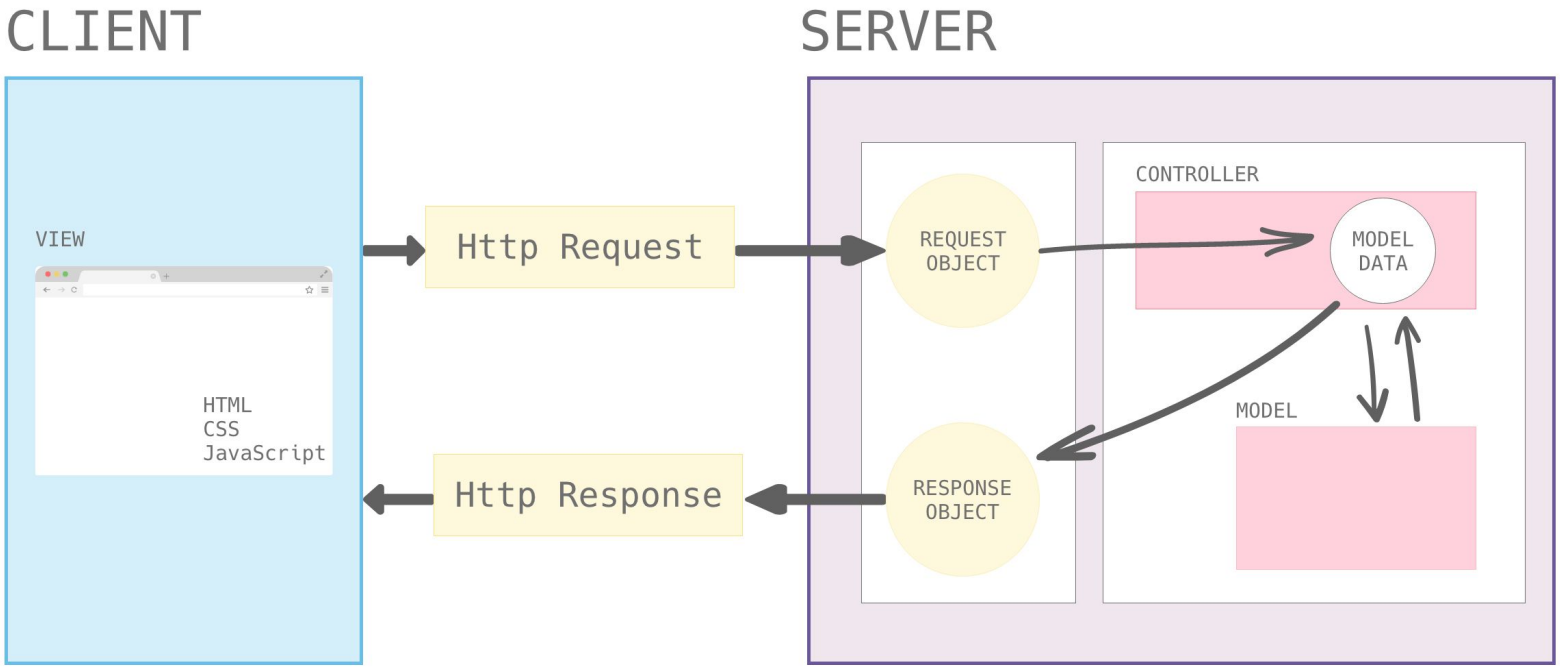
Module 2 - Lecture 14

Server-Side APIs: Part 1

MVC Pattern

- MVC Model-View-Controller
- Why? Separation of concerns.
- **M**odel (Data, Validation Logic, Business Rules)
- **V**iew (Presentation)
- **C**ontroller (Public face of your API. Facilitator)


CLIENT
SERVER
VIEW
HTML
CSS
JavaScript
Http Request
Http Response
REQUEST OBJECT
RESPONSE OBJECT
CONTROLLER
MODEL DATA
MODEL

Model

- A class that represents an entity.
  - City, Country, Hotel, Reservation, etc.

- Models often are a direct mapping from a database table.

- Contains getters, setters, and data validation.

View

- Meant to handle the presentation of our model.

- Commonly, the View takes in a Model and uses it to create an HTML page that can be rendered by a web browser.

- For now, we are only passing our Model data back in its raw form (JSON).

Controller

- Facilitates the Request/Response.
- Entry point for your Web Application.
- Get the Model and give it to the View.


spring®

Routing / Handler Mapping

- How does Spring know which handler method to call?

- **@RestController**
  - Let’s Spring know that a Java class is a Controller

- **@RequestMapping**
  - Define the URL and HTTP Method

Data Binding / Model Binding

- The process by which Spring takes the parameters we pass in an HTTP request and supplies them to our handler methods.

- Several options. Each use a different annotation.
  - Parameters in the querystring.

  - Parameters in the URL path.

  - Data in the request's body

RequestParam

http://localhost:3000/hotels?id=1

```
@RequestMapping(path = "/hotels", method = RequestMethod.GET)
    public Hotel get(@RequestParam int id) {
        return hotelDAO.get(id);
    }
```

These can be set as optional

@RequestParam(required = false)

PathVariable

http://localhost:3000/hotels/1

```
@RequestMapping(path = "/hotels/{id}", method = RequestMethod.GET)
  public Hotel get(@PathVariable int id) {
     return hotelDAO.get(id);
  }
```

These can be set as optional

@PathVariable(required = false)

RequestBody

http://localhost:3000/hotels

Request Body:

```
{
        “id”: 2,
        “name”: “Hilton Columbus Downtown”,
        “stars”: 4,
        “roomsAvailable”: 500,
        “costPerNight”: 190
}
```

```
@RequestMapping( path = "/hotels", method = RequestMethod.POST)
  public Reservation addHotel(@RequestBody Hotel hotel) {
     return hotelDAO.create(hotel);
  }
```

QUESTIONS?