MVC 5 ASP.NET

Framework:

* An abstraction in which common code providing generic functionality can be selectively overridden or specialized by user code providing specific functionality
* Collection of programs that do something useful and which you can use to develop your own applications; guides you on how to do something
* Provides functionalities/solution to the particular problem area

Platform:

* Any hardware or software used to host an application or service

Cookie:

* Small pieces of data stored as text on the client's computer
* Normallyused only to store small amounts of data, including user preferences, time and more
* An HTTP cookie (also called web **cookie**, **Internet cookie**, browser **cookie**, or simply **cookie**) is a small piece of data sent from a website and stored on the user's computer by the user's web browser while the user is browsing.

State:

* Stateful: keeps / tracks the state of interactions; may create a connection object that tracks the information requested
* Stateless: no record of previous interactions; each interaction request has to be handled by the information that came with it
* \*\*\*Virtually Stateful Platform:

API: Application Programming Interface

* A set of routines, protocols, and tools for building software applications

🡪 good API’s makes it easier to develop a program by providing all the building blocks

* A software intermediary that allows two applications to talk to each other

🡪 the messenger that delivers your request to the provider that you’re requesting it from and then delivers the response back to you.

* make often repeated yet complex processes highly reusable with a little bit of code

ASP.NET Web API:

* Framework / Platform to build HTTP services that reach broad..
* For building restful applications on the .NET framework

**UNSURE OF:**

HTTP requests / handlers

Bundling & minification 🡪 Sematic Analysis

Authentication & authorization filters; filter overrides

Componentized

**.NET PLATFORM:**

A general-purpose development platform

* Several key features:
* support for multiple programming languages
* asynchronous and concurrent programming models
* native interoperability

🡪 which enable a wide range of scenarios across multiple platforms

* Microsoft actively develops and supports three .NET languages:
* C#
* F#
* Visual Basic (VB)

**ASP.NET PLATFORM:**

A free web framework for building great websites and web applications using HTML, CSS, and JavaScript. You can also create Web APIs and use real-time technologies like Web Sockets.

* Offers 3 frameworks for creating web applications – each targeting a different development style
* Web Forms
* ASP.NET Web Pages
* **ASP.NET MVC** – combination of the ASP.NET web framework & the MVC software pattern

**ASP.NET MVC 5:**

One ASP.NET:

* Only one ASP.NET project type / web application option exists
* Can then add in support for other frameworks as you develop
* --> Because the tooling & features are delivered as NuGet packages

ASP.NET Identity:

Bootstrap Templates:

Attribute Routing, ASP.NET Scaffolding, Authentication Filters, Filter Overrides

Installation and shit

**MVC APPLICATION STRUCTURE:**

When creating an ASP.NET MVC application w/ Visual Studio – automatically adds several files & directories to the project 🡪 these default files provide you w/ a basic structure for a working application, complete with home page, about page, account login/ logout/ registration pages, & an unhandled error page

Conventions:

ASP.NET MVC applications, by default, rely heavily on conventions

* MVC is designed around some sensible convention-based defaults that can be overridden as needed;
* Ex. uses a convention-based directory-naming structure when resolving View templates 🡪 allows you to omit the location path when referencing views from w/in a Controller class
* “Convention over Configuration”:
* Ruby on Rails – “We know, by now, how to build a web application. Let’s roll that experience into the framework so that we don’t have to configure absolutely everything again.”
* Expectations of the application’s structure

Conventions Simplify Communication:

While you write code to communicate, you are speaking to two very different audiences

* The Computer 🡪 clear & unambiguous instructions for the computer to execute
* The Developer 🡪 to allow for navigation & reading of your code for later maintenance, debugging, and enhancement
* Following common conventions allows MVC developers worldwide to share a common baseline for all our applications
* Advantage of software design patterns 🡪 the way they establish a standard language

Chapter 2 – Controllers

**CONTROLLER’S ROLE:**

Within the MVC pattern, controllers are responsible for responding to user input, often making changes to the model in response to that user input

\*ASP.NET MVC implements the front controller variant of the MVC pattern, & the controllers sits on front of everything except the routing subsystem\*

Front Controller:

* Software Design Pattern
* A controller that handles all requests for a website
* Often used in web applications to implement workflows