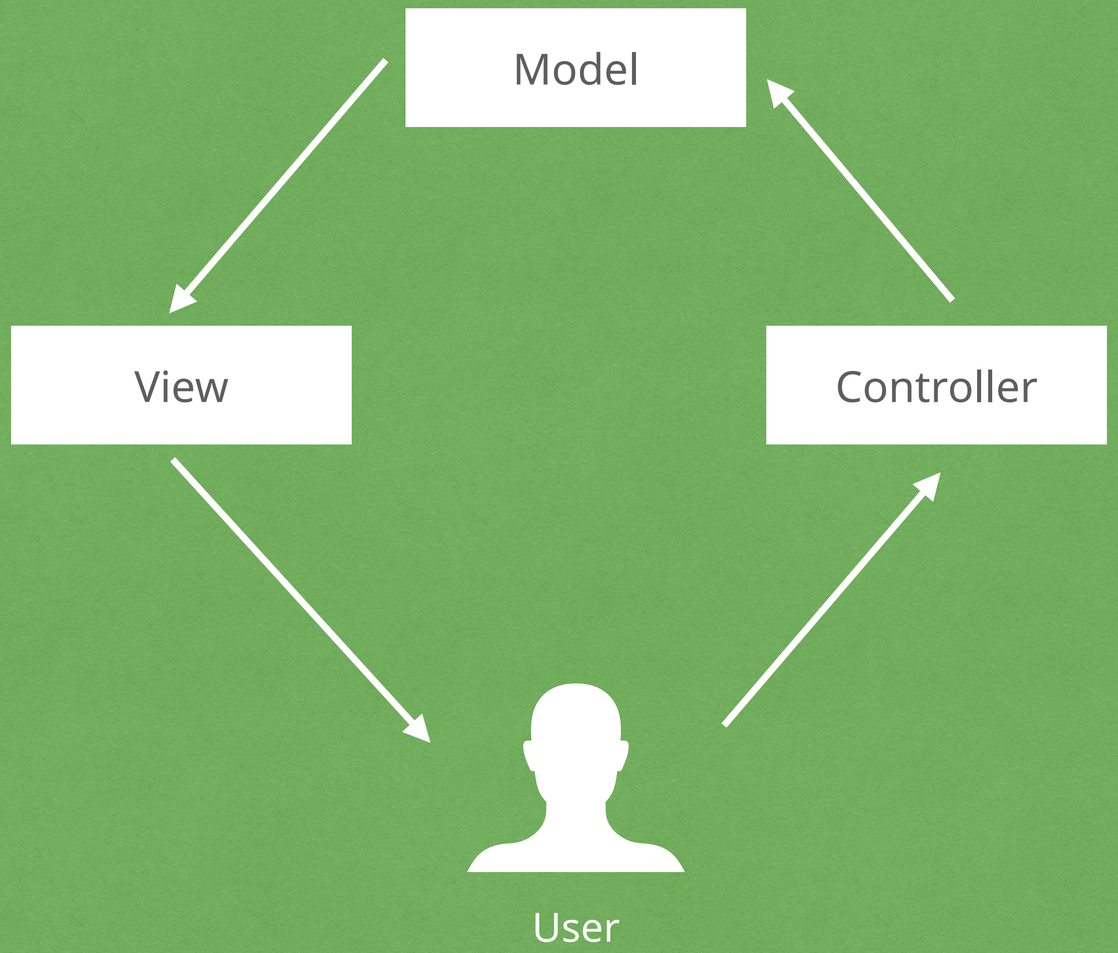
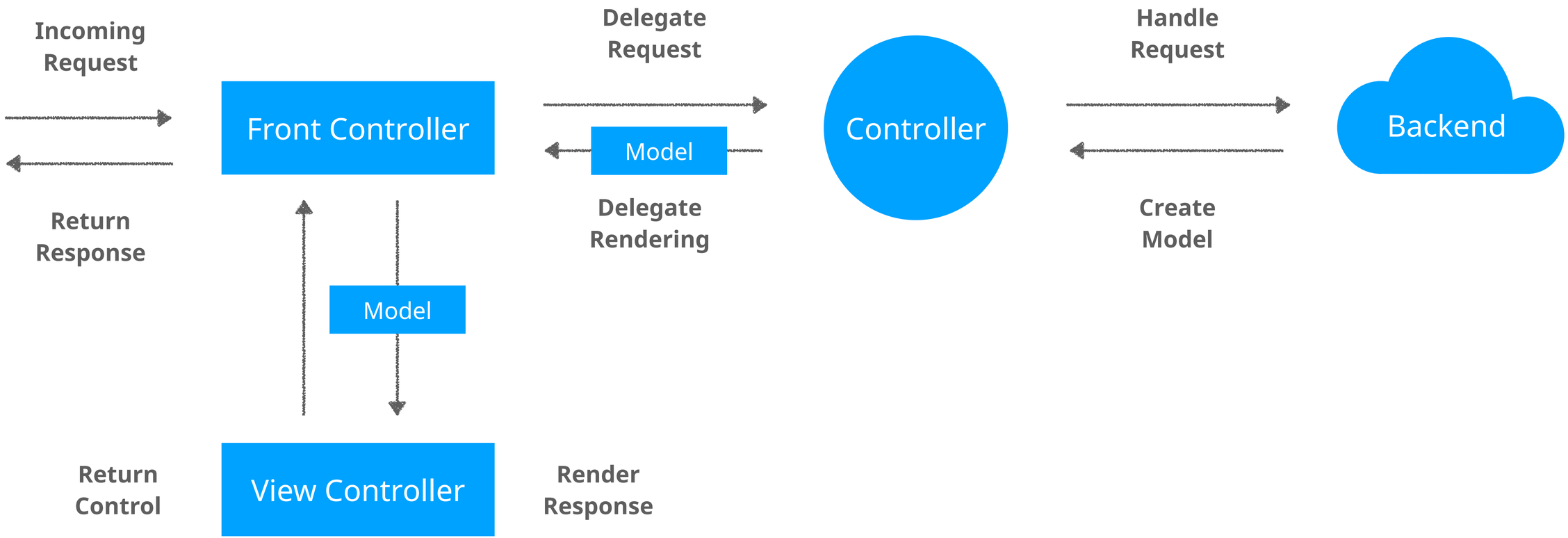
Spring Framework

Why Spring MVC?

Spring MVC Architecture



Request / Response Lifecycle



Terminologies

DispatcherServlet

Entry Point

Controller

Command Pattern Handler

RequestMapping

URL and Request Type

ViewResolver

Locates View to serve

servlet-config

Configuration file

POJO

Plain Old Java Object

Bean

Spring configured POJO

Prerequisites

Java

Maven (Basic knowledge)

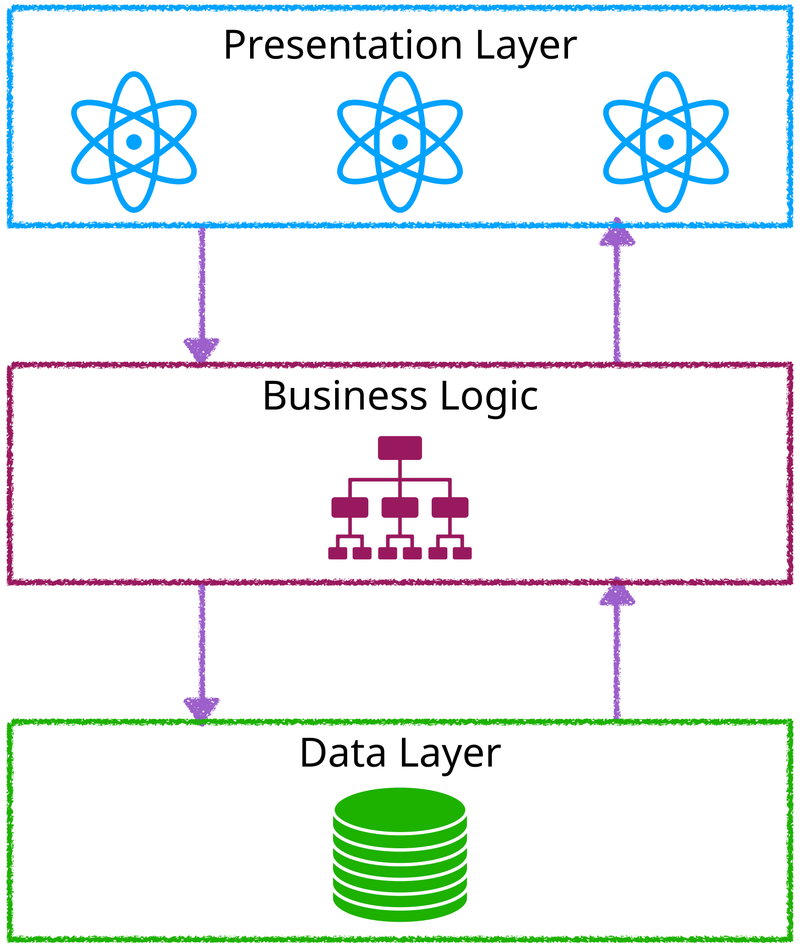
Tools

JDK 11

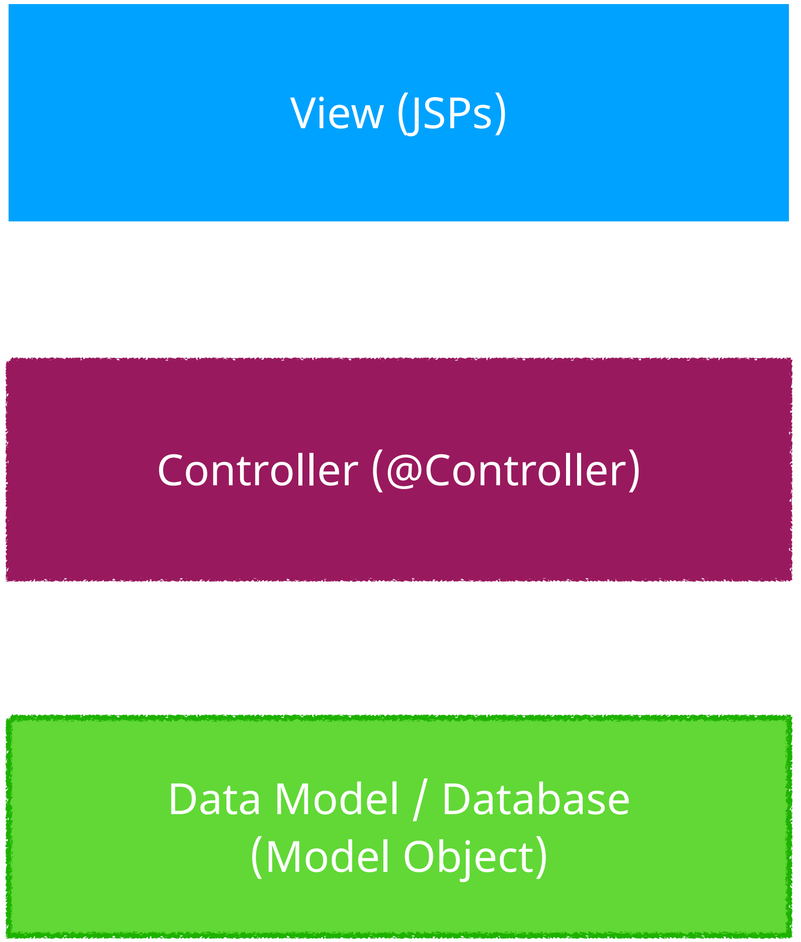
IDE (STS4 / VSCode / IntellijJ IDEA)

Tomcat (for deployment)

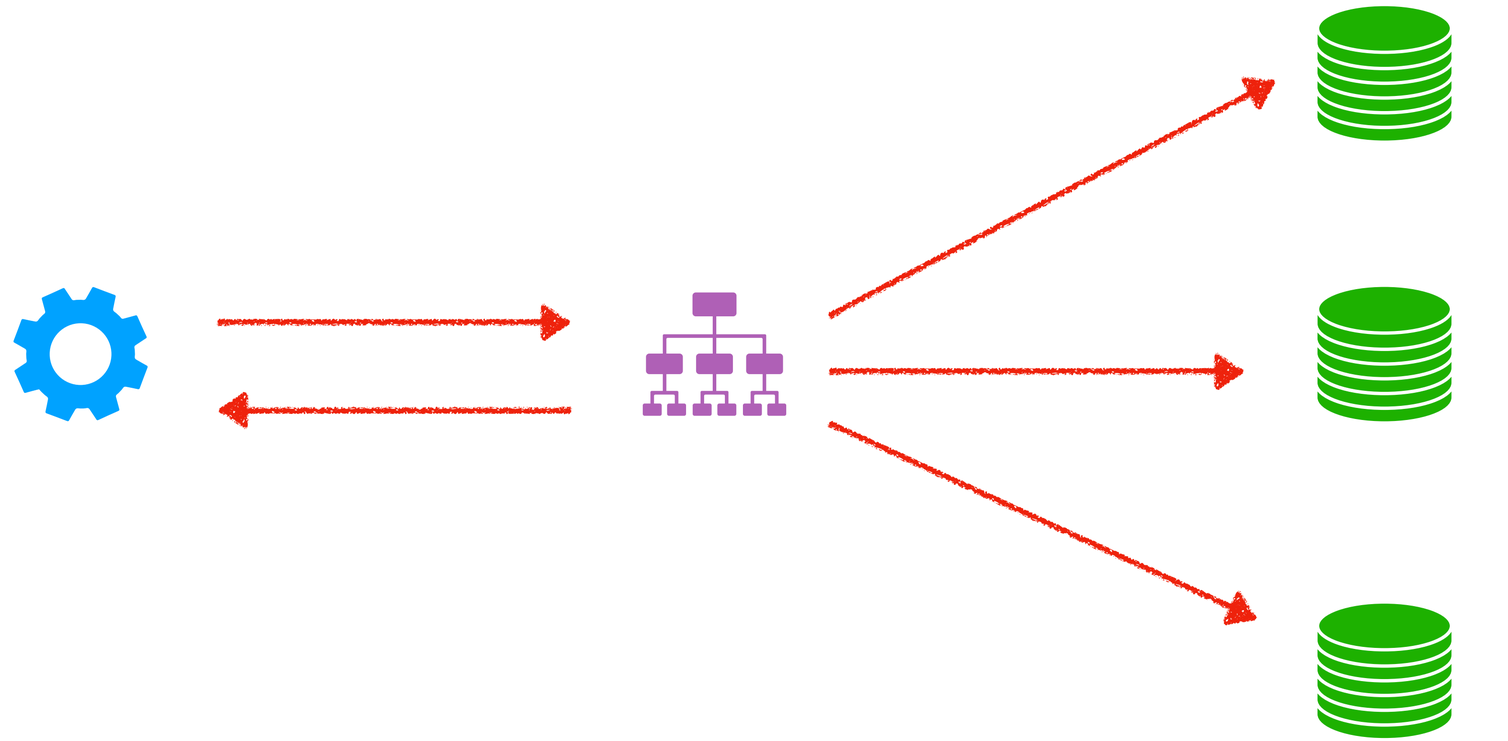
Application Layers



Application Layers



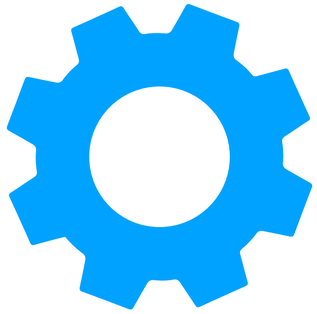
Components



Controller

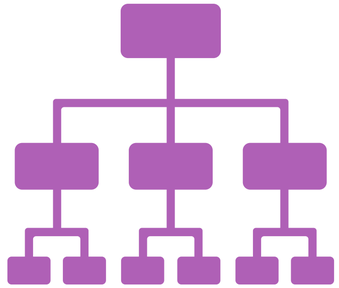
Service

Repositories



Controller

- Light-weight
- Handles requests/response
- No business logic
- Coordinate with service & repository
- Annotated with @Controller
- Handles exceptions and view routing



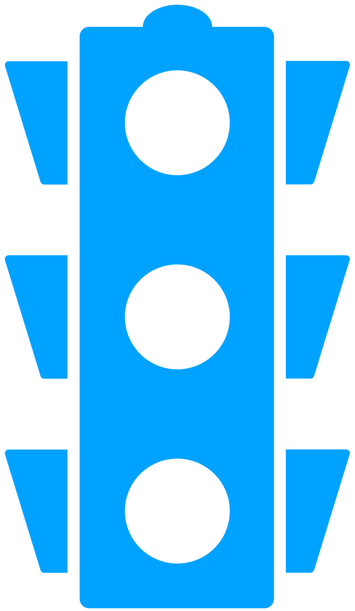
Service

- Heavy-weight
- Annotated with @Service
- Describes verb/actions of system
- Business logic belongs here
- Ensures business object state
- Transactional
- Often same methods as repository

Repository

- Annotated with @Repository
- Nouns (data) of the system
- Database interaction
- One-to-one object mapping
- Often one-to-one database table mapping

Duties of Controller



- Interpret and transform
- Access business logic
- Determines views or response type
- Interpret exceptions

@Controller

```
@Controller
public class GreetingController {
    @GetMapping("greeting")
    public String greeting(Map<String, Object> model) {
```