# Introduction to MVC

* Stands for Model-View-Controller
* It is a design pattern we used to separate our application into three main groups
* This will help us achieve separation of concern
  + Makes it easier to scale our application
  + Easier to code, debug, and test
  + Make your application easier to digest

## Model

* The model process data / user input
* It represents the current state of application

## View

* It is what the end-user sees and interacts with
* Essentially what the webpage you see when visiting a website
* So it is composed of your HTML, CSS, JS, or other assets

## Controller

* The controller handles the client request
* It will call upon appropriate business logic to process what the client wants from the database/server
* The controller will then choose the appropriate view that the user should interact with
  + It will also fill in any model data needed to display that view

# ASP.NET MVC

* It is the web application framework developed by Microsoft that implements that MVC design pattern
* Since it uses the MVC design pattern, your web application is separated into a model, view and a controller

# Communication between the server and the client

* Basically, how writing a URL address in your browser and magically brings you a nice-looking webpage.
* One thing to note: we will mainly focus on HTTP/HTTPS
  + Hyper Text Transfer Protocol Secure

## Request response life cycle

1. Client (your browser) will send a request (the url you sent)
2. The server will receive that request and will do some process
3. The server will send an appropriate response (html, css, js, etc.) and a status code
4. The client will receive the response and the browser will process that response

# DNS

* It stands for Domain Name System
* It is essentially a directory of names and ip address
* It translates our pretty name of a website (google.com) into a numerical ip address (0.0.0.0 – 255.255.255.255) for locating the right server that is connected to that website
* Main reason is so people can easily find your website since it is easier to memorize a name verses a combination of numbers.

# HTTP Request

* It has 3 requests
  + Request Line
    - Method – describes what action will be perform
    - Target – describes where to send that request
  + Header
    - Meta data
    - Ex: content type – what data type the body has
  + Body
    - Data you want the server to process

# HTTP Response

* It has 3 parts
  + First line
    - Gives a **status code**
  + Middle subsequent lines
    - Meta data
    - Like HTTP Request Header
  + Final block of lines
    - The data block (it is the data that the http response gives to you)

# HTTP Verb/Methods

* Describes what action the client wants the server to perform on a given resources
* Common Verbs
  + GET – Used to retrieve data from the server
  + POST – Used to submit data to the server
  + PUT – Replaces/updated the existing data on the server
  + DELETE – Deletes existing data on the server
  + HEAD – it is like a get method, but I will only give you the header

## Safe methods

* A method is safe if it doesn’t change/alter your data in your server
* Ex: GET or HEAD

## Idempotent methods

* If you call on the same method repeatedly, it should give the same result/effect
* So, image you created a DELETE http method that will delete the last entry in your server
  + Try calling the method 4 times and it will delete different row each time
  + This will make the DELETE http method not idempotent only because it gives different result each time
  + To fix this, make sure your delete http method uses the customer id to find its row and try deleting the same row every time.

# HTTP status code

* Gives the result of an HTTP request
* They are group of 5 different levels:
  + 1XX – Informational
  + 2XX – Successful
  + 3XX – Redirection
  + 4XX – Client error – Your frontend needs work
  + 5XX – Server error – Your backend needs work

# Controller

## Controller Actions

* An action is essentially a method inside a controller that will be called depending on what url http request the user is asking to be displayed on their web browser.

## Request Parameter

* Controller actions has the capability of adding parameters into the methods to essentially pass data from the url to the controller action.
* Common ways of application are getting data from your database and displaying to the view

# Model

## Model Binding

* It is a way to bind data coming from your HTTP request direct into a model
* It is essentially the fancy way of converting the JSON file into objects that C# understands

## Data Annotation

* Great way to include validation for your models
* Essentially instead of adding implementation details in your properties to enforce certain data to only be stored, data annotation can be used to enforce it.

## ViewModel

* This is how we pass model information to our views
* This is done by using our controllers and having the controller’s action pass some model to the view

# View

* It is basically html with c# coding mixed into it
* Allows us to have dynamically changing views

## Partial-View

* It is an incomplete view that is mostly used with other views
* Useful if you want a view that needs to be displayed in numerous other views

## Strongly-View

* It is when you pass in the model itself in the controller’s action
* It enforces strongly typed checking in the view itself

## Weakly-View

* It is weakly-typed checking since datatype are resolved at runtime
* Try to avoid since intellisense won’t help you see certain datatypes cannot be stored in certain properties of a model
  + i.e. storing a string into an int property
* More error-prone