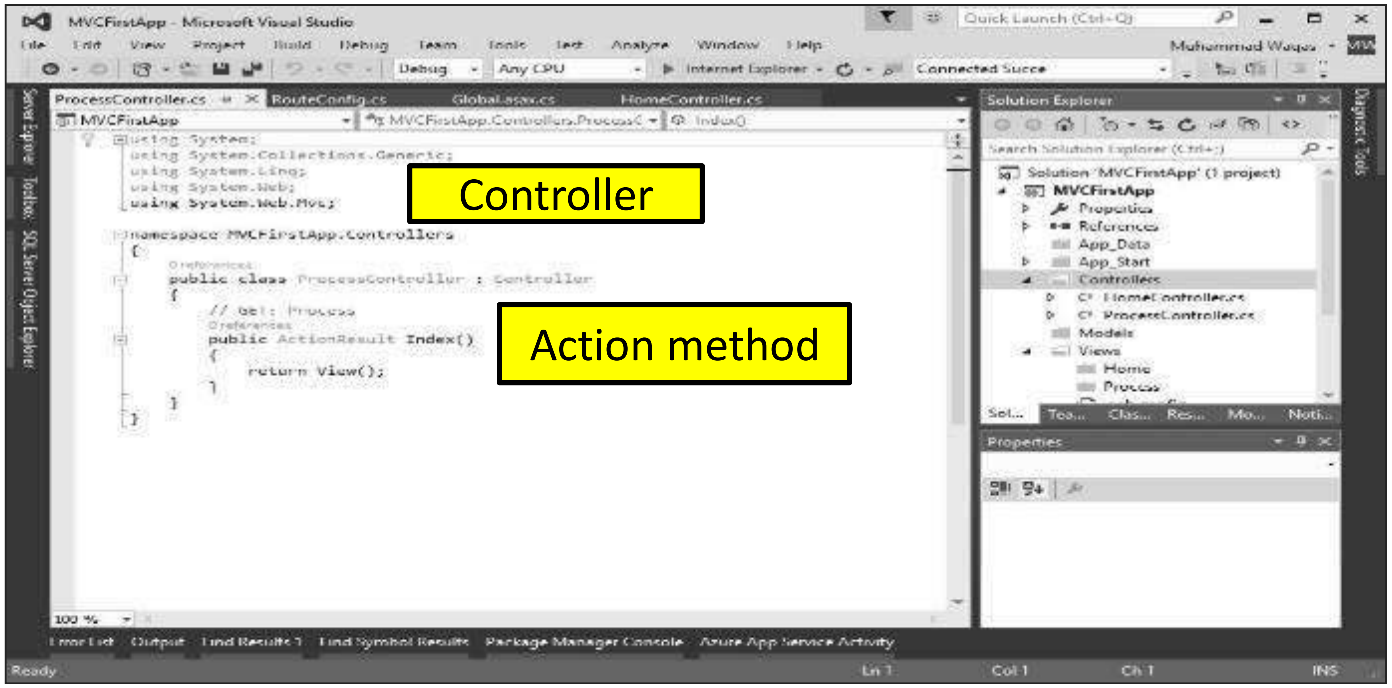
- Receives request sent by the client.
- Calls necessary Model that will interact with database and fetch data back to Controller.
- Controller further calls required View and pass the data which is further rendered to client as a response of the request.



Action Methods

- Action Methods are public methods inside a controller class.
- A Controller class can have multiple action methods that are responsible for performing certain operations depending upon the client action.
- Actions can return anything but often returns an instance of IActionResult that produces a response.

ProcessController has only one default Action Method called **Index**



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

namespace MVCFirstApp.Controllers
{
    public class ProcessController : Controller
    {
        // GET: Process
        public ActionResult Index()
        {
            return View();
        }
    }
}
```

Different response types

Action Result	Helper Method	Short Description
ViewResult	View	It renders a view as a web page
PartialViewResult	PartialView	It renders a PartialView.
RedirectToRouteResult	RedirectToAction or RedirectToRoute	It redirects to another action method.
JavaScriptResult	JavaScript	It returns a script that can be executed on client.
FileResult	File	It returns a binary output to for response.
ContentResult	Content	To return user-defined content type.

View & different view engines

Views

•Responsible for providing the user interface to the user.

•The view transforms a model into a format ready to be presented to the user.

•The view examines the model object and transforms the contents to HTML

•A parent folder contains all HTML "Views" with each controller name as a folder. Each folder will contain several cshtml files relating to the methods in that folder's controller. Example: http://www.xyzcompany.com/Products/List we have a Products folder with a **List.cshtml** file. In the Controllers folder, open the **ProductsController.cs** and look for the List method.

•**Views/Shared** – Meant for any shared cshtml files needed across the website.

View Engines

- Views created using Razor view engine.
- This engine works between view and browser to provide HTML to browser.
- What is Razor?
  - Introduced in ASP.NET MVC 3 and is the default view engine moving forward
  - Provides a clean, lightweight, simple view engine.
  - Provides a streamlined syntax for expressing views
  - Minimizes the amount of syntax and extra characters.
- Razor View Engine has .cshtml (with C#) and .vbhtml (with VB) extension for views.

Different view engines

There are many types of view engines and these are:

- ASPX
- Razor
- Spark
- NHaml
- NDJango
- Hasic
- Brail

Books

1. Beginning ASP.NET 4: in C# and VB; Imar Spaanjaars,2010
2. Beginning ASP.NET 3.5 in C# 2008: From Novice to Professional; 2nd edition, Matthew MacDonald,2007
3. ASP.NET 3.5 Unleashed by Stephen Walther, 2008
4. Pro ASP.NET 3.5 in C# 2008: Includes Silverlight 2 by Matthew MacDonald,2008
5. ASP.NET 3.5 For Dummies by Ken Cox,2008

References

1. ASP.NET; URL: https://dotnet.microsoft.com/apps/aspnet
2. URL: https://www. tutorialspoint.com/index.htm
3. URL: http://www.webdevelopmenthelp.net/2014
4. View Engines; URL: https://www.tutorialride.com/asp-net-mvc

Thank you!

Introduction to ASP.Net MVC

Course Code: CSC 4164 Course Title: ADVANCED PROGRAMMING WITH .NET

Dept. of Computer Science
Faculty of Science and Technology

Lecture No:	**04**	**Week No:**	**02**	**Semester:**	**Fall 2020-21**
Lecturer:	*Victor Stany Rozario, stany@aiub.edu*				

Lecture Outline

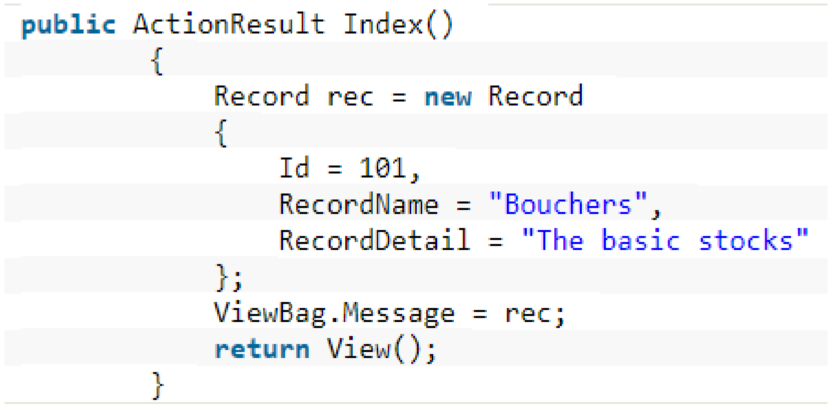
- Controller finding a view
- Passing data from Controller to View
- Models
- Strongly typed views

Controller finding a view

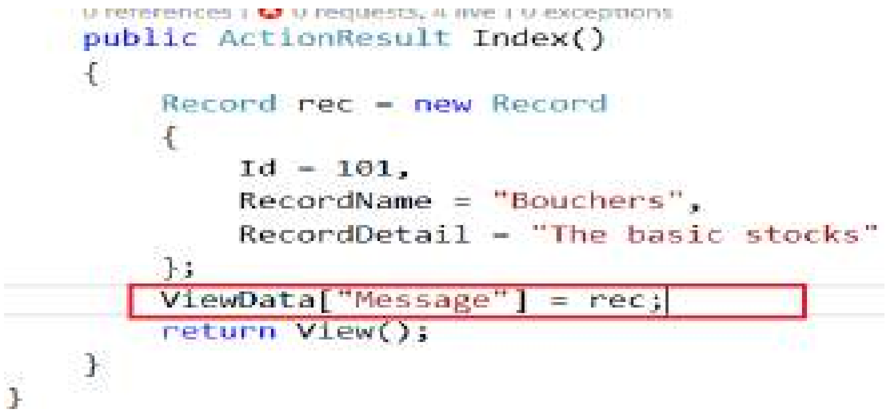
- The URL tells the routing mechanism which controller class to instantiate and which action method to call and supplies the required arguments to that method.
- The controller's method then decides which view to use, and that view then renders the HTML.
- The Views directory contains a folder for your controller, with the same name as the controller.
- Within controller folder there's a view file for each action method, named the same as the action method.
- This is how views are associated to an action method.

Passing data from Controller to View

- **ViewBag**: Is a type object and a dynamic property under the controller base class.
  - o It is ideally temporary data.
  - o It doesn't have typecasting and null checks.
- **ViewData**: Is a dictionary which can contain key-value pairs where each key must be string. Requires typecasting as well as null checks.

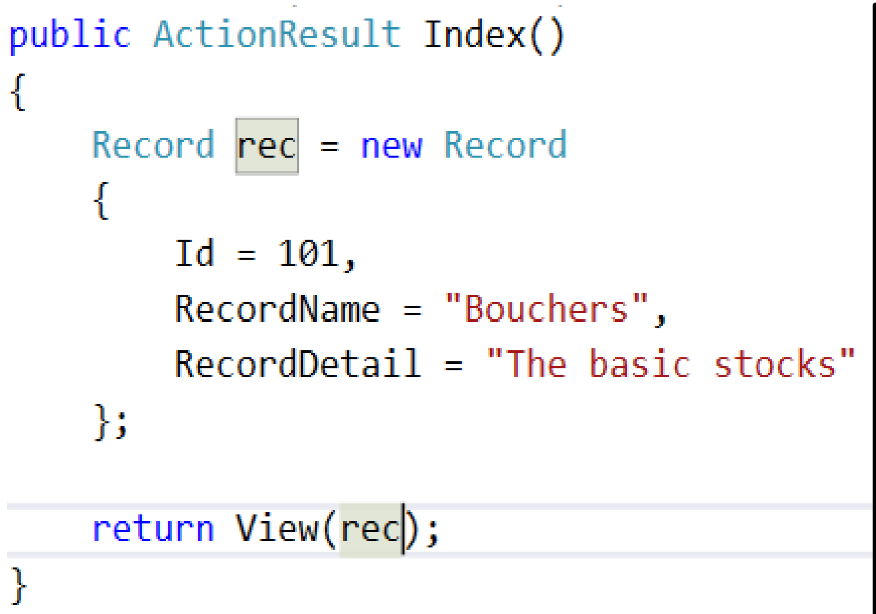


```
public ActionResult Index()
        {
            Record rec = new Record
            {
                Id = 101,
                RecordName = "Bouchers",
                RecordDetail = "The basic stocks"
            };
            ViewBag.Message = rec;
            return View();
        }
```



```
0 references | 0 requests, 4 live | 0 exceptions
public ActionResult Index()
{
    Record rec = new Record
    {
        Id = 101,
        RecordName = "Bouchers",
        RecordDetail = "The basic stocks"
    };
    ViewData["Message"] = rec;
    return View();
}
}
```

- Passing an **object** of the model class to the View.
- **TempData**: Stays for a subsequent HTTP Request as opposed to other options (ViewBag and ViewData) those stay only for current request.
  - Maintain data between controller actions as well as redirects.
  - Typecasting and null checks required in order to avoid errors.



```
public ActionResult Index()
{
    Record rec = new Record
    {
        Id = 101,
        RecordName = "Bouchers",
        RecordDetail = "The basic stocks"
    };

    return View(rec);
}
```

```
public ActionResult TemporaryEmployee()
{
        Employee employee = new Employee
        {
        EmpID = “121”,
        EmpFirstName = “John”,
        EmpLastName = “Nguyen”
        };
        TempData[“Employee”] = employee;
        return
        RedirectToAction(“PermanentEmployee”);
}
```

Models

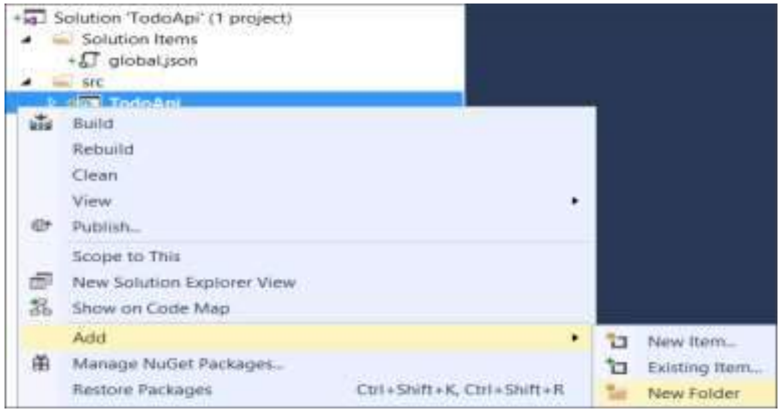
What is Model?

- Business/domain logic
- Model objects, retrieve and store model state in a persistent storage (database).

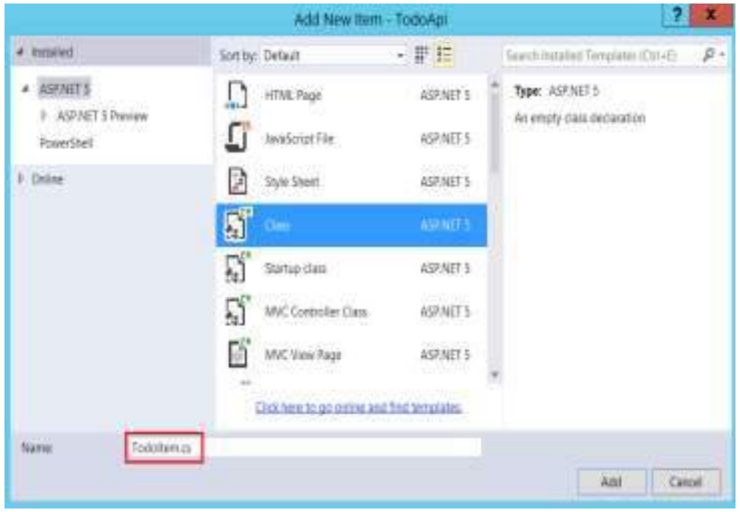
```
public class Customer{
         public string FirstName { get; set; }
        public string LastName { get; set; }
        public string Company { get; set; }
        public IEnumerable Orders { get; set; }
        }
```

Creating Models

Add a folder named “**Models**”. In Solution Explorer, right-click the project. Select Add → New Folder. Name the folder Models.



Right-click the **Models** folder and select Add → New Item. In Add New Item, select the Class. Name the class **TodoItem** and click **OK**.



```
publicclassTodoItem
{
public string Key { get; set; }
public string Name { get; set; }
public bool IsComplete { get; set; }
}
```

Strongly typed views

- The view which binds to a specific type of **View Model** by passing the model object as a parameter to the View() method is called as Strongly Typed View.

```
public class HomeController : Controller
   {
       public ActionResult Index()
       {
           EmployeeBusinessLayer employeeBL = new EmployeeBusinessLayer();
           Employee employee = employeeBL.GetEmployeeDetails(101);
           ViewBag.Header = "Employee Details";

           return View(employee);
       }
   }
```

Changes in **Index.cshtml** View:

```
@model FirstMVCDemo.Models.Employee
<!DOCTYPE html>
<html>
<head>
  <meta name="viewport"
content="width=device-width" />
  <title>Page Title</title>
</head>
<body>
  <h2>@ViewBag.Header</h2>
  <table style="font-family:Arial">
    <tr>
      <td>Employee ID:</td>
      <td>@Model.EmployeeId </td>
    </tr>
    <tr>
      <td>Name:</td>
      <td>@Model.Name</td>
    </tr>  </table>
</body>
</html>
```

Advantages of Strongly-typed views:

- Strongly Typed View in ASP.NET MVC provides compile-time error checking as well as intelligence support.
- If we misspell the property name, then it comes to know at compile time rather than at runtime.

Books

1. Beginning ASP.NET 4: in C# and VB; Imar Spaanjaars,2010
2. Beginning ASP.NET 3.5 in C# 2008: From Novice to Professional; 2nd edition, Matthew MacDonald,2007
3. ASP.NET 3.5 Unleashed by Stephen Walther, 2008
4. Pro ASP.NET 3.5 in C# 2008: Includes Silverlight 2 by Matthew MacDonald,2008
5. ASP.NET 3.5 For Dummies by Ken Cox,2008

References

1. ASP.NET; URL: https://dotnet.microsoft.com/apps/aspnet
2. URL: https://www. tutorialspoint.com/index.htm
3. URL: http://www.webdevelopmenthelp.net/2014
4. View Engines; URL: https://www.tutorialride.com/asp-net-mvc
5. URL: https://www.tutorialsteacher.com/mvc
6. Strongly typed views; URL: https://dotnettutorials.net/lesson

Thank you!

Introduction to ASP.Net MVC

Course Code: CSC 4164 Course Title: ADVANCED PROGRAMMING WITH .NET

Dept. of Computer Science
Faculty of Science and Technology

Lecture No:	**05**	**Week No:**	**03**	**Semester:**	**Fall 2020-21**
Lecturer:	*Victor Stany Rozario, stany@aiub.edu*				

Lecture Outline

- Methods of receiving form data.
- Mapping action method with GET and POST request.
- Using annotations.
- Form data validation using annotation (metadata) in models.
- Session Management

Methods of receiving form data

Request object

- Traditional approach
- Uses request object of the **HttpRequestBase** class. Object contains input field name and values as name-value pairs in case of the form submit.
- Get the values of the controls by their names using as indexer from the request object in the controller.

```
string strName = Request["txtName"].ToString();
```

Here ‘txtName’ is the name of the input in the form. Therefore, its values can be retrieved in controller from request object like shown above.

Form collection

FormCollection object also has requested data as the name/value collection as the **Request** object.

```
[HttpPost]
public ActionResult Calculate(FormCollection form){
        string strName = form["txtName"].ToString();
        . . . . . . . . . . . . . . . . . . . . . }
```

Through Parameters

Pass the input field names as parameters to the post action method by keeping the names same as the input field names. These parameters will have the values for those fields and the parameter types should be string. Also, there is no need to define the parameters in any specific sequence.

```
[HttpPost]
public ActionResult Calculate(FormCollection form){
        public ActionResult Calculate(string txtName)
       {
        string strName = Convert.ToString(txtName);
        . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
       }
```

Strongly typed model

Strongly typed model binding to view: Here, we need to create a strongly typed view which will bind directly the model data to the various fields of the page.

I. Create a model with the required member variables.
   Let's say we have a model named 'Person' with member variable named as 'Name’

II. Now pass the empty model to the view as parameter in the controller action.

```
public ActionResult GetName()
{
 Person person = new Person();
 return View(person);
}
```

Strongly typed model

III. Prepare the strongly typed view to display the model property values through html elements as below:

```
<div><%= Html.Encode(person.Name)%></div>
```

IV. Create the action method that handles the POST request & processes the data.

```
[HttpPost]
public ActionResult GetPersonName(Person person)
{
        return Content(person.Name.ToString());
}
```

Mapping action method with GET and POST request

To receive and process submitted form data, add a Form action method in order to create the following:

• A method that responds to **HTTP GET** requests: A GET request is what a browser issues normally each time someone clicks a link. This version of the action will be responsible for displaying the initial blank form when someone first visits /Home/Form.

•A method that responds to **HTTP POST** requests: By default, forms rendered using Html.BeginForm() are submitted by the browser as a POST request. This version of the action will be responsible for receiving submitted data and deciding what to do with it.

Handing GET and POST requests in separate C# methods helps to keep my controller code tidy, since the two methods have different responsibilities. Both action methods are invoked by the same URL, but MVC makes sure that the appropriate method is called, based on whether I am dealing with a GET or POST request.

HTTP GET & POST

- HTTP GET method- Add the HttpGet attribute to existing Form action method. This tells MVC that this method should be used only for GET requests.
- HTTP POST method- Add an overloaded version of Form, which takes a GuestResponse parameter and applies the HttpPost attribute. The attribute tells MVC that the new method will deal with POST requests. Also import the Models namespace—this is just so to refer to the **GuestResponse** model type without needing to qualify the class name.

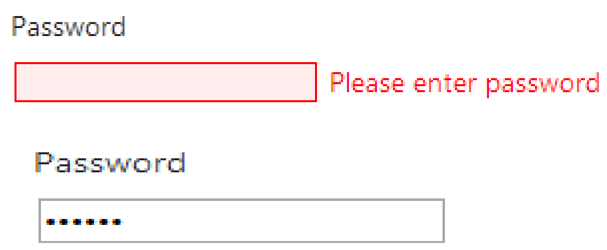
```
[HttpGet]
public ViewResult Form()
{
        return View();
}
```

```
[HttpPost]
 public ViewResult Form(GuestResponse response)
{
        // TODO
        return View("Thanks", guestResponse);
 }
```

Using annotations

In asp.net mvc data **Annotations** attribute is a simple rule that can be applied to Model to validate Model data.
The Data Annotations validation attributes are kind of rules to apply to our model class fields and need to provide our error message .It will validate input one the user gives the input and provide error message according to it. Some common Data Annotations validation are required fields like checking fields Range, StringLength etc.

```
[DataType(DataType.Password)]

[Required(ErrorMessage = "Please enter password")]

public string Password { get; set; }

[Required(ErrorMessage = "Please enter ConfirmPassword")]
```


Password

Please enter password

Password

•••••••

Some examples

```
[DisplayFormat(DataFormatString = "{0:dd.MM.yyyy}")]
[Required(ErrorMessage = "Please enter Date of Birth")]
public DateTime StudentDOB { get; set; }
[Range(5000, 15000, ErrorMessage = "Please enter valid range")]
[Required(ErrorMessage = "Please enter Student Fees")]
public decimal StudentFees { get; set; }
[Required(ErrorMessage = "Please enter Student Address")]
[StringLength(50, ErrorMessage = "Only 50 character are allowed")]
public string StudentAddress { get; set; }
[DataType(DataType.Password)]
[Required(ErrorMessage = "Please enter password")]
public string Password { get; set; }
[Required(ErrorMessage = "Please enter ConfirmPassword")]
[DataType(DataType.Password)]
[Compare("Password", ErrorMessage = "Password not matching")]
public string ConfirmPassword { get; set; }
```

Attribute	Description
Required	Indicates that the property is a required field
StringLength	Defines a maximum length for string field
Range	Defines a maximum and minimum value for a numeric field
RegularExpression	Specifies that the field value must match with specified Regular Expression
CreditCard	Specifies that the specified field is a credit card number
CustomValidation	Specified custom validation method to validate the field
EmailAddress	Validates with email address format
FileExtension	Validates with file extension
MaxLength	Specifies maximum length for a string field
MinLength	Specifies minimum length for a string field
Phone	Specifies that the field is a phone number using regular expression for phone numbers

Form data validation using annotation (metadata) in models

ASP.NET MVC uses DataAnnotations attributes to implement validations. DataAnnotations includes built-in validation attributes for different validation rules, which can be applied to the properties of model class. ASP.NET MVC framework will automatically enforce these validation rules and display validation messages in the view. Following will show the steps of form validation:

Step 1: Apply DataAnnotation attribute on the properties of the model class.
Step 2: Create the GET and POST Edit Action method in the same as previous section. The GET action method will render Edit view to edit the selected object and the POST Edit method will save edited one.
In the POST Edit method, we first check if the ModelState is valid or not. If ModelState is valid then update the information into database, if not then return Edit view again with the same data.

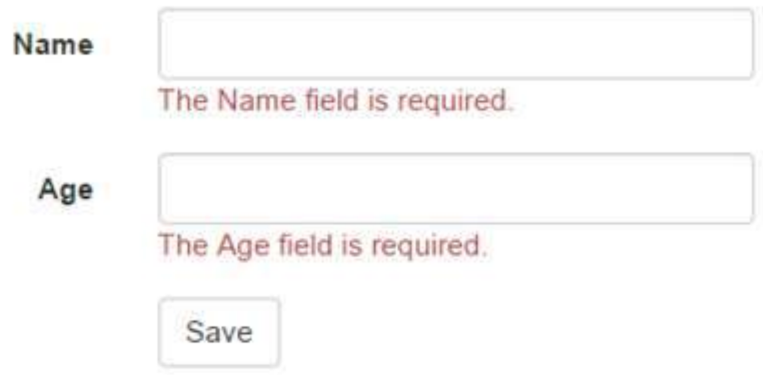
Examples of Step 1 & 2:

```
public class Student
{
   [Required]
   public string Name { get; set; }

   [Range(5,50)]
   public int Age { get; set; }
}
```

```
public class StudentController : Controller
   {
      public ActionResult Edit(int id)
      {
         var std = studentList.Where(s => s.StudentId == 
StudentId).FirstOrDefault();
         return View(std);
      }
      [HttpPost]
      public ActionResult Edit(Student std)
      {
         if (ModelState.IsValid) {
            //write code to update student
            return RedirectToAction("Index");
         }
        return View(std);
      }
```

Step 3: Create an Edit view. Generating Edit view under View/**Class** folder. Edit.cshtml will be generated as shown below.
In the Edit.cshtml, it calls Html Helper method **ValidationMessageFor** for every field and **ValidationSummary** method at the top. ValidationMessageFor is responsible to display error message for the specified field. ValidationSummary displays a list of all the error messages at once.

So now, it will display default validation message when you submit an Edit form without entering the required information. The image below shows the error generated for the previous Student example shown.


Name

The Name field is required.

Age

The Age field is required.

```
@Html.ValidationSummary(true, "", new { @class = "text-danger" })
    @Html.HiddenFor(model => model.StudentId)
    <div class="form-group">
      @Html.LabelFor(model => model.StudentName, htmlAttributes: new {
@class = "control-label col-md-2" })
      <div class="col-md-10">
        @Html.EditorFor(model => model.StudentName, new { htmlAttributes
= new { @class = "form-control" } })
        @Html.ValidationMessageFor(model => model.StudentName, "", new {
@class = "text-danger" })
      </div>
    </div>

    <div class="form-group">
      @Html.LabelFor(model => model.Age, htmlAttributes: new { @class =
"control-label col-md-2" })
      <div class="col-md-10">
        @Html.EditorFor(model => model.Age, new { htmlAttributes = new {
@class = "form-control" } })
        @Html.ValidationMessageFor(model => model.Age, "", new { @class =
"text-danger" })
      </div>
    </div>
```

The cshtml page for the previous example.

Session Management

The HTTP protocol on which all web applications work is a stateless protocol. Here stateless means that information is not retained from one request to another.
For example, if you had a login page which has 3 textboxes, one for the name and the other for the password and email. When you click the Login button on that page, the application needs to ensure that the username, email and password get passed onto the next page. In ASP.Net, this is done in a variety of ways. The first way is via a concept called ViewState. Here ASP.Net automatically stores the contents of all the controls. It also ensures this is passed onto the next page. This is done via a property called the **ViewState**.
The other way is to use an object called a "**Session Object**." The Session object is available throughout the lifecycle of the application. You can store any number of key-value pairs in the Session object. On any page, you can store a value in the Session object via the below line of code.
Session["Key"]=value

Session Management

This stores the value in a Session object and the 'key' part is used to give the value a name. This allows the value to be retrieved at a later point in time. To retrieve a value, you can simply issue the below statement “**Session["Key“]**”

Storing in the session object

```
Session[“UserName"]=txtUName.Text;
Response.Write(Session[“UserName“]);
```

Retrieving from the session object

Books

1. Beginning ASP.NET 4: in C# and VB; Imar Spaanjaars,2010
2. Beginning ASP.NET 3.5 in C# 2008: From Novice to Professional; 2nd edition, Matthew MacDonald,2007
3. ASP.NET 3.5 Unleashed by Stephen Walther, 2008
4. Pro ASP.NET 3.5 in C# 2008: Includes Silverlight 2 by Matthew MacDonald,2008
5. ASP.NET 3.5 For Dummies by Ken Cox,2008

References

1. ASP.NET; URL: https://www.guru99.com/
2. View Engines; URL: https://www.tutorialride.com/asp-net-mvc
3. URL: https://www.tutorialsteacher.com/mvc
4. Strongly typed views; URL: https://dotnettutorials.net/lesson
5. URL: https://www.tutlane.com/tutorial/aspnet-mv
6. URL: https://www.tutorialsteacher.com/mvc

Thank you!

Introduction to ASP.Net MVC

Course Code: CSC 4164 Course Title: ADVANCED PROGRAMMING WITH .NET

Dept. of Computer Science
Faculty of Science and Technology

Lecture No:	**06**	**Week No:**	**03**	**Semester:**	**Fall 2020-21**
Lecturer:	*Victor Stany Rozario, stany@aiub.edu*				

Lecture Outline

- Use of session in User Authentication
- HTML Helpers

Authorization in ASP.NET

Authorization in MVC is controlled through the **AuthorizeAttribute** attribute and its various parameters. At its simplest, applying the AuthorizeAttribute attribute to a controller or action limits access to the controller or action to any authenticated user.

For example, the following code limits access to the AccountController to any authenticated user.

```
[Authorize]
public class AccountController : Controller
{
   public ActionResult Login()
   {
   }
   public ActionResult Logout()
   {
   }
}
```

If you want to apply authorization to an action rather than the controller, apply the AuthorizeAttribute attribute to the action itself:

```
public class AccountController : Controller
{
  public ActionResult Login()
  {
  }

  [Authorize]
  public ActionResult Logout()
  {
  }
}
```

Now only authenticated users can access the Logout function.

You can also use the AllowAnonymous attribute to allow access by non-authenticated users to individual actions.

This allowS only authenticated users to the AccountController, except for the Login action, which is accessible by everyone, regardless of their authenticated or unauthenticated / anonymous status.

```
[Authorize]
public class AccountController : Controller
{
  [AllowAnonymous]
   public ActionResult Login()
   {………..}
   public ActionResult Logout()
   {………..} }
```

Use of session in User Authentication

Override base class method

ASP.NET has a nice session feature that uses cookies or URL rewriting to associate multiple requests from a user together to form a single browsing session. A related feature is session state, which associates data with a session. Data associated with a session is deleted when a session expires (typically because a user has not made a request for a while).

- Create custom class
- Override base class method in custom class (with appropriate logic)

```
public class CustomizeAuthorize : AuthorizeAttribute
   {
      protected override bool AuthorizeCore(HttpContextBase httpContext)
      {
         if (httpContext == null)
         {
            throw new ArgumentNullException("httpContext");
         }
         IPrincipal user = httpContext.User;
         if (!user.Identity.IsAuthenticated)
         {
            return false;
         }
if ((this.Roles.Length > 0) && (!this.Roles.Contains(ReturnUserRole(user.Identity.Name))))
         {
            return false;
         }
         return true;
      }
```

Create custom class

Override base class method

```
[CustomizeAuthorize(Roles = "Admin")]
```

use the attribute to decorate role name.

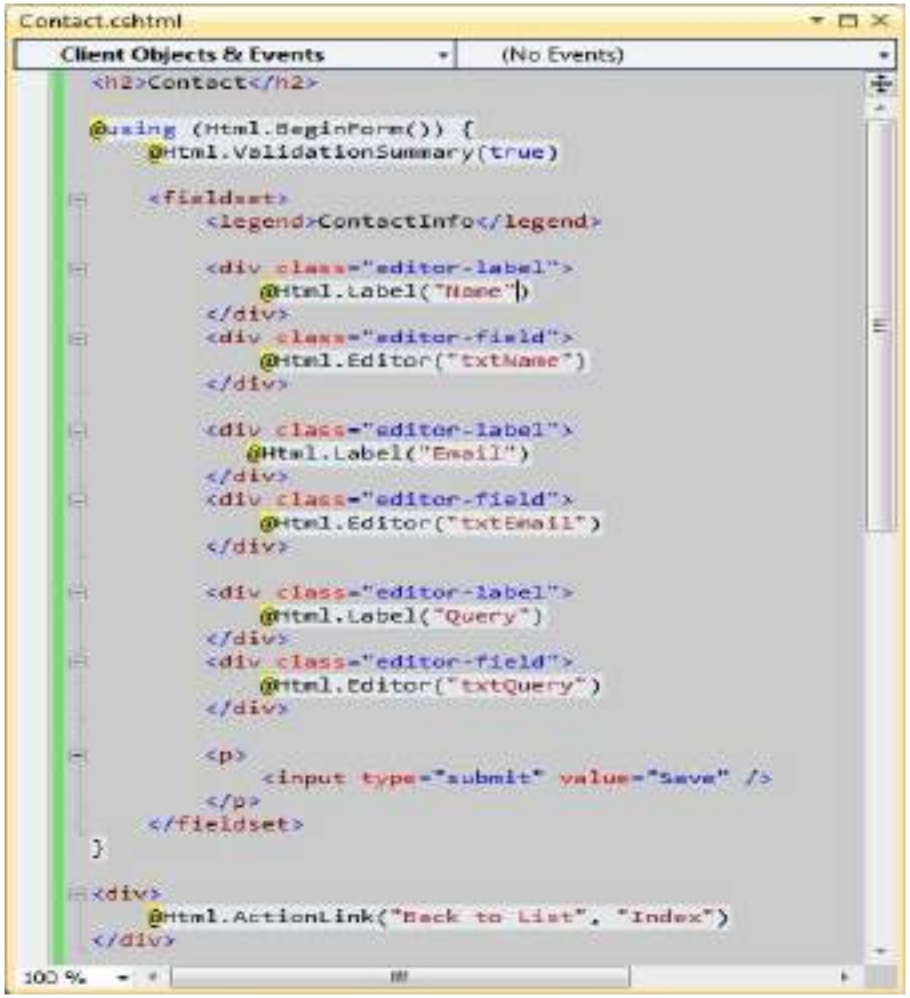
HTML Helpers

People coming from the asp.net web forms background are used to putting the ASP.NET server control on the page using the toolbox. When we work with ASP.NET MVC application there is no toolbox available to us from where we can drag and drop HTML controls on the view. In MVC, if we want to create a view it should contain HTML code for specifying the mark up. MVC Beginners(specially with Web forms background) finds this a little troubling.

ASP.NET MVC team must have anticipated this problem and thus to ease this problem, the ASP.NET MVC framework comes with a set of HTML Helper methods. These helpers are simple functions that let the developer to specify the type of HTML needed on the view. This is done in C#. The final HTML will be generated by these functions at the runtime i.e. We don't have to worry about the correctness of generated HTML.

Built in HTML Helpers

Create a simple contact us form that will ask the user for his name, email id and his query. we can design this form in simple HTML easily, let us see how we can utilize HTML helper to achieve the same.



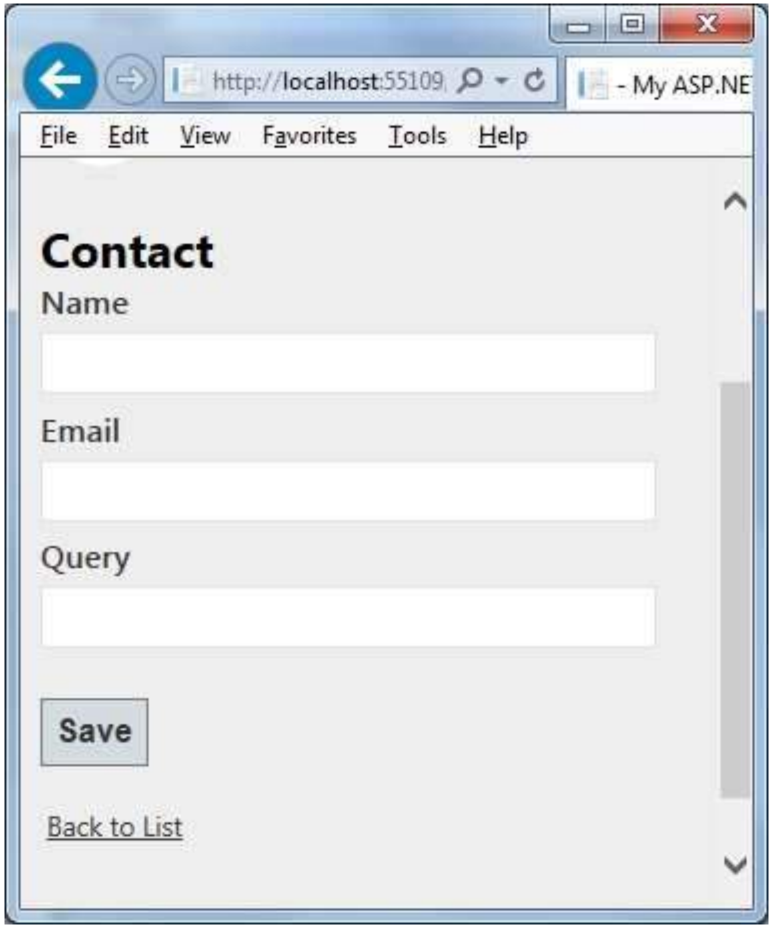
Contact.cshtml

Client Objects & Events | (No Events)

```
<h2>Contact</h2>

@using (Html.BeginForm()) {
    @Html.ValidationSummary(true)

    <fieldset>
        <legend>ContactInfo</legend>

        <div class="editor-label">
            @Html.Label("Name")
        </div>
        <div class="editor-field">
            @Html.Editor("txtName")
        </div>

        <div class="editor-label">
           @Html.Label("Email")
        </div>
        <div class="editor-field">
            @Html.Editor("txtEmail")
        </div>

        <div class="editor-label">
            @Html.Label("Query")
        </div>
        <div class="editor-field">
            @Html.Editor("txtQuery")
        </div>

        <p>
            <input type="submit" value="Save" />
        </p>
    </fieldset>
}

<div>
    @Html.ActionLink("Back to List", "Index")
</div>
```

100 %

In this screenshot we can see that the HTML form is created using a HTML helper function, all the labels on the page are created using helper functions and all the textboxes are also created using helper functions. Now run the application and try to see the result.


http://localhost:55109

- My ASP.NE

File Edit View Favorites Tools Help

Contact

Name

Email

Query

Save

Back to List

This is the page we see after we run the application.

Following HTML helpers are built into the ASP.NET MVC framework:

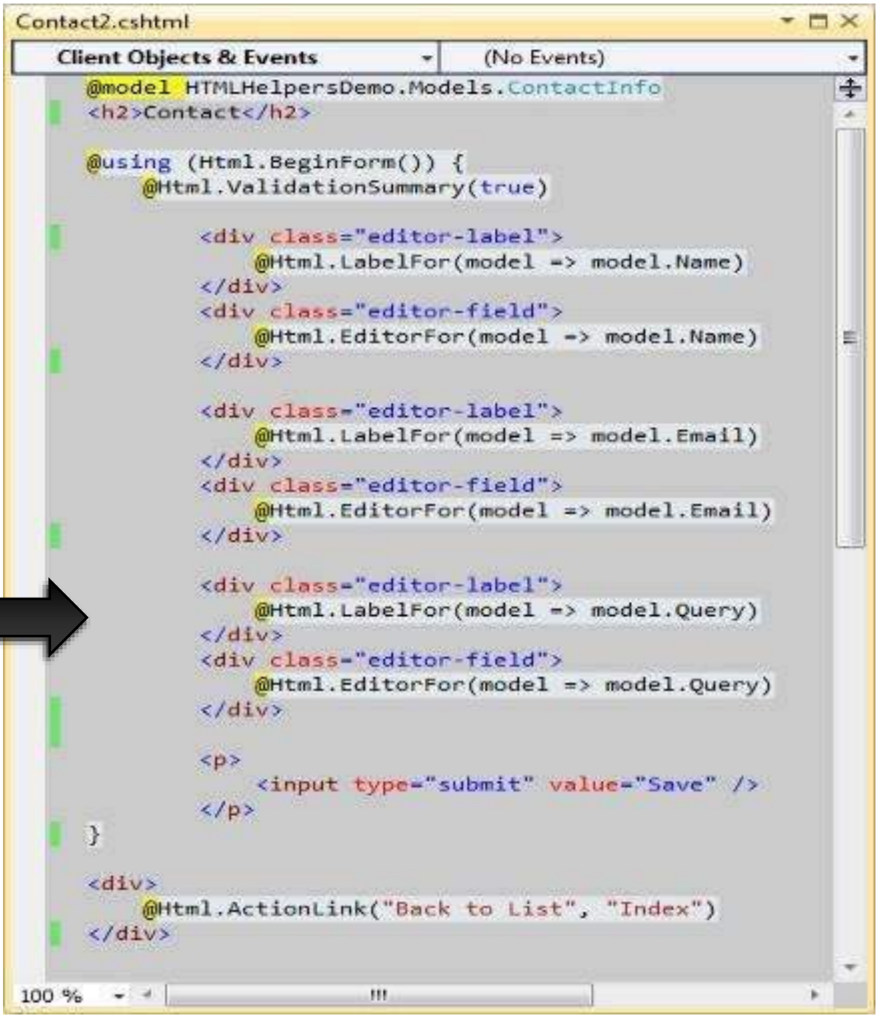
- Html.BeginForm
- Html.EndForm
- Html.TextBox
- Html.TextArea
- Html.Password
- Html.Hidden
- Html.CheckBox
- Html.RadioButton
- Html.DropDownList
- Html.ListBox

HTML Helpers for strongly typed views

If we are creating a strongly typed view then it is also possible to use the HTML helpers methods with the model class. Let us create a model “ContactInfo”for the contact us page:

```
public class ContactInfo
{
    public string Name { get; set; }
    public string Email { get; set; }
    public string Query { get; set; }
}
```

*Strongly typed view for contact us page using the **Html helpers**.*



```
@model HTMLHelpersDemo.Models.ContactInfo
<h2>Contact</h2>

@using (Html.BeginForm()) {
    @Html.ValidationSummary(true)

        <div class="editor-label">
            @Html.LabelFor(model => model.Name)
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.Name)
        </div>

        <div class="editor-label">
            @Html.LabelFor(model => model.Email)
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.Email)
        </div>

        <div class="editor-label">
            @Html.LabelFor(model => model.Query)
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.Query)
        </div>

        <p>
            <input type="submit" value="Save" />
        </p>
}

<div>
    @Html.ActionLink("Back to List", "Index")
</div>
```

These helper methods create the output HTML elements based on model properties. The property to be used to create the HTML is passed to the method as a lambda expression. It could also be possible to specify id, name and various other HTML attributes using these helper methods. Following HTML helpers are available to be used with strongly typed views:

- Html.TextBoxFor
- Html.TextAreaFor
- Html.PasswordFor
- Html.HiddenFor
- Html.CheckBoxFor
- Html.RadioButtonFor
- Html.DropDownListFor
- Html.ListBoxFor

Books

1. Beginning ASP.NET 4: in C# and VB; Imar Spaanjaars,2010
2. Beginning ASP.NET 3.5 in C# 2008: From Novice to Professional; 2nd edition, Matthew MacDonald,2007
3. ASP.NET 3.5 Unleashed by Stephen Walther, 2008
4. Pro ASP.NET 3.5 in C# 2008: Includes Silverlight 2 by Matthew MacDonald,2008
5. ASP.NET 3.5 For Dummies by Ken Cox,2008

References

1. ASP.NET; URL: https://www.guru99.com/
2. URL: https://docs.microsoft.com/
3. Strongly typed views; URL: https://dotnettutorials.net/lesson
4. URL: https://www.tutlane.com/tutorial/aspnet-mv
5. URL: https://www.tutorialsteacher.com/mvc

Thank you!

ASP.NET MVC BASICS

Course Code: CSC 4164 Course Title: ADVANCED PROGRAMMING WITH .NET

Dept. of Computer Science
Faculty of Science and Technology

Lecture No:	**07**	**Week No:**	**04**	**Semester:**	**Fall 2020-21**
Lecturer:	*Victor Stany Rozario, stany@aiub.edu*				

Lecture Outline

Hands-on:

- ➢ Simple data access using dummy repository class
- ➢ Data Access using scaffold templates (List, Create, Edit, Update, Delete)
- ➢ Modifying scaffolded templates (Dropdown list for Edit view)
- ➢ Preventing unintended update using update model, try update model, bind attribute and interface
- ➢ Disadvantages of delete using a GET request and its solution, Layouts in razor view

Let's see the Hands-on demonstration

Books

1. Beginning ASP.NET 4: in C# and VB; Imar Spaanjaars,2010
2. Beginning ASP.NET 3.5 in C# 2008: From Novice to Professional; 2nd edition, Matthew MacDonald,2007
3. ASP.NET 3.5 Unleashed by Stephen Walther, 2008
4. Pro ASP.NET 3.5 in C# 2008: Includes Silverlight 2 by Matthew MacDonald,2008
5. ASP.NET 3.5 For Dummies by Ken Cox,2008

References

1. ASP.NET; URL: https://www.guru99.com/
2. URL: https://docs.microsoft.com/
3. Strongly typed views; URL: https://dotnettutorials.net/lesson
4. URL: https://www.tutlane.com/tutorial/aspnet-mv
5. URL: https://www.tutorialsteacher.com/mvc

Thank you!

Entity Framework

Course Code: CSC 4164 Course Title: ADVANCED PROGRAMMING WITH .NET

Dept. of Computer Science
Faculty of Science and Technology

Lecture No:	**08**	**Week No:**	**05**	**Semester:**	**Fall 2020-21**
Lecturer:	*Victor Stany Rozario, stany@aiub.edu*				

Lecture Outline

- Definition of EF
- ADO.NET VS EF
- Installing and configuring EF
- Introduction to LINQ and Lambda Expression
- CRUD operations using EF Schema-First approach
- CRUD operations using EF Code-First approach

Entity Framework

Definition of EF

- Microsoft has provided an O/RM framework called "Entity Framework" to automate database related activities for your application.
- ADO.NET entity is an ORM (object relational mapping) which creates a higher abstract object model over ADO.NET components. So rather than getting into dataset, datatables, command, and connection objects as shown, you work on higher level domain objects like customers, suppliers, etc.
- ADO.NET is the oldest Microsoft data access framework. It shipped in the original release of .NET Framework and has proven to be stable and dependable. ADO.NET is a set of libraries that supports interactions with many data sources.

- Writing and managing ADO.Net code for data access is a tedious and monotonous job.
- As developers work with strongly typed .NET objects called entities, there was a big gap between the object models used in the Object-Oriented Programming (OOP) and the data storage, which is in a relational model. Much code was needed to deal with this gap. ORM was created to resolve this issue. It is a technique for converting data stored in a relational database to domain-specific classes.
- Entity Framework offers many benefits compared to previous technology. One of its advantages is the excellent tooling support where we can build and maintain data access in a much shorter time. We can focus on solving business problems without worrying about the underlying data storage.

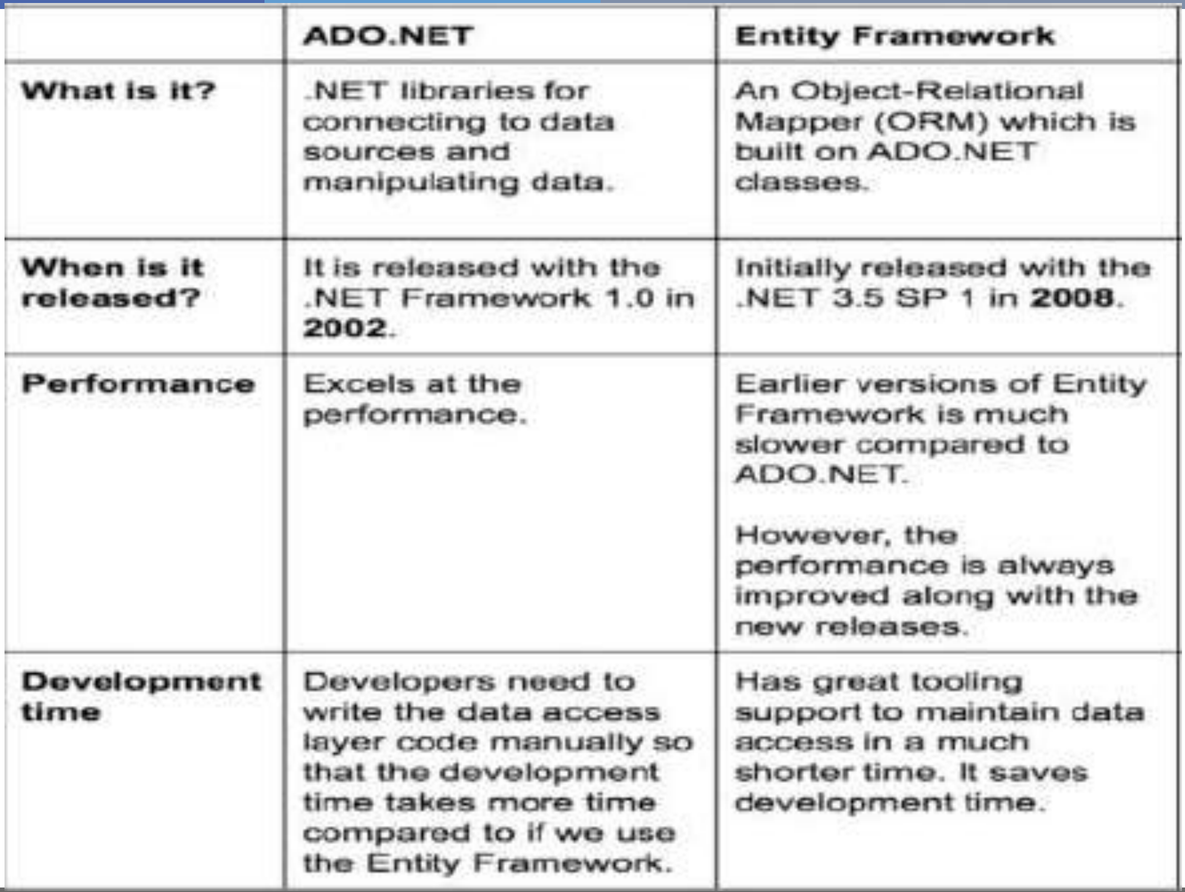
Entity Framework Features

- **Cross-platform**: EF Core is a cross-platform framework which can run on Windows, Linux and Mac.
- **Modelling**: EF (Entity Framework) creates an EDM (Entity Data Model) based on POCO (Plain Old CLR Object) entities with get/set properties of different data types. It uses this model when querying or saving entity data to the underlying database.
- **Querying**: EF allows us to use LINQ queries (C#/VB.NET) to retrieve data from the underlying database. The database provider will translate this LINQ queries to the database-specific query language (e.g. SQL for a relational database). EF also allows us to execute raw SQL queries directly to the database.
- **Change Tracking**: EF keeps track of changes occurred to instances of your entities (Property values) which need to be submitted to the database.
- **Saving**: EF executes INSERT, UPDATE, and DELETE commands to the database based on the changes occurred to your entities when you call the SaveChanges() method. EF also provides the asynchronous SaveChangesAsync() method.

Entity Framework Features

- **Concurrency**: EF uses Optimistic Concurrency by default to protect overwriting changes made by another user since data was fetched from the database.
- **Transactions**: EF performs automatic transaction management while querying or saving data. It also provides options to customize transaction management.
- **Caching**: EF includes first level of caching out of the box. So, repeated querying will return data from the cache instead of hitting the database.
- **Built-in Conventions**: EF follows conventions over the configuration programming pattern and includes a set of default rules which automatically configure the EF model.
- **Configurations**: EF allows us to configure the EF model by using data annotation attributes or Fluent API to override default conventions.
- **Migrations**: EF provides a set of migration commands that can be executed on the NuGet Package Manager Console or the Command Line Interface to create or manage underlying database Schema.

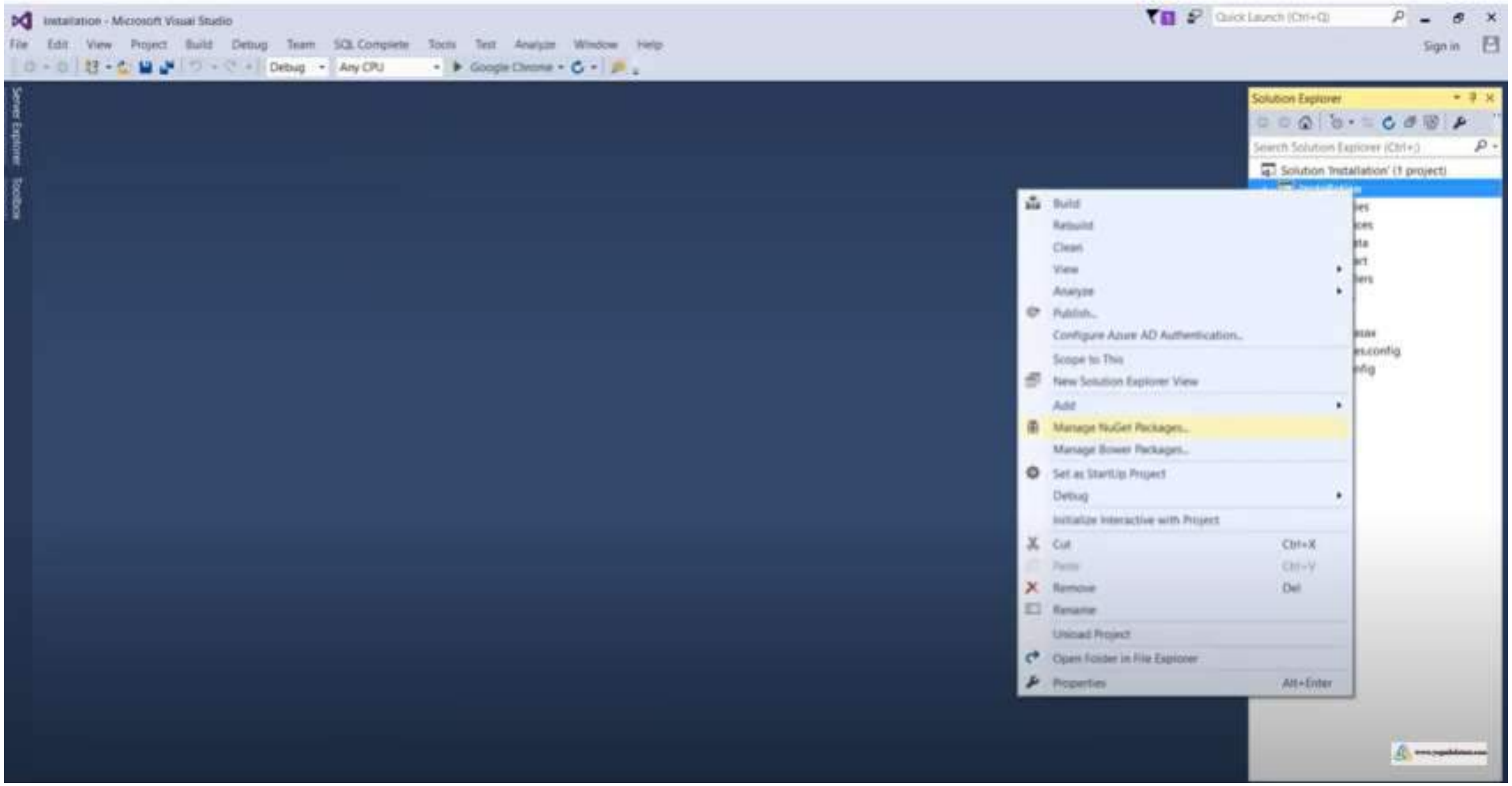
ADO.NET VS EF



	ADO.NET	Entity Framework
What is it?	.NET libraries for connecting to data sources and manipulating data.	An Object-Relational Mapper (ORM) which is built on ADO.NET classes.
When is it released?	It is released with the .NET Framework 1.0 in **2002**.	Initially released with the .NET 3.5 SP 1 in **2008**.
Performance	Excels at the performance.	Earlier versions of Entity Framework is much slower compared to ADO.NET. However, the performance is always improved along with the new releases.
Development time	Developers need to write the data access layer code manually so that the development time takes more time compared to if we use the Entity Framework.	Has great tooling support to maintain data access in a much shorter time. It saves development time.

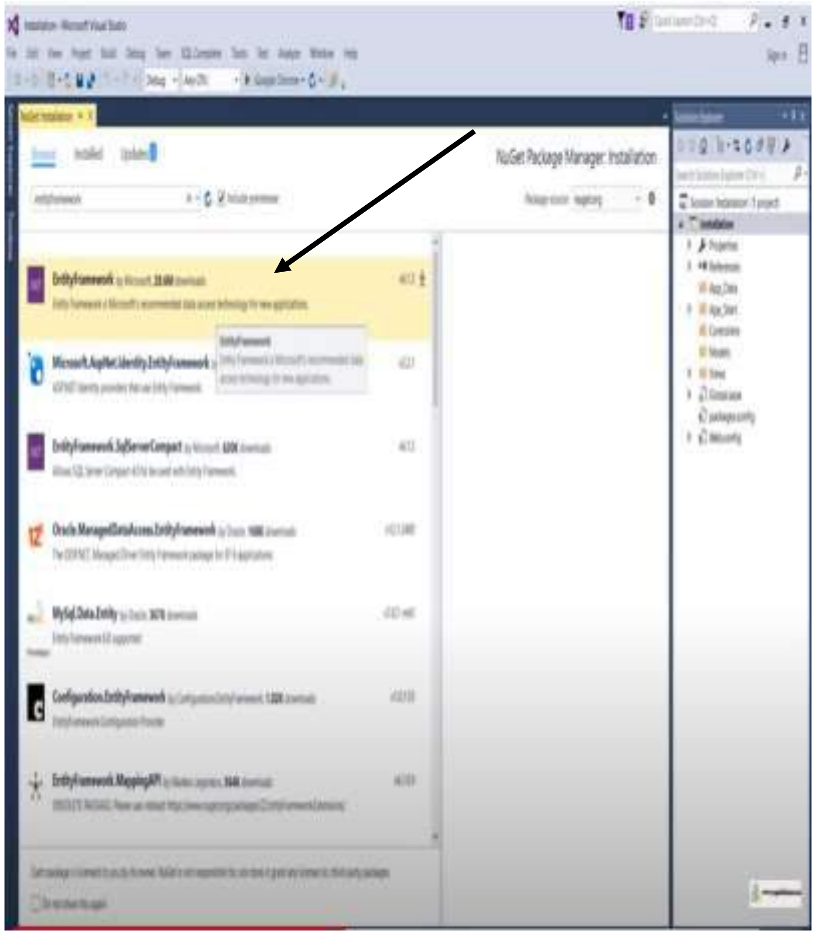
Installing EF

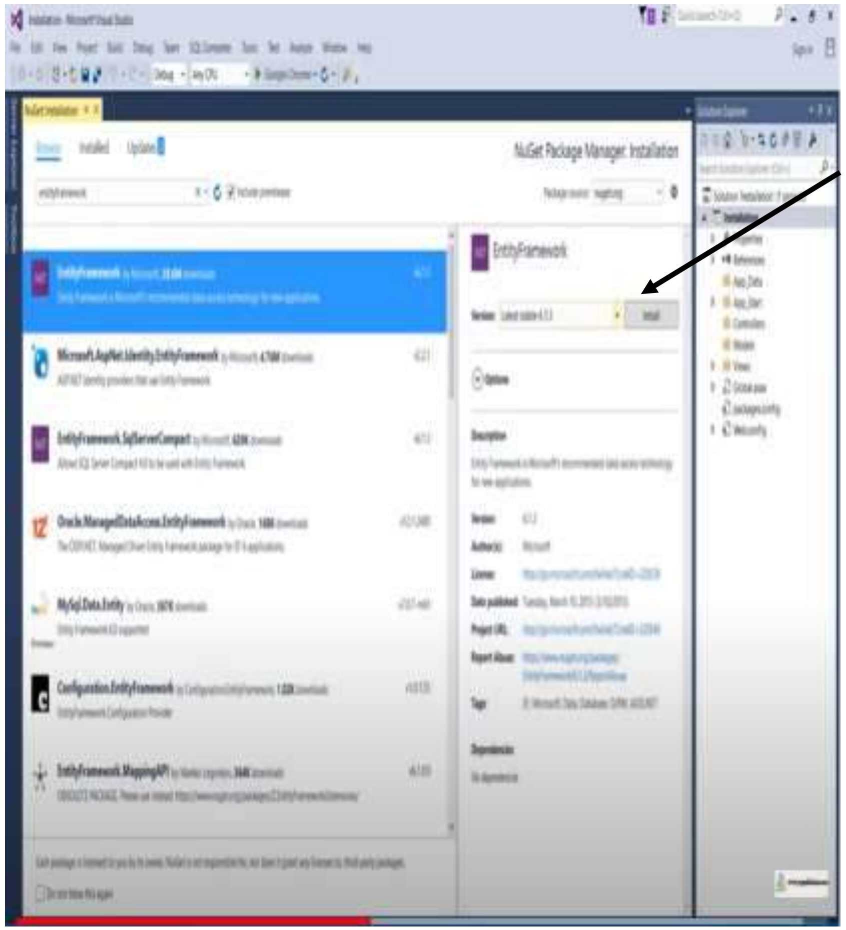
- Right click on the project in the Solution Explorer in Visual Studio and select Manage NuGet Packages.. (or select on the menu: Tools -> NuGet Package Manager -> Manage NuGet Packages For Solution).


Installation - Microsoft Visual Studio
File Edit View Project Build Debug Team SQL Complete Tools Test Analyze Window Help
Solution Explorer
Solution 'Installation' (1 project)
Build
Rebuild
Clean
View
Analyze
Publish...
Configure Azure AD Authentication...
Scope to This
New Solution Explorer View
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

Installing EF

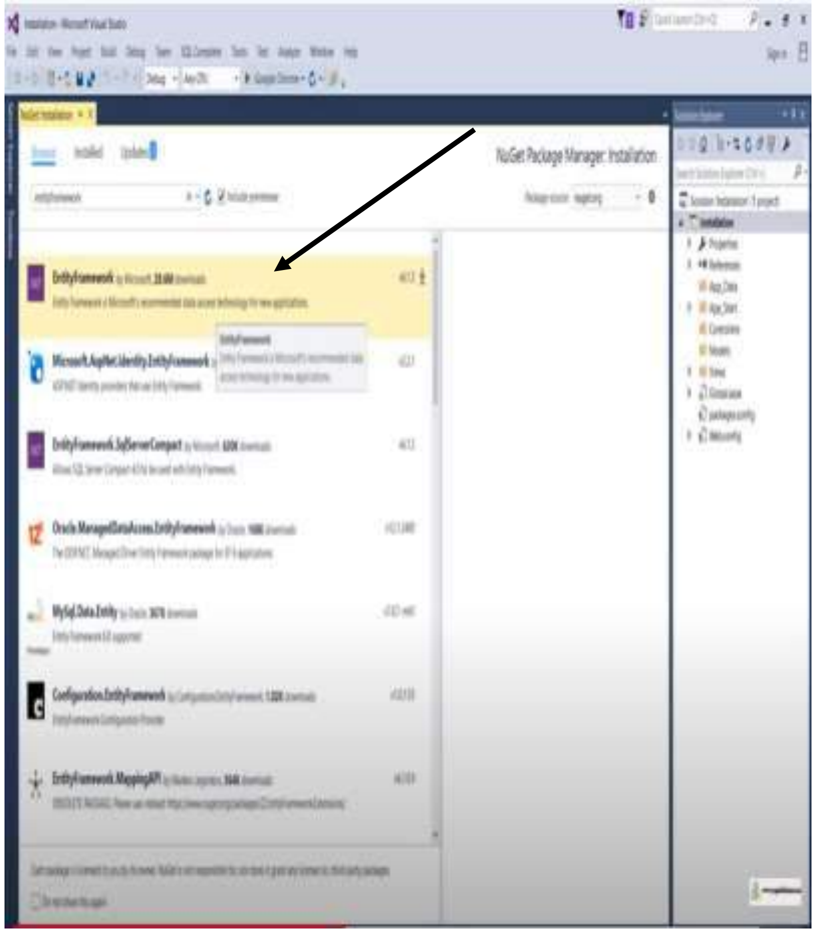
- Select **Browse** on the top and search **Entity framework** on the search list. Select Entity framework and then click “Install”.

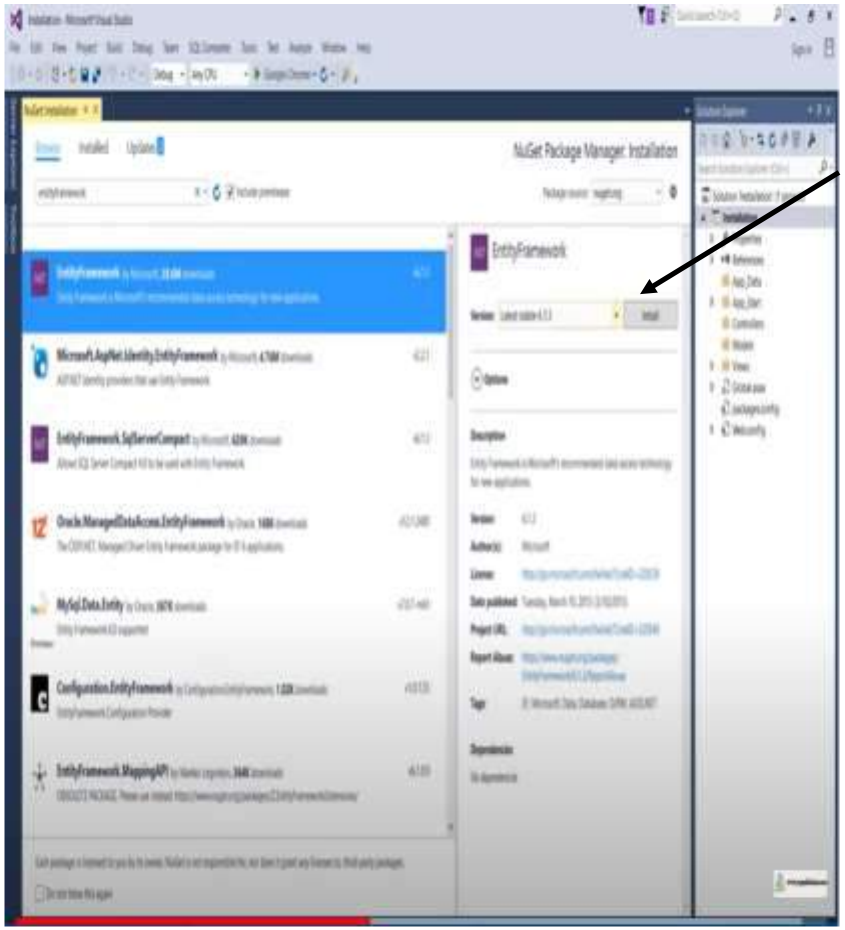

Browse
Installed
Updates
NuGet Package Manager: Installation
EntityFramework
Microsoft.AspNet.Identity.EntityFramework
EntityFramework.SqlServerCompact
Oracle.ManagedDataAccess.EntityFramework
MySql.Data.Entity
Configuration.EntityFramework
EntityFramework.MappingAPI


Browse
Installed
Updates
NuGet Package Manager: Installation
EntityFramework
Install
Microsoft.AspNet.Identity.EntityFramework
EntityFramework.SqlServerCompact
Oracle.ManagedDataAccess.EntityFramework
MySql.Data.Entity
Configuration.EntityFramework
EntityFramework.MappingAPI

Installing EF

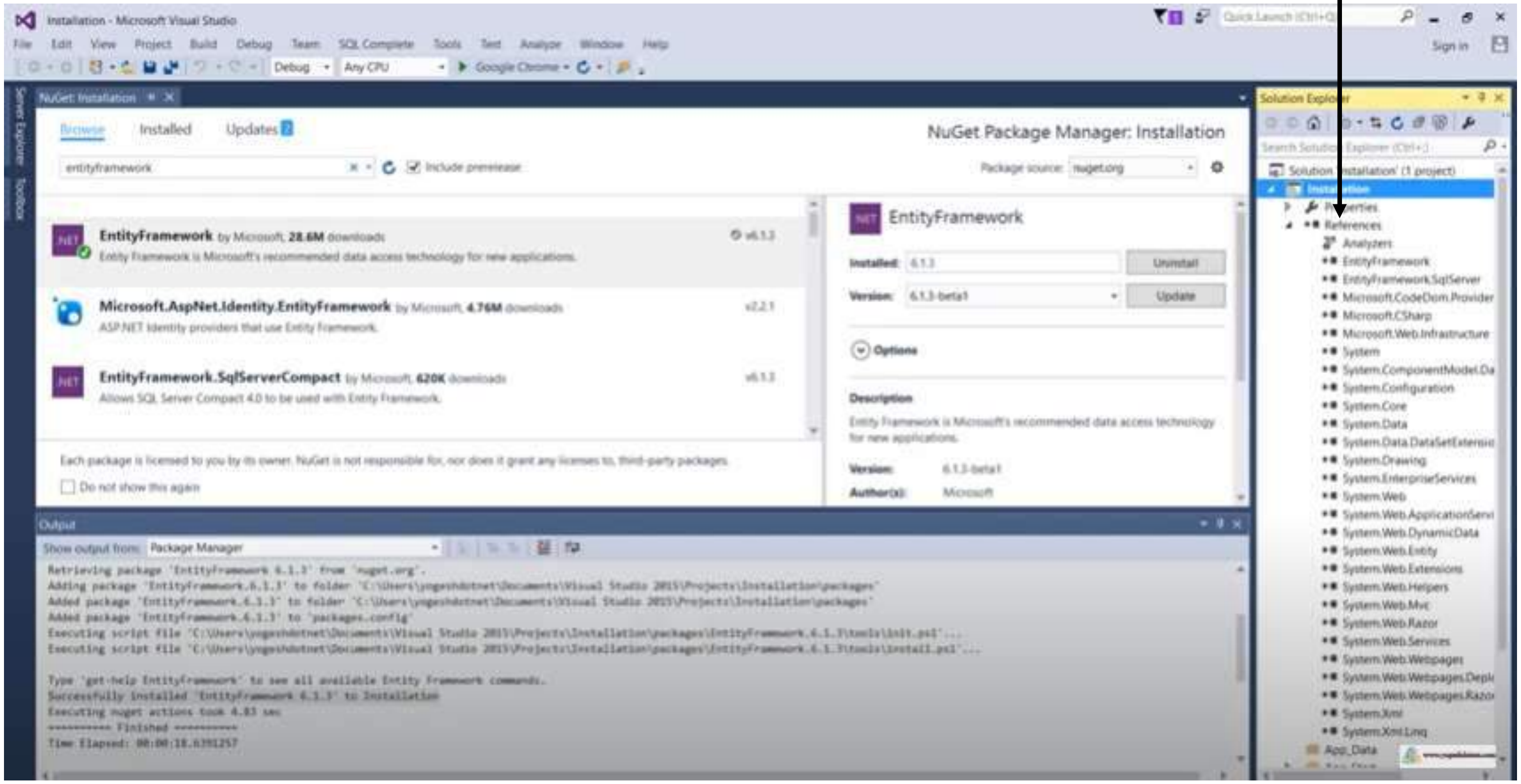
- Select **Browse** on the top and search **Entity framework** on the search list. Select Entity framework and then click “Install”. After that it again asks for approval so just click onto “OK”. Afterwards accept the conditions.





Installing EF

- You will then find the package install and to check it, click onto “References” from the Solution Explorer. You will see the references of **Entity Framework** and **Entity Framework SqlServer**.



NuGet Package Manager: Installation

entityframework — Include prerelease — Package source: nuget.org

EntityFramework by Microsoft, **28.6M** downloads — v6.1.3
Entity Framework is Microsoft's recommended data access technology for new applications.

Microsoft.AspNet.Identity.EntityFramework by Microsoft, **4.76M** downloads — v2.2.1
ASP.NET Identity providers that use Entity Framework.

EntityFramework.SqlServerCompact by Microsoft, **620K** downloads — v6.1.3
Allows SQL Server Compact 4.0 to be used with Entity Framework.

Each package is licensed to you by its owner. NuGet is not responsible for, nor does it grant any licenses to, third-party packages.

Do not show this again

EntityFramework
Installed: 6.1.3
Version: 6.1.3-beta1

Description
Entity Framework is Microsoft's recommended data access technology for new applications.

Version: 6.1.3-beta1
Author(s): Microsoft

Show output from: Package Manager

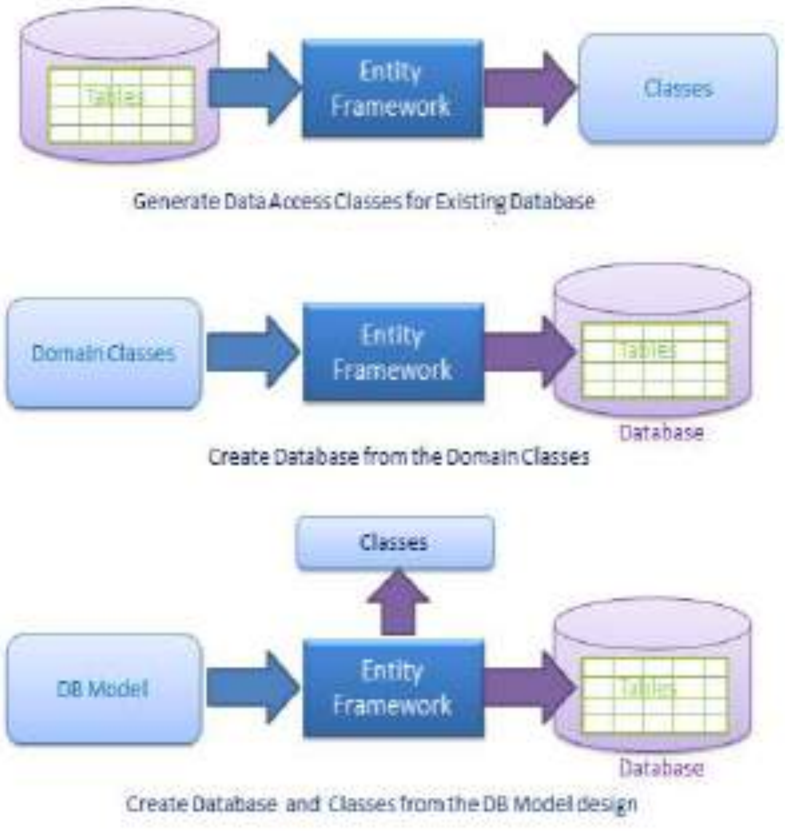
```
Retrieving package 'EntityFramework 6.1.3' from 'nuget.org'.
Adding package 'EntityFramework.6.1.3' to folder 'C:\Users\yogeshdotnet\Documents\Visual Studio 2015\Projects\Installation\packages'
Added package 'EntityFramework.6.1.3' to folder 'C:\Users\yogeshdotnet\Documents\Visual Studio 2015\Projects\Installation\packages'
Added package 'EntityFramework.6.1.3' to 'packages.config'
Executing script file 'C:\Users\yogeshdotnet\Documents\Visual Studio 2015\Projects\Installation\packages\EntityFramework.6.1.3\tools\init.ps1'...
Executing script file 'C:\Users\yogeshdotnet\Documents\Visual Studio 2015\Projects\Installation\packages\EntityFramework.6.1.3\tools\install.ps1'...

Type 'get-help EntityFramework' to see all available Entity Framework commands.
Successfully installed 'EntityFramework 6.1.3' to Installation
Executing nuget actions took 4.83 sec
========== Finished ==========
Time Elapsed: 00:00:18.6391257
```

Solution Explorer

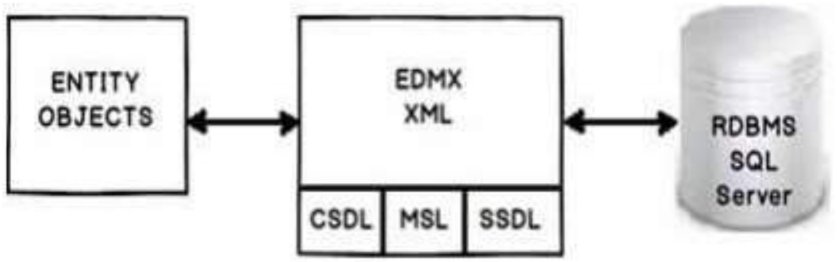
```
Solution 'Installation' (1 project)
  Installation
    Properties
    References
      Analyzers
      EntityFramework
      EntityFramework.SqlServer
      Microsoft.CodeDom.Provider
      Microsoft.CSharp
      Microsoft.Web.Infrastructure
      System
      System.ComponentModel.Da
      System.Configuration
      System.Core
      System.Data
      System.Data.DataSetExtensio
      System.Drawing
      System.EnterpriseServices
      System.Web
      System.Web.ApplicationServi
      System.Web.DynamicData
      System.Web.Entity
      System.Web.Extensions
      System.Web.Helpers
      System.Web.Mvc
      System.Web.Razor
      System.Web.Services
      System.Web.Webpages
      System.Web.Webpages.Deplo
      System.Web.Webpages.Razor
      System.Xml
      System.Xml.Linq
    App_Data
```

Below is the code for Entity Framework in which we are working on higher level domain objects like customer rather than with base level ADO.NET components (like dataset, datareader, command, connection objects, etc.).

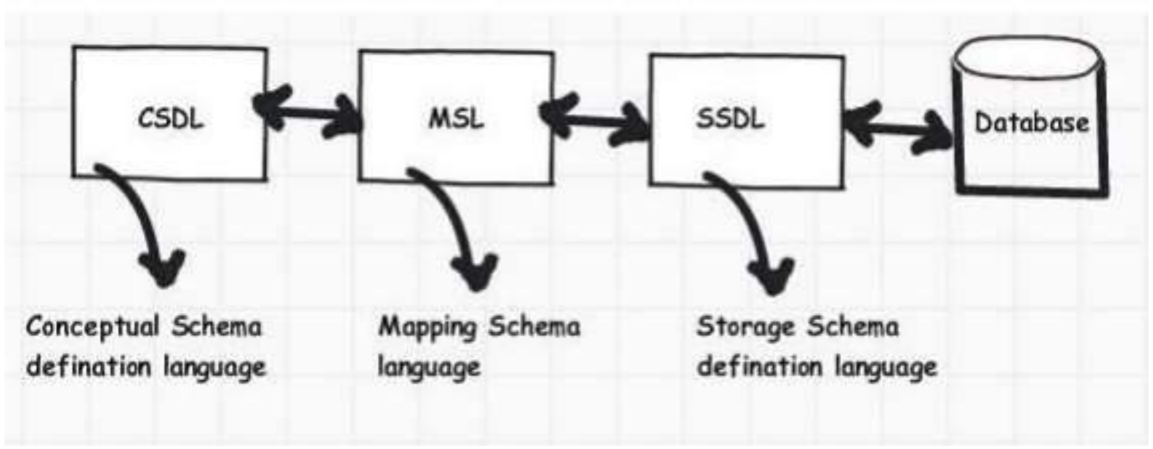
```
DataTable table = adoDs.Tables[0];
for (int j = 0; j < table.Rows.Count; j++)
{
	DataRow row = table.Rows[j];
	// Get the values of the fields
	string CustomerName =
	(string)row["Customername"];
	string CustomerCode =
	(string)row["CustomerCode"];
}

foreach (Customer objCust in obj.Customers)
{ }
```


Tables
Entity Framework
Classes
Generate Data Access Classes for Existing Database
Domain Classes
Entity Framework
Tables
Database
Create Database from the Domain Classes
Classes
DB Model
Entity Framework
Tables
Database
Create Database and Classes from the DB Model design

EDMX file



•CSDL (Conceptual Schema definition language) is the conceptual abstraction which is exposed to the application.
•SSDL (Storage Schema Definition Language) defines the mapping with your RDBMS data structure.
•MSL (Mapping Schema Language) connects the CSDL and SSDL.


CSDL
MSL
SSDL
Database
Conceptual Schema defination language
Mapping Schema language
Storage Schema defination language

Introduction to LINQ

C# provides a mechanism for querying collections known as **LINQ-**Language Integrated Query.

- LINQ enables access to collections (and databases) using query expressions which are similar to SQL queries
    - This allows the retrieval of information from a wide variety of data sources.
- .NET also provides LINQ providers for :
    - LINQ to SQL
        - For querying databases
    - LINQ to XML
        - For querying xml documents
- We can design a simple LINQ query which filters the contents of an array

Querying an array

A simple query object comprises from, where and select clauses

- Also, we can make use of the keyword var which is an implicit type

```
var filteredArray =
        from ...... // range variable and data source
        where ..... // boolean expression
        select..... // which value appears in the results
```

In the next slide we will see an example of accessing an array by LINQ.

- IEnumerable<T> is an interface implemented by arrays and collections
  - It is a generic type
  - We replace the T by a real type (such as an int)

```
using System;
using System.Collections.Generic;
using System.Linq;
class LINQtoArray
{
        static void Main(string[] args)
        {
                int[] array = { 2, 6, 4, 12, 7, 8, 9, 13, 2 };
                var filteredArray = // LINQ query
                from element in array
                where element < 7
                select element;
                PrintArray(filteredArray, "All values less than 7:");
        }
        public static void PrintArray(IEnumerable<int> arr, string message)
        {
                Console.Write("{0}",message);
                foreach (var element in arr)
                Console.Write(" {0}", element);
                Console.WriteLine();
        }
}
```

We can add the orderby (descending) clause to our query to sort ouT filtered array into ascending (descending) order.

```
var filteredArray =
        from ...... // range variable and data source
        where ..... // boolean expression
        orderby ..... (descending) // sort
        select..... // which value appears in the results
```

It's important to understand a feature of LINQ known as deferred execution

- The result of a LINQ query expression is not a sequence or collection of objects but a query object
- It represents the commands needed to execute the query
- The query does not execute until the program requests data from the query object
- Deferred execution is a powerful feature of LINQ as it allows applications to pass queries around as data
- In our simple example, the query is not run until it is passed to the PrintArray method

Introduction to Lambda Expression

A lambda expression is an anonymous function that you can use to create delegates or expression tree types. By using lambda expressions, you can write local functions that can be passed as arguments or returned as the value of function calls. Lambda expressions are particularly helpful for writing LINQ query expressions. To create a lambda expression, you specify input parameters (if any) on the left side of the lambda operator =>, and you put the expression or statement block on the other side. For example, the lambda expression x => x * x specifies a parameter that's named x and returns the value of x squared. You can assign this expression to a delegate type, as the following example shows:

```
delegate int del(int i);
static void Main(string[] args)
{
        del myDelegate = x => x * x;
        int j = myDelegate(5); //j = 25
}
```

Expression Lambdas

A lambda expression with an expression on the right side of the => operator is called an expression lambda. Expression lambdas are used extensively in the construction of Expression Trees (C# and Visual Basic). An expression lambda returns the result of the expression and takes the following basic form:

(input parameters) => expression

The parentheses are optional only if the lambda has one input parameter; otherwise they are required. Two or more input parameters are separated by commas enclosed in parentheses:

(x, y) => x == y

Sometimes it is difficult or impossible for the compiler to infer the input types. When this occurs, you can specify the types explicitly.

Statement Lambdas

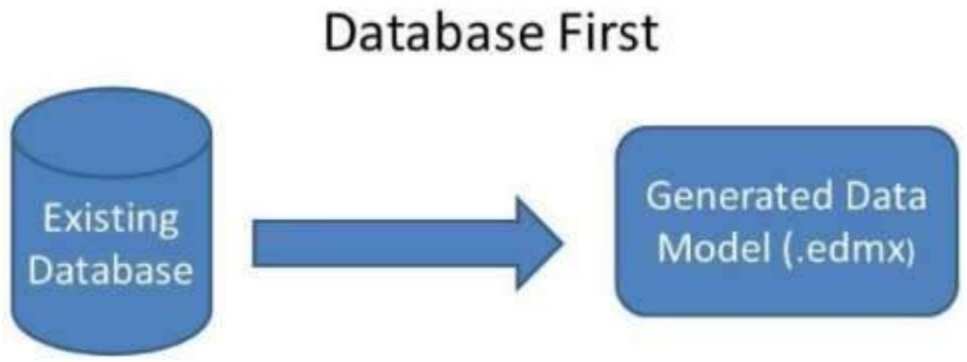
A statement lambda resembles an expression lambda except that the statement(s) is enclosed in braces:

(input parameters) => {statement;}

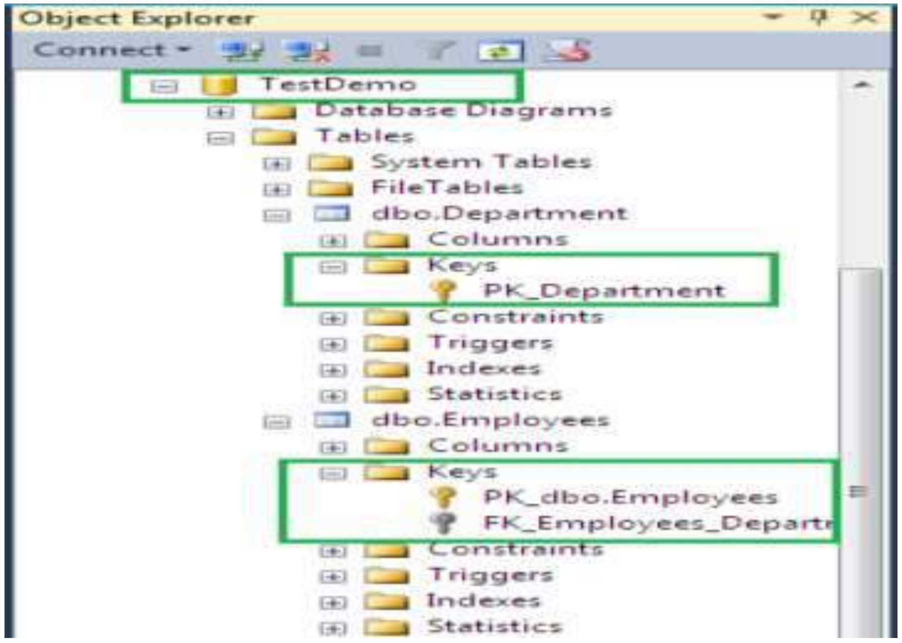
Details will be covered in the notes.

CRUD operations using EF Schema-First approach

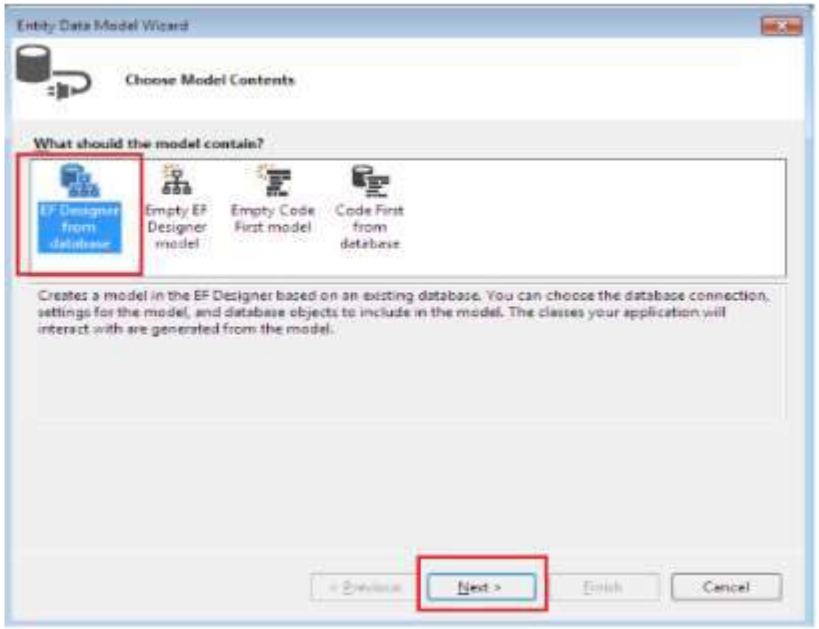
Database First is nothing but only a approach to create web application where database is available first and can interact with database. Database is created first and after that we manage the code. The Entity Framework is able to generate a business model based on the tables and columns in a relational database.



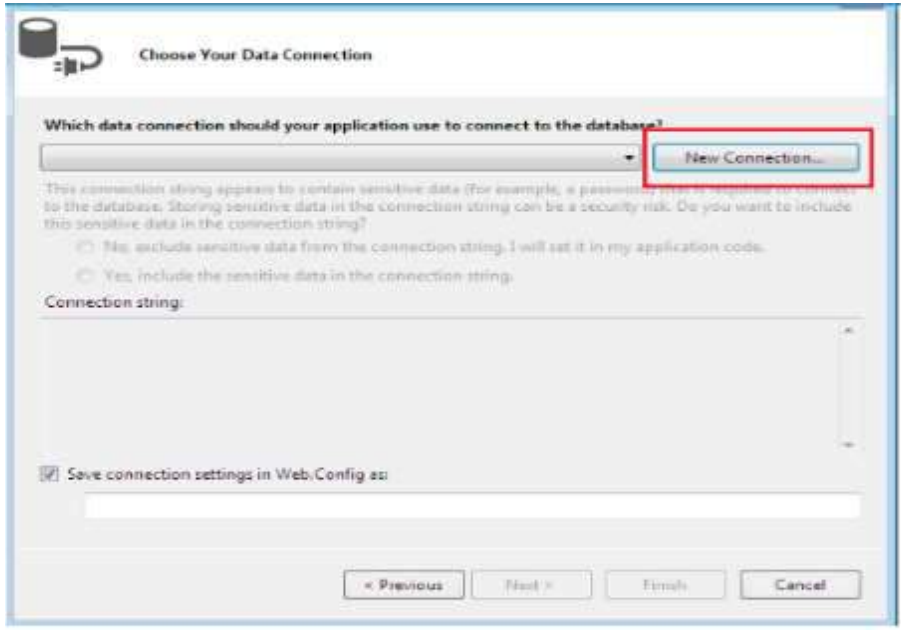
- To create a new database first open "**Microsoft SQL Server Management Studio**" and Right click on **Database** and choose **New Database**.
- It will open a New Database Dialog where you can define your database structure. You need to provide the database name "**TestDemo**" and click to **OK**. It will add a new database for you. You can check it into the Object Explorer.
- Create some table which will participate in CRUD operations. We are going to create two tables "Employee" and "Department" and make relationship between both.



- Create new ASP.NET Web application in Visual Studio.
- Create models from existing database. To Add Models, Right Click on Models folder and choose Add and then choose New Item. Choose **Data** node then **ADO.NET Entity Data Model**, provide the valid name and click **OK**.
- In the **Entity Data Model Wizard**, select **EF Designer from database** and Click **Next**.
- Click the **New Connection** button. where you will define the database connection.



Creates a model in the EF Designer based on an existing database. You can choose the database connection, settings for the model, and database objects to include in the model. The classes your application will interact with are generated from the model.



Which data connection should your application use to connect to the database?

This connection string appears to contain sensitive data (for example, a password) that is required to connect to the database. Storing sensitive data in the connection string can be a security risk. Do you want to include this sensitive data in the connection string?

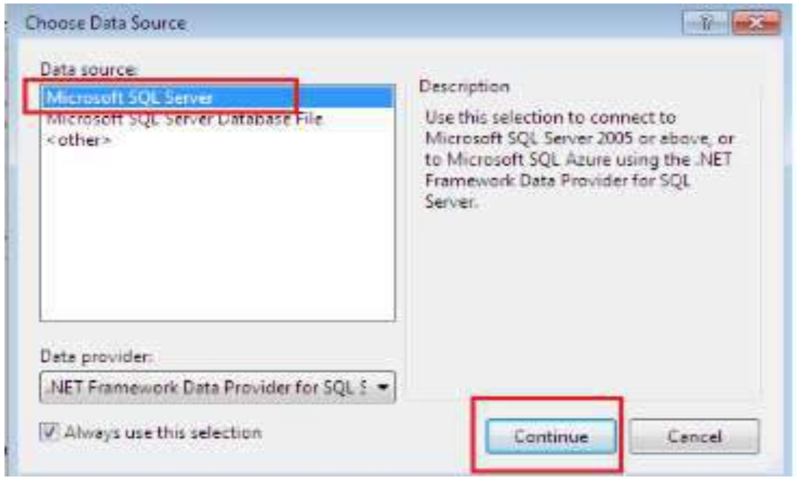
No, exclude sensitive data from the connection string. I will set it in my application code.

Yes, include the sensitive data in the connection string.

Connection string:

[✓] Save connection settings in Web.Config as:

- In the **Choose Data Source** window, we need to choose Data Source and click to **Continue**.
- It will open a new dialog where you need to specify everything about database connection such as Server name. Here choose **Windows Authentication** and also choose the database. Select our database name.
- It will create database connection string in the Entity Data Model Wizard. Click to **Next**.
- From the next Entity Data Model Wizard, we can choose database items like **Tables**, **Views**, **Stored Procedures** and **Function**. Choose as per your requirement and pass the **Model Name** and click to **Finish**..


Choose Data Source

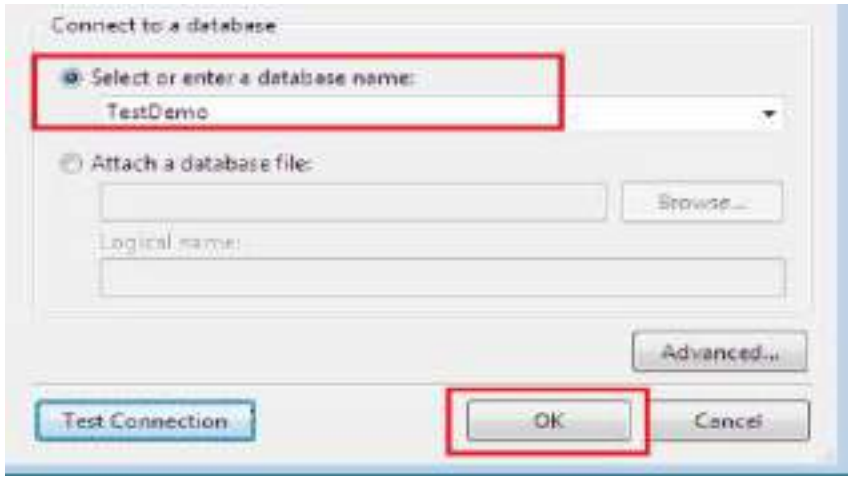
Data source:
Microsoft SQL Server
Microsoft SQL Server Database File
<other>

Description
Use this selection to connect to Microsoft SQL Server 2005 or above, or to Microsoft SQL Azure using the .NET Framework Data Provider for SQL Server.

Data provider: .NET Framework Data Provider for SQL S

[✓] Always use this selection

Continue Cancel

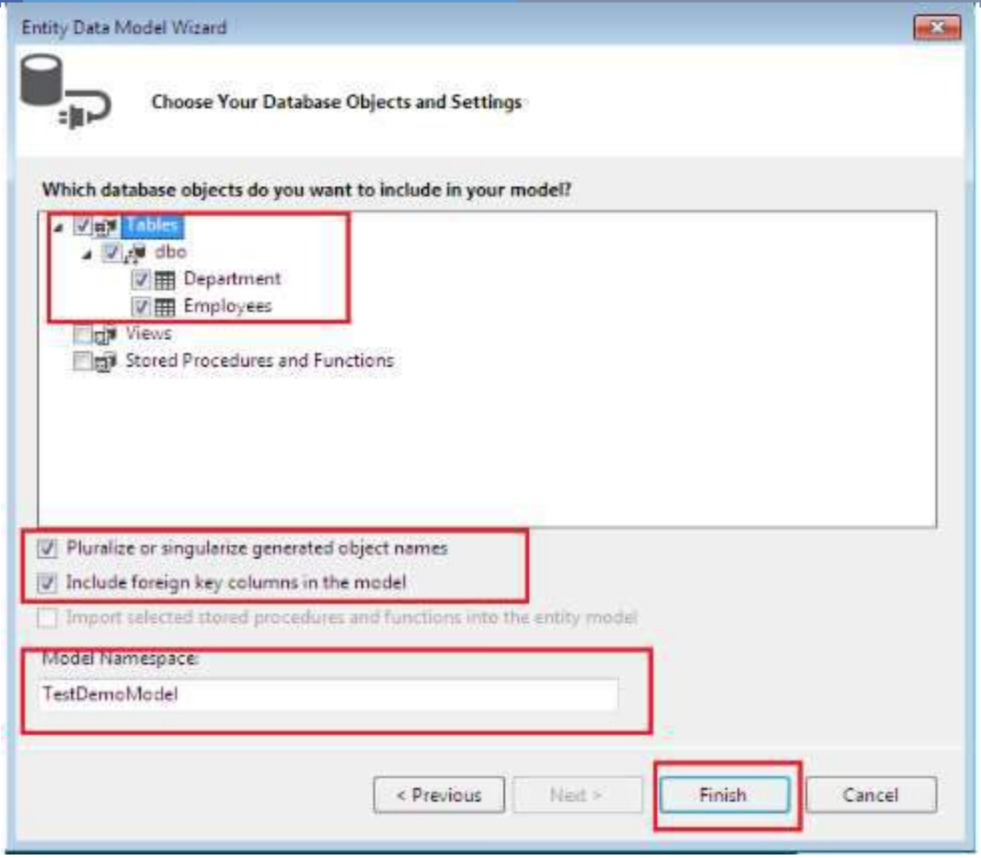

Connect to a database

● Select or enter a database name:
TestDemo

○ Attach a database file:
Browse...
Logical name:

Advanced...

Test Connection OK Cancel


Entity Data Model Wizard

Choose Your Database Objects and Settings

Which database objects do you want to include in your model?

- [✓] Tables
  - [✓] dbo
    - [✓] Department
    - [✓] Employees
- [ ] Views
- [ ] Stored Procedures and Functions

[✓] Pluralize or singularize generated object names

[✓] Include foreign key columns in the model

[ ] Import selected stored procedures and functions into the entity model

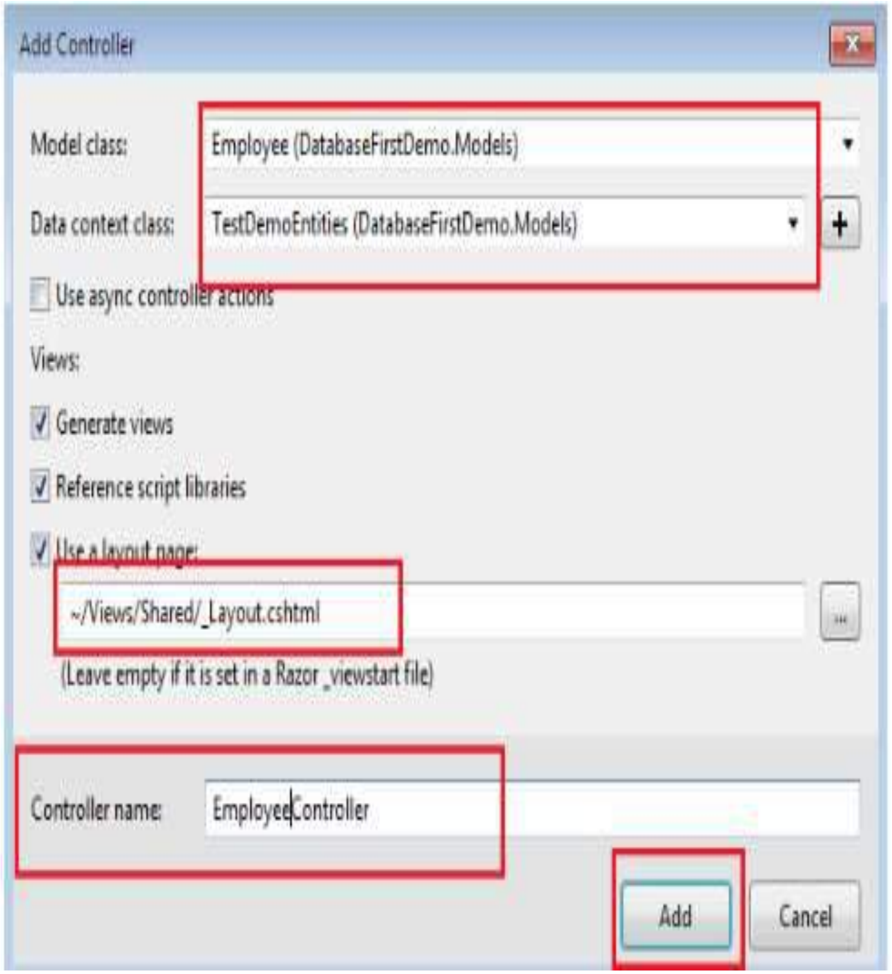
Model Namespace:

TestDemoModel

The **DemoDataModel.Context.cs** file contains a class that derives from the **DbContext** class. It also provides a property for each model class that corresponds to a database table. The **Department.cs** and **Employee.cs** files contain the model classes that represent the databases tables. Here you can see Department.cs and Employee.cs represent the database the database table. They contain all columns as properties.

- Before proceeding to move forward build the application. We have added database with tables and implemented it with Entity Data Model which has been created model classes. So, it is time to create User Interface [Views] which will used to perform user operation like here we can add data, select data, edit data and delete data.
- To add new controller, Right click on Controller folder and choose **Add** and choose Add **Scaffolded** Item.
- It will open a Add Scaffold window, Here we need to choose the Controller type. Here we need to choose **MVC5 Controller with views, using Entity Framework and choose Add.**
- From the Add Controller window, we need to select **the Model Class and Data Context Class**. We can also select layout page. In the Controller Name section, we can provide the specific name for the controller "**EmployeeController**".
- When we click on **Ok**. It will Scaffold and create the EmployeeController as well as Views for Employee Controller.
- In the controller you can see all the operation has been defined by default. From database instance creation to getting data, deleting data, editing data has defined by the code.

- After adding EmployeeController. We can see here a Employee folder has been added inside the Views and all the view like Index.cshtml, edit.cshtml etc also has added.



Add Controller

Model class: Employee (DatabaseFirstDemo.Models)

Data context class: TestDemoEntities (DatabaseFirstDemo.Models)

Use async controller actions

Views:

Generate views

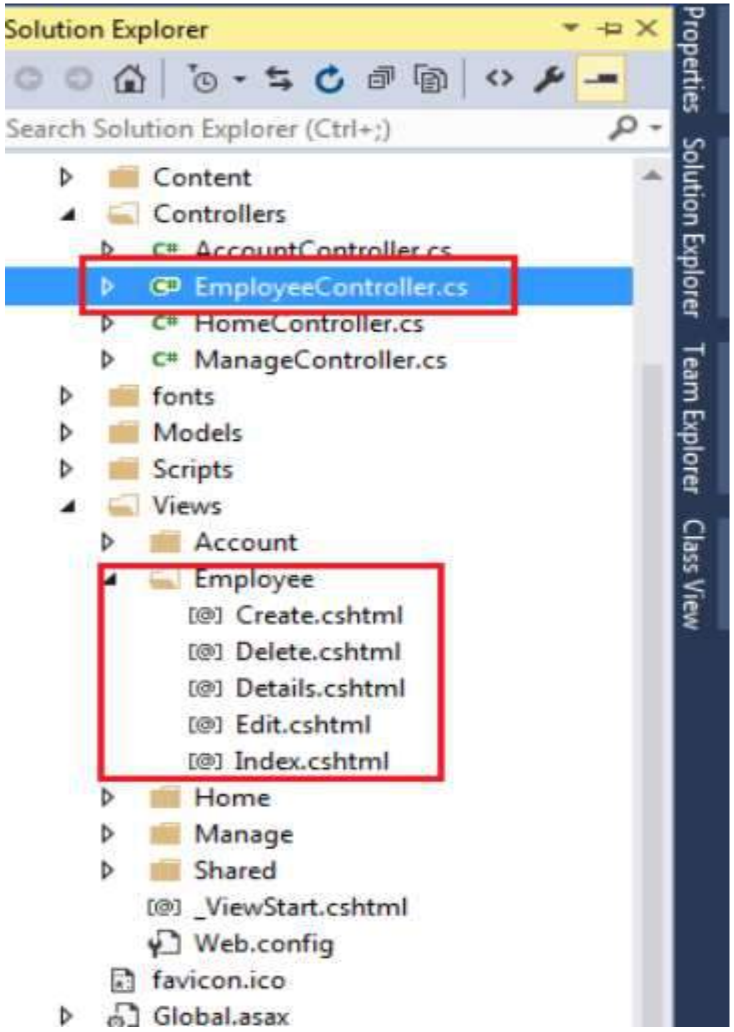
Reference script libraries

Use a layout page:

~/Views/Shared/_Layout.cshtml

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

Controller name: EmployeeController



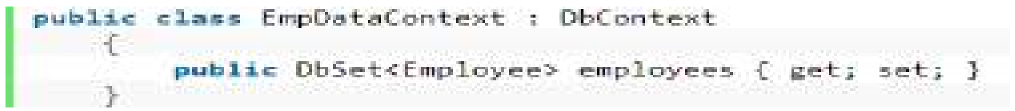
Solution Explorer

Search Solution Explorer (Ctrl+;)

```
Content
Controllers
    AccountController.cs
    EmployeeController.cs
    HomeController.cs
    ManageController.cs
fonts
Models
Scripts
Views
    Account
    Employee
        Create.cshtml
        Delete.cshtml
        Details.cshtml
        Edit.cshtml
        Index.cshtml
    Home
    Manage
    Shared
    _ViewStart.cshtml
    Web.config
favicon.ico
Global.asax
```

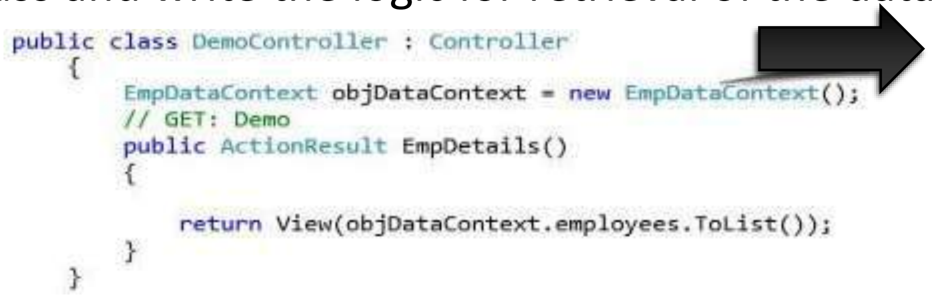
CRUD operations using EF Code-First approach

- Create an ASP.NET Web application then, we have to create a domain class. Right-click on Models folder and add a class like an Employee class.
- Add a DbContext class, for that, right-click the Models folder and add a class and give the name EmpDataContext



```
public class EmpDataContext : DbContext
    {
        public DbSet<Employee> employees { get; set; }
    }
```

- Next, we have to add a Controller. Go to Controllers folder and add a controller.
- The controller has an Index Action method which is automatically created. Change it to a user-friendly method name. Now, create the object of your EmpDataContext class and write the logic for retrieval of the data.



```
public class DemoController : Controller
    {
        EmpDataContext objDataContext = new EmpDataContext();
        // GET: Demo
        public ActionResult EmpDetails()
        {

            return View(objDataContext.employees.ToList());
        }
    }
```

- Add a View for displaying the employee records. Right-click on action method and add a View.
- Now, before the retrieval of data, we have to set our connection string in **web.config** file.

```
<connectionStrings>
      <add name="MySqlConnection" connectionString="Data
     Source=MyPC;database=MyDemoDB;User Id=sa;Password=123;"
     providerName="System.Data.SqlClient" />
</connectionStrings>
```

- Set the connection string in EmpDataContext class. If we don’t want to give manual connection string, then when running the project, it will automatically create the connection string with the same name as the class name of EmpDataContext class. If we create manually, then we have to pass our Connection String name in base class parameter.
- Now, before the retrieval of data, we have to set our connection string in **web.config** file.

- In Code First approach when we will run the project, at that moment automatically, it will create the Database and the Table; table name will be same as our domain Class name; as below my class name is Employee. Also, we can set the Primary key in SQL table from our program by using the Key class attribute in “[ ]” brackets. For using this functionality, we have to import the namespace like below.

```
using System.ComponentModel.DataAnnotations;        (Import namespace)
using System.ComponentModel.DataAnnotations.Schema;
   [Table("TblEmployee")]
   public class Employee
   {
      [Key]
      public int EmpId { get; set; }
      public string Name { get; set; }
      public string Address { get; set; }
      public string Email { get; set; }
      public string MobileNo { get; set; }
}
```

- In Code First approach when we will run the project, at that moment automatically, it will create the Database and the Table; table name will be same as our domain Class name; as below my class name is Employee. Also, we can set the Primary key in SQL table from our program by using the Key class attribute in “[ ]” brackets. For using this functionality, we have to import the namespace like below.
- Now, we will run the project and then check in SQL. Create the controllers and views manually and write down the CRUD operations to see the result.

```
using System.ComponentModel.DataAnnotations;         (Import namespace)
using System.ComponentModel.DataAnnotations.Schema;
   [Table("TblEmployee")]
   public class Employee
   {
      [Key]
      public int EmpId { get; set; }
      public string Name { get; set; }
      public string Address { get; set; }
      public string Email { get; set; }
      public string MobileNo { get; set; }
}
```

Books

1. Beginning ASP.NET 4: in C# and VB; Imar Spaanjaars,2010
2. Beginning ASP.NET 3.5 in C# 2008: From Novice to Professional; 2nd edition, Matthew MacDonald,2007
3. ASP.NET 3.5 Unleashed by Stephen Walther, 2008
4. Pro ASP.NET 3.5 in C# 2008: Includes Silverlight 2 by Matthew MacDonald,2008
5. ASP.NET 3.5 For Dummies by Ken Cox,2008

References

1. ASP.NET; URL: https://www.got-it.ai/solutions
2. URL: https://www.tutorialspoint.com/
3. URL: https://www.c-sharpcorner.com/aURL: https://www.tutorialsteacher.com/mvc
4. Entity framework; URL:https://www.entityframeworktutorial.net/
5. URL: https://www.tutlane.com/tutorial/aspnet-mv
6. URL: http://www.mukeshkumar.net/

Thank you!