



Școala
informală
de IT

Introduction to ASP.NET MVC Core



Curriculum

1. The HTTP Protocol

2. The MVC Pattern

- Model, View, Controller
- The MVC Pattern for Web and Examples

3. ASP.NET MVC Core

- ASP.NET MVC Core Advantages

4. Creating ASP.NET MVC Project

5. NuGet Package Management

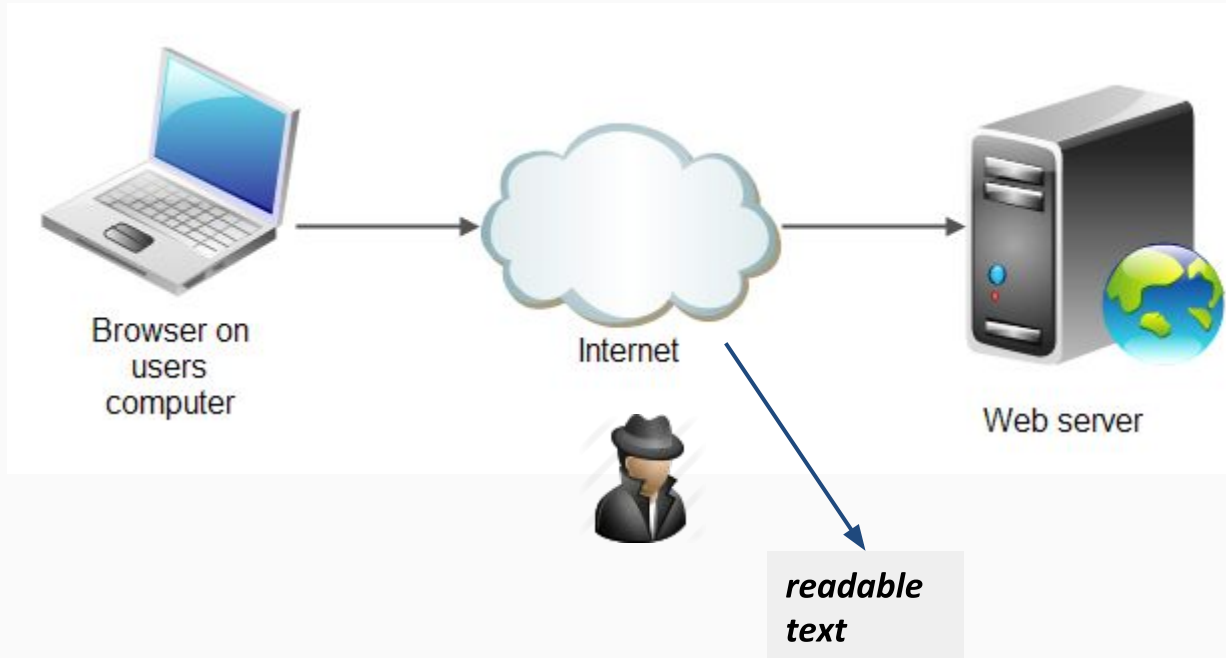


The HTTP Protocol



HTTP Protocol

HTTP = HyperText Transfer Protocol



HTTP Protocol

- Client-server protocol for transferring Web resources (HTML files, images, styles, data, etc.)
- Important properties of HTTP
 - Request-response model
 - Text-based format
 - Connectionless (after a request the client disconnects from the server and waits for a response)
 - Relies on a unique resource URLs
 - **Stateless** (browser can retain information between different requests across the web pages)

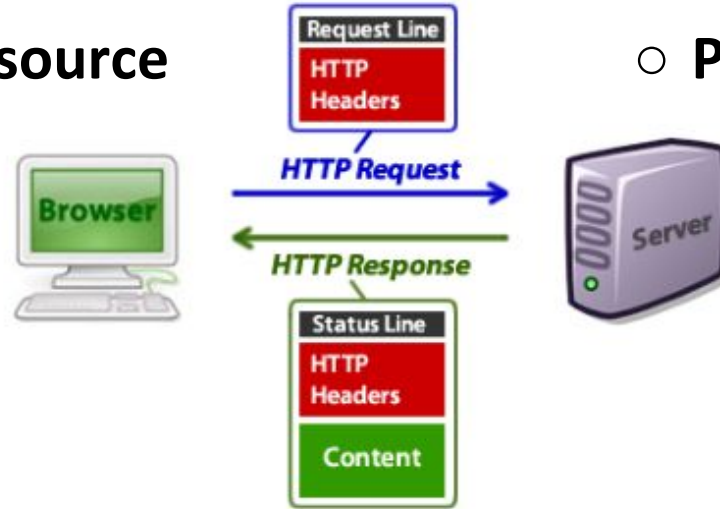
HTTP: Request - Response Protocol

- Client program

- Running on end host
- E.g. Web browser
- Requests a resource

- Server program

- Running at the server
- E.g. Web server
- Provides resources



HTTP: Request - Response Protocol

- HTTP request (GET, POST):

```
GET /Products HTTP/1.1
Host: www.mysite.com
User-Agent: Mozilla/5.0
...
```

```
POST /Product/Edit/3 HTTP/1.1
Host: www.mysite.com
Pragma: no-cache
User-Agent: Mozilla/5.0
...
```

- HTTP response:

```
HTTP/1.1 200 OK
Cache-Control: private
Content-Type: text/html; charset=utf-8
Server: Microsoft-IIS/7.5
Date: Wed, 13 Sep 2017 19:21:58 GMT
Content-Length: 2138
...
<html><title>Hello</title>
Welcome to our site</html>
```

HTTP: Request Message

- Request message sent by a client consists of
 - Request line – request method (GET, POST, PUT, DELETE, HEAD, ...), resource URI, and protocol version
 - Request headers – additional parameters
 - Body – optional data for some verbs
 - E.g. posted form data, files, etc.
 - GET, HEAD verbs don't use request body

```
<request method> <resource> <HTTP/version>  
<headers>  
<request body>
```


HTTP: Response Message

- **Response message sent by the server**
 - **Status line – protocol version, status code, status phrase**
 - **Response headers – provide metadata**
 - **Body – the contents of the response of the requested resource)**

```
<HTTP/version> <status code> <status text>  
<headers>  
<response body>
```

HTTP: Response Codes

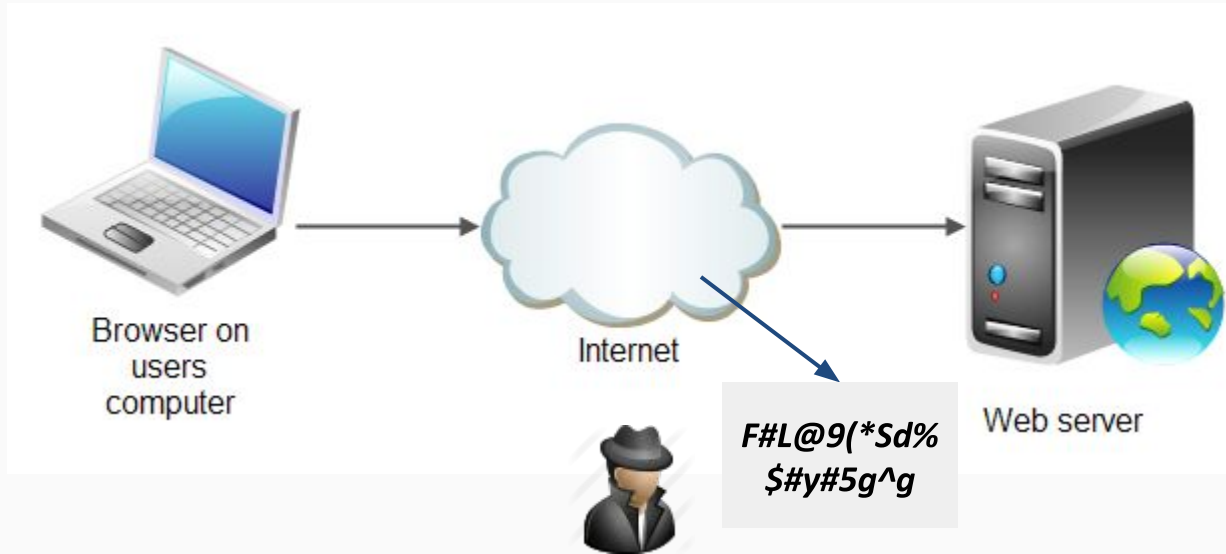
- HTTP response code classes
 - **1xx**: informational (e.g., "**100 Continue**")
 - **2xx**: success (e.g., "**200 OK**")
 - **3xx**: redirection (e.g., "**304 Not Modified**", "**302 Found**")
 - **4xx**: client error (e.g., "**404 Not Found**")
 - **5xx**: server error (e.g., "**503 Service Unavailable**")
- "**302 Found**" is used for redirecting the Web browser to another URL

HTTPS Protocol

HTTPS - Hyper Text Transfer Protocol Secured

SSL - Secure Sockets Layer

The SSL protocol is used to encrypt data for secure data transmission.



New Web Project



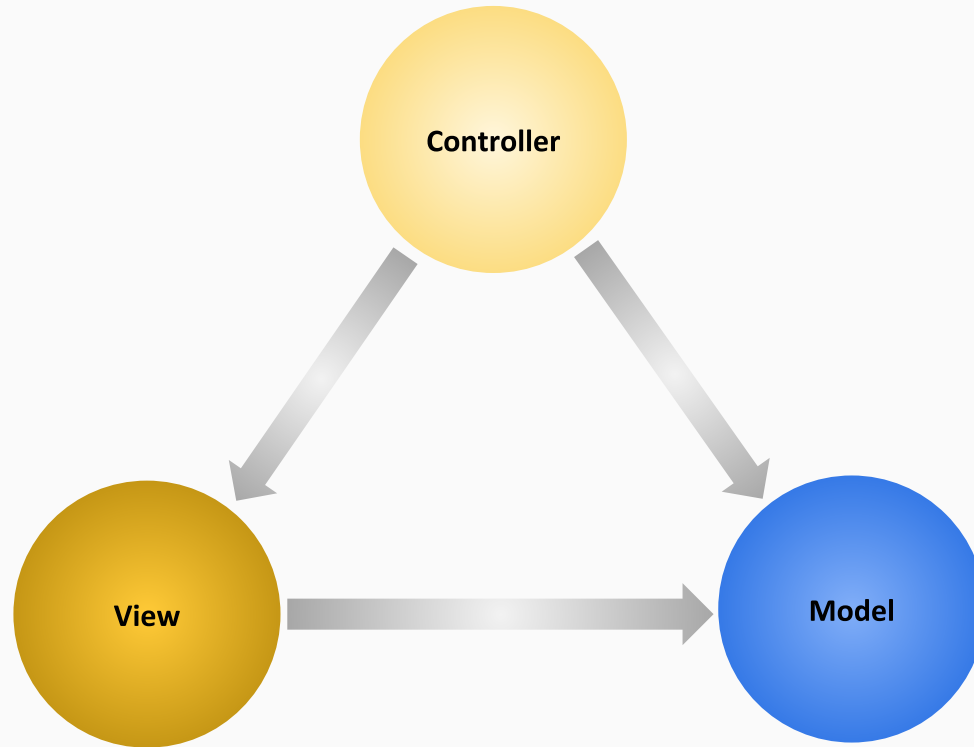
The MVC Pattern

- **New empty web project**
 - Run it F5 and dotnet run
- **New web project**
- **.csproj, Startup.cs, Program.cs**
 - **TargetFramework**

The MVC Pattern



The MVC Pattern



The MVC Pattern

- **Model–view–controller (MVC) is a software architecture pattern**
- **Originally formulated in the late 1970s by Trygve Reenskaug as part of the Smalltalk**
- **Code reusability and separation of concerns**
- **Originally developed for desktop, then adapted for internet applications**

Model

- **Set of classes that describes the data we are working with as well as the business**
- **Rules for how the data can be changed and manipulated**
- **May contain data validation rules**
- **Often encapsulate data stored in a database as well as code used to manipulate the data**
- **Most likely a Data Access Layer of some kind**
- **Apart from giving the data objects, it doesn't have significance in the framework**

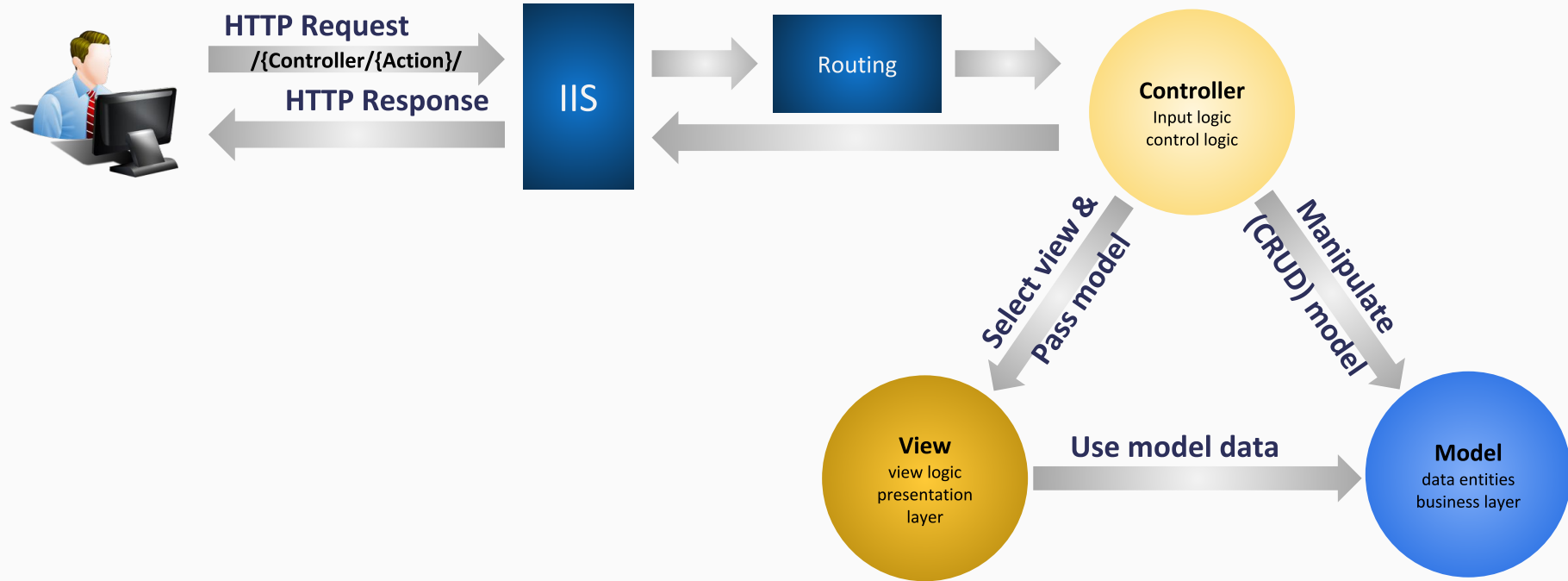
View

- **Defines how the application's user interface (UI) will be displayed**
- **May support master views (layouts) and sub-views (partial views or controls)**
- **Web: Template to dynamically generate HTML**

Controller

- **The core MVC component**
- **Process the requests with the help of views and models**
- **A set of classes that handles**
 - **Communication from the user**
 - **Overall application flow**
 - **Application-specific logic**
- **Every controller has one or more "Actions"**

The MVC Pattern



MVC Steps

- Incoming HTTP request routed to **Controller**
- **Controller** processes request and creates presentation **Model**
 - Controller also selects appropriate result (view)
- **Model** is passed to **View**
- **View** transforms **Model** into appropriate output format (HTML)
- Response is rendered (HTTP Response)

ASP.NET MVC Core



ASP.NET MVC Core

- **Embrace the web**
 - **User/SEO friendly URLs, HTML 5, SPA**
 - **Adopt REST concepts**
- **Uses MVC pattern**
 - **Conventions and Guidance**
 - **Separation of concerns**

ASP.NET MVC Core

- **Tight control over markup**
- **Testable**
- **Loosely coupled and extensible**
- **Convention over configuration**
- **Razor view engine**
 - **One of the greatest view engines**
 - **With intellisense, integrated in Visual Studio**
- **Reuse of current skills (C#, EF, LINQ, JS, etc.)**
- **Application-based**

Extensible

- **Replace any component of the system**
 - **Interface-based architecture**
- **Almost anything can be replaced or extended**
 - **Model binders (request data to CLR objects)**
 - **Action/result filters (e.g. OnActionExecuting)**
 - **Custom action result types**
 - **View engine (Razor, WebForms)**
 - **View helpers (HTML, AJAX, URL, etc.)**
 - **Custom data providers (ADO.NET), etc.**

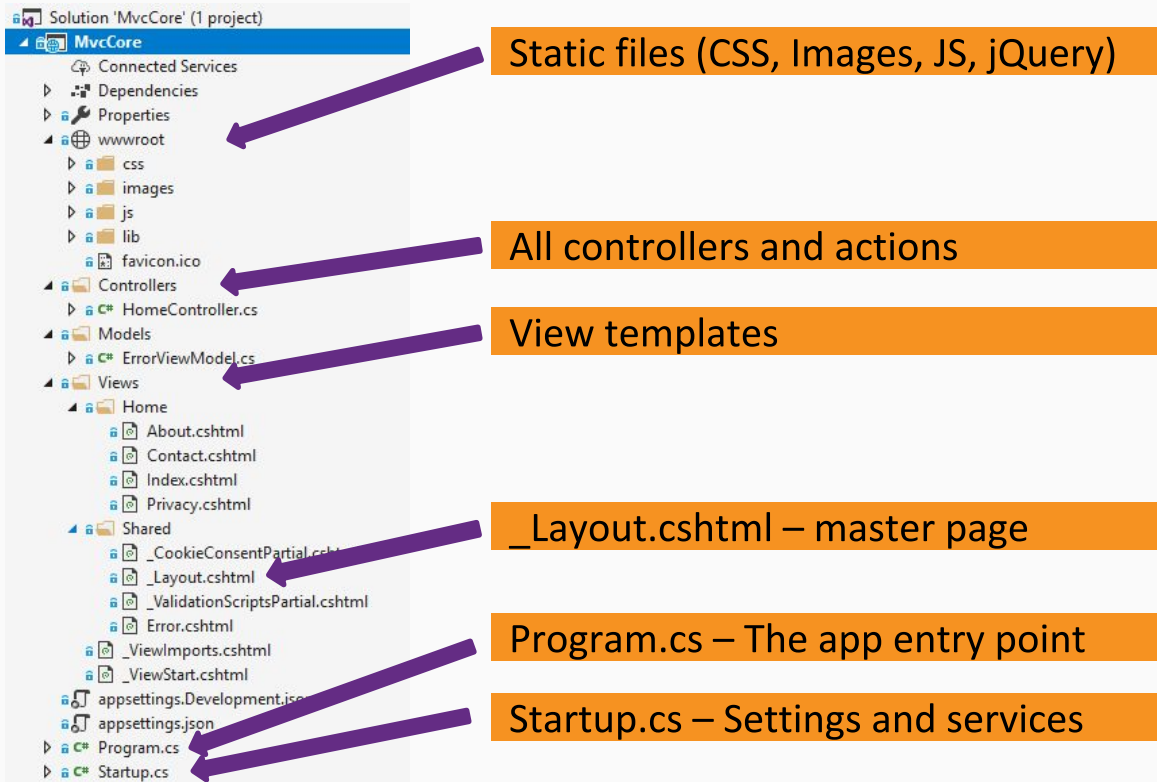
Creating ASP.NET MVC Core Project



The Technologies

- **Technologies that ASP.NET MVC Core uses**
 - **C# (OOP, unit testing, async, etc.)**
 - **ASP.NET Core**
 - **HTML 5 and CSS 3, Bootstrap**
 - **JavaScript (jQuery, AngularJS, ReactJS etc.)**
 - **AJAX, SPA (Single-page apps)**
 - **Databases (MS SQL)**
 - **ORM (Entity Framework and LINQ)**
 - **Web and HTTP**

Internet App Project Files



The image shows a file explorer for a project named 'MvcCore'. The project structure is as follows:

- Solution 'MvcCore' (1 project)
 - MvcCore
 - Connected Services
 - Dependencies
 - Properties
 - wwwroot
 - css
 - images
 - js
 - lib
 - favicon.ico
 - Controllers
 - HomeController.cs
 - Models
 - ErrorViewModel.cs
 - Views
 - Home
 - About.cshtml
 - Contact.cshtml
 - Index.cshtml
 - Privacy.cshtml
 - Shared
 - _CookieConsentPartial.cshtml
 - _Layout.cshtml
 - _ValidationScriptsPartial.cshtml
 - Error.cshtml
 - _ViewImports.cshtml
 - _ViewStart.cshtml
 - appsettings.Development.json
 - appsettings.json
 - Program.cs
 - Startup.cs

Annotations with arrows pointing to specific files or folders:

- Static files (CSS, Images, JS, jQuery) points to the **wwwroot** folder.
- All controllers and actions points to the **Controllers** folder.
- View templates points to the **Views** folder.
- _Layout.cshtml – master page** points to the **_Layout.cshtml** file in the **Shared** subfolder of **Views**.
- Program.cs – The app entry point** points to the **Program.cs** file.
- Startup.cs – Settings and services** points to the **Startup.cs** file.

NuGet Package Management



NuGet Package Management

- Free, open source package management
- Makes it easy to install and update open source libraries and tools
- Part of Visual Studio 2012+
- Configurable package sources
- Simple as adding a reference
- GUI-based package installer
- Package manager console

