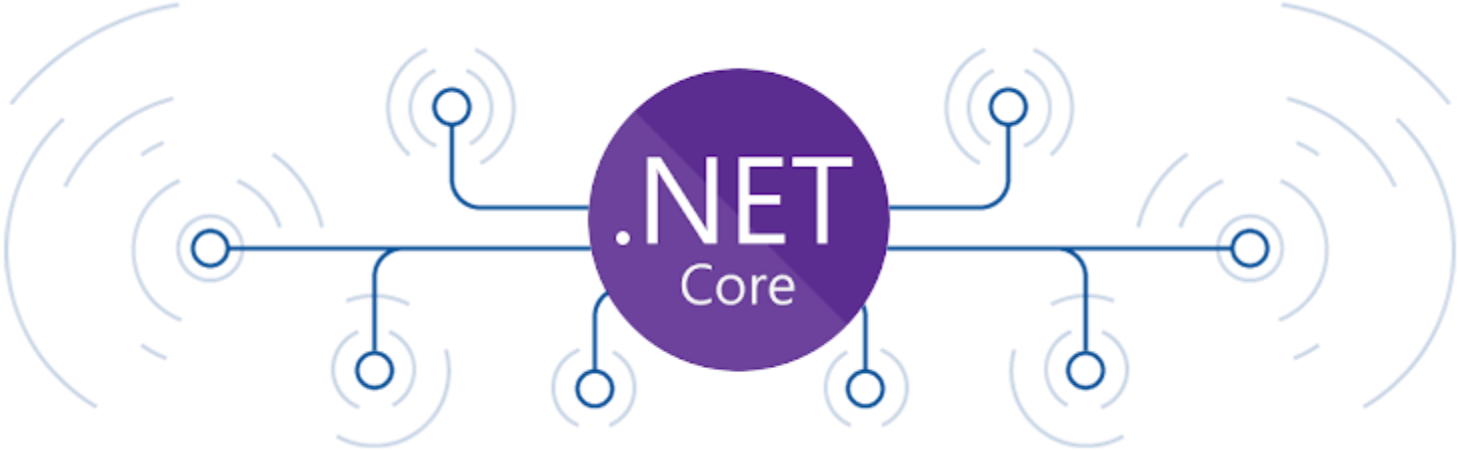
Application Development

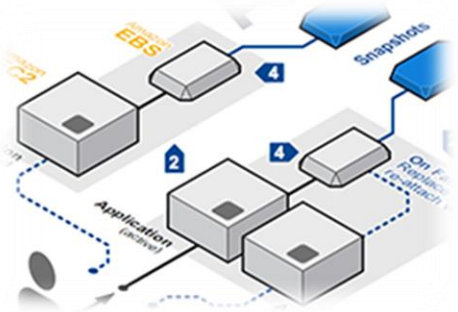


https://bit.ly/niiemvc

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

1. Introduction to ASP.NET Core
2. MVC
3. Layout
4. Entity Framework Core
5. Web API
6. 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# languague
- 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

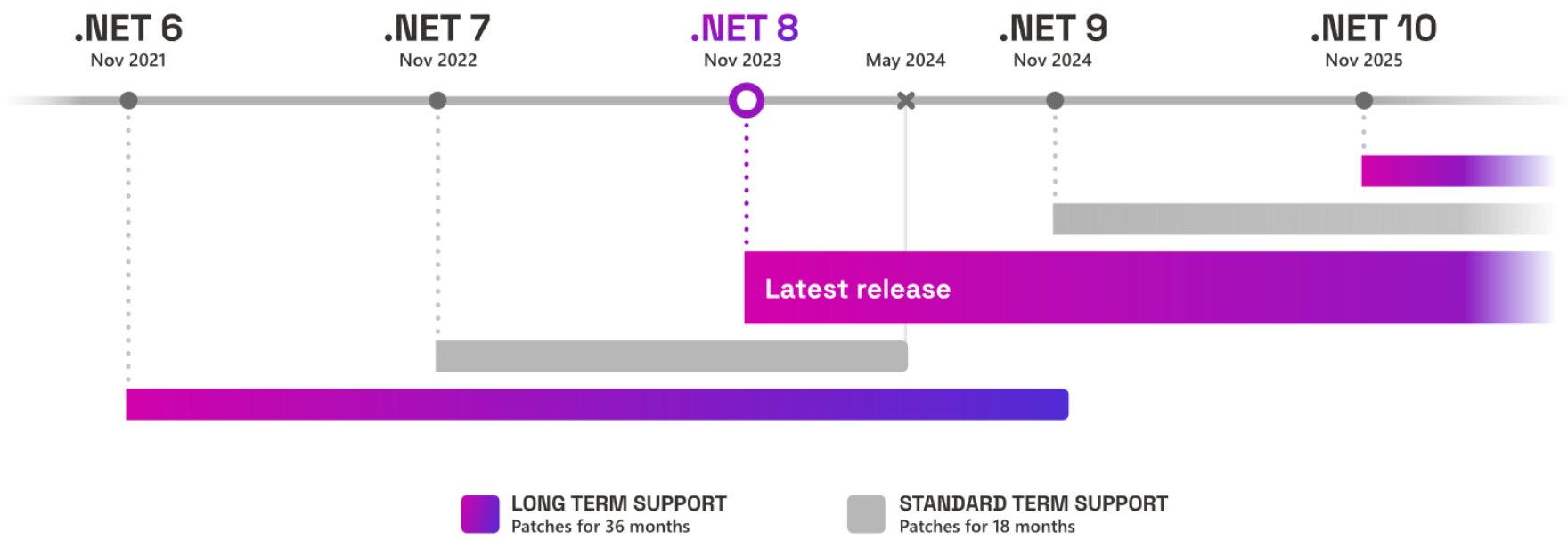

Choose your Editors
and Tools

Open Source
with Contributions


Cross-Platform


Fast

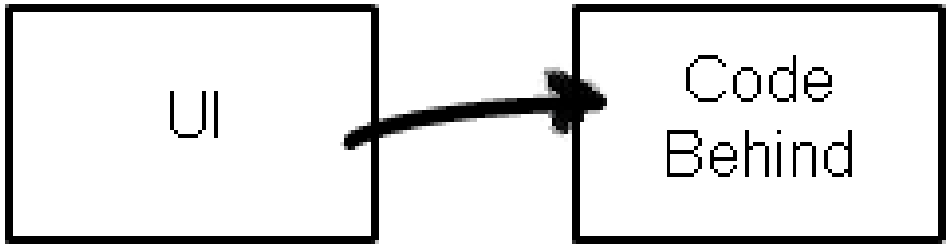
Net core version



https://dotnet.microsoft.com/en-us/platform/support/policy/dotnet-core

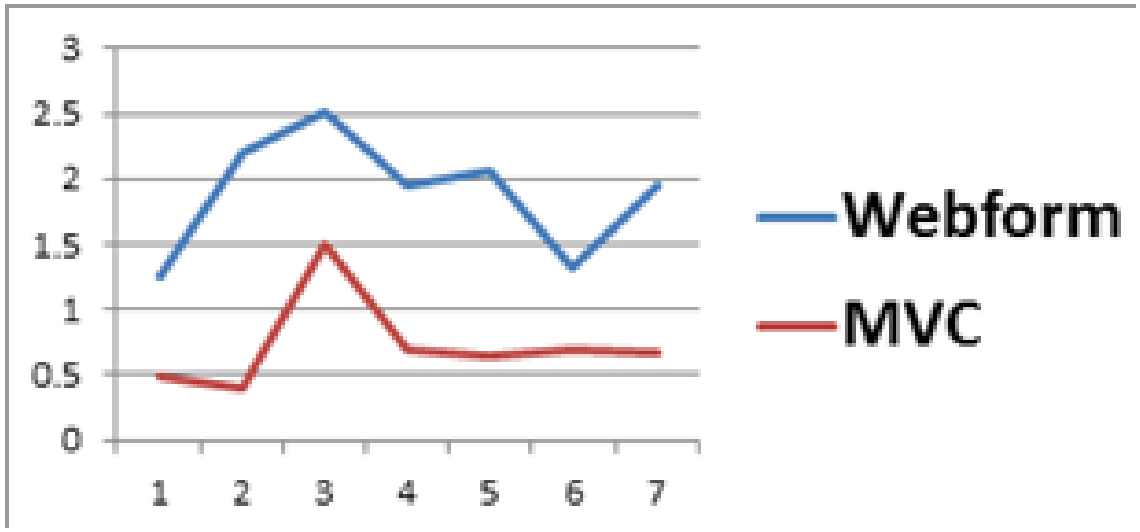
Why?

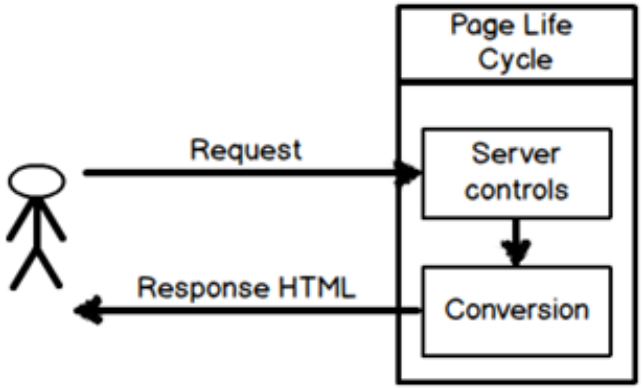
- Web Form



Problems with Asp.Net Web Forms

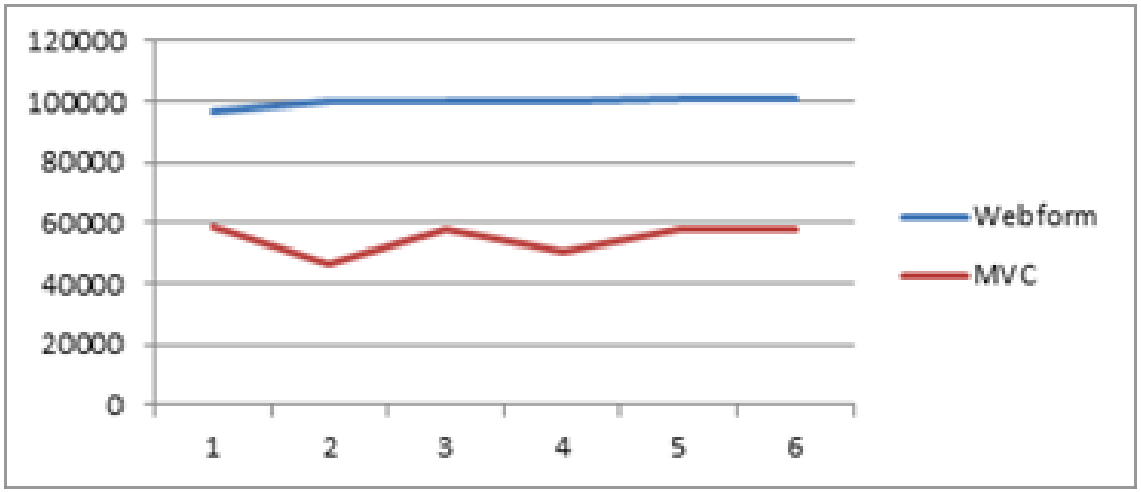
- Response time issues





Problems with Asp.Net Web Forms

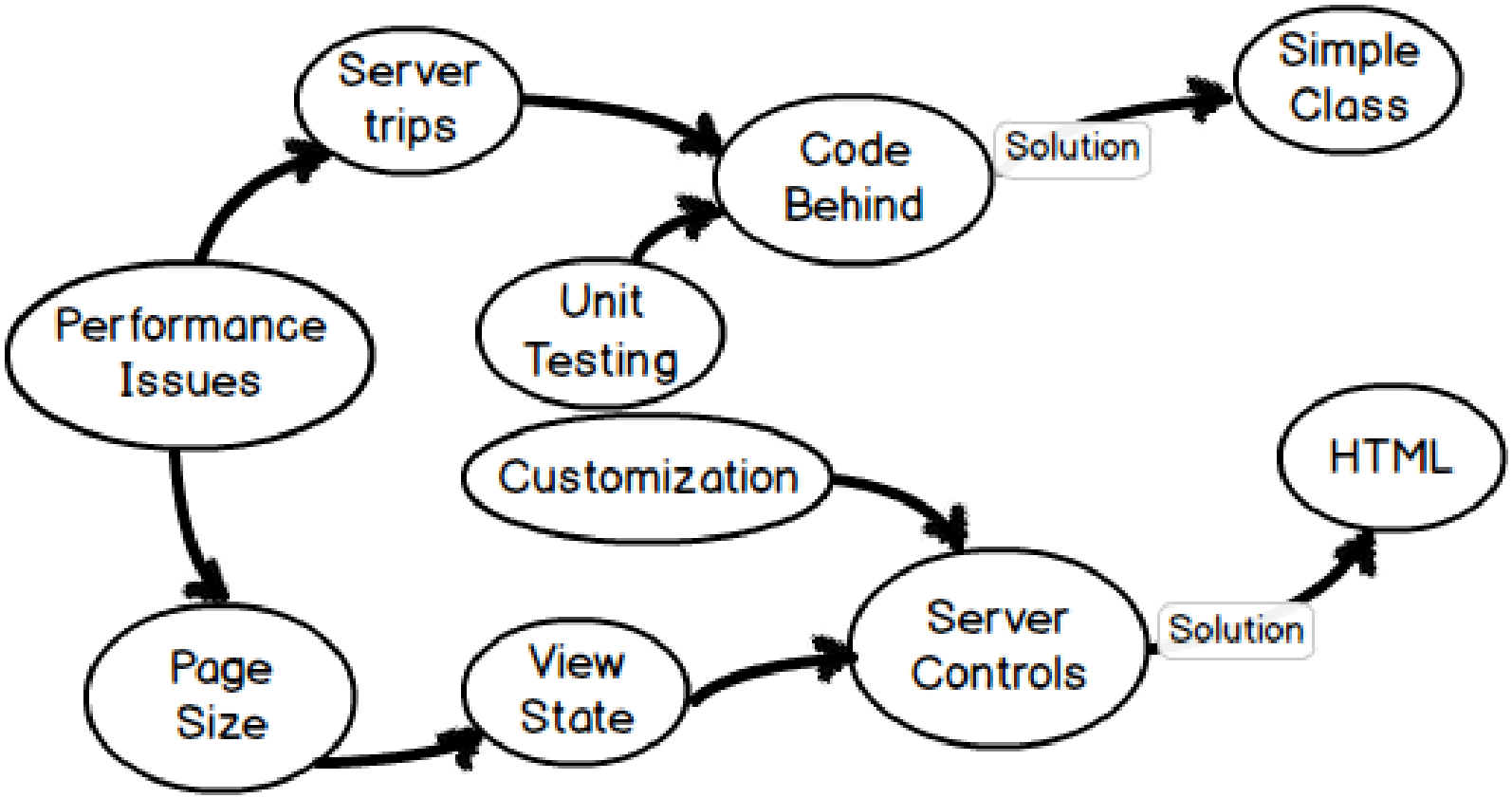
- Bandwidth consumption

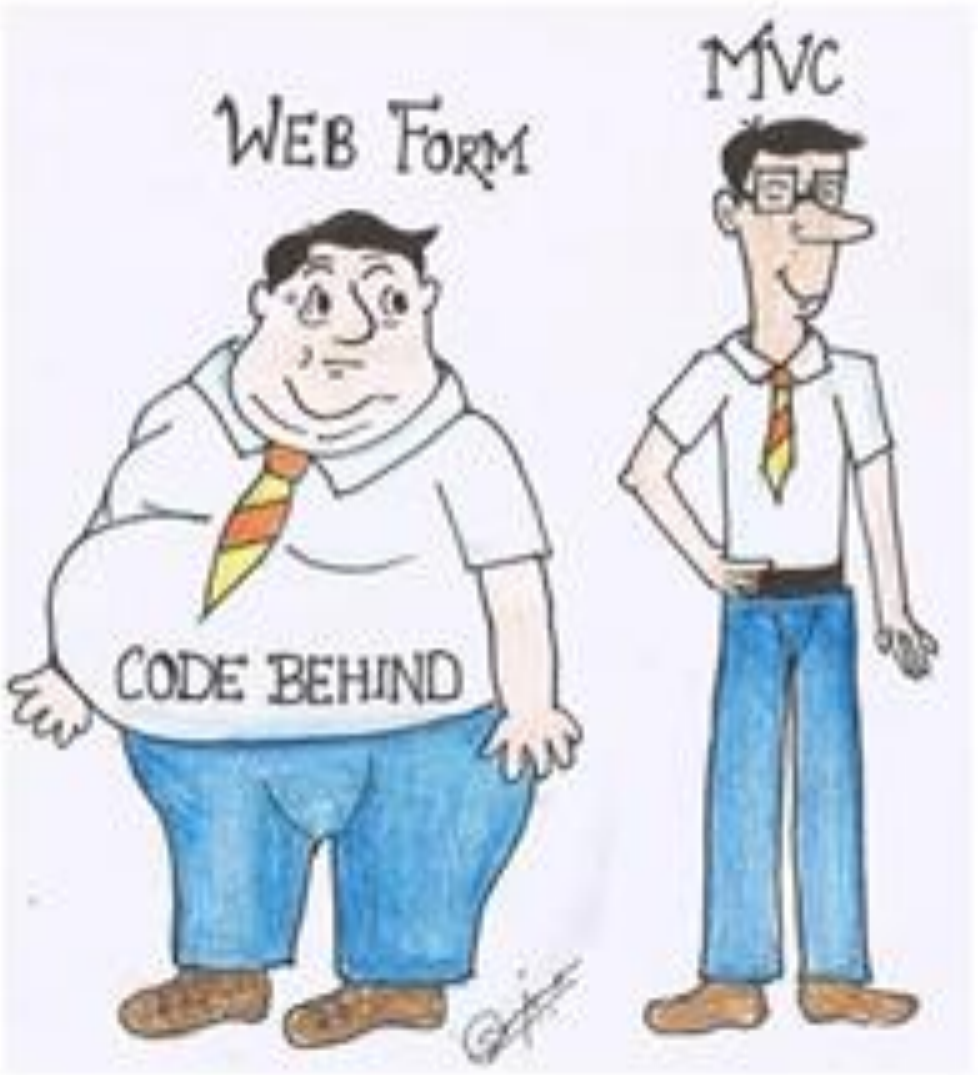


ViewState : Web Form

Problems with Asp.Net Web Forms

- Unit Test




WEB FORM
MVC
CODE BEHIND

Introduction

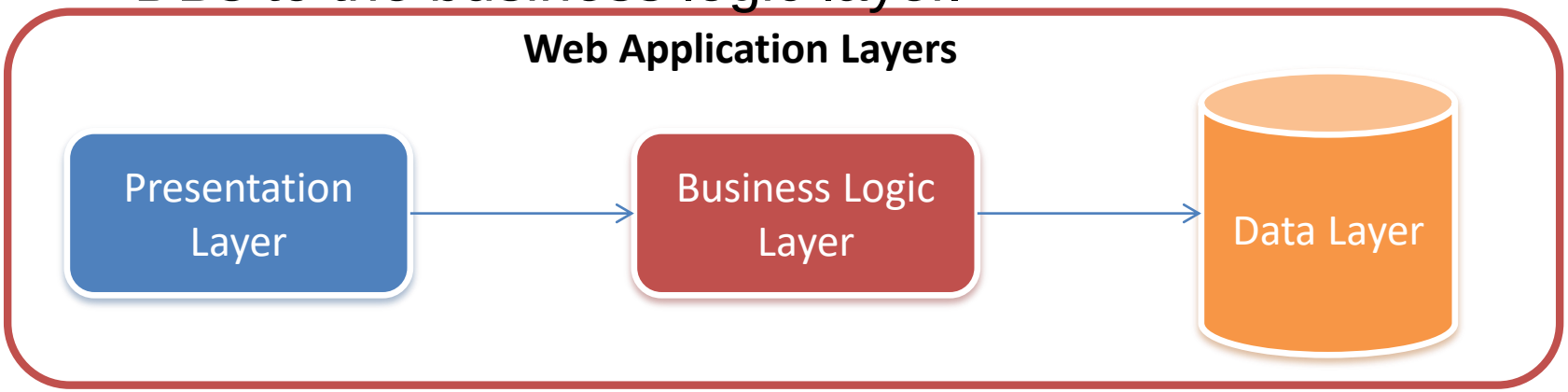
- Overview of Web Application Development
- Introduction to ASP.NET MVC
- Architecture of ASP.NET MVC App
- Creating an ASP.NET MVC Project
- Structure of an ASP.NET MVC Project

Overview of Web Application Development

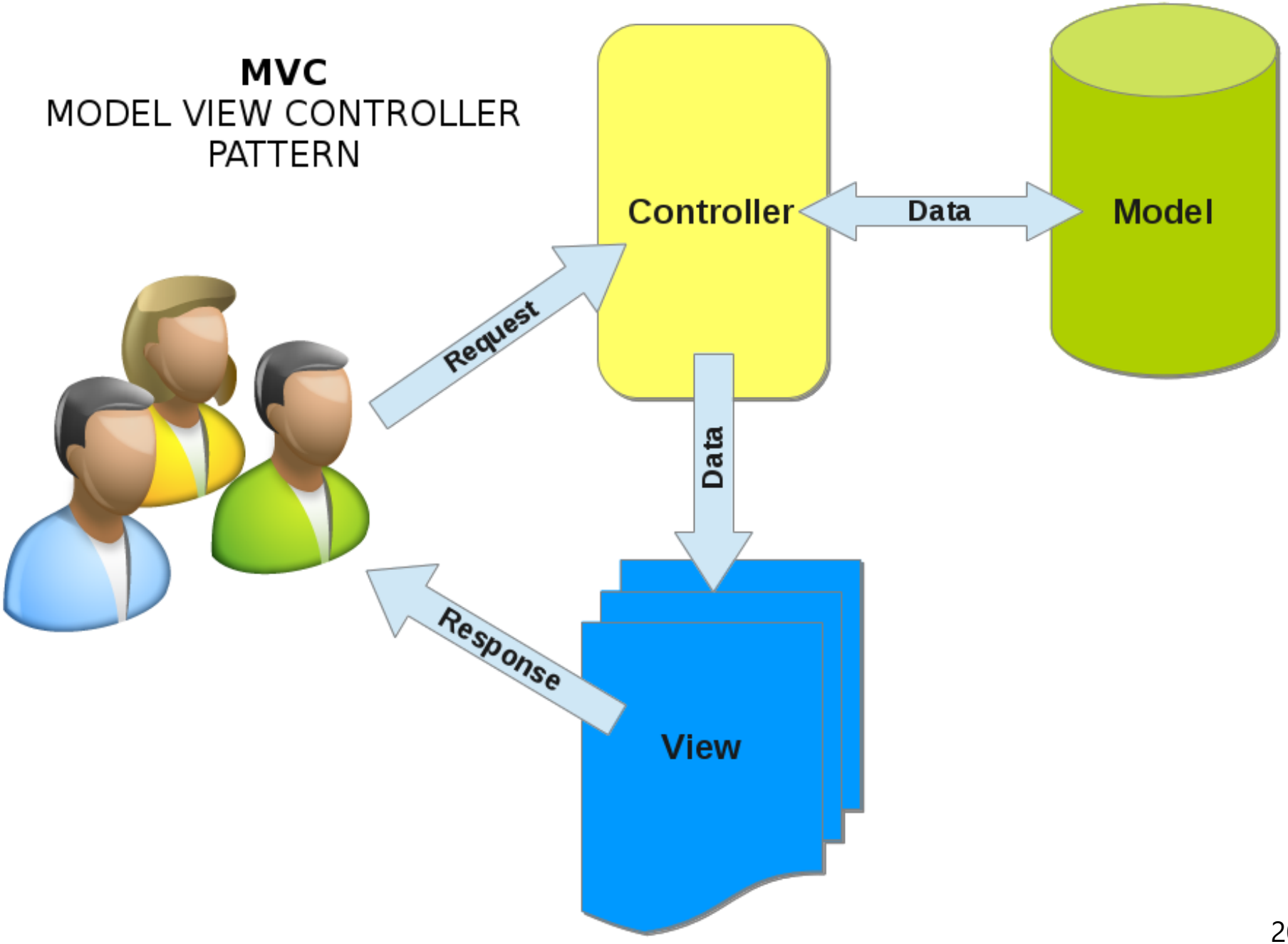
- Web Application
  - Are programs that are executed on a Web server and accessed from a Web browser.
  - Allows you to share and access information over the Internet that can be accessed globally at any time.
  - Enable you to perform commercial transactions known as E-commerce application.

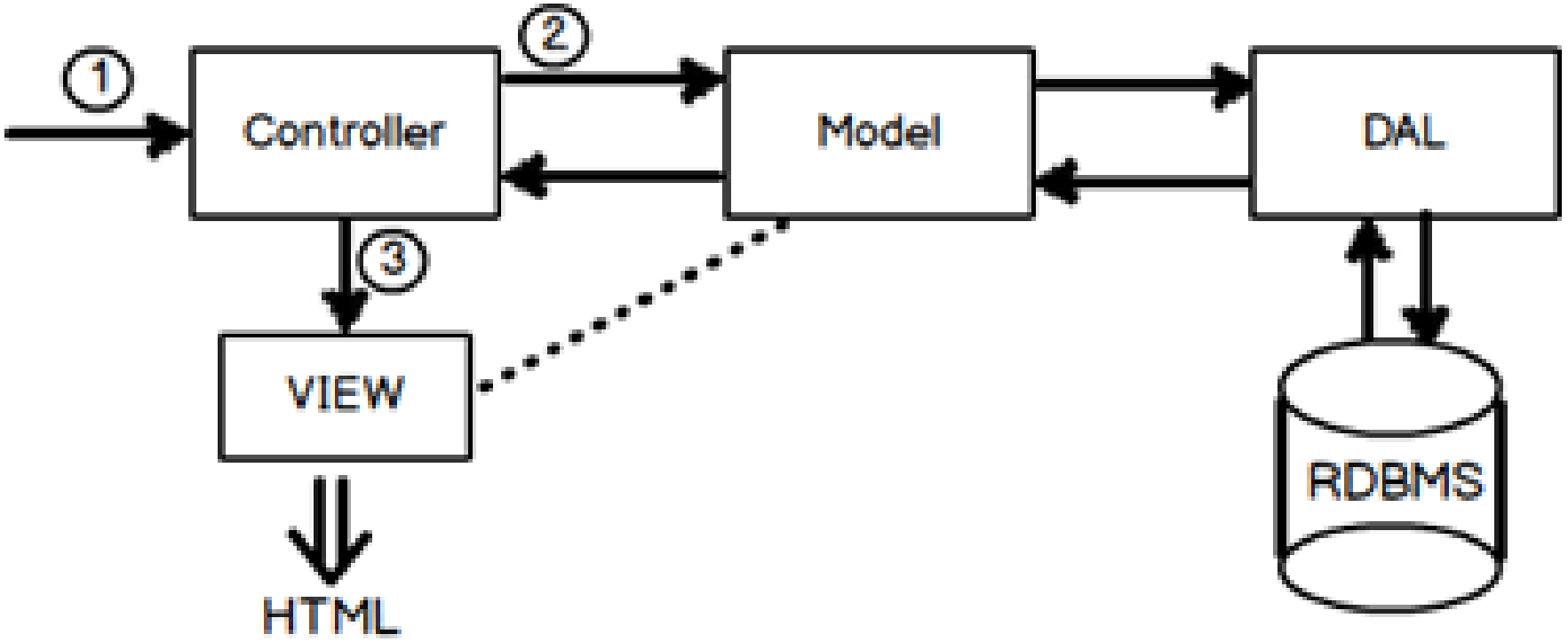
Web Application Layers

- 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




1
2
Controller
Model
DAL
3
VIEW
RDBMS
HTML

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.

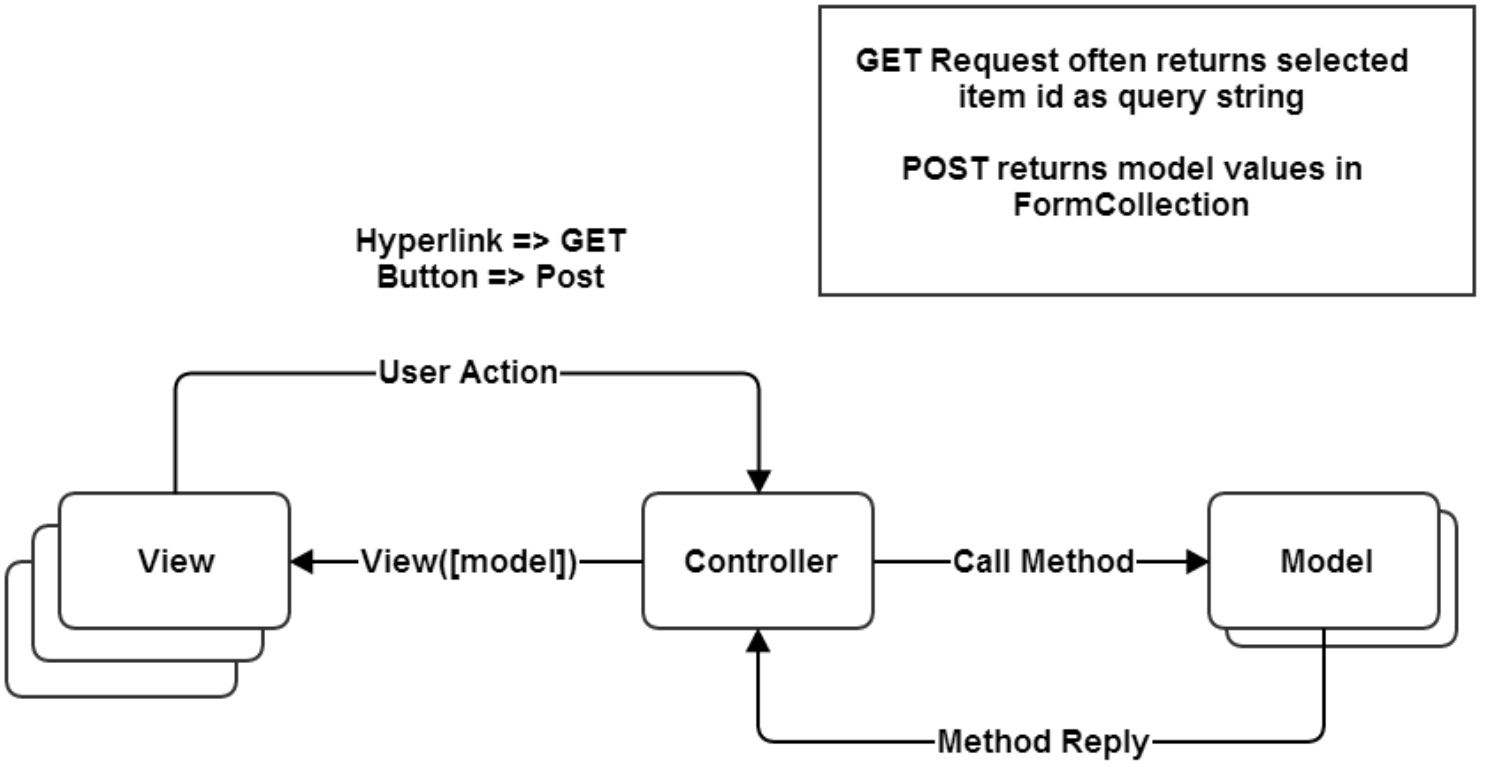
Types of Web Pages

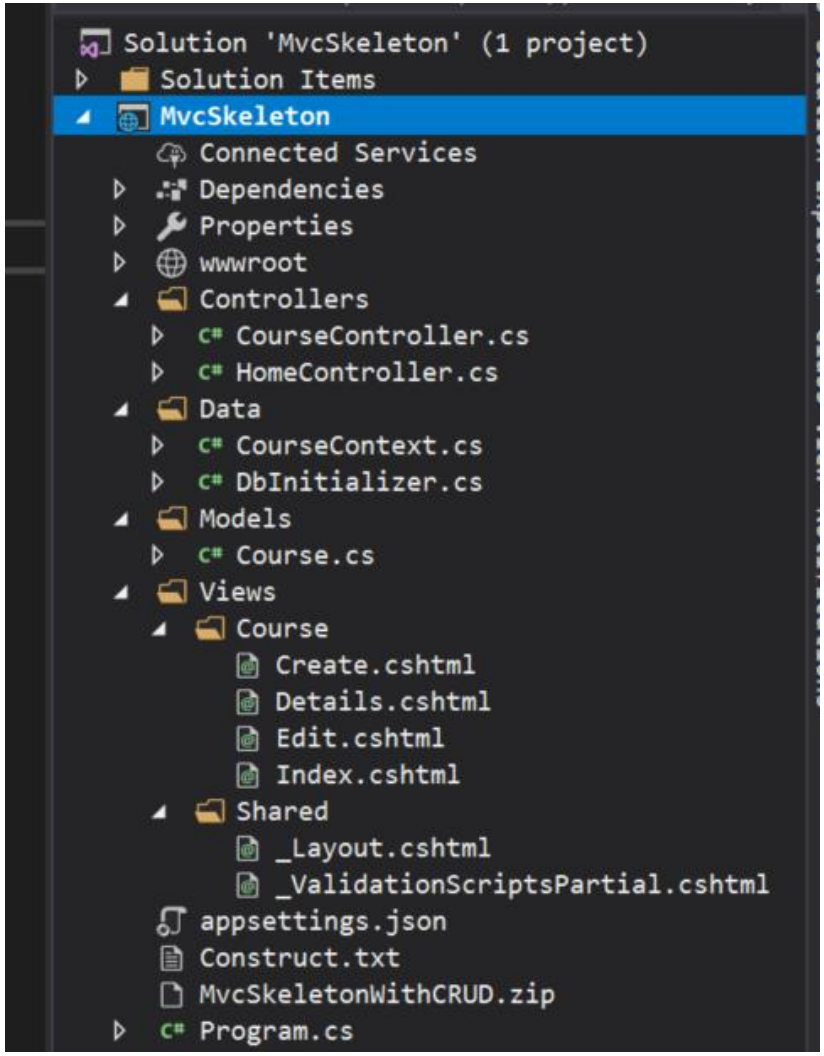
- A Web app consists of Web pages.
- A Web page can be categorized on the following two types:
  - **Static Web page**: Consist of only HTML to present content to users
  - **Dynamic Web page**: Consists of HTML in combination with server-side and client-side scripts to respond to user actions.

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)

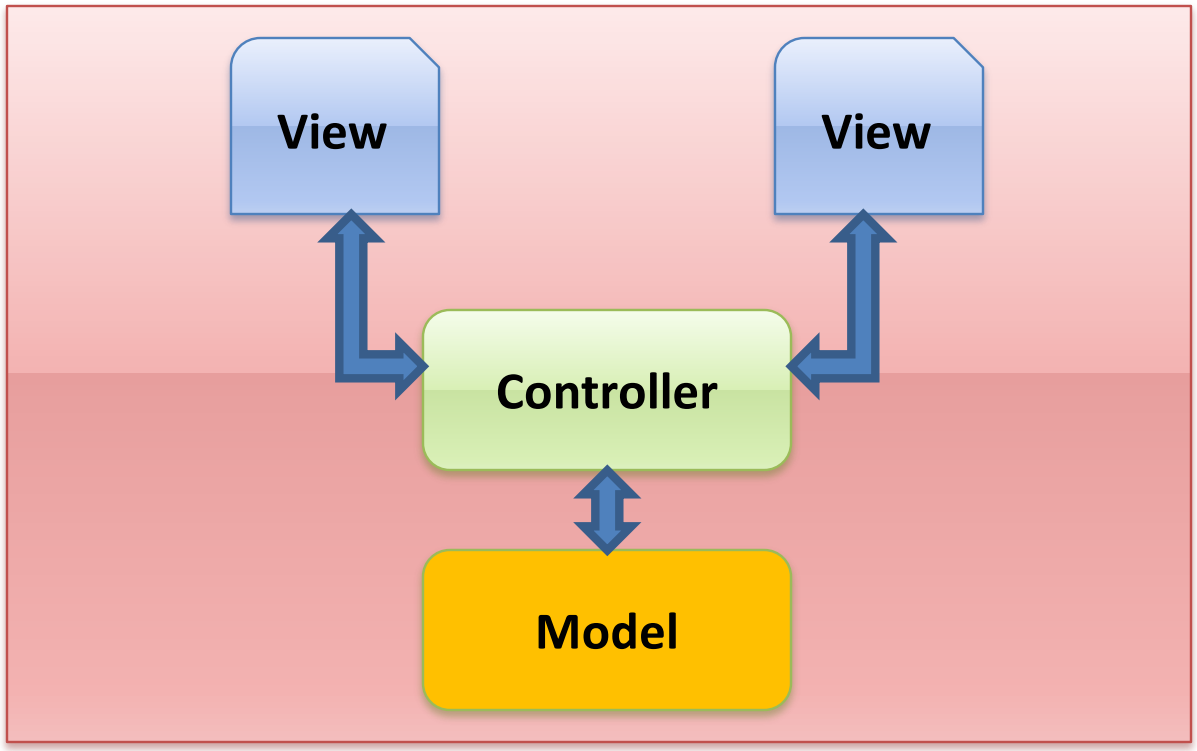
```
[HttpGet]
public IActionResult Edit(int? id)
{
    if (id == null)
    {
        return StatusCode(Microsoft.AspNetCore.Http.StatusCodes.Status400BadRequest);
    }
    Course course = context_.Courses.Find(id);
    if (course == null)
    {
        return StatusCode(StatusCodes.Status404NotFound);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



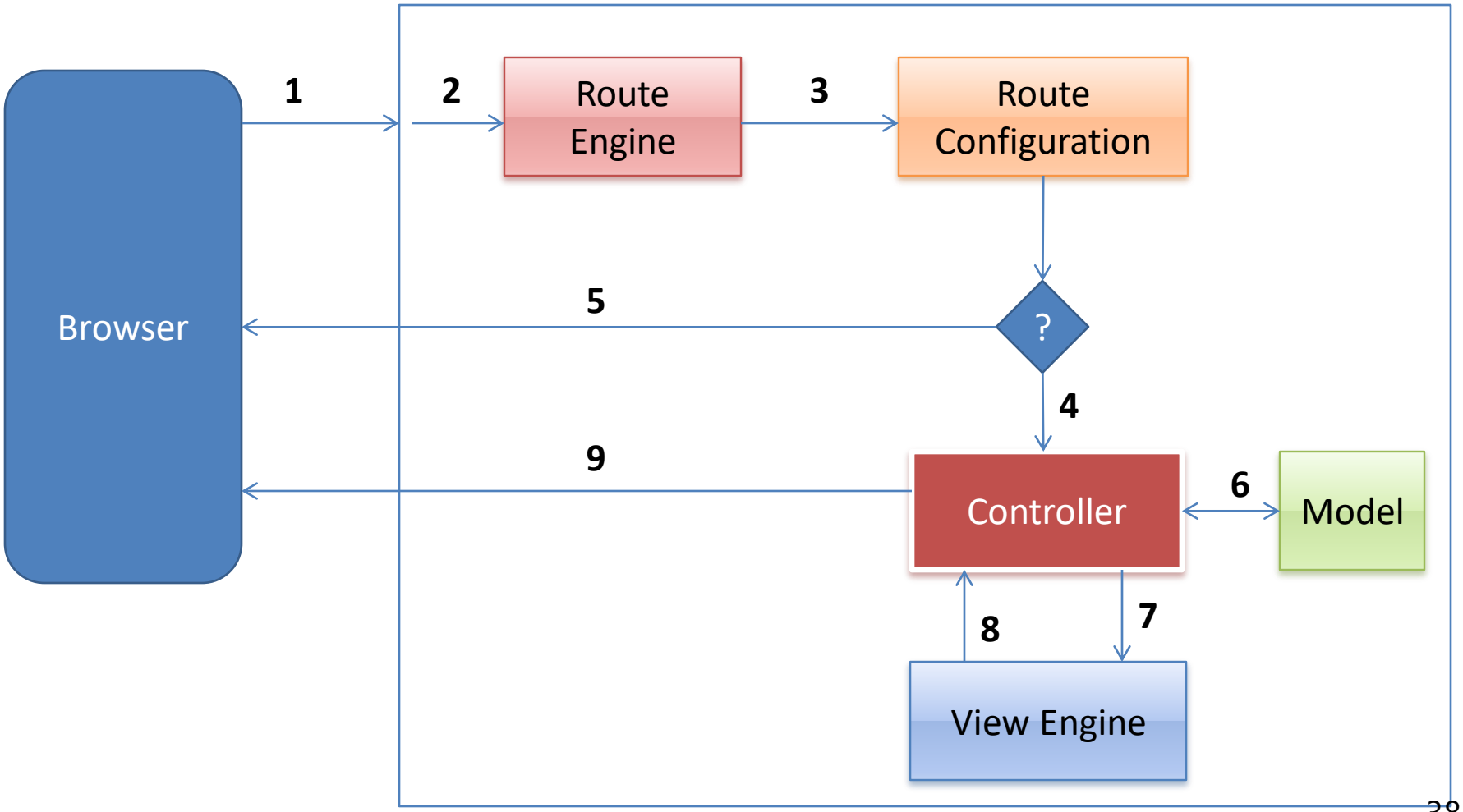
ASP.NET MVC

- 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 an 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