AIF233201 - Pemrograman Berbasis Web

Server-Side Programming

by Raymond Chandra Putra
Pascal Alfadian Nugroho
Keenan Adiwijaya Leman

14-10-2024

INFORMATIKA
UNPAR

Overview

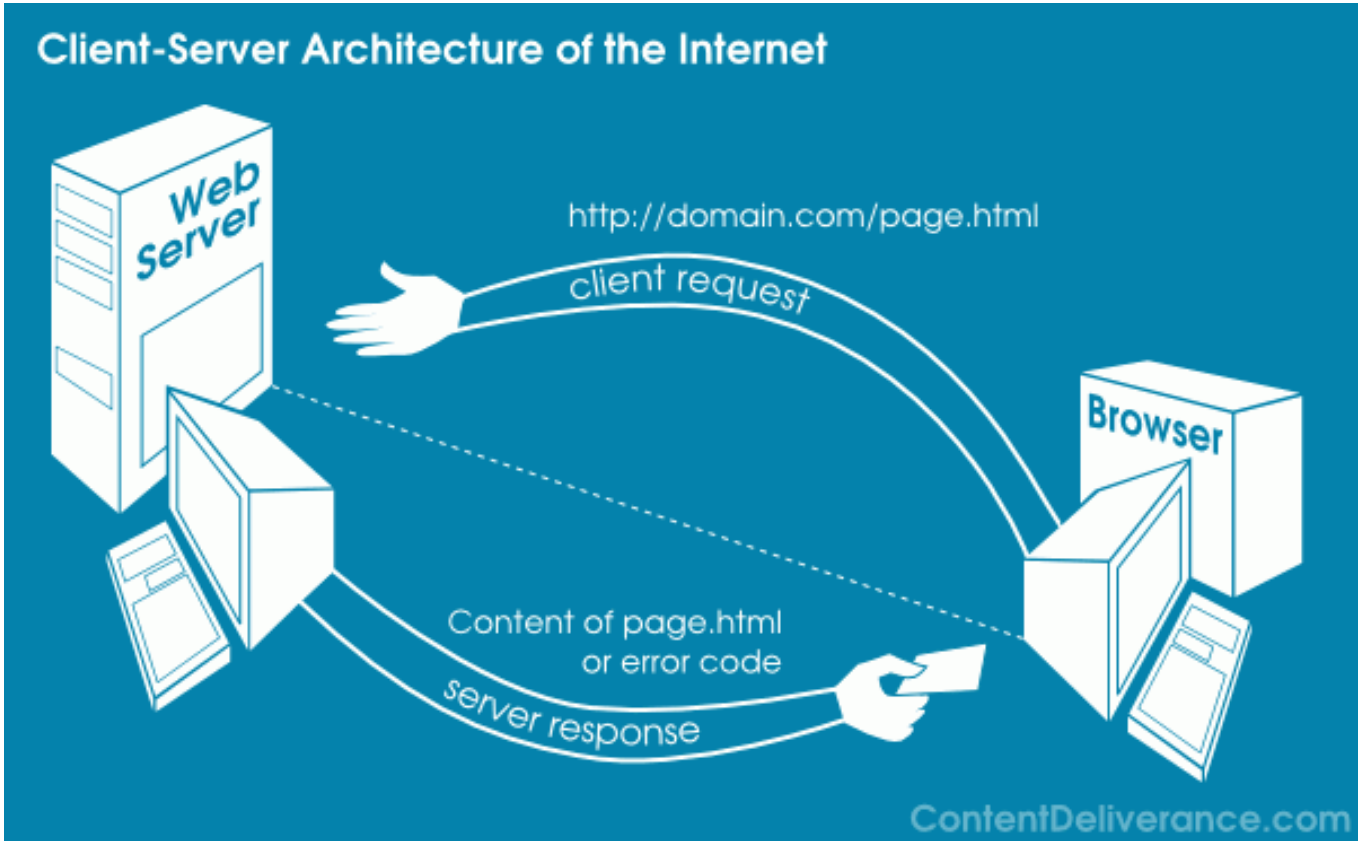
- Server-Side Programming
- Java Spring Boot
  - Introduction
  - Hello World
  - Bean
  - Handling Request

Client-Side Programming

- Up until this point in the course, we've been using JavaScript on the client-side.

- JavaScript on the client-side means the code is **sent** by a server and **ran** by a browser.

Client Server Architecture



Server-Side Programming

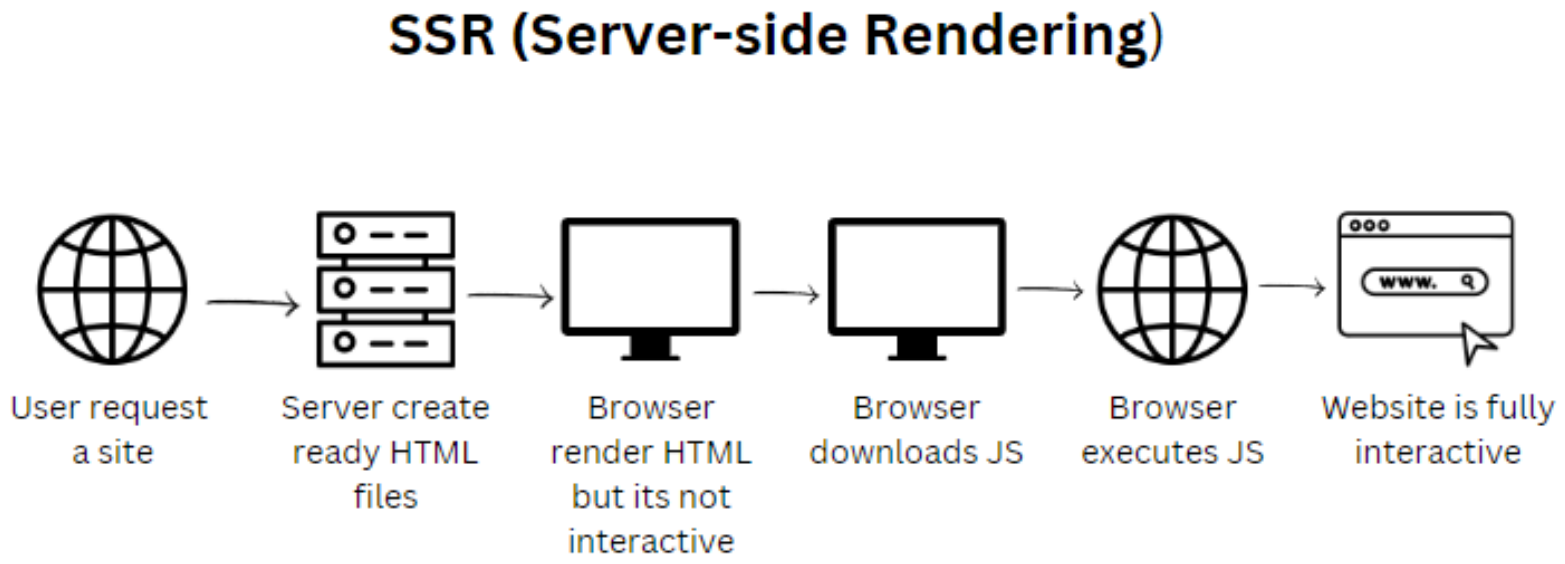
- Server-side web programming means the code is executed on the server-side.

- There's many language options, few common ones including:
  - Java
  - PHP
  - Python
  - Perl
  - And, since 2009, JavaScript.
  - etc

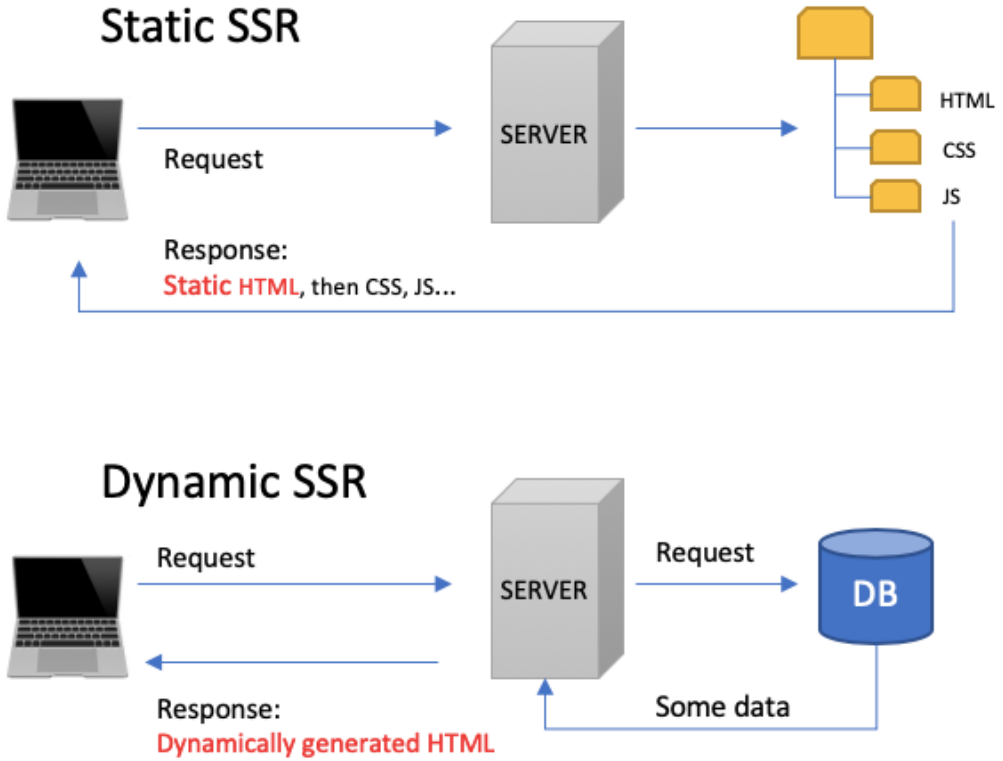
Web Server

- Opening .html file from your computer is not equals to running website.
- We need web server to serve our HTML document.
- Example of Web Server:
  - PHP : Apache, Nginx, etc
  - ASP.NET : IIS
  - JS : Node JS (V8 Engine)
  - Java : Tomcat

Server-Side Rendering

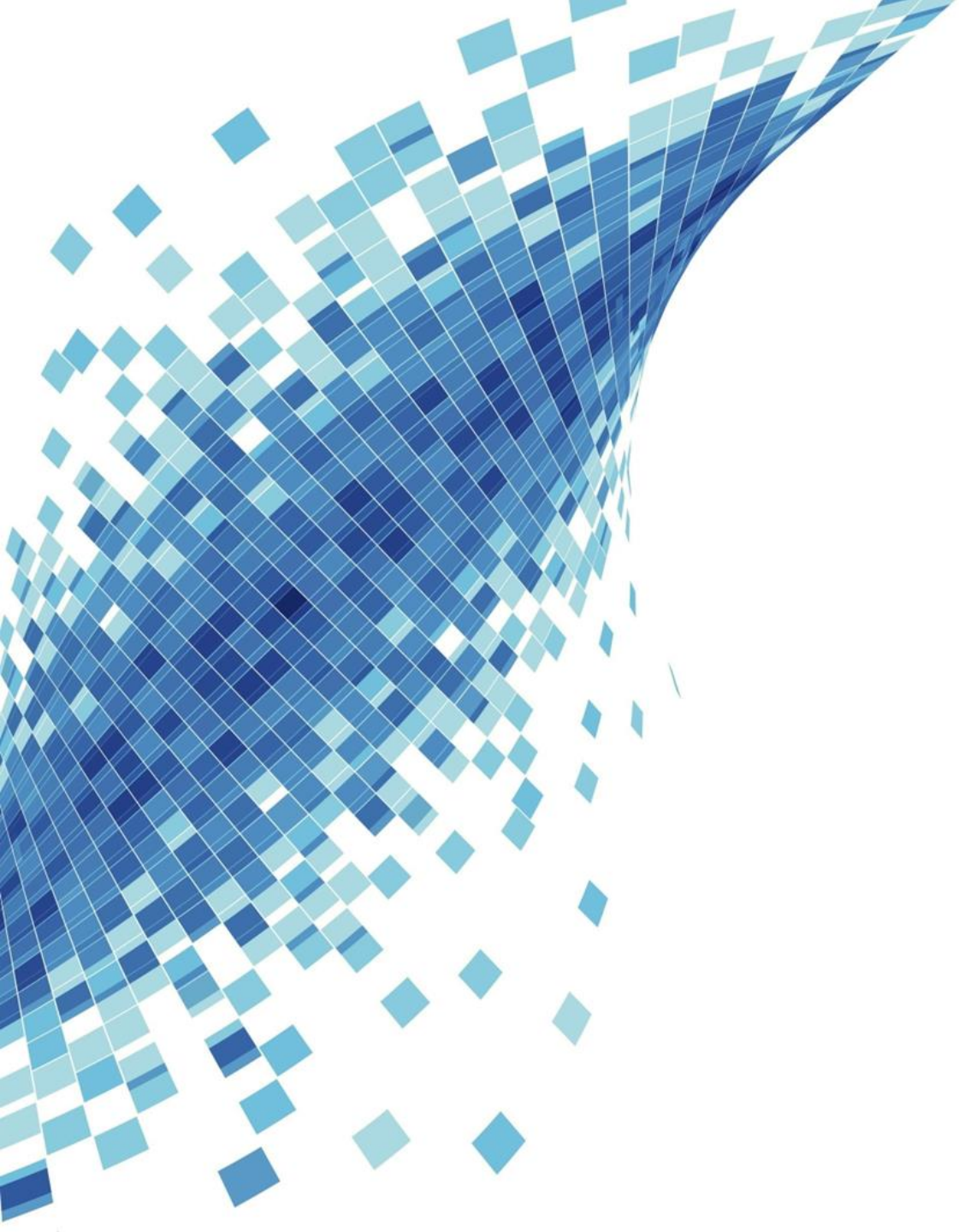


Static vs Dynamic SSR



HTTP Request Code

- 200 OK
- 300 Multiple Choices
- 400 Bad Request
- 401 Unauthorized
- 403 Forbidden
- 404 Not Found
- 500 Internal Server Error
- 501 Not Implemented
- 503 Service Unavailable



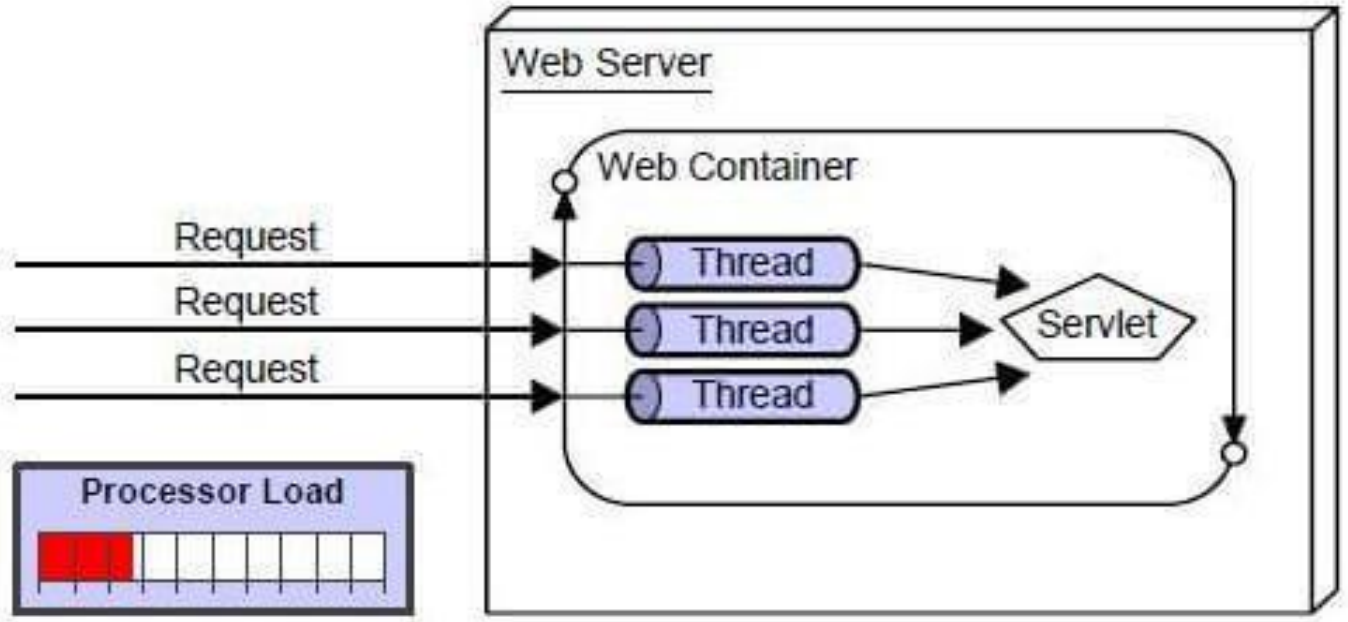
Java Spring Boot

INFORMATIKA
UNPAR

Java: A Versatile Language for Web Development

- Java has been a cornerstone of web development for decades, offering a robust, scalable, and secure platform for building complex applications. Its versatility, cross-platform compatibility, and extensive ecosystem make it a popular choice among developers.
- Advantages:
  - Platform Independence
  - Object-Oriented Programming (OOP)
  - Robustness and Security
  - Extensive Ecosystem
  - Large Community and Support

Java Servlet



Java Framework

- Java EE
- Spring
  - Also include Spring Boot
- Play Framework
- Struts

Spring Framework

- Spring is a comprehensive framework that simplifies the development of Java applications, especially web applications.
- Core Features:
  - Dependency Injection (DI)
  - Aspect-Oriented Programming (AOP)
  - Data Access Abstraction
  - Transaction Management
  - MVC Framework

Problem using Spring Framework

- Spring provides a comprehensive set of features for building Java Application.
- Spring offers **high flexibility** and customization options, allowing developers to tailor the framework to their specific needs.
- Have extensive configuration options and complex XML-based configuration.
- All of them sometime can overwhelm a newer developer and undeniably create steep learning curve.

Java Spring Boot

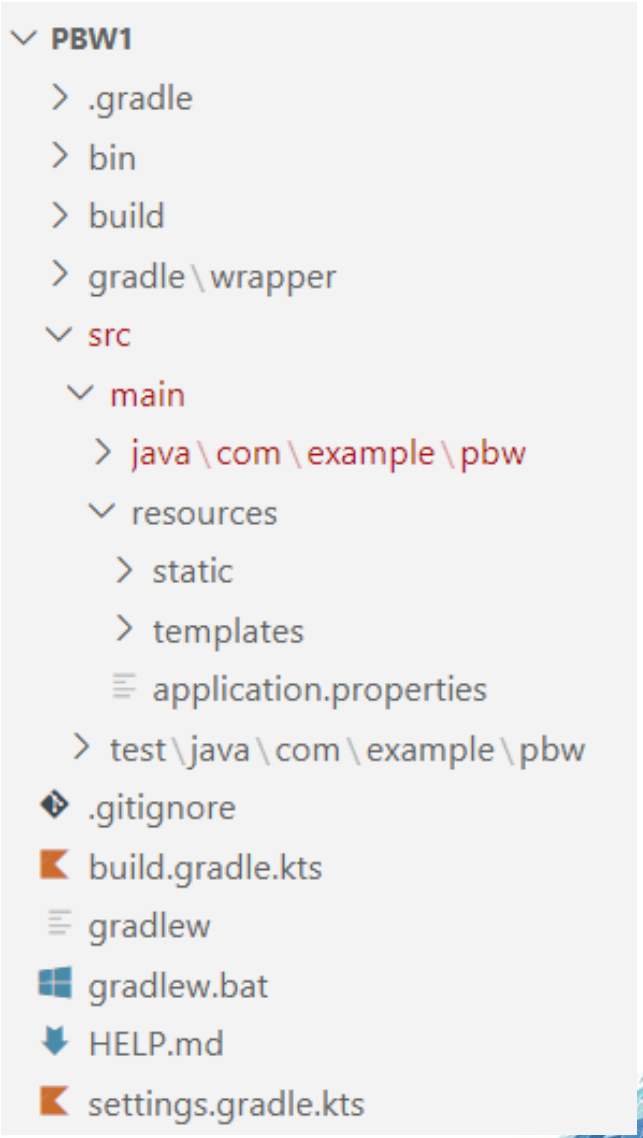
- Opinioned Framework for Spring that automatically configure common need for web development.
  - But less flexibility than Spring.
- Offering rapid development because we can allocate more for developing rather than configuring project.
  - automatic configuration has gone well beyond component scanning and autowiring.
- By default, also embedded Tomcat web server.

Creating Spring Boot Application

- Options:
  - Use Spring Initializr:
    - https://start.spring.io/
  - Manually from command line
    - Maven or Gradle project
  - New project from preferred IDE

Project Structure

- Example using gradle project
- Structure can be different according different approach



INFORMATIKA UNPAR

Hello World

```java
package com.example.pbw;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class PbwApplication {
    Run | Debug
    public static void main(String[] args) {
        SpringApplication.run(PbwApplication.class, args);
    }
}
```

Creating Controller

```java
package com.example.pbw;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.ResponseBody;

@Controller
public class HelloController {

    @GetMapping("/hello")
    @ResponseBody
    public String hello() {
        return "Greetings from Spring Boot!";
    }
}
```

Application Properties

- Used for configuration basic config such as:
  - Application name
  - Port used (default: 8080)
  - Context path (default: /)
  - DB configuration
  - Logging
  - Cache
  - Environment (dev/prod)
  - Etc.

Package Dependency Used

- This course use following dependencies :
  - spring-boot-starter-web
  - spring-boot-devtools
  - spring-boot-starter-thymeleaf
  - spring-boot-starter-jdbc
  - spring-boot-starter-data-jdbc
  - Lombok
  - postgresql
- Coordinator notes:
  - TBU: The dependencies still tentative depending on how we goes with this course. ☹

Spring Bean

- Bean: Powering the Spring Framework (also spring boot)
- It is a fundamental concept in Spring Framework that provides a flexible and powerful way to manage the lifecycle and dependencies of objects in your application.
- In Spring, we don’t need to explicitly instantiate the object and the following dependencies.

Create Spring Bean

- There are several ways to create Spring beans:
  - XML configuration:
    - Define beans in an XML configuration file using <bean> elements.
  - Annotations:
    - Annotate your classes with @Component, @Service, @Repository, or @Controller to make them Spring beans Java-based configuration.
  - Create a configuration class annotated with @Configuration and use @Bean annotations to define beans within the class.

Bean in Spring Boot

- @SpringBootApplication Annotation, includes:
  - @EnableAutoConfiguration: enable Spring Boot's auto-configuration mechanism
  - @ComponentScan: enable @Component scan on the package where the application is located
  - @Configuration: allow to register extra beans in the context or import additional configuration classes
- Spring Boot make the Bean configuration easier using component scanning and autowiring.

Component Scan

- How Component Scan Works:
  - Base Package: You specify a base package where Spring Boot will search for components.
  - Annotation Discovery: Spring Boot looks for classes annotated with @Component, @Service, @Repository, or @Controller within the specified package and its subpackages.
  - Bean Registration: If a class is annotated with one of these components, it is automatically registered as a bean in the application context.

Autowiring

- Autowiring allow Spring Boot automatically inject dependencies into bean based on their type or name. (DI)

```java
@Service
public class MyService {

    private final MyRepository repository;

    @Autowired
    public MyService(MyRepository repository) {
        this.repository = repository;


    }

    // ...
}
```

Handling Request

- Requests are mapped to controller using @RequestMapping:
  - `@RequestMapping("/test1")`

    short for:
  - `@RequestMapping(value = "/test1", method = RequestMethod.GET)`
- There are others annotation derived from RequestMapping:
  - @GetMapping
  - @PostMapping
  - @PutMapping
  - @DeleteMapping
  - @PatchMapping

Grouping Request

- @RequestMapping when put on class level can be used for grouping request.
- @RequestMapping(“person”)
  - /person/
  - /person/update
  - /person/delete
  - etc

Request Parameters

- Sometimes we need to include some parameters in requests:

```
@GetMapping("/hello2")
@ResponseBody
public String hello2(@RequestParam(name = "name", required = false,
                     defaultValue = "World") String name){
    return "Hello " + name;
}
```

Redirect

```
@GetMapping("/redirect-to-google")
public String redirectToGoogle() {
    return "redirect:https://www.google.com";
}
```

- OR

```
@GetMapping("/redirect-to-google2")
public void redirectToGoogle2(HttpServletResponse response)
                                    throws IOException{
    response.sendRedirect("https://google.com");
}
```

Argument Resolver

- In previous example, we can add parameter in controller method. For example: RequestParam or HttpServletResponse.
- But how Spring Boot knows what method should be called if the implementation differ for each request?
  - The answer is Argument Resolver.
- **Argument Resolvers** are components in Spring Boot that are responsible for resolving method arguments during the invocation of controller methods.

Argument Resolver (2)

- They provide a flexible mechanism for customizing how arguments are obtained, allowing you to integrate various data sources and processing logic into your web applications.
- Later on, we will see other kinds of parameter that can be added to our controller's methods.
- If there's no argument resolver suitable for the parameter, we also can add custom argument resolver.

Static Files

- Static files are files that we use as it is without modification (Static SSR).
- Examples
  - CSS
  - Javascript
  - Other assets (font, image, etc)
  - Static HTML
- Static files are put in resources/static/ and Java Spring Boot automatically route them. (No Controller needed)

Thanks

INFORMATIKA
UNPAR

INFORMATIKA
UNPAR

informatika.unpar.ac.id
informatika@unpar.ac.id
if.unpar
if.unpar