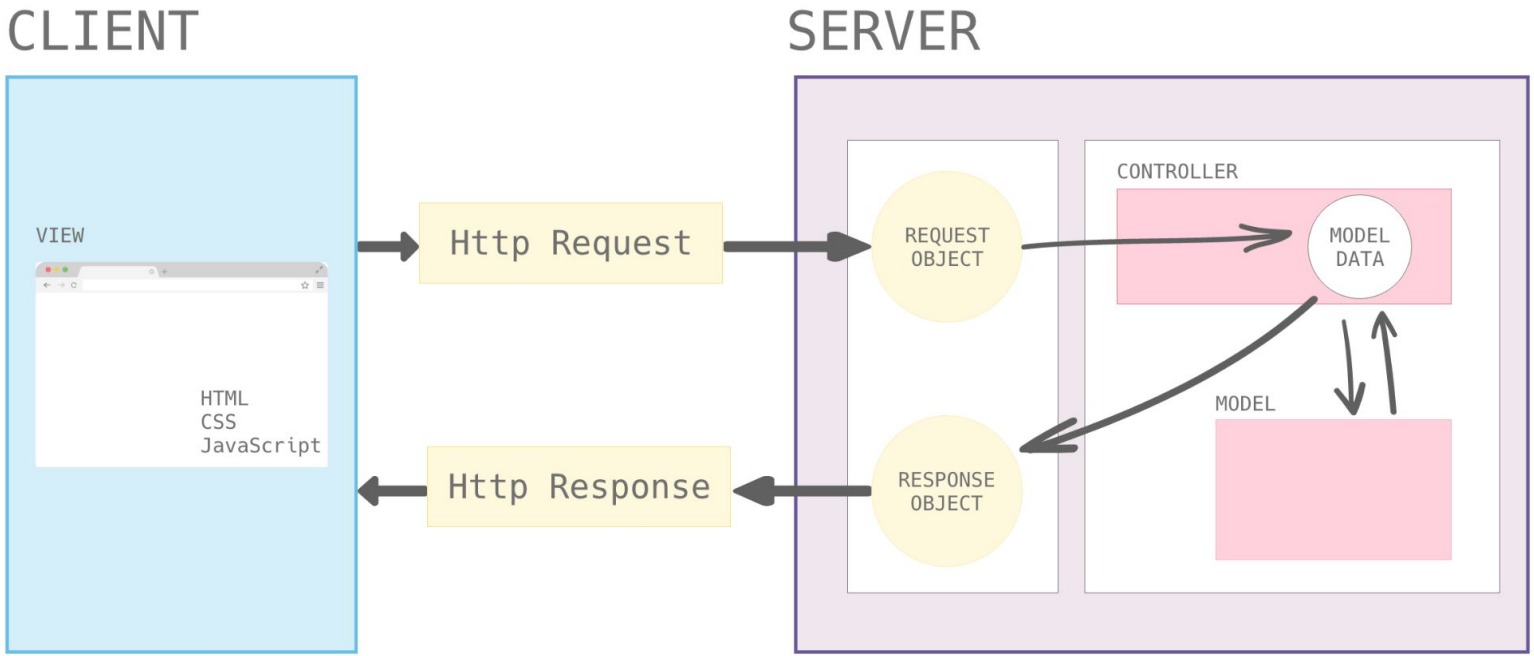
Module 2 - Lecture 13

Server-side APIs

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.
  - e.g. City, Country, Hotel, Reservation, etc.

- Models may or may not be a direct mapping from a database table.

- Contains getters, setters, data validation, etc.

View

- Handles the presentation of a model.

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

- For APIs, however, we will pass our Model data back in its raw form as JSON.

Controller

- Facilitates the Request/Response.
- Entry point for your Web Application / API.
- Obtains the Model and passes it to the View.

Spring Framework / Spring Boot

Routing / Handler Mapping

- How does Spring know which handler method to call?

- **@RestController**
  - Signals to Spring that this class is a Controller.

- **@RequestMapping**
  - Defines a unique URL and HTTP method.

Data Binding / Model Binding

- The process by which Spring takes 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

- GET 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

- GET 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

- POST http://localhost:3000/hotels
- Body:

```
{
    "name": "Marriot Tech Elevator",
    "stars": 5,
    "roomsAvailable": 100,
    "costPerNight": 150,
}
```

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

LET'S CODE