Introduction to Spring Boot

CS F213: Object Oriented Programming

Dipanjan Chakraborty
Department of CS&IS

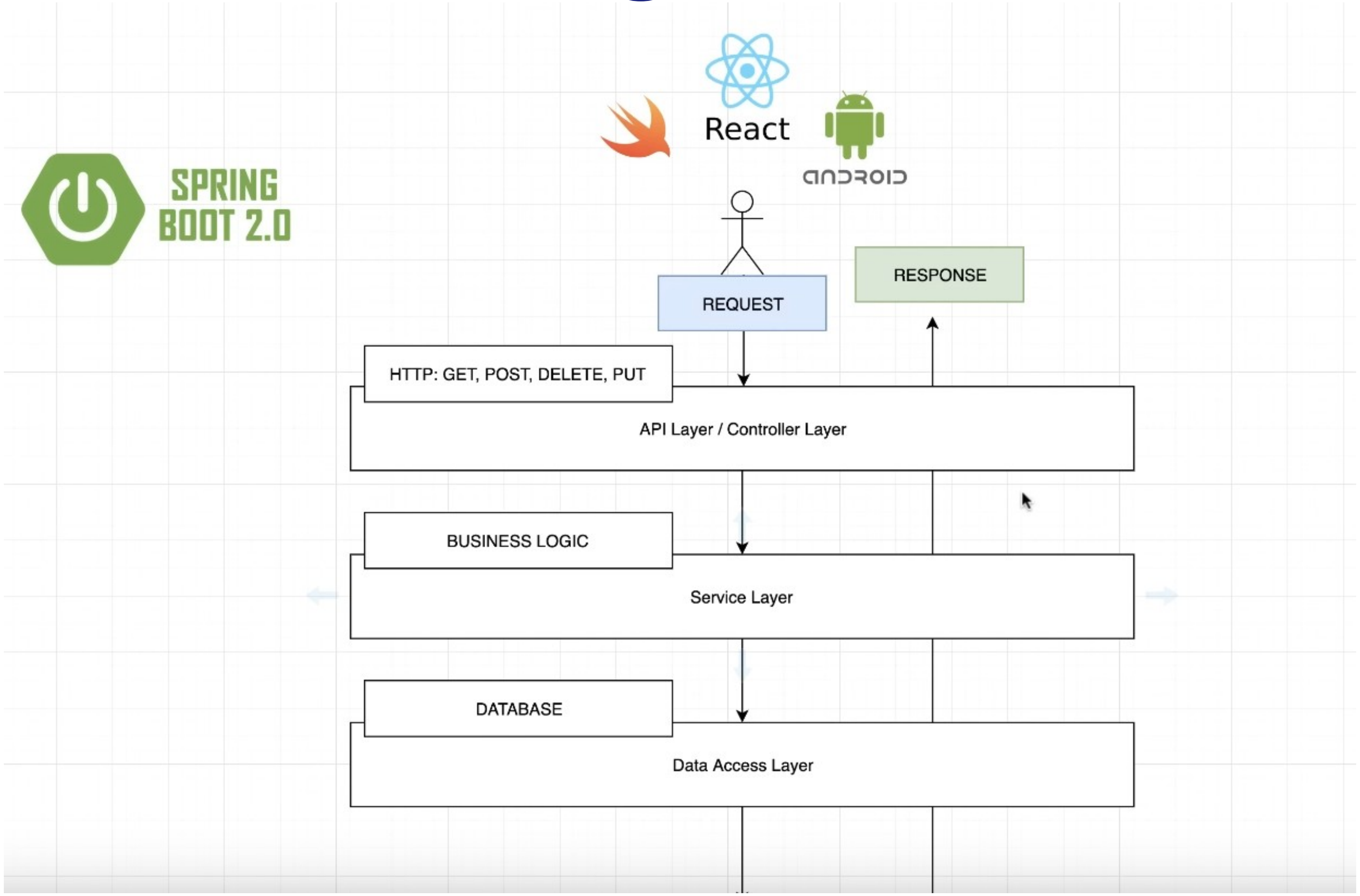
What is Spring Boot

- Spring Boot is an application development framework for the JVM
  - Support for Java and other JVM languages like Kotlin and Groovy
- Based on the Spring framework
- The Spring framework allows a lot of flexibility in the configurations
- Spring Boot is an ‘opinionated’ version of Spring: it defaults to some conventions (over configuration) and allows a programmer to get started quickly
- Most Spring Boot applications need minimal Spring configuration.
- You can focus more on business features and less on infrastructure.

Getting Started

- Let’s start with an example
- We will use Intellij Idea and Java 21/22

Getting Started



Getting Started

- There are several annotations which help jumpstart development
- `@SpringBootApplication`
  - Combines three annotations: `@EnableAutoConfiguration`, `@ComponentScan`, `@Configuration`
- `@RestController`
  - tells Spring that this code describes an endpoint that should be made available over the web
- `@GetMapping("/hello")`
  - tells Spring to use our hello() method to answer requests that get sent to the `http://localhost:8080/hello` address
- `@RequestParam(value = "name", defaultValue = "World")`
  - tells Spring to expect a `name` value in the request, but if it's not there, it will use the word `"World"` by default

Getting Started

- The `@RestController` annotation marks the class as a controller, where every method returns a domain object instead of a view.
- `@Repository`: indicates that the class provides the mechanism for storage, retrieval, update, delete and search operation on objects
- `@Service`: marks that the class provides some business functionalities / is a service provider
- `@Autowired`: automatically wire the required beans (dependencies) into the classes, eliminating manual configuration
- `@Qualifier`: differentiates between bean objects

The Java Stream API

- Introduced in Java 8 (not the same as I/O streams)
- Combined with Lambdas, ability to perform very sophisticated operations that search, filter, map, or otherwise manipulate data
- One can construct sequences of actions that resemble, in concept, the type of database queries for which one might use SQL
- Such actions can be performed in parallel, thus providing a high level of efficiency, especially when large data sets are involved

The Java Stream API

- A stream is a conduit for data: represents a sequence of objects
- Operates on a data source, such as an array or a collection
  - Itself does not provide storage for data
- Methods in the stream interface are either *terminal* or *intermediate*
- Intermediate operations can be *stateful* (e.g. sorting) or *stateless* (e.g. filtering)
- Let's look at an example with filtering and mapping

The Optional Class

- Introduced in Java 8
- A way to handle situations in which a value may or may not be present
- Before, one would normally use the value *null* to indicate that no value is present.
  - This can lead to null pointer exceptions if an attempt is made to dereference a null reference.
  - Frequent checks for a null value were necessary to avoid generating an exception.

Resources Used

- https://spring.io/quickstart
- First part of:
  https://www.youtube.com/watch?v=vtPkZShrvXQ

- https://spring.io/guides/gs/accessing-data-mysql