MVC:

To understand how Model-View-Controller works, you'd need to know what role each part fulfills.

The model is a blueprint for how an object is made. If you were building a house, you'd know what it will need to be considered a house. I know it will need a door, a roof, and some walls. These are the parameters of my object. The blueprint then stores the exact type of door, roof, and walls you want. Your house could have a blue door, panel roof, and brick walls. All would be kept in the blueprint of my house.

The controller is the brains of the program, think of it like your contractor. You can tell it what you want, and it can deliver it to you. It's the middleman between the what users see and what data is stored in an object. It will serve you up what you request and can perform complicated tasks. Say you wanted to change your blue door to red, you'd tell the controller and it can update the blueprints.

The view is what you as a user would see when your done building your house. You'll see the blue or red door you chose. Or see a whole different iteration of a house. Either way, you'd use that controller to get the data from the model. This is the interface of your application.

It's a pretty important concept for Object oriented program, and it's widely used in web applications. It divides responsibilities and allows object blueprints to be reused. So, no matter where I'm at, I know my house is going to need a door, walls, and a roof. When you stick to a design pattern like this, it makes it easy for other developers to modify existing code. It keeps everything clean, concise, and well organized.