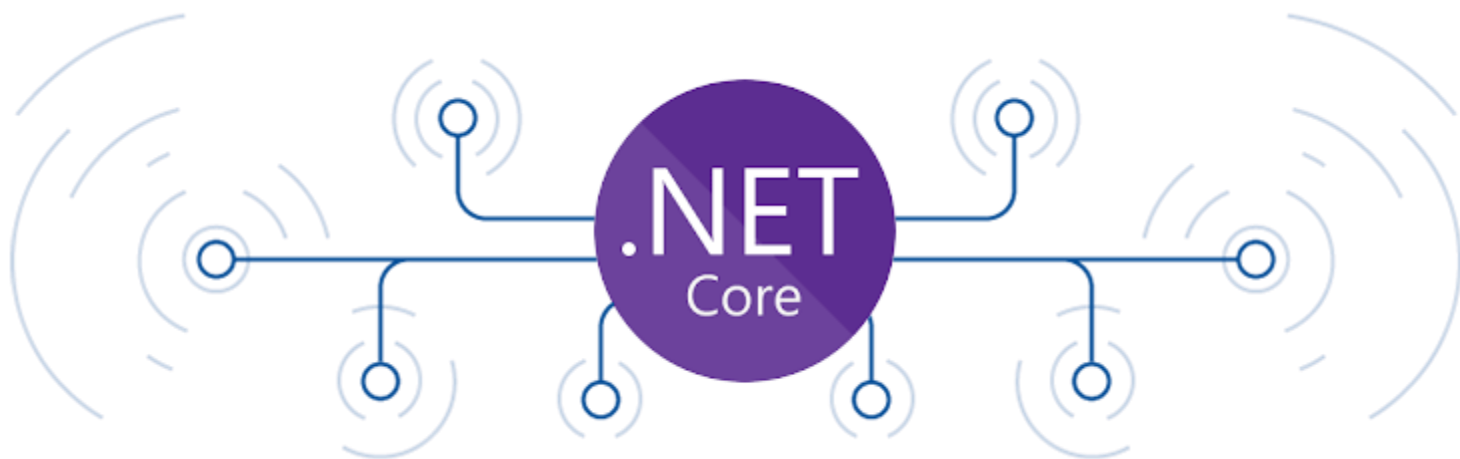


Application Development



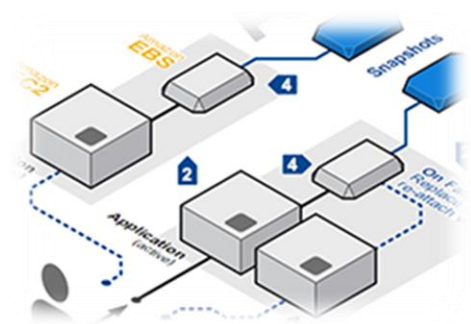
<https://bit.ly/niiemvc>

ThS Lương Trần Hy Hiến

Course Objectives

❑ The C# MVC Frameworks course provides

- In depth knowledge in the ASP.NET Core Framework
- Knowledge for creating REST Services
- Deploying Applications
- Architectural ASP.NET Core Application Design
- Best practices in modern ASP.NET Core applications



C# Recommended Software

- ❑ Software needed for this course:
 - Microsoft Windows 10+
 - Visual Studio 2022 – <https://www.visualstudio.com>
 - MS SQL Server - <https://www.microsoft.com/en-cy/sql-server/sql-server-downloads>



Course content



Introduction to ASP.NET Core



MVC



Layout



Entity Framework Core



Web API



Final project guideline

General subject requirements

- ❑ Attend all theoretical sessions
- ❑ Do hands-on labs seriously!
- ❑ Complete all requirements:
 - Read reference materials (e-books) for the subject
 - Homework exercises
 - Practical exercise
 - Project

Prerequisite

- ❑ Web development knowledges:
 - HTML(5), CSS(3), BootStrap(5),...
 - Client-side scripting: javascript,...
 - Distinguish POST/GET
- ❑ C# language
- ❑ Database and SQL
 - Stored procedure in SQL Server

Assessment Brief

- ☐ 20% - Attendances + Labs
- ☐ 20% - Mid term
- ☐ 60% - Final project

Exercises in class

- ☐ Case studies
- ☐ Quiz

ASP.NET Core

A new open-source and cross-platform
framework for building modern cloud-based
Web applications using .NET

<https://learn.microsoft.com/en-us/aspnet/core/introduction-to-aspnet-core>

ASP.NET Core and the Modern Web



Totally Modular



Faster Development Cycle



Seamless transition
from on-premises to cloud



Choose your Editors
and Tools



Open Source
with Contributions

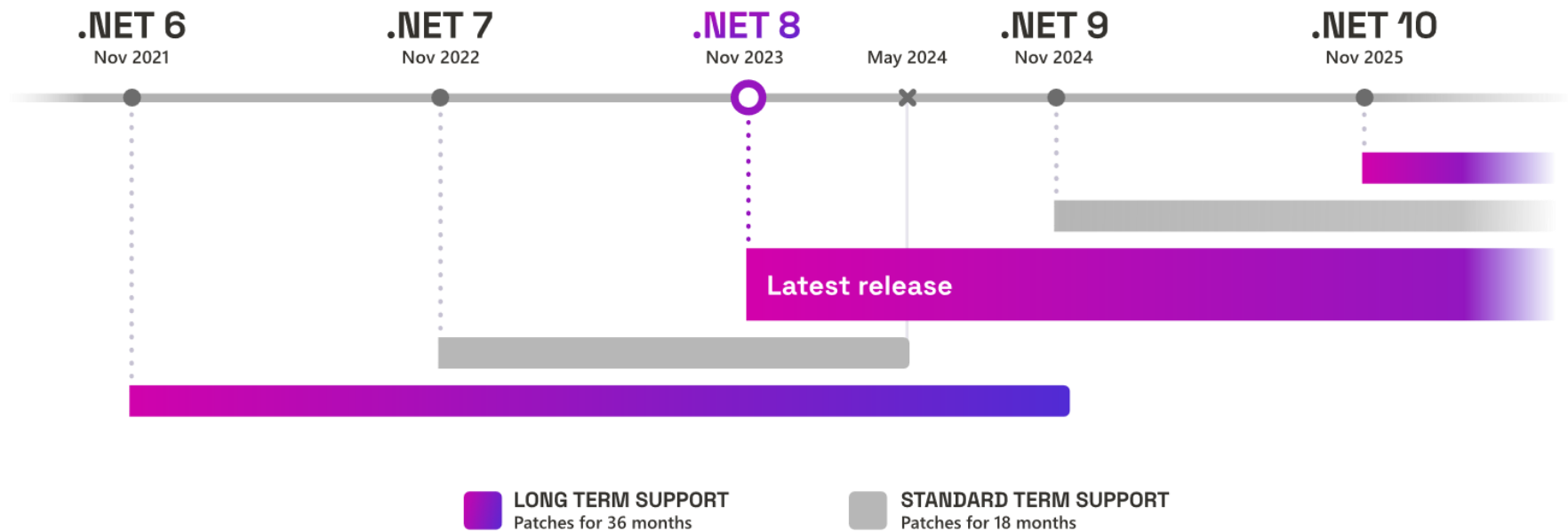


Cross-Platform



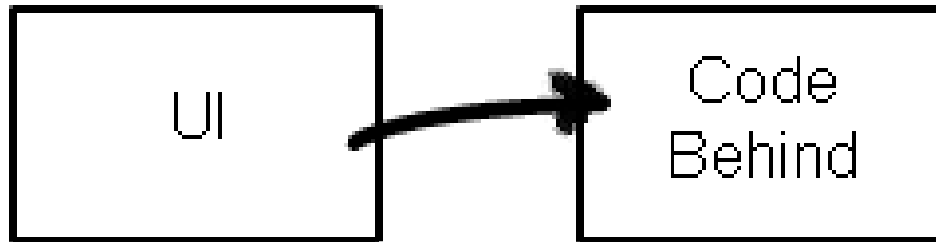
Fast

Net core version



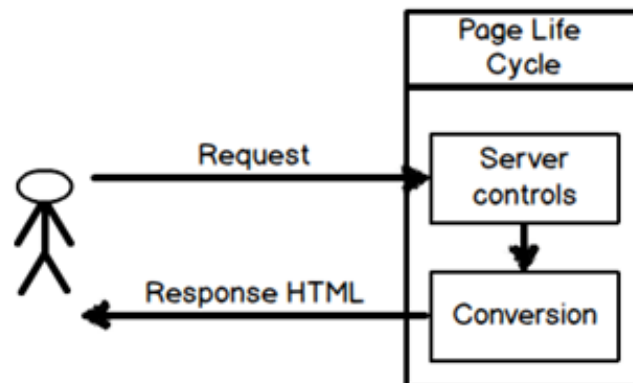
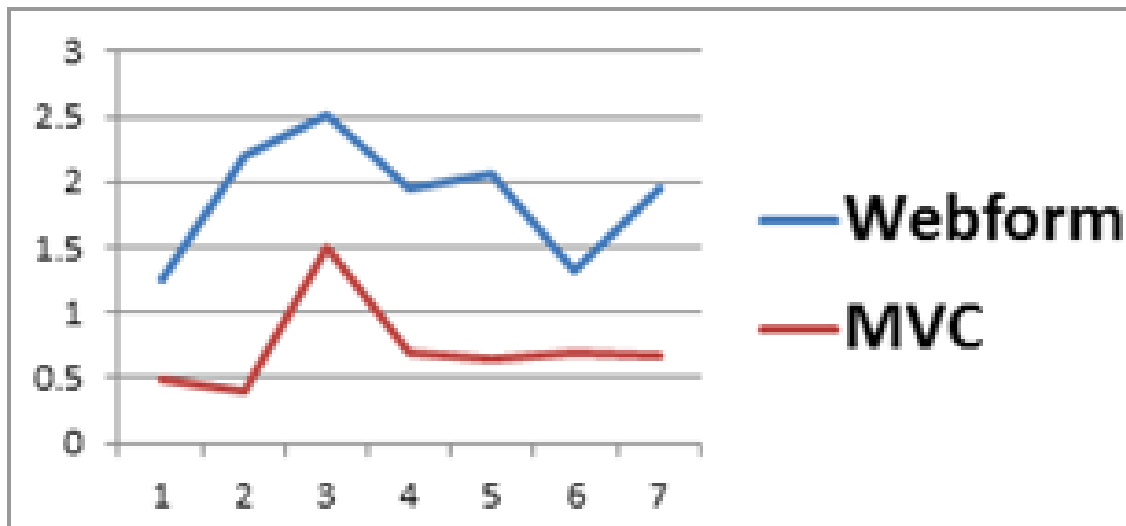
Why?

☐ Web Form



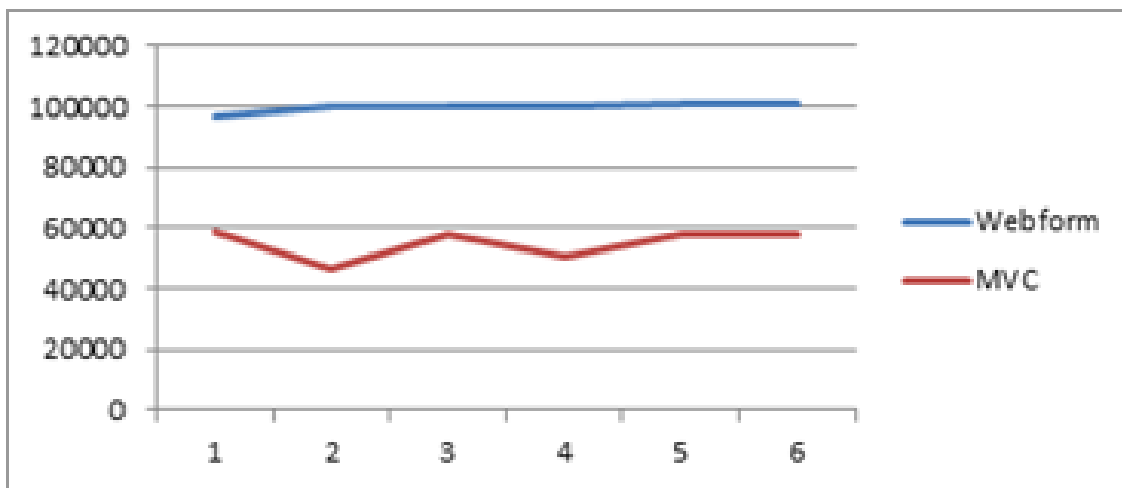
Problems with Asp.Net Web Forms

□ Response time issues



Problems with Asp.Net Web Forms

❑ Bandwidth consumption

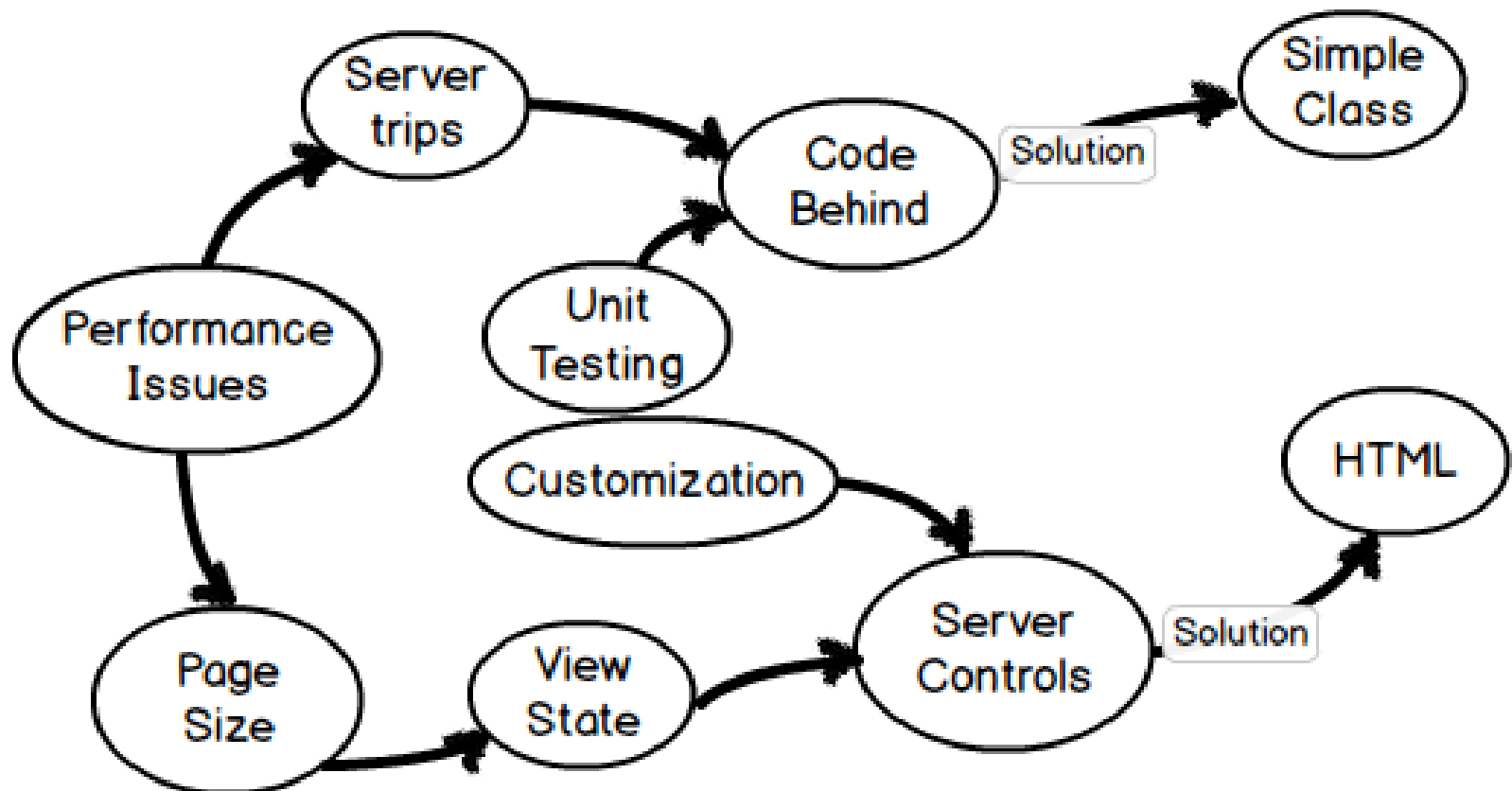


ViewState : Web Form

```
<body>
  <form method="post" action="WebForm1.aspx" id="form1">
    <div class="aspNetHidden">
      <input type="hidden" name="__VIEWSTATE" id="__VIEWSTATE"
        value="hFRayhQ99cEK735FA/bisys7ds1d0NoQ8BnfjU1l+1tuCvwpP60QLVr1h/RxioE0epEz6XyqChdz3j5BNP9LRFQzDz+SAH3KzF3vELxp9IHCkRZATU8Na3y0NaXgP0uBvRu61xkXkQ6q8i7F71
        6kx9L0D0vzU35c1PD8IXgUgOr47NbdL3HT8QsLH3Zf18hAXetBYUTAEaBHRK1koactcnVd8c8C8UABVjfn8zo9ofWu11x3/G3E8Wan/8TXsM4jr8zLoQ9KcV7yQ0sshIYD1XGpG/Q17LesvBncPe27
        cmQ0qenF0KKK7L1X5KzIzbx+HwJyehqH8CzX0EUQ66X2R5SUIqnpOILA88McE/66eLQ1VPGUqQCF5NDQv8v8UHS1VU75u9yul101H23IQ5gE1tw004CmoD1Pvq+HAIEd20Tcf3xH84zR9ax1zqlUw4
        7Kqz8p0iPhVB709H4p4k5GqCEVjDg8z1s0r3Q1d9qLU0wpUvGNaVcIeSut+P10i0yqdIdtHTMRnOrbd5bWzJoellmTnDhC06z08/3y20kYn9z8FgVH898y2w0ldtzw1PwlvThZVrd+EArlfkyrk3u1xN7:
        r5YHnna+PLD2nP1Ly9bW63PjnhUVM13ddfhJ5ySR1AG0u4yF8HcM6fcdtrSC1K26vFznfw0PE1f5z3bh8Rpd6y0GKJH+u0Qy3Y0trB488PYSash80xpgll4Axf7ckSymLNIzqnrsx04vH2pZUT1MyF
        QduuH7t1cVhd046E6fv3QL5aIKK6F3H213ly5a5Vks0nassQ/p8IgtYaeQF24i9rcvFX0P6y5XntucP8Ru8Ib8hgRplnRu0fHn7+/6jse8XU50RReVH456my2ofYhKXF/s6vXthChc17Q87gv21/0v
        P2h30906Ra3LURdsEgk579wCYW0+svQ00EKA/EmdIta+11Ac+HwKxrtY3HT3y25QFb1a8efqJ4X72K7D5z5XK52C0u8B8ps5FvJ11CRTq05ktXYCKHukHhHA7EpTX0pkip452AD0e0ZvN73j0XQ+fu
        hA0YCoGvzrH98bDen8Xg9uH57T77Tg8ing7QqPb87VAtuuH8HY/7aDKrvqGfRG09gbPZvc4uIaaC57ZnXFL8w2Bq1eaZ3CA3CIu+3xpfErs1zC68YV1Hb5sDfanXc0eoTC1g18fuAtazR5b6k79KXK8ti
        1X0fnnY0A1hwa69G6fk04VhV0V2tq/Umecc8pcDeec4qht56t8nd05UaPshTRPW1rtahv0rc8F8eh1n0ov13gdI000Pn/wb9t305FuIF51AXhg9E5aUa/IodA3f+/nd12guLvx46pUcR9d0vH208K1b0r
        mRkK1Iahr19aGhd8m3sojw+SEIcsjFucxQU88o6uy5hC48KIdJusdCnh57PC9t1+b07U0ecy8iF8X1Yfk10vyx09T35F30frjstiefzZxfGkZgsdt11QmbsnCQu+uYtDw9G2TP9Qcs/8s77XjoYjc08si
        yH84rPhf9o6vXJ00uKakqW8Ks9vH0QazQ8B0z28H0K+2V8G2IPJESV95ChkQ0heI2K3J+YH0XPKFus2905IRn1Lex8d5/8/zbfw82TubAkRyR3uH67n2NuhDf8/VLSbe+29oq4z821V3DjG1F4c
        4g4Xx/01VfDnckIdbvcBI1p8kdxsHFjYX35Y0Lrfs0o2LAuy0Ujy+B+khv1hws1gh0ckX3Aab8YTdrIp+43A/9etzhxeUjR/dh5kUeg4ta7X+qkDu08K5gpcDw1WY9AZV25FP9/ZDQ0d5gAsqL05fj
        3611c4QpT11tYayQ2m6NfHdHMEHr755z2m7sLLV0Z05rJ1ef/aAYC6G65UPRf93coara3PQ200C4Mv/zfucJt203NaD7Hr5vLCOnc4m1CnBryanR2Pg8Xtq+4NakX17VRYv154E2u841j7L37A140rCl
        jXzNRLf1dQJd8KAp68iqZt9IIasv20qcX0e6KfCtQ5EFIJEHRUXVsp1jxkfoto730/Knj1o1HUSQ520G2dTckK8ePVe2He1gtR4cvk7AF1yH2F5srFSU7I10vgqOPEIdc1w68ITQ1F2TK5Y94DQ8t
        E80HnVh0zJ08tgdUSREtHmLv0V+YtbJ2758xy00Q+A5Jp66/PmQXaunrveA1J0uPK1oX1Idq1/+1y51281k+0uoqj+X2sAHT0Y7x0KsP0BwmdEHF1RzbeHf0JcatuJ140018e0e+/L7t9:/z0X0f
        tLx0+dtb071kj70oTKF429AX/GhdTK6XD9sKcTuuXxSroqD7P8TnIVB569xydx+eHKLvUTty0Zq56V550F8H23Q1qoHqTKLH6gRzj+anu2j5lG26CyhQ9213LYB157XFO91bH7TgdR5wqePz9Nyq9CquU
        16iAN420Mn54xun3Fsn0/p456IPIYk0Ty+Om/q4C6LHdGLXtInkTISHC0TU40vvHfjmsE3U/v0uR08zPglr6b2V1Q/1Czsp4BP0dg257jp7R1H8sssc2RL0hE157PsuY4eL9XrO2HmJhaoj2Op8
        C9p6jgs1x885o/c2Nvfc7X4RQ7P1Ks0ty+Om/q4C6LHdGLXtInkTISHC0TU40vvHfjmsE3U/v0uR08zPglr6b2V1Q/1Czsp4BP0dg257jp7R1H8sssc2RL0hE157PsuY4eL9XrO2HmJhaoj2Op8
        KIS/M0Cobyf83d5LH7xqD09v0j5C3EE09NgU8G1ChP7a0L7cb/s5n/7T+v2wahpdga1l0pp/bf4P/PF65+mc2bdfCw6FvQEcHLfxTnRenOH2RQX511IhtK0GwF5Tnt7eHGXdtIT8rnk/bJHrsIU
        VHMdIvxx1pt20ppeHfrJ5R1tHKECk16k0L3u7f0ZAN01c41EY26e120VrCPUC1BayaG8H08z285/RAEH+8cwc0mc5f86d0z0d5d0+tkUknc15u0R2V2Uu1E5/NUJPzhfK0BwU/+3evCX6RPhz8
```

Problems with Asp.Net Web Forms

□ Unit Test





Introduction

- ❑ Overview of Web Application Development
- ❑ Introduction to ASP.NET MVC
- ❑ Architecture of ASP.NET MVC App
- ❑ Creating an ASP.NET MVC Project
- ❑ Structure of an ASP.NET MVC Project

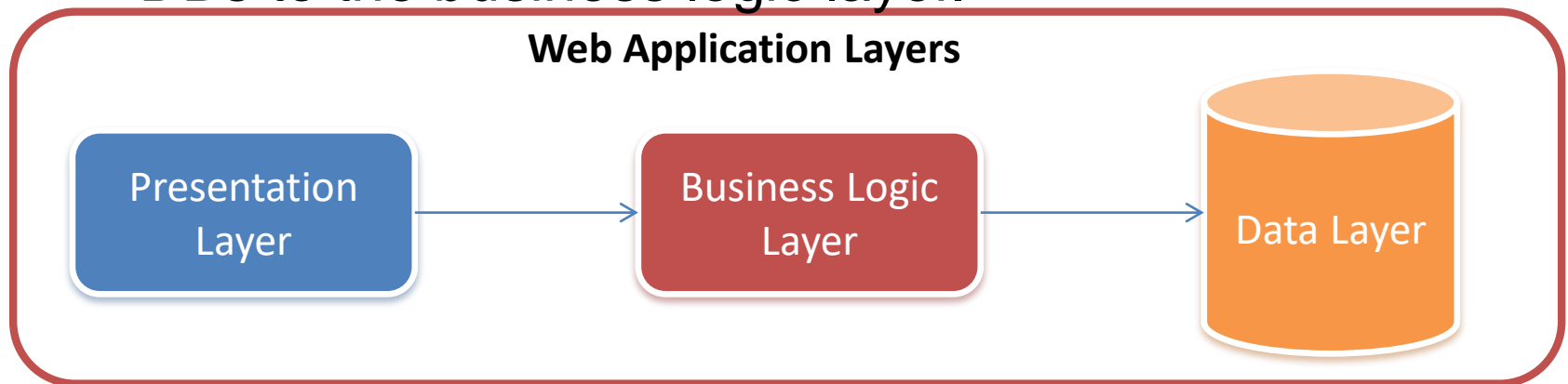
Overview of Web Application Development

□ Web Application

- Are programs that are executed on a Web server and accessed from a Web browser.
- Allows you to share and access information over the Internet that can be accessed globally at any time.
- Enable you to perform commercial transactions known as E-commerce application.

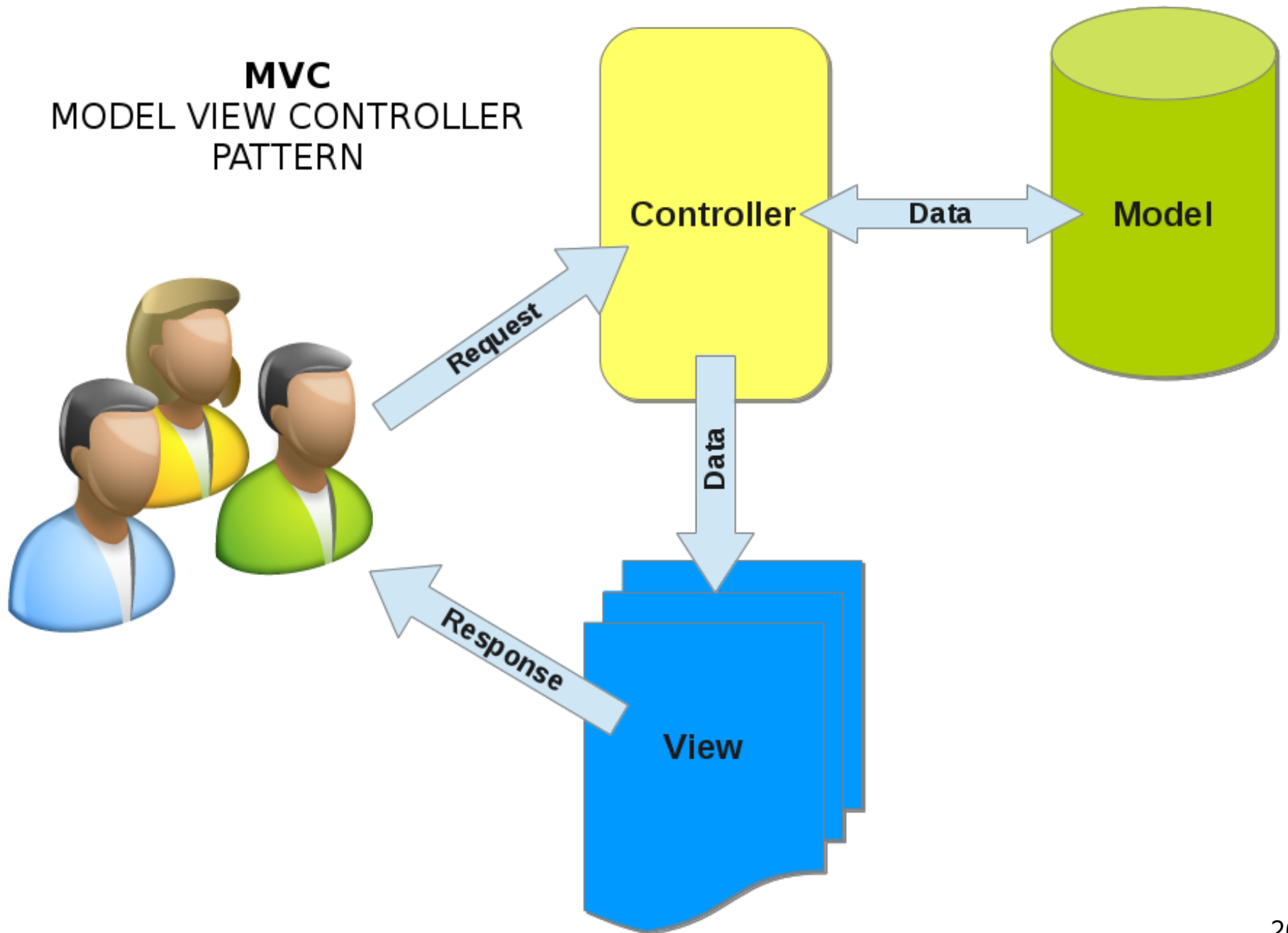
Web Application Layers

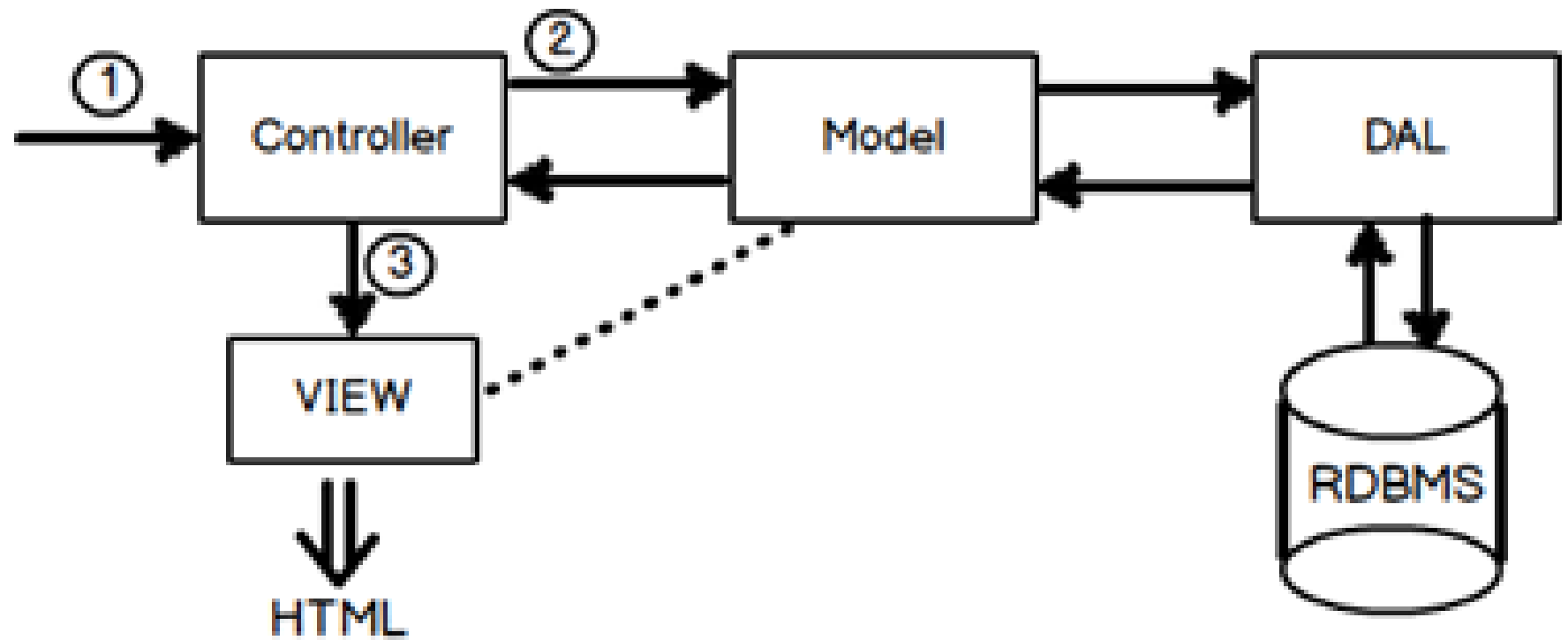
- ❑ Web apps are typically divided into 3 layers:
 - **Presentation layer:** Enable users to interact with the app
 - **Business logic layer:** Enables to control the flow of execution and communication between the presentation layer and data layer.
 - **Data layer:** Enables to provide the app data stored in DBs to the business logic layer.



MVC

MODEL VIEW CONTROLLER
PATTERN





Web App Architectures

- ❑ Architecture of an application depends on the system in which the layers of the application are distributed and communicated to each other.
- ❑ An application can be based on one of the following types of architectures:
 - **Single-tier:** In this architecture, all the three layers are integrated together and installed on a single computer.
 - **Two-tier:** In this architecture, the three layers are distributed over two tiers, a client and a server.
 - **Three-tier:** In this architecture, the three layers of the application are distributed across different computers.
 - **N-tier:** In this architecture, the components of the three-tier are further separated.

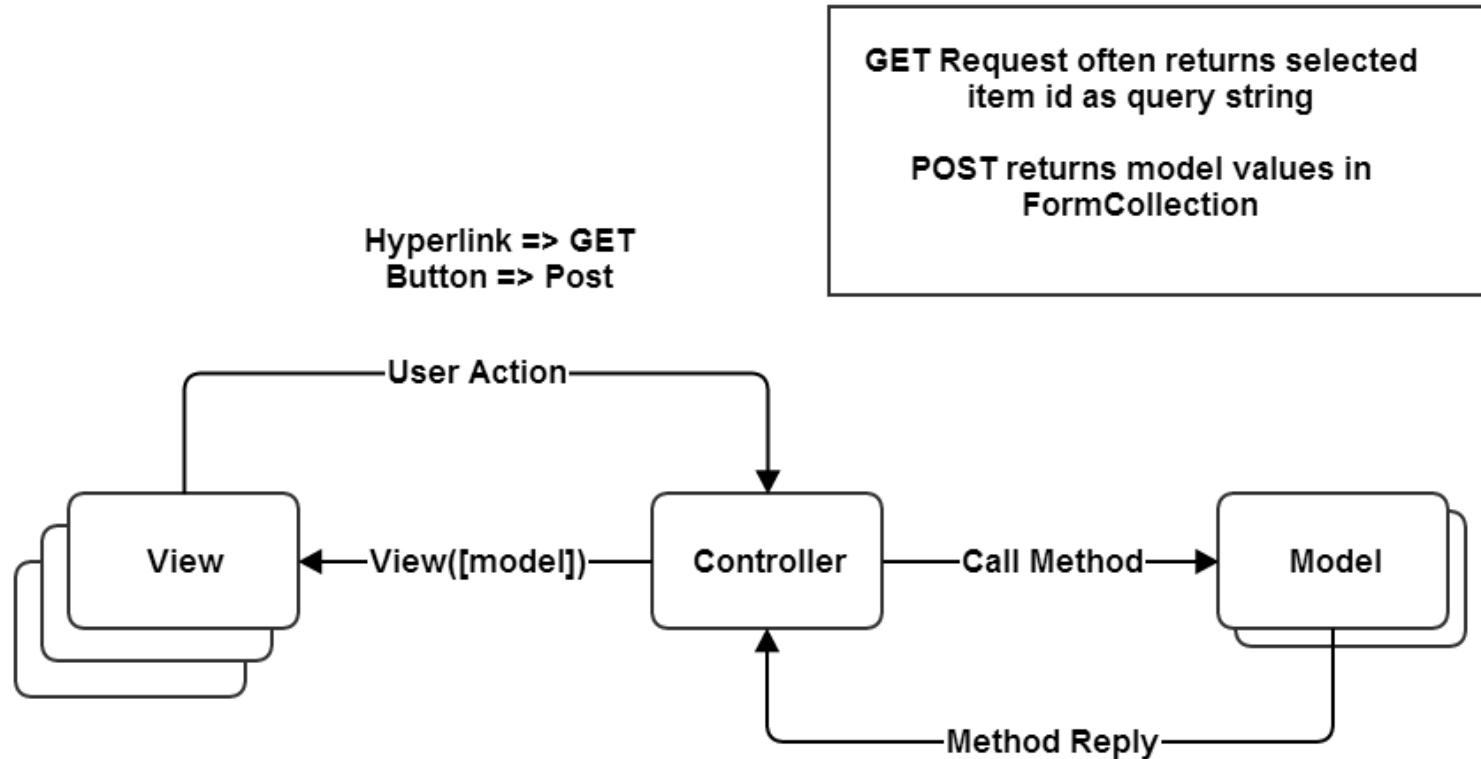
Types of Web Pages

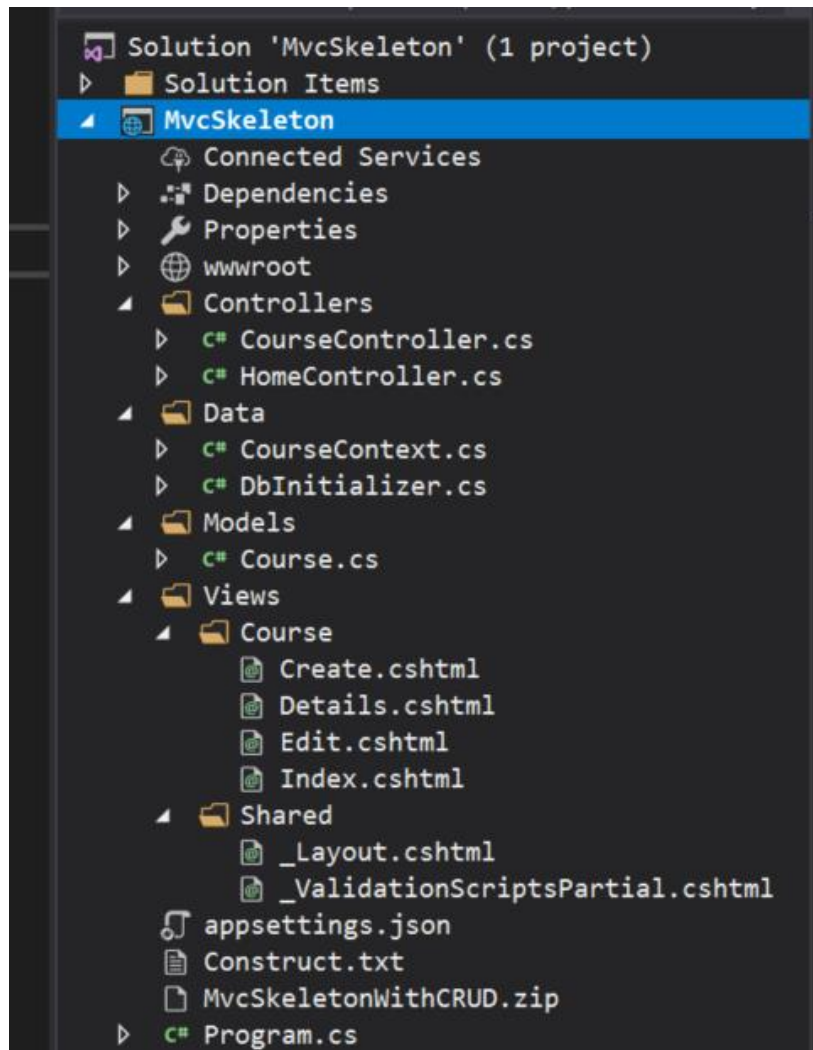
- ❑ A Web app consists of Web pages.
- ❑ A Web page can be categorized on the following two types:
 - **Static Web page:** Consist of only HTML to present content to users
 - **Dynamic Web page:** Consists of HTML in combination with server-side and client-side scripts to respond to user actions.

What is Asp.Net Core MVC?

- Framework for building web applications
- Based on Model-View-Controller pattern
 - Model manages the application data and enforces constraints on that model.
 - Often accessed through persistent objects
 - Views are mostly passive presentations of application state.
 - Views generate requests sent to a controller based on client actions.
 - Controllers translate requests into actions on the data model and generate subsequent views.

MVC Structure





Mvc Structure

- Controllers
 - Connect Views to Data
- Models
 - Provide structured data, usually persisted to a db
 - Accessed through C# class instances
- Views
 - Combine markup and C# code to display and accept data.

MVC Life Cycle

- ❑ Clients request a named action on a specified controller, e.g.:
 - <http://localhost/aController/anAction>
- ❑ The request is routed to aController's anAction method.
 - That method decides how to handle the request, perhaps by accessing a model's state and returning some information in a view.
 - User actions in the view, e.g., data entered, button presses, result in get (ActionLink) or post (Button) requests to a specific controller action.
 - That process may repeat for many cycles.

What is a Model?

- A model is a file of C# code and often an associated data store, e.g., an SQL database or XML file.
 - The file of C# code manages all access to the application's data through objects.
 - Linq to SQL and Linq to XML can be used to create queries into these data stores
 - This can be direct
 - More often it is done through objects that wrap db tables or XML files and have one public property for each attribute column of the table.

MvcSkeleton with CRUD Model

```
namespace MvcSkeleton.Models
{
    //////////////////////////////////////
    // Course class - an item for CourseList

    public class Course
    {
        [DatabaseGenerated(DatabaseGeneratedOption.Identity)]
        public int Id { get; set; }
        public string Number { get; set; }
        public string Name { get; set; }
        public string Instructor { get; set; }
    }
}
```

Adding a Model

- ❑ Right-click on Model folder and select Add Class.
 - Populate the model class with public properties that represent data to be managed.
 - Usually, the model is persisted to an XML file or SQL database using LINQ or the Entity Data Framework.

What is a View?

- Views are cshtml files with only HTML and inline C# code, e.g., `<td>@crs.Number, @crs.Name`
 - Code is used just to support presentation and does no application processing.
 - The HTML is augmented by HTML Helpers, provided by Asp.Net Core MVC that provide shortcuts for commonly used HTML constructs, e.g.:

```
@Html.ActionLink("Edit", "Edit", new { id = crs.Id })
```
 - Asp.Net MVC also provides tag helpers that translate into pure markup, e.g.:

```
<input asp-for="Name" />
```

Create View

```
<div class="indent">
  <p>
    <a asp-action="Create">Create New</a>
  </p>
  <table class="table">
    <tbody>
      @foreach (var crs in Model)
      {
        <tr>
          <td>
            @crs.Number, @crs.Name
          </td>
          <td>
            @Html.ActionLink("Edit", "Edit", new { id = crs.Id /* id=item.PrimaryKey */ }) |
            @Html.ActionLink("Details", "Details", new { id = crs.Id /* id=item.PrimaryKey */ }) |
            @Html.ActionLink("Delete", "Delete", new { id = crs.Id /* id=item.PrimaryKey */ })
          </td>
        </tr>
      }
    </tbody>
  </table>
</div>
```


Views are results of Controller actions (methods)

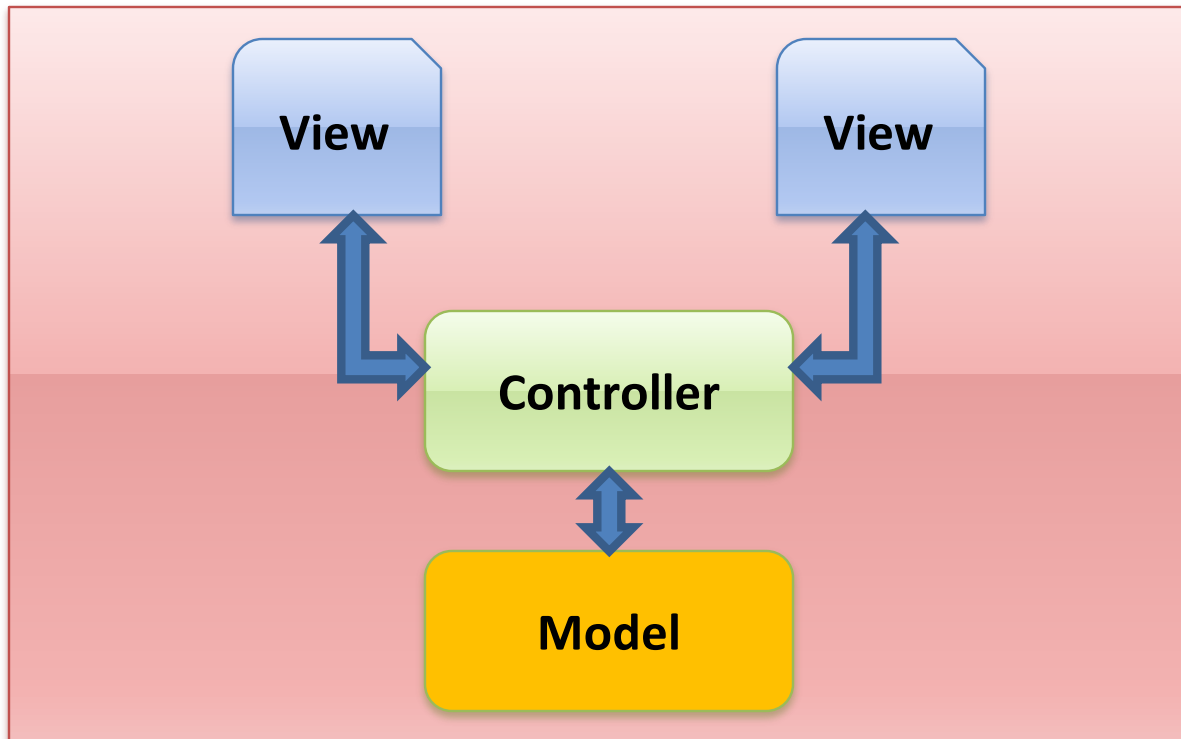
```
[HttpGet]
public IActionResult Edit(int? id)
{
    if (id == null)
    {
        return StatusCode(Microsoft.AspNetCore.Http.StatusCodes.Status400BadRequest);
    }
    Course course = context_.Courses.Find(id);
    if (course == null)
    {
        return StatusCode(StatusCodes.Status404NotFound);
    }
    return View(course);
}
```

ASP.NET MVC

- ❑ Based on the MVC design pattern that allows you to develop software solutions.
- ❑ MVC design pattern:
 - Allows you to develop Web App with loosely coupled components
 - Enable separating data access, business, and presentation logic from each other.
- ❑ While using the MVC design pattern, a Web app can be divided into following types:
 - Model: Represents information about a domain that can be the app data of a Web app.
 - View: Represents the presentation logic to provide the data of the model.
 - Controller: Represents the logic responsible for coordination between the view and model classes.

ASP.NET MVC

- ❑ Following figure shows the communications between the model, view and controller components:



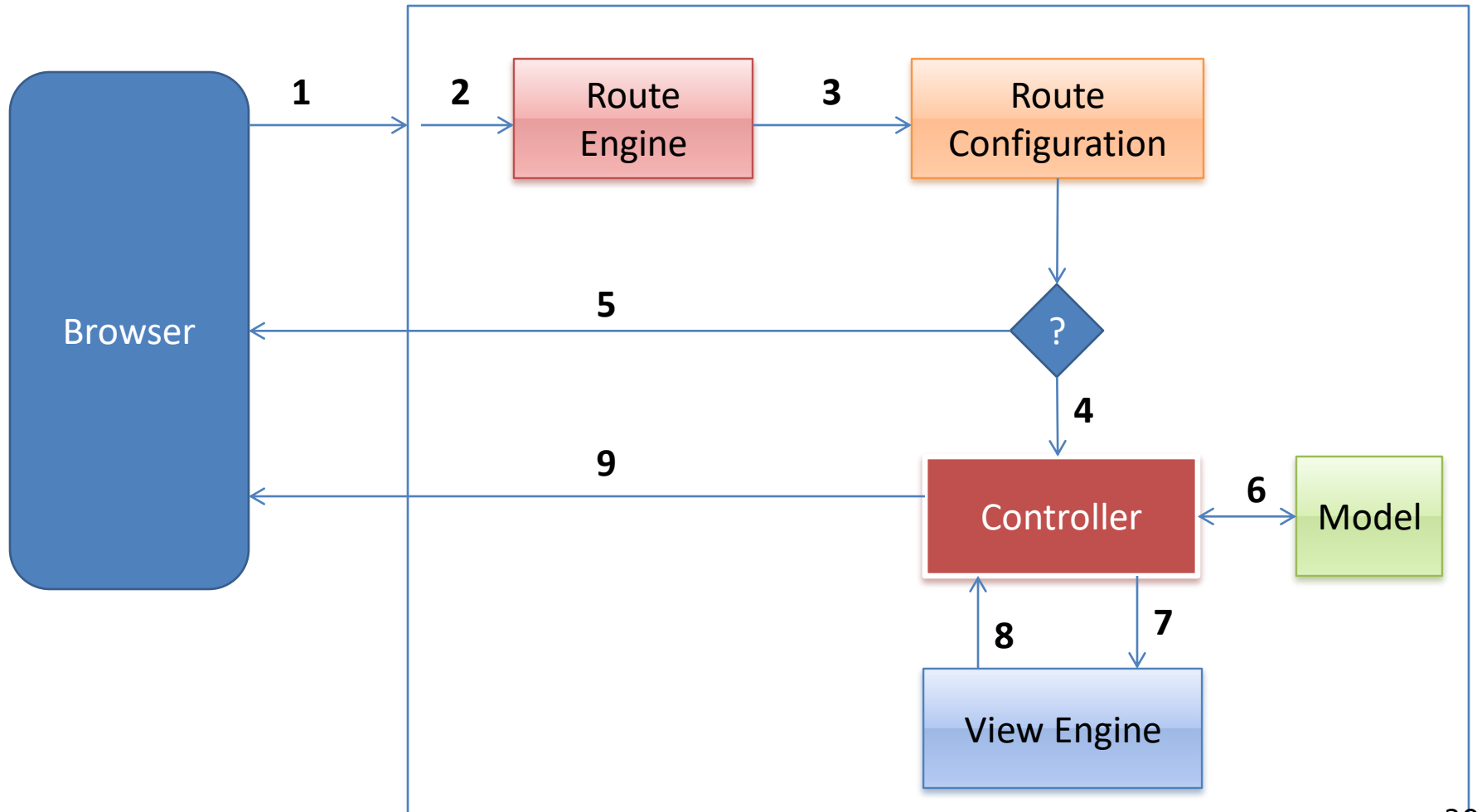
ASP.NET MVC

- ❑ As ASP.NET MVC is based on the MVC design pattern, it provides the following benefits:
 - Separation of concerns: Enables you to ensure that various app concerns into different and independent software components.
 - Simplified testing and maintenance: Enable you to test each component independently. Thus, it allows you to ensure that it is working as per the requirement of the app and then, simplifies the process of testing, maintenance, and troubleshooting procedures.
 - Extensibility: Enables the model to include a set of independent components that you can easily modify or replace based on the app requirement.

Architecture of ASP.NET MVC App

- ❑ The steps that the components of the ASP.NET MVC Framework performs while handling an incoming request includes:
 - The browser sends a request to an ASP.NET MVC app.
 - The MVC Framework forwards the request to the routing engine.
 - The route engine checks the route configuration of the app for an appropriate controller to handler the request.
 - When a controller is found it is invoked
 - When a controller is not found \Rightarrow an error to the browser
 - The controller communicates with the model
 - The controller requests a view engine for a view based on the data of the model.
 - The view engine returns the result to the controller
 - The controller sends back the result as an HTTP response to the browser

Architecture of ASP.NET MVC App



Supporting Technologies

- ❑ ASP.NET MVC app support various technologies to create dynamic and responsive Web app.
- ❑ Some of the supporting technologies that you can use while creating an ASP.NET MVC application are as follows:
 - JavaScript
 - JQuery
 - AJAX
 - IIS
 - Windows Azure

Q&A

