Spring MVC with Thymeleaf Basics

SoftUni Team

Technical Trainers

Software University

Software University

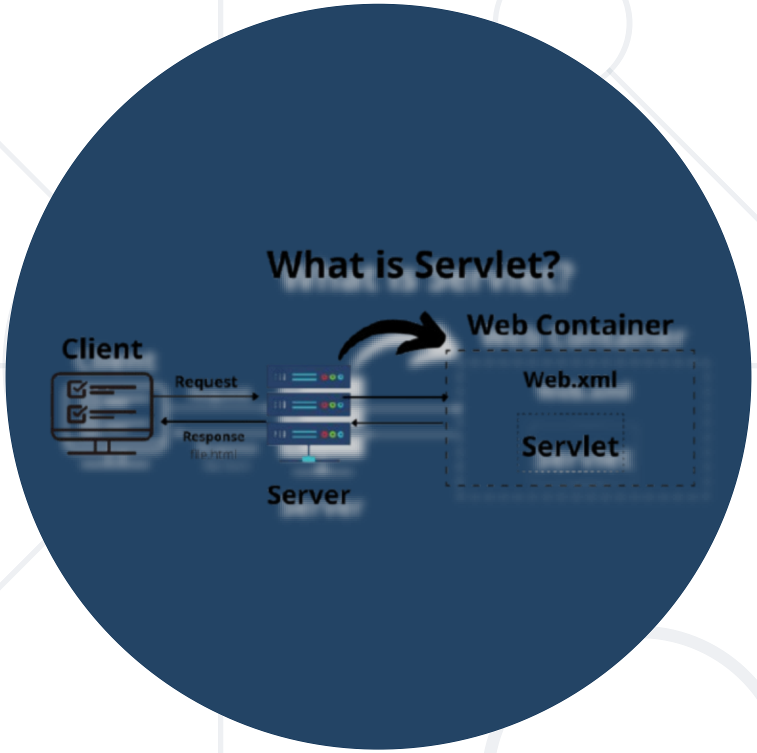
https://softuni.bg

Have a Question?

sli.do

#java-web

Table of Content

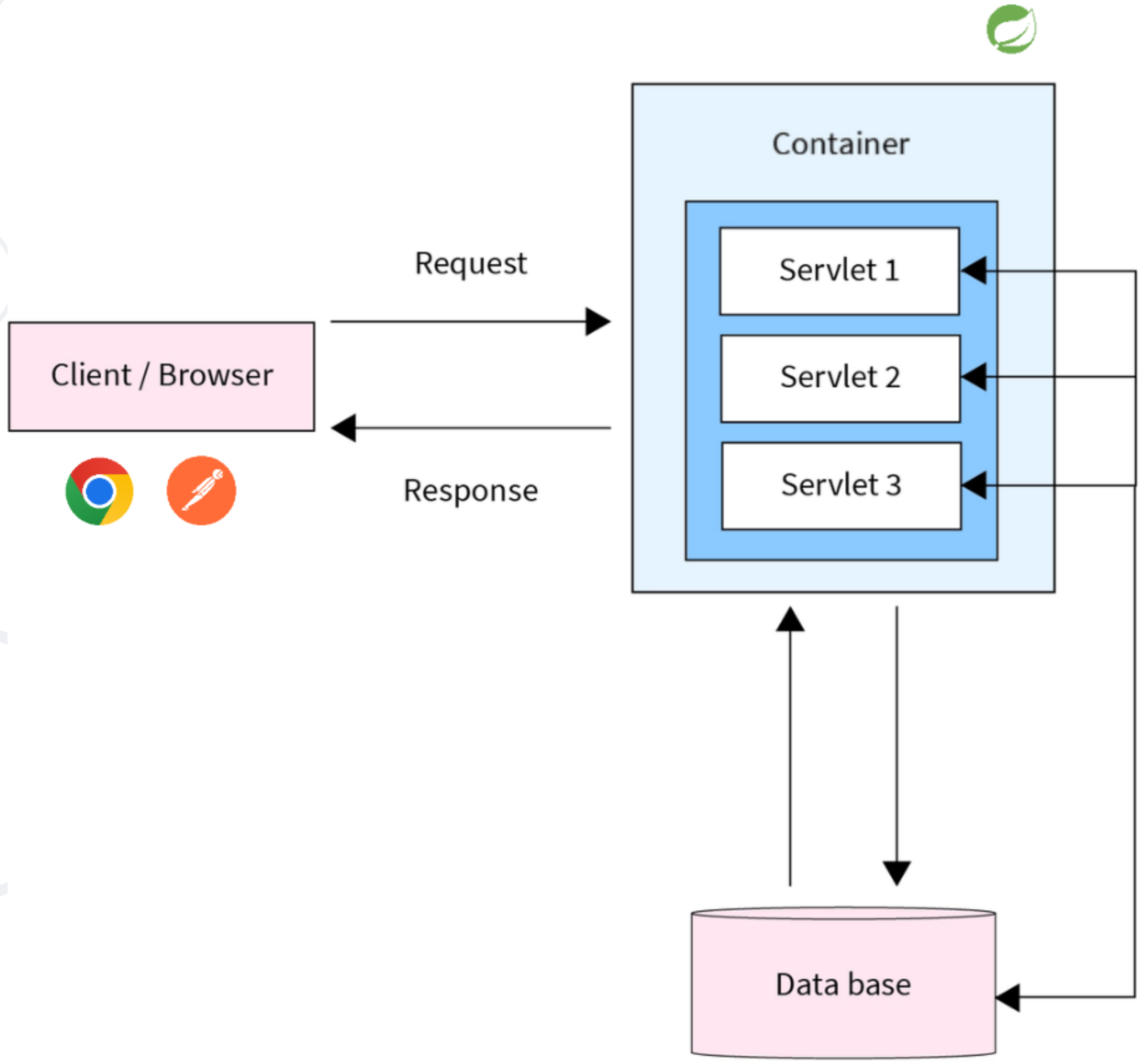


What is Servlet?

What is Servlet?

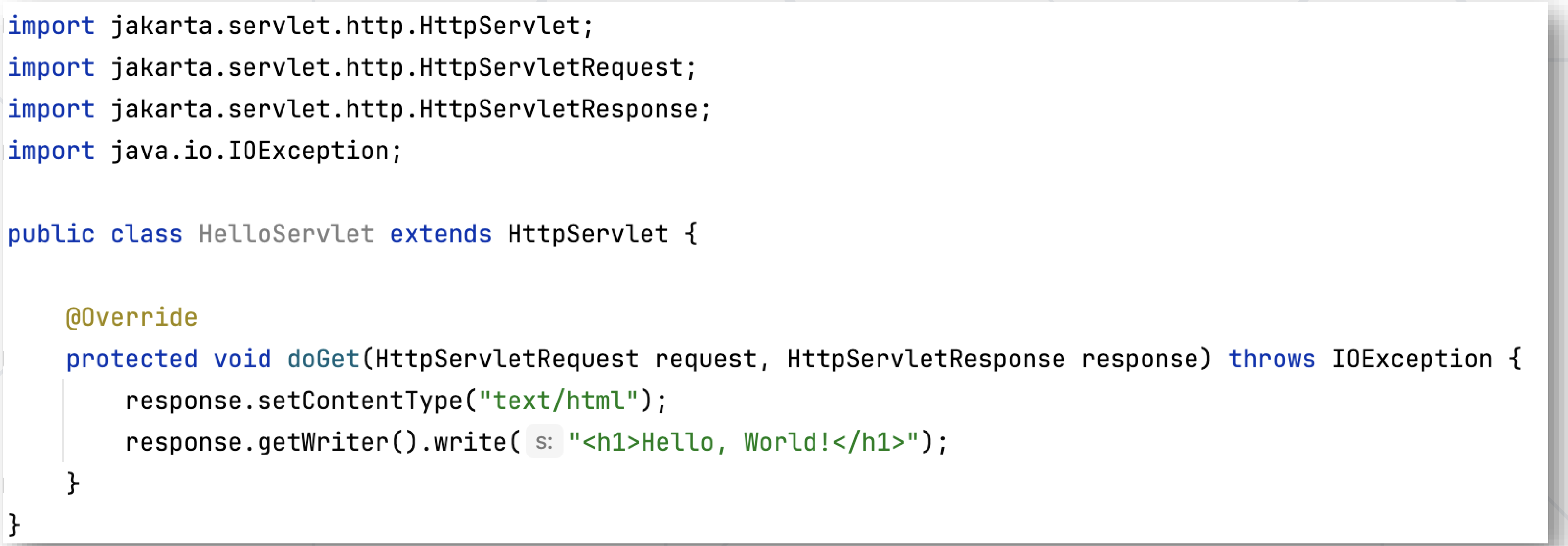
- A **Java class** that processes requests (usually HTTP) and generates responses
- **Workflow**:
  - Client (e.g., browser) **sends** a request (e.g., /orders/1)
  - The **Servlet Container** (e.g., Tomcat) maps the request to the appropriate Servlet
  - The Servlet processes the request and sends back a response

Servlets Overview



Basic Web App Java EE + Servlets

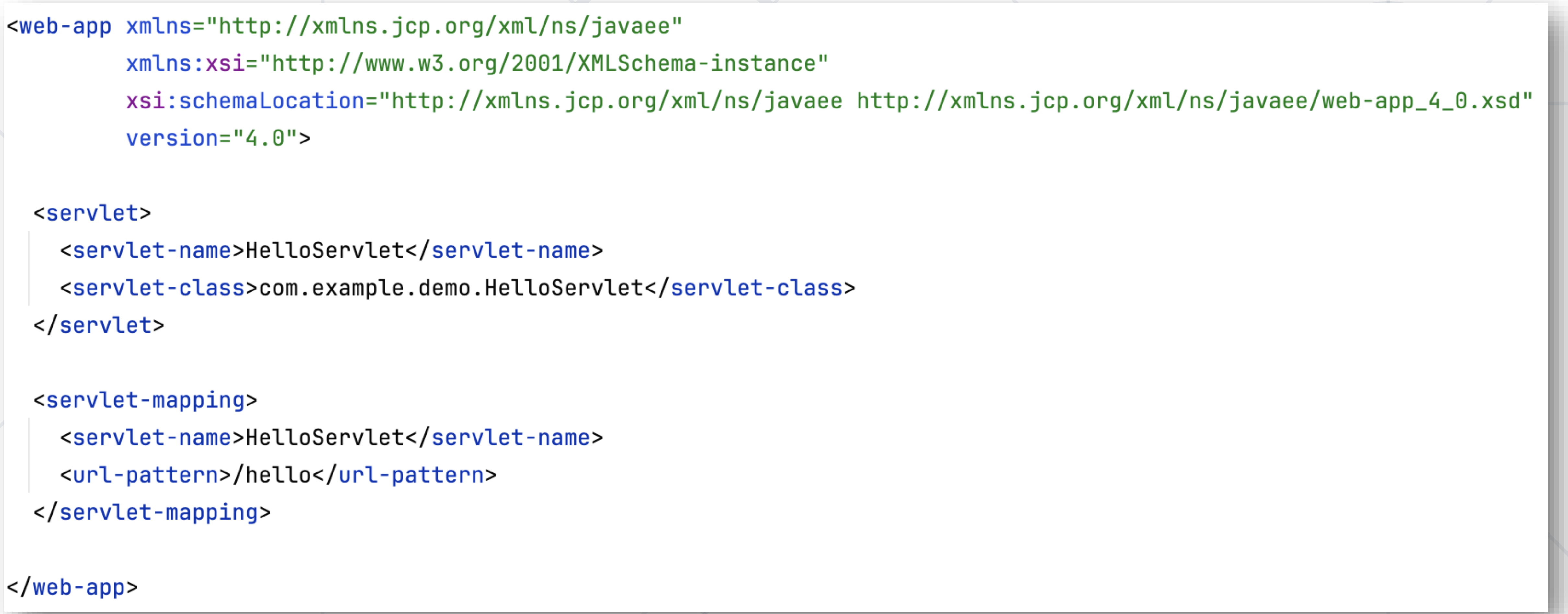
- Write the **Servlet**



```java
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import java.io.IOException;

public class HelloServlet extends HttpServlet {

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws IOException {
        response.setContentType("text/html");
        response.getWriter().write( s: "<h1>Hello, World!</h1>");
    }
}
```

Basic Web App Java EE + Servlets

- Servlet **configuration** (web.xml)



```xml
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app_4_0.xsd"
         version="4.0">

  <servlet>
    <servlet-name>HelloServlet</servlet-name>
    <servlet-class>com.example.demo.HelloServlet</servlet-class>
  </servlet>

  <servlet-mapping>
    <servlet-name>HelloServlet</servlet-name>
    <url-pattern>/hello</url-pattern>
  </servlet-mapping>

</web-app>
```

Basic Web App Java EE + Servlets

- **WAR File Packaging**: Use **Maven** or **Gradle** to package the application into a WAR file
- Deploy to Servlet Container **(Tomcat)**:
  - Copy the WAR file to Tomcat's "**webapps**/" directory
  - Start Tomcat and access **http://localhost:8080/hello**

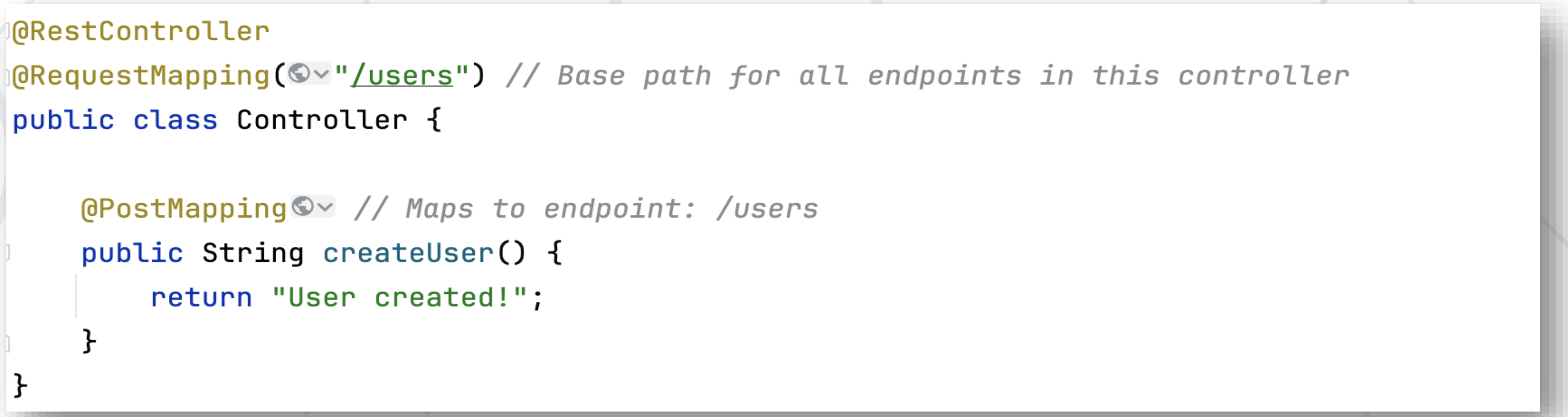
Key Pain Points with Servlets

- **Manual Configuration**
  - web.xml or via annotations
- **Complex Deployment**
  - Packaging and deploying a WAR file
- **Limited Features**
  - Manually handle advanced features
- **An easier alternative? Yes - Spring Controllers**

Spring Controllers

What are Spring Controllers?

- Java classes that handle HTTP requests and send back responses
- Simplify handling web requests compared to raw Servlets
- Two Types:
  - @Controller: Used for returning views (like HTML pages or templates)
  - @RestController: Used for returning raw data (like JSON or XML), typically in REST APIs

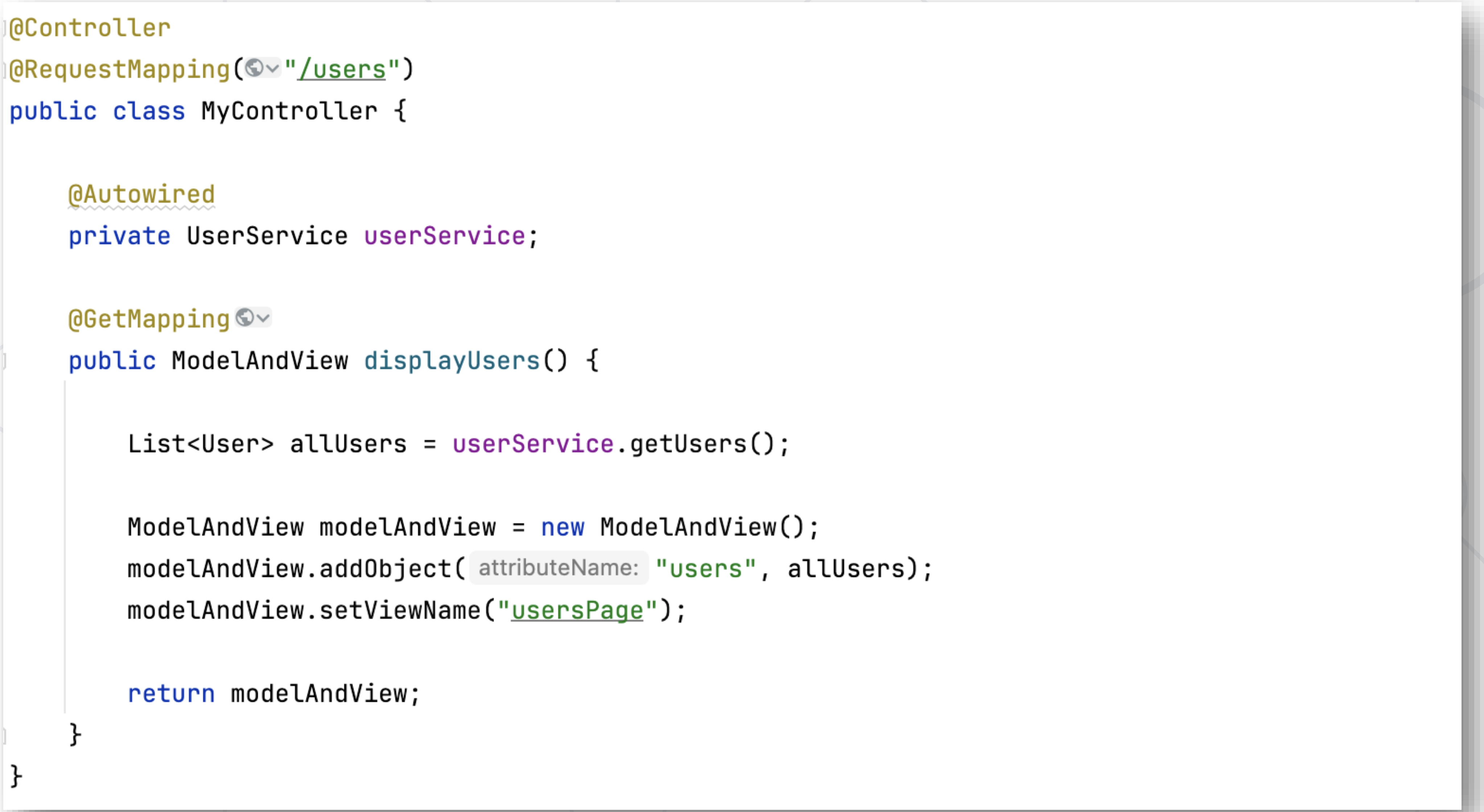


```java
@RestController
@RequestMapping("/users") // Base path for all endpoints in this controller
public class Controller {

    @PostMapping // Maps to endpoint: /users
    public String createUser() {
        return "User created!";
    }
}
```

@Controller

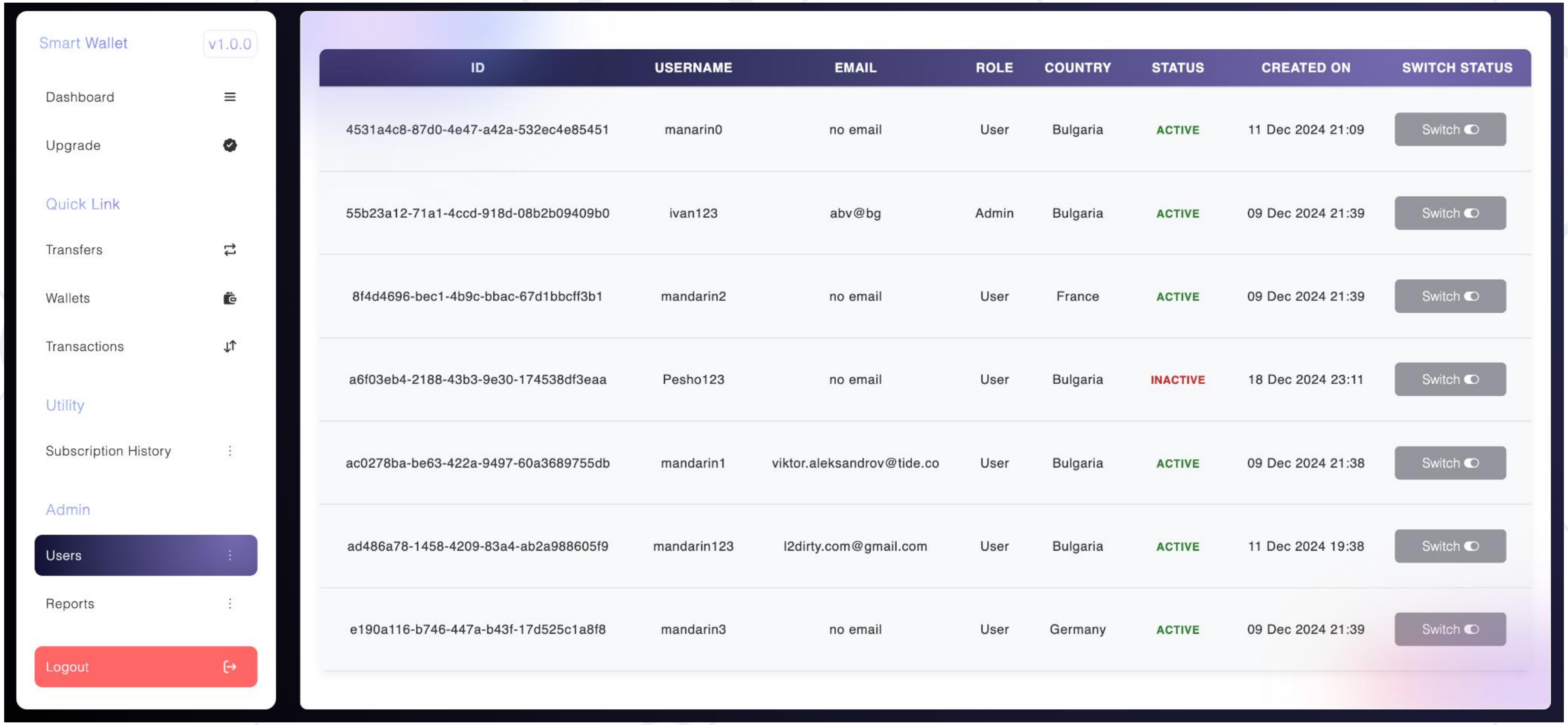
- Marks a class as a **Spring MVC** Controller
- **Spring-managed bean**
- Return views (HTML pages) rendered by a view technology (e.g., Thymeleaf, JSP)

@Controller



```
@Controller
@RequestMapping("/users")
public class MyController {

    @Autowired
    private UserService userService;

    @GetMapping
    public ModelAndView displayUsers() {

        List<User> allUsers = userService.getUsers();

        ModelAndView modelAndView = new ModelAndView();
        modelAndView.addObject( attributeName: "users", allUsers);
        modelAndView.setViewName("usersPage");

        return modelAndView;
    }
}
```

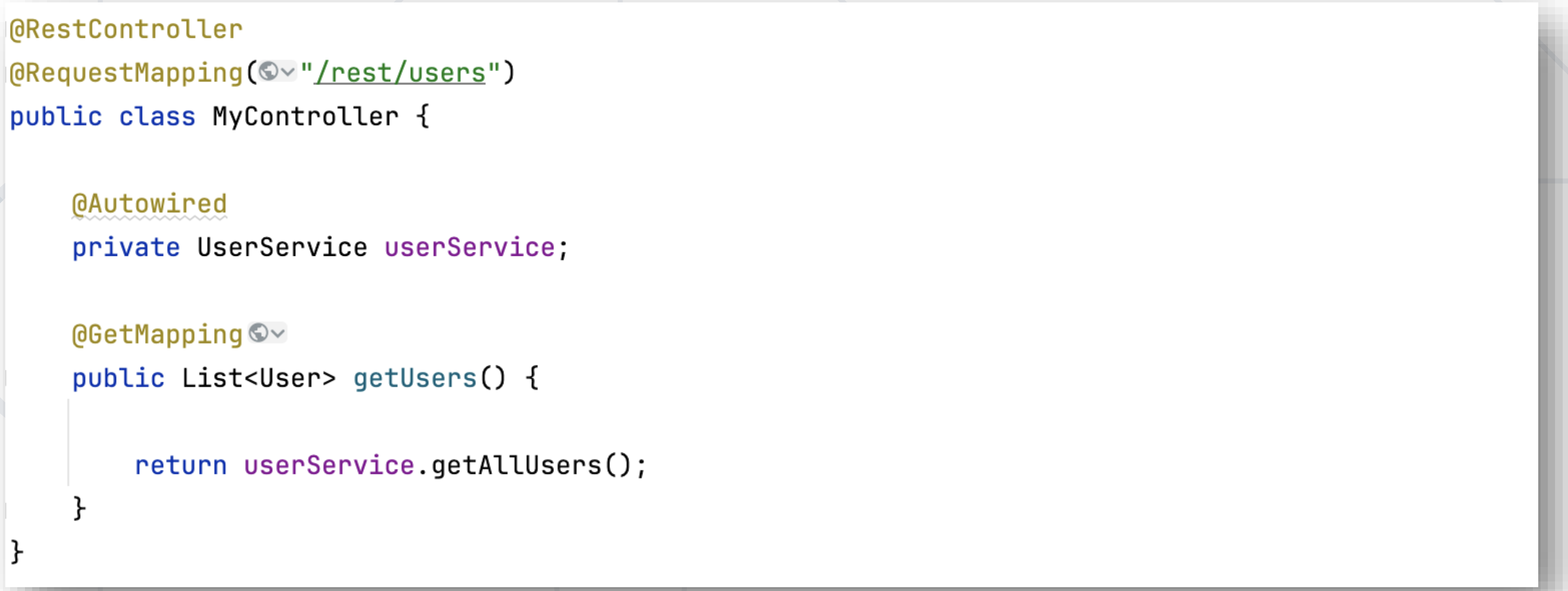
@Controller



ID	USERNAME	EMAIL	ROLE	COUNTRY	STATUS	CREATED ON	SWITCH STATUS
4531a4c8-87d0-4e47-a42a-532ec4e85451	manarin0	no email	User	Bulgaria	ACTIVE	11 Dec 2024 21:09	Switch
55b23a12-71a1-4ccd-918d-08b2b09409b0	ivan123	abv@bg	Admin	Bulgaria	ACTIVE	09 Dec 2024 21:39	Switch
8f4d4696-bec1-4b9c-bbac-67d1bbcff3b1	mandarin2	no email	User	France	ACTIVE	09 Dec 2024 21:39	Switch
a6f03eb4-2188-43b3-9e30-174538df3eaa	Pesho123	no email	User	Bulgaria	INACTIVE	18 Dec 2024 23:11	Switch
ac0278ba-be63-422a-9497-60a3689755db	mandarin1	viktor.aleksandrov@tide.co	User	Bulgaria	ACTIVE	09 Dec 2024 21:38	Switch
ad486a78-1458-4209-83a4-ab2a988605f9	mandarin123	l2dirty.com@gmail.com	User	Bulgaria	ACTIVE	11 Dec 2024 19:38	Switch
e190a116-b746-447a-b43f-17d525c1a8f8	mandarin3	no email	User	Germany	ACTIVE	09 Dec 2024 21:39	Switch

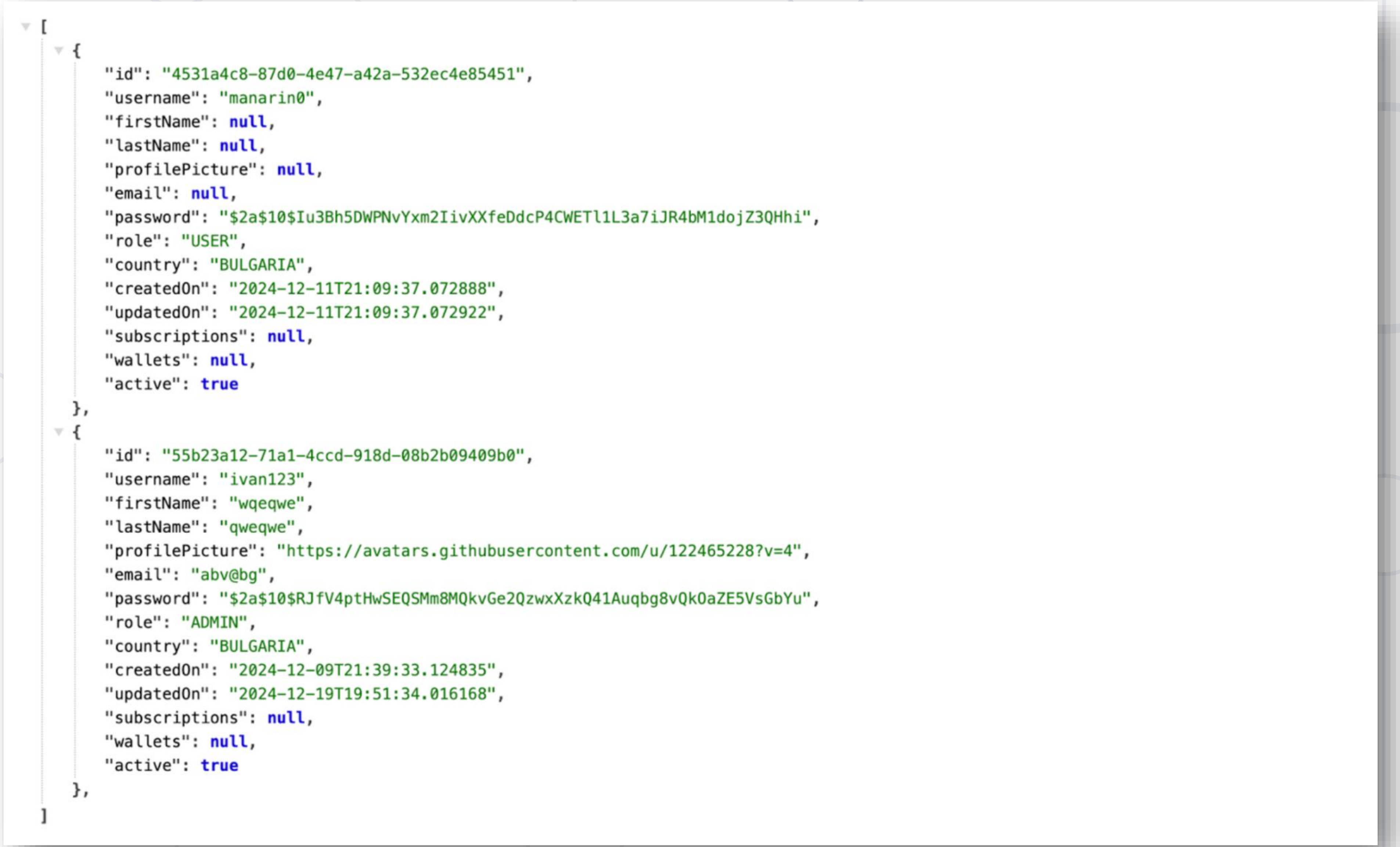
@RestController

- **Spring-managed bean**
- Building **REST** APIs that return raw data (e.g., JSON, XML)
- **Does not return views**, instead it directly serializes data to the response



```
@RestController
@RequestMapping("/rest/users")
public class MyController {

    @Autowired
    private UserService userService;

    @GetMapping
    public List<User> getUsers() {

        return userService.getAllUsers();
    }
}
```

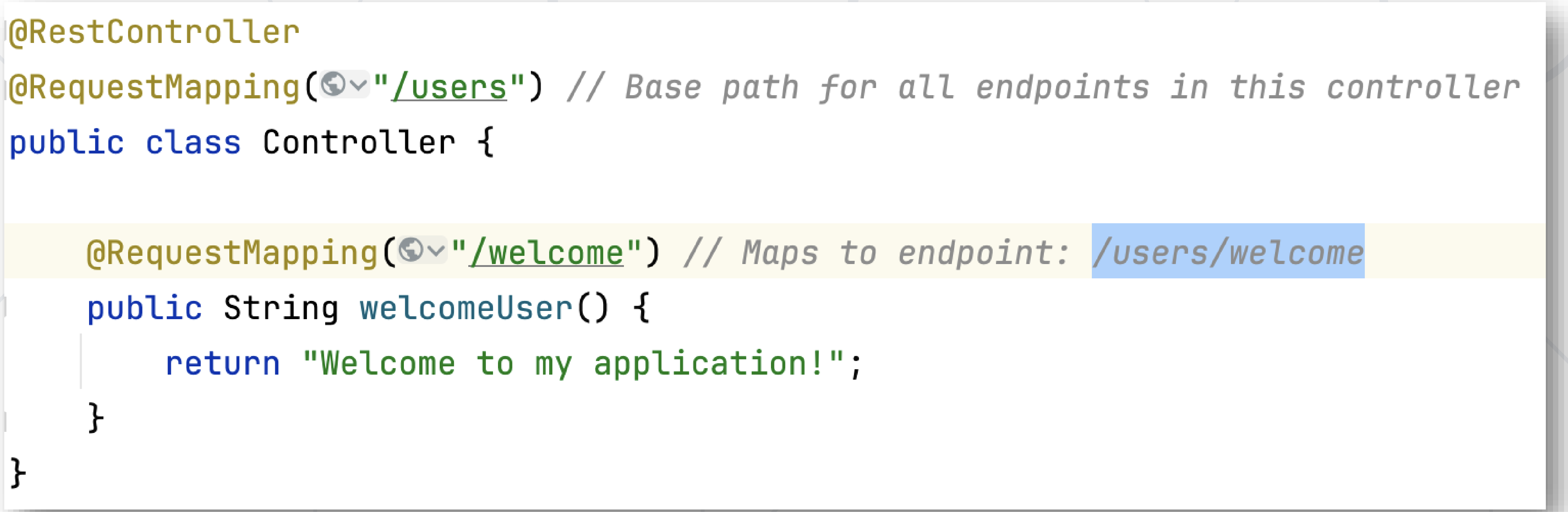
@RestController



```
[
  {
    "id": "4531a4c8-87d0-4e47-a42a-532ec4e85451",
    "username": "manarin0",
    "firstName": null,
    "lastName": null,
    "profilePicture": null,
    "email": null,
    "password": "$2a$10$Iu3Bh5DWPNvYxm2IivXXfeDdcP4CWETl1L3a7iJR4bM1dojZ3QHhi",
    "role": "USER",
    "country": "BULGARIA",
    "createdOn": "2024-12-11T21:09:37.072888",
    "updatedOn": "2024-12-11T21:09:37.072922",
    "subscriptions": null,
    "wallets": null,
    "active": true
  },
  {
    "id": "55b23a12-71a1-4ccd-918d-08b2b09409b0",
    "username": "ivan123",
    "firstName": "wqeqwe",
    "lastName": "qweqwe",
    "profilePicture": "https://avatars.githubusercontent.com/u/122465228?v=4",
    "email": "abv@bg",
    "password": "$2a$10$RJfV4ptHwSEQSMm8MQkvGe2QzwxXzkQ41Auqbg8vQkOaZE5VsGbYu",
    "role": "ADMIN",
    "country": "BULGARIA",
    "createdOn": "2024-12-09T21:39:33.124835",
    "updatedOn": "2024-12-19T19:51:34.016168",
    "subscriptions": null,
    "wallets": null,
    "active": true
  },
]
```

@RequestMapping

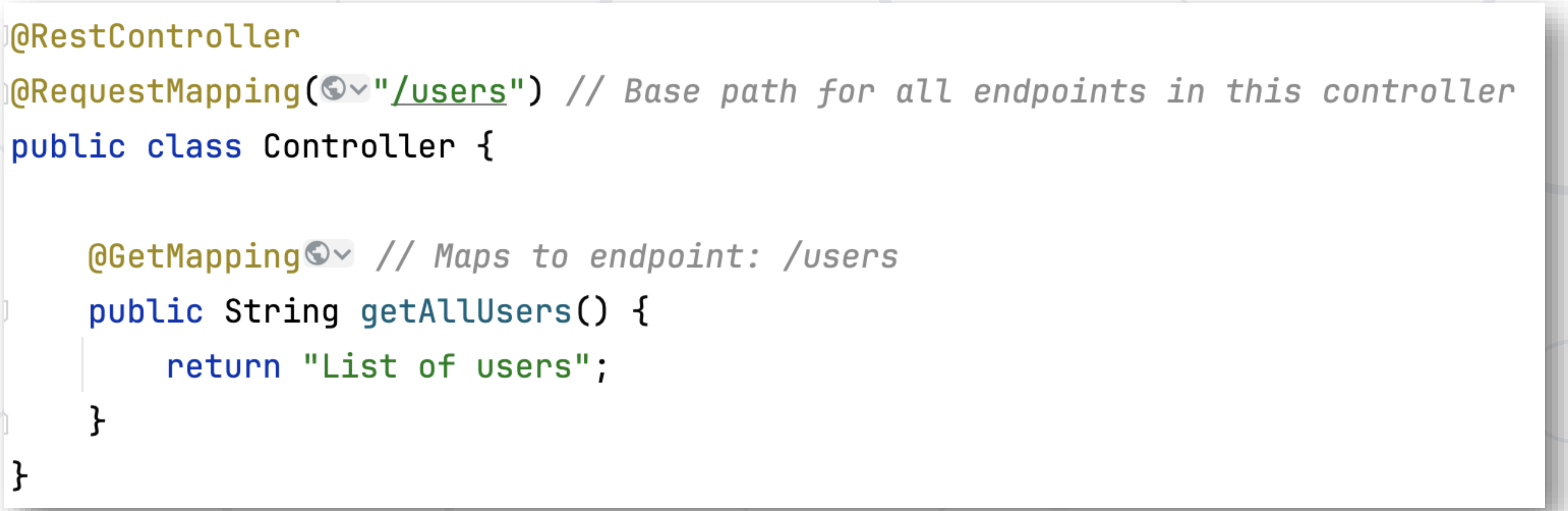
- Maps specific **URLs** to **controller** methods
- Defines **base paths** (on classes) or **specific endpoints** (on methods)



```java
@RestController
@RequestMapping("/users") // Base path for all endpoints in this controller
public class Controller {

    @RequestMapping("/welcome") // Maps to endpoint: /users/welcome
    public String welcomeUser() {
        return "Welcome to my application!";
    }
}
```

@GetMapping

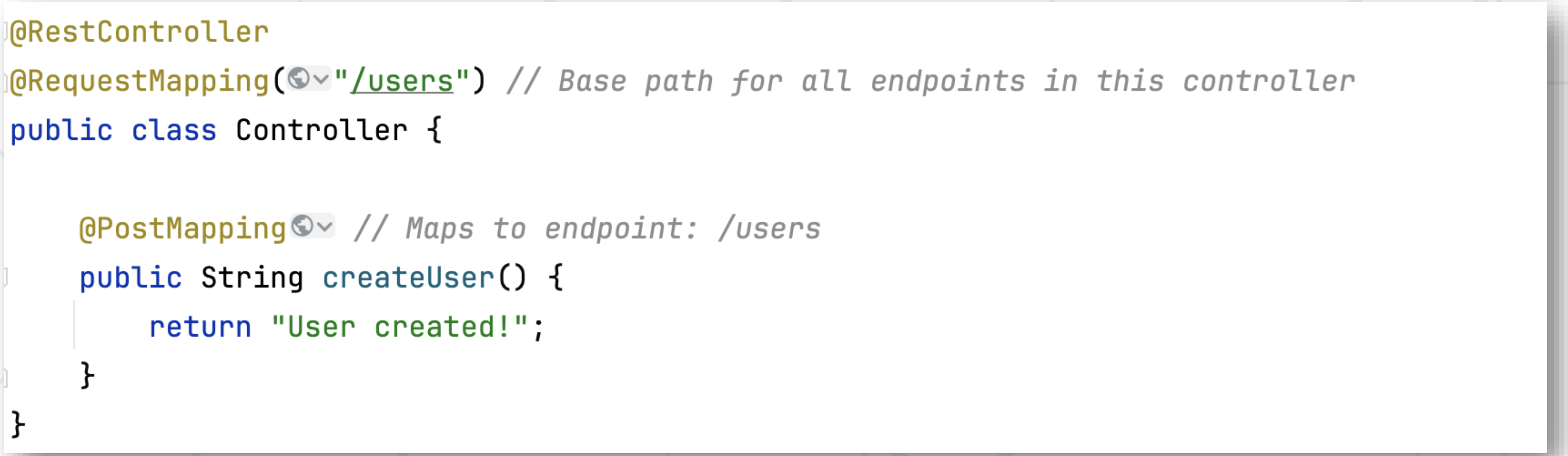
- Handles HTTP **GET** requests, used for retrieving data



```java
@RestController
@RequestMapping("/users") // Base path for all endpoints in this controller
public class Controller {

    @GetMapping // Maps to endpoint: /users
    public String getAllUsers() {
        return "List of users";
    }
}
```

@PostMapping

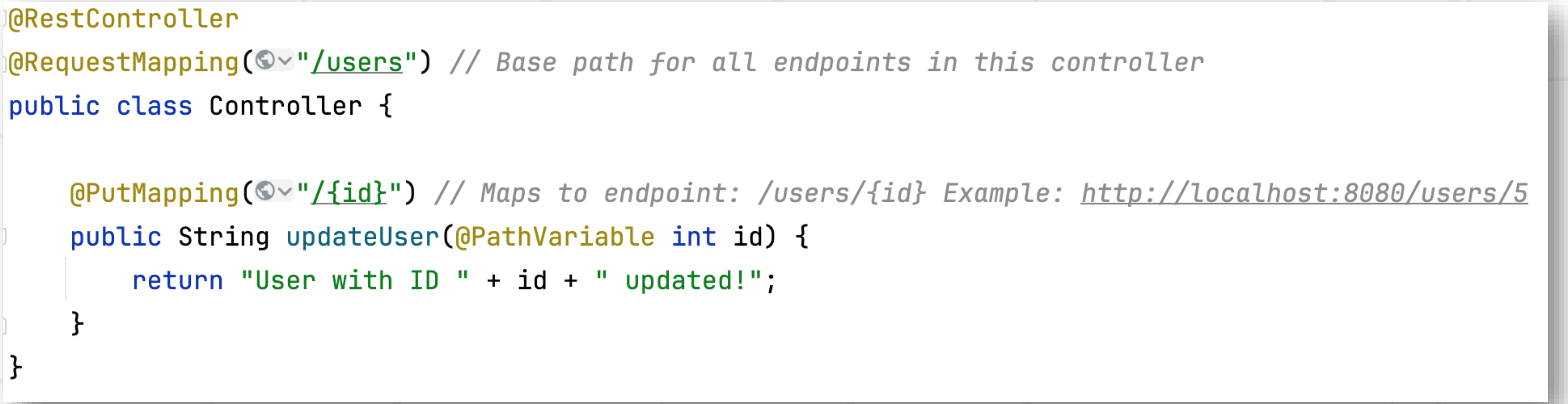
- Handles HTTP **POST** requests, used for creating resources



```java
@RestController
@RequestMapping("/users") // Base path for all endpoints in this controller
public class Controller {

    @PostMapping // Maps to endpoint: /users
    public String createUser() {
        return "User created!";
    }
}
```

@PutMapping

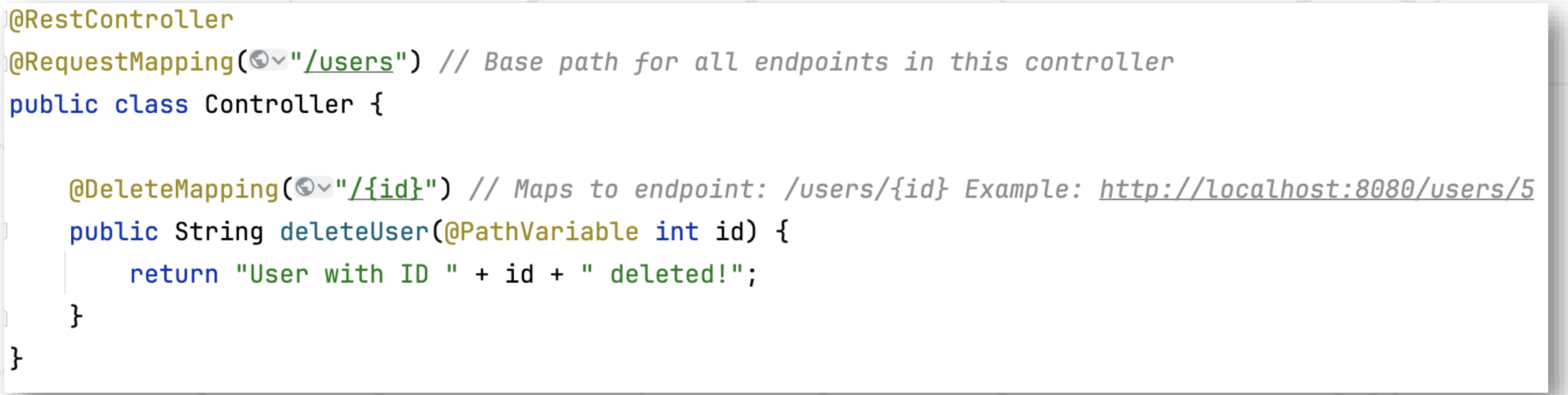
- Handles HTTP **PUT** requests, used for updating resources



```java
@RestController
@RequestMapping("/users") // Base path for all endpoints in this controller
public class Controller {

    @PutMapping("/{id}") // Maps to endpoint: /users/{id} Example: http://localhost:8080/users/5
    public String updateUser(@PathVariable int id) {
        return "User with ID " + id + " updated!";
    }
}
```

@DeleteMapping

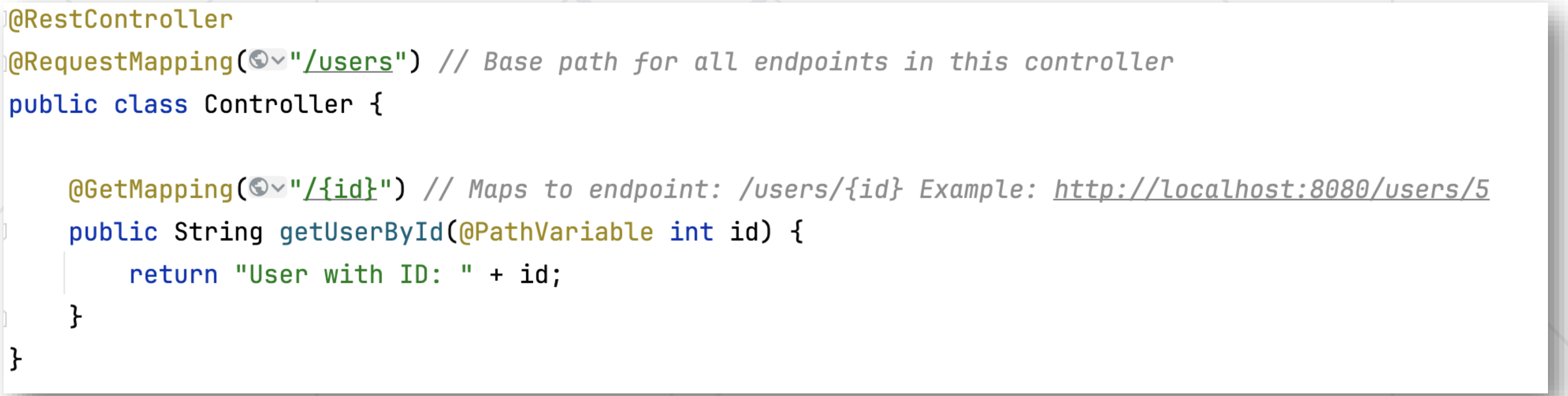
- Handles HTTP **DELETE** requests, used for deleting resources



```java
@RestController
@RequestMapping("/users") // Base path for all endpoints in this controller
public class Controller {

    @DeleteMapping("/{id}") // Maps to endpoint: /users/{id} Example: http://localhost:8080/users/5
    public String deleteUser(@PathVariable int id) {
        return "User with ID " + id + " deleted!";
    }
}
```

Path Variables: @PathVariable

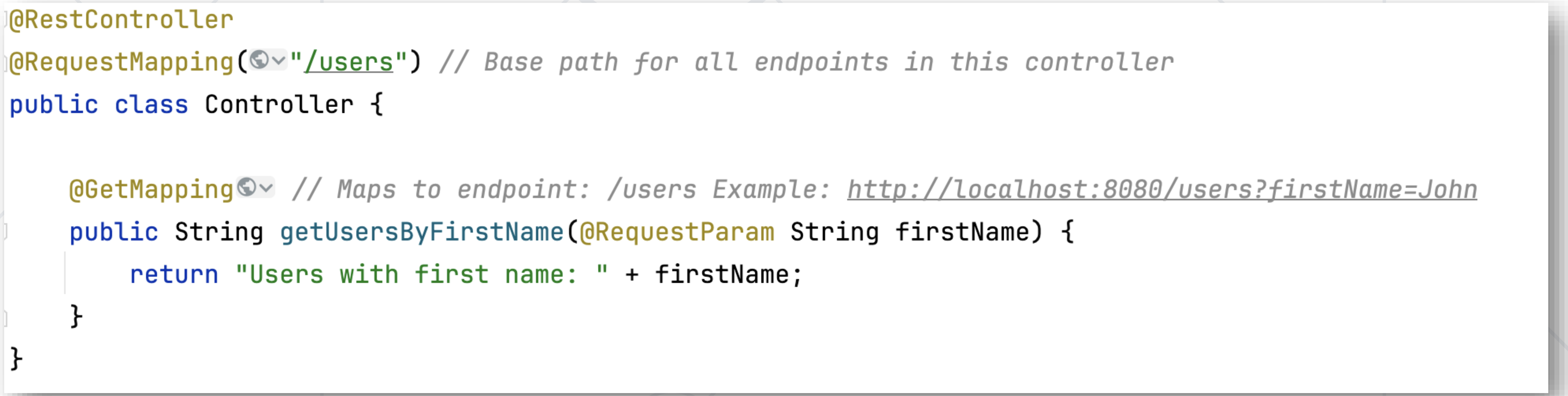
- Extracts **values** from URL path parameters
  - URL: localhost:8080/users/**5**
  - Path Variable: id = 5



```java
@RestController
@RequestMapping("/users") // Base path for all endpoints in this controller
public class Controller {

    @GetMapping("/{id}") // Maps to endpoint: /users/{id} Example: http://localhost:8080/users/5
    public String getUserById(@PathVariable int id) {
        return "User with ID: " + id;
    }
}
```

Query Parameters: @RequestParam

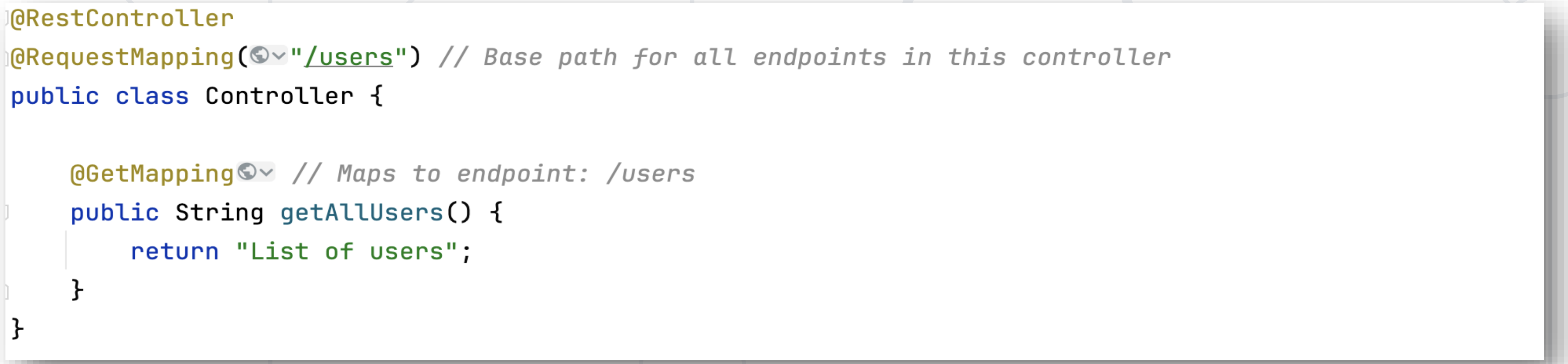
- Extracts **query parameters** from the URL
  - URL: localhost:8080/users**?**firstName=**John**
  - Query Parameters: firstName = John



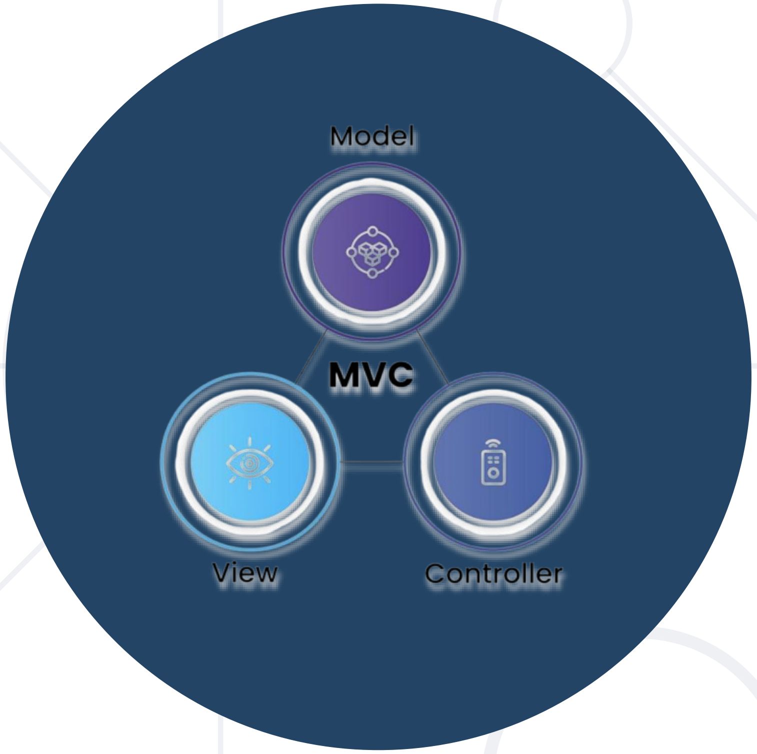
```java
@RestController
@RequestMapping("/users") // Base path for all endpoints in this controller
public class Controller {

    @GetMapping // Maps to endpoint: /users Example: http://localhost:8080/users?firstName=John
    public String getUsersByFirstName(@RequestParam String firstName) {
        return "Users with first name: " + firstName;
    }
}
```

What is an Endpoint?

- A specific **URL pattern** where a client interacts with a server
- Example: Running a Spring Boot app locally on port 8080
  - **Base URL:** http://localhost:8080/
  - **Endpoint:** /**users**
  - **Full URL:** http://localhost:8080/**users**



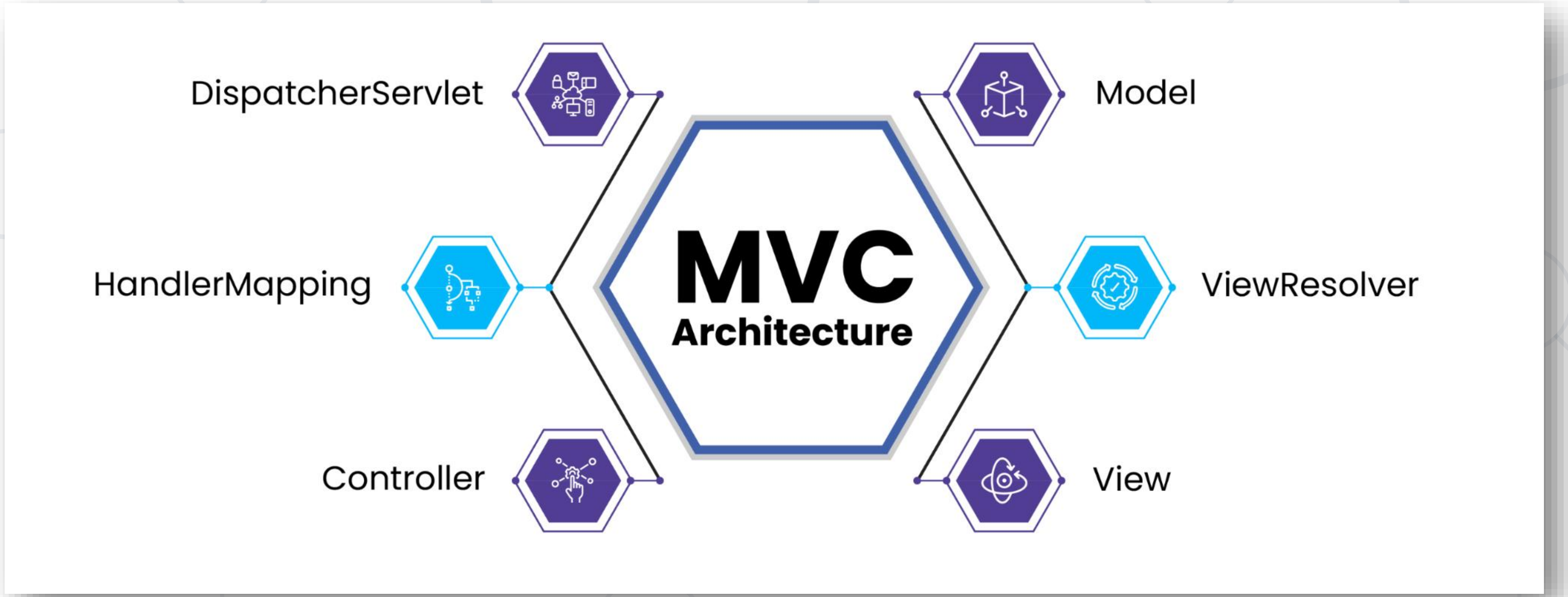
```
@RestController
@RequestMapping("/users") // Base path for all endpoints in this controller
public class Controller {

    @GetMapping // Maps to endpoint: /users
    public String getAllUsers() {
        return "List of users";
    }
}
```



Spring MVC

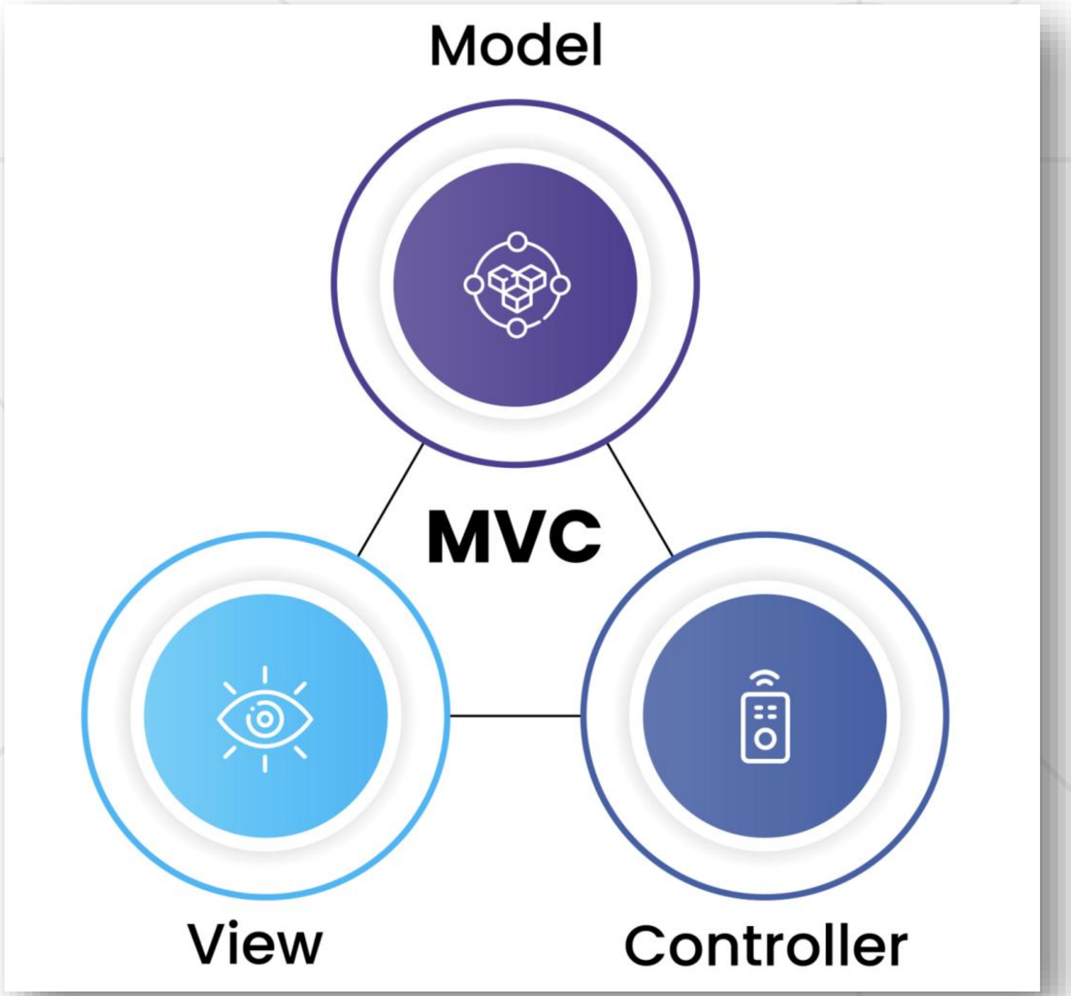
Spring MVC

- Spring **MVC** is a **Spring** module for creating **web** applications
- It follows the **Model-View-Controller** architectural pattern



Three Main Components

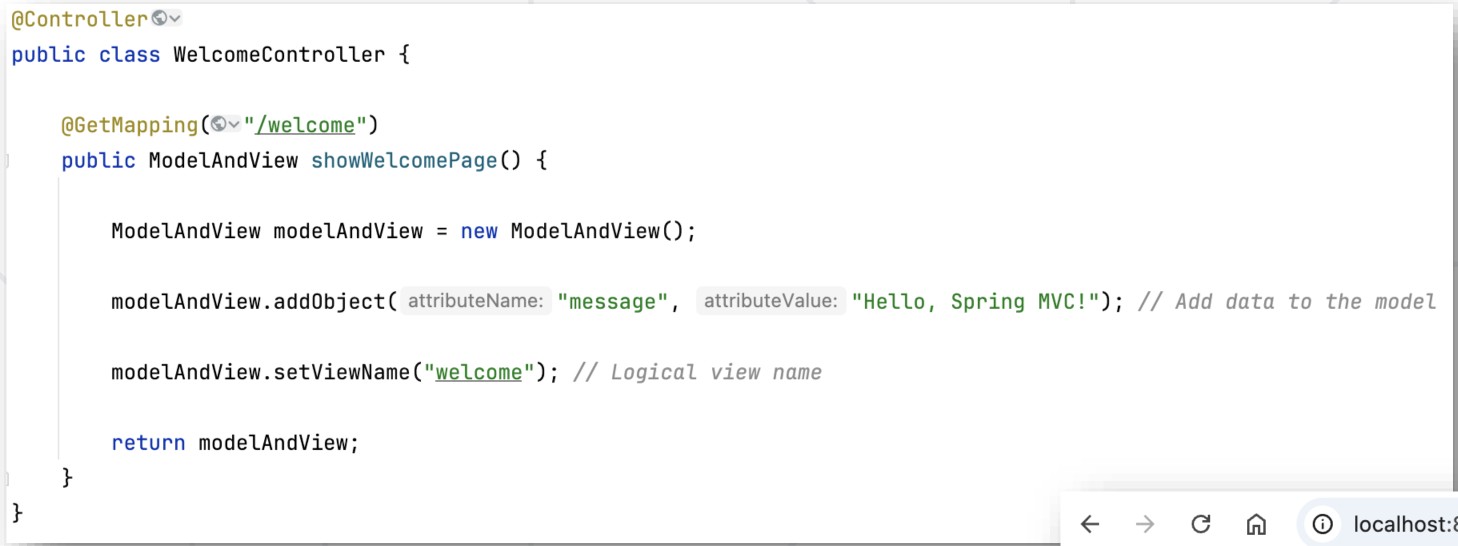
- **Model**: A container for data sent from the **Controller** to the View
- **View**: Defines what the user sees (HTML pages)
- **Controller**: Manages incoming requests and decides what data to send to which **View**



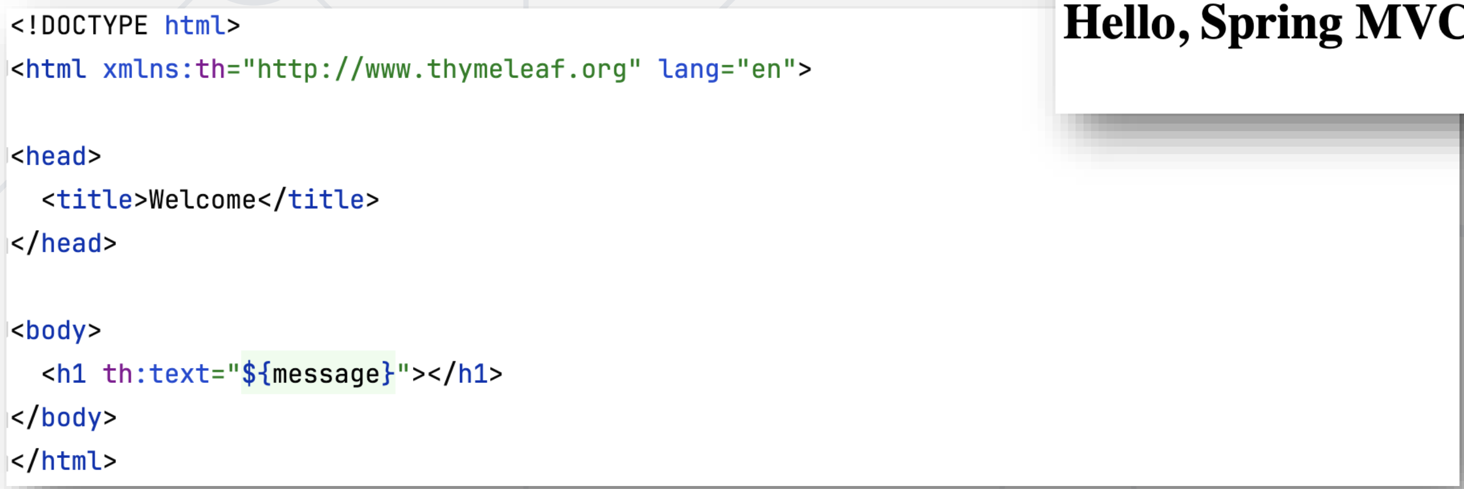
Model

- It holds data that the **Controller** prepares
- It carries information like user details, form submissions, or any other data to the **View**
- **Key-Value Data Structure**: The data in a Model is usually passed as key-value pairs

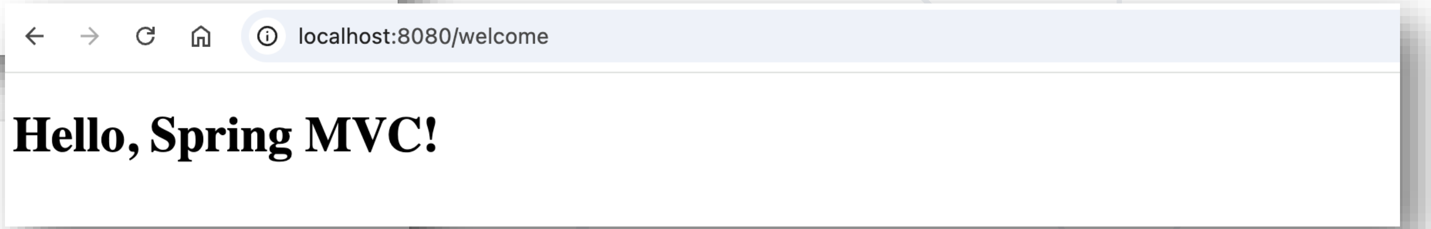
Model



```java
@Controller
public class WelcomeController {

    @GetMapping("/welcome")
    public ModelAndView showWelcomePage() {

        ModelAndView modelAndView = new ModelAndView();

        modelAndView.addObject( attributeName: "message",  attributeValue: "Hello, Spring MVC!"); // Add data to the model

        modelAndView.setViewName("welcome"); // Logical view name

        return modelAndView;
    }
}
```



```html
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org" lang="en">

<head>
  <title>Welcome</title>
</head>

<body>
  <h1 th:text="${message}"></h1>
</body>
</html>
```



localhost:8080/welcome

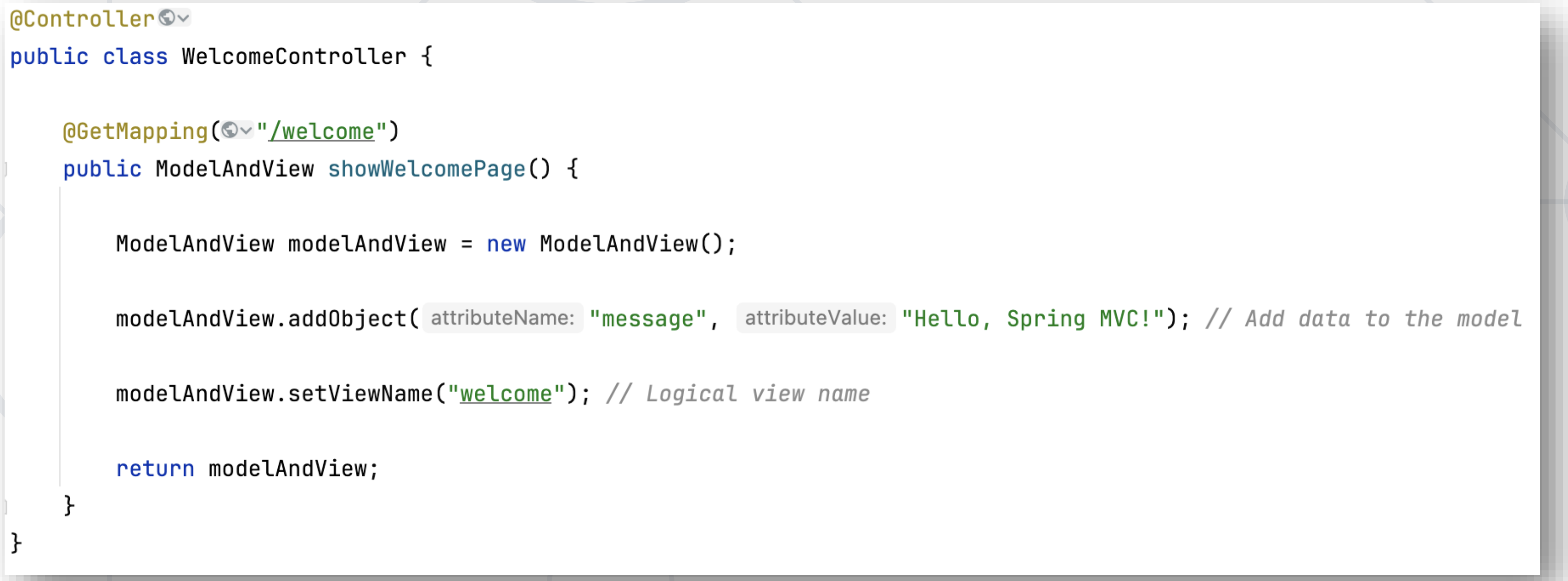
Hello, Spring MVC!

View

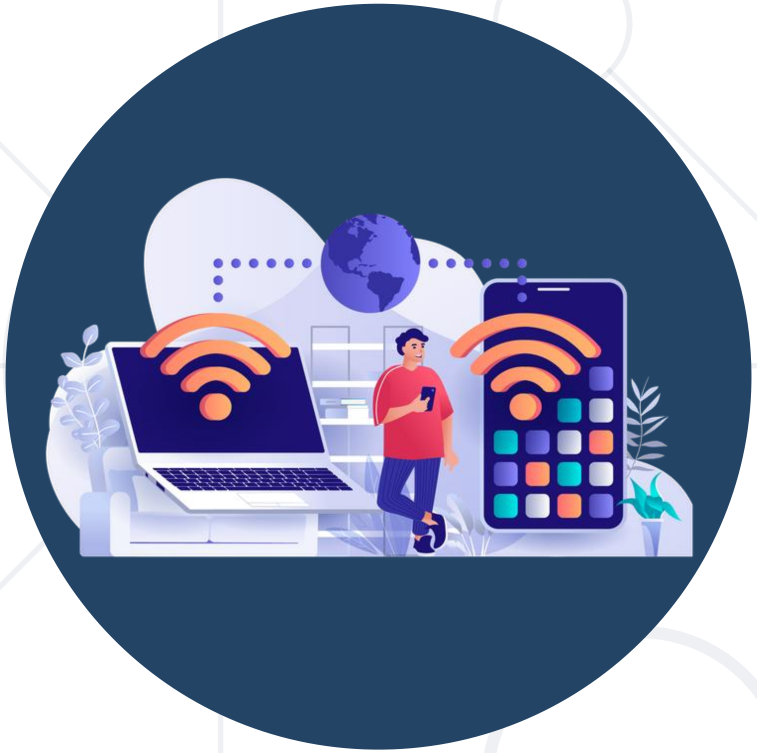
- Is simply the **HTML page** that is sent to the browser
- It's where the user sees the data prepared by the **Controller**
- **Static and Dynamic Views**:
  - **Static View**: Plain HTML page
  - **Dynamic View**: An HTML template with placeholders that are replaced with data from the **Model**

Controller

- Responsible for handling user requests
- choosing a **View** that will be sent to the user
- **Controller** passes data to the view



```java
@Controller
public class WelcomeController {

    @GetMapping("/welcome")
    public ModelAndView showWelcomePage() {

        ModelAndView modelAndView = new ModelAndView();

        modelAndView.addObject( attributeName: "message",  attributeValue: "Hello, Spring MVC!"); // Add data to the model

        modelAndView.setViewName("welcome"); // Logical view name

        return modelAndView;
    }
}
```



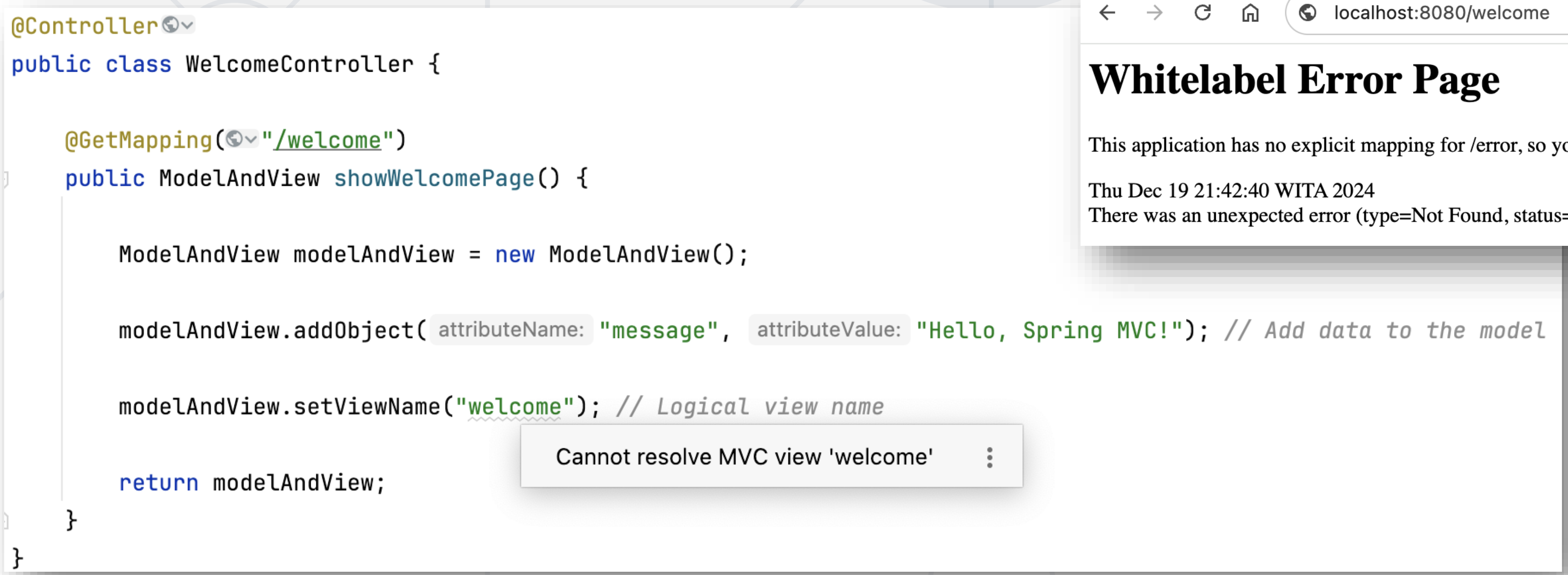
View Technology

View Technology

- View technologies are **libraries** that help generate web **pages (e.g. HTML)** from data
- They work with **Spring MVC** to render the **UI**
- Examples of **View Technologies**:
  - **Thymeleaf** (modern)
  - **JSP** (legacy)
  - **Freemarker**, **Mustache**, etc.

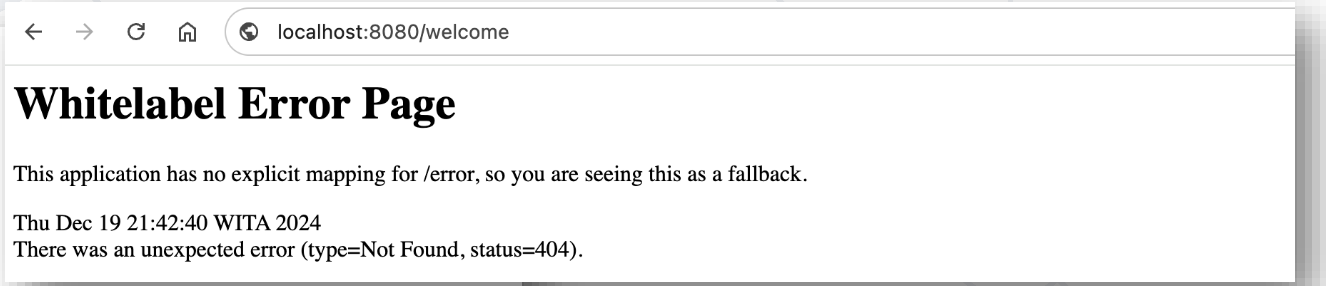
Why Do We Need View Technology?

- Without a view technology, Spring MVC **cannot resolve view names** (e.g., "**welcome**") to the actual **HTML** file (e.g., "**welcome.html**")
- The browser would get a **404 error**



```java
@Controller
public class WelcomeController {

    @GetMapping("/welcome")
    public ModelAndView showWelcomePage() {

        ModelAndView modelAndView = new ModelAndView();

        modelAndView.addObject( attributeName: "message", attributeValue: "Hello, Spring MVC!"); // Add data to the model

        modelAndView.setViewName("welcome"); // Logical view name

        return modelAndView;
    }
}
```

Cannot resolve MVC view 'welcome'



localhost:8080/welcome

Whitelabel Error Page

This application has no explicit mapping for /error, so you are seeing this as a fallback.

Thu Dec 19 21:42:40 WITA 2024
There was an unexpected error (type=Not Found, status=404).

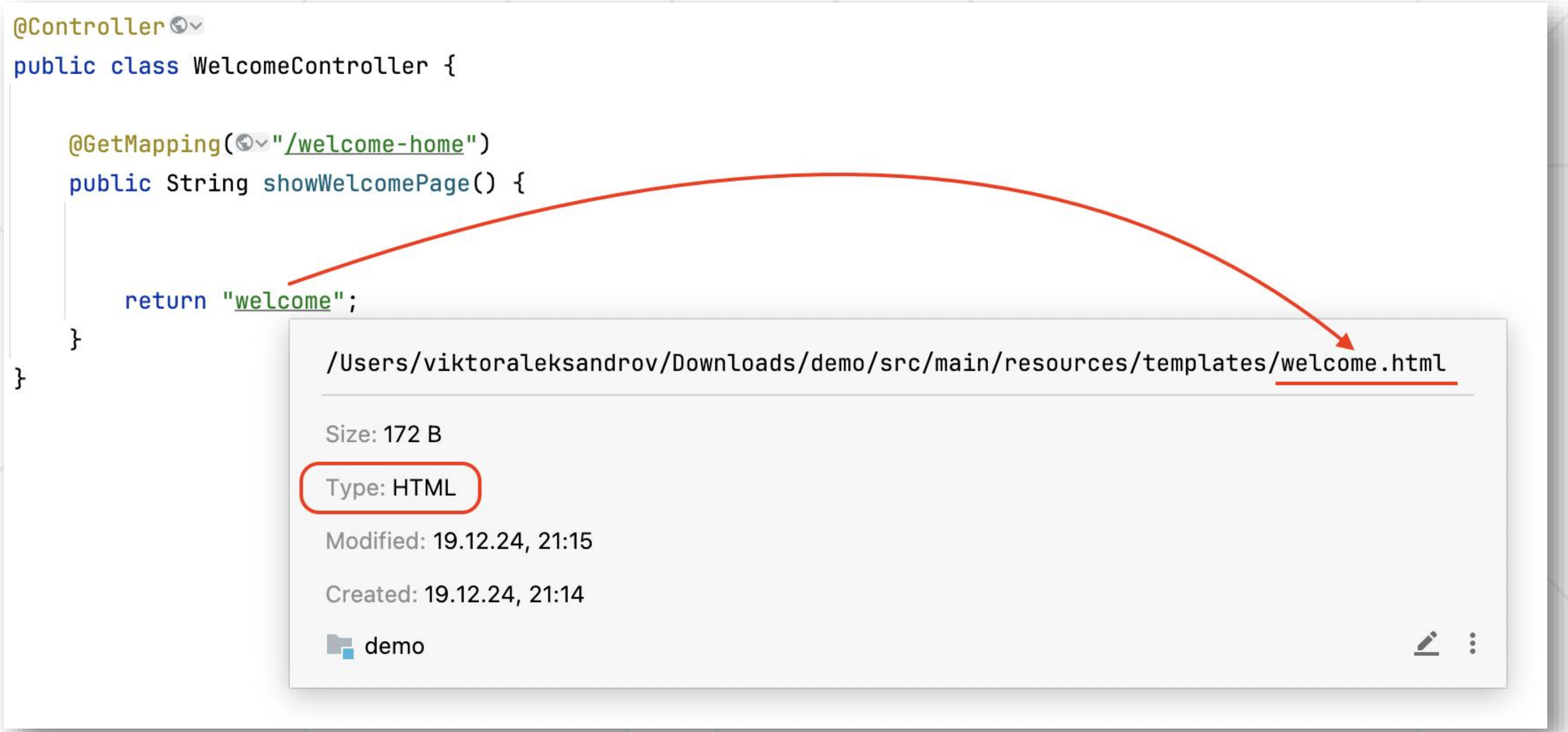
Thymeleaf

What is Thymeleaf?

- **Thymeleaf** is a **server-side** Java **template engine** (View Technology) for rendering dynamic HTML
- **Setting Up Thymeleaf**:
  - Add **spring-boot-starter-thymeleaf** dependency
  - Create new templates/views in the **resources/templates** folder

+

What is Thymeleaf?



```
@Controller
public class WelcomeController {

    @GetMapping("/welcome-home")
    public String showWelcomePage() {


        return "welcome";
    }
}
```

/Users/viktoraleksandrov/Downloads/demo/src/main/resources/templates/welcome.html

Size: 172 B

Type: HTML

Modified: 19.12.24, 21:15

Created: 19.12.24, 21:14

demo

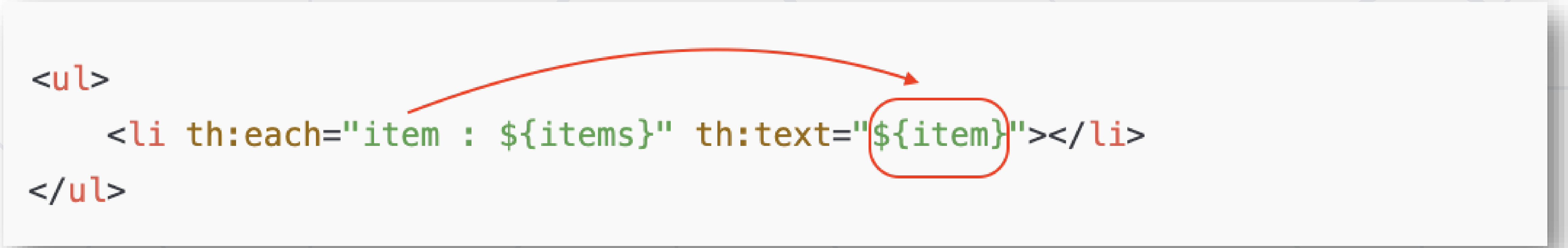
Key Thymeleaf Functions

Key Thymeleaf Functions

- **th:text**: Replaces text **dynamically**
  - Example: <p th:text="${**message**}">Default Text</p>
- **th:href**: Generates a **hyperlink**
  - Example: <a th:href="@{/**profile**}">Profile</a>
- **th:if**: Conditionally displays elements based on a **condition**
  - Example: <p th:if="${**user.loggedIn**}">Welcome back!</p>
  - Example: <p th:if="${**user.age** > 24}">You are adult!</p>

Key Thymeleaf Functions

- **th:each**: Iterates over a collection



```
<ul>
    <li th:each="item : ${items}" th:text="${item}"></li>
</ul>
```

Key Thymeleaf Functions

- **th:case:** Used in a switch-like structure with th:switch

```html
<div th:switch="${role}">
    <p th:case="'admin'">Welcome, Admin!</p>
    <p th:case="'user'">Hello, User!</p>
</div>
```

Thymeleaf Utility Objects

Thymeleaf Utility Objects

- **#strings**: Provides string manipulation utilities
  - `<p th:text="${#strings.capitalize('[placeholder value]')}">[Capitalized Value]</p>`
  - `<p th:text="${#strings.substring('[placeholder value]', 0, 5)}">[Substring]</p>`

- **#temporals** (or #dates): Performs date and time operations
  - `<p th:text="${#temporals.format(myDate, 'yyyy-MM-dd')}">[Formatted Date]</p>`
  - `<p th:text="${#temporals.dayOfWeekName(myDate)}">[Day of the Week]</p>`

Thymeleaf Utility Objects

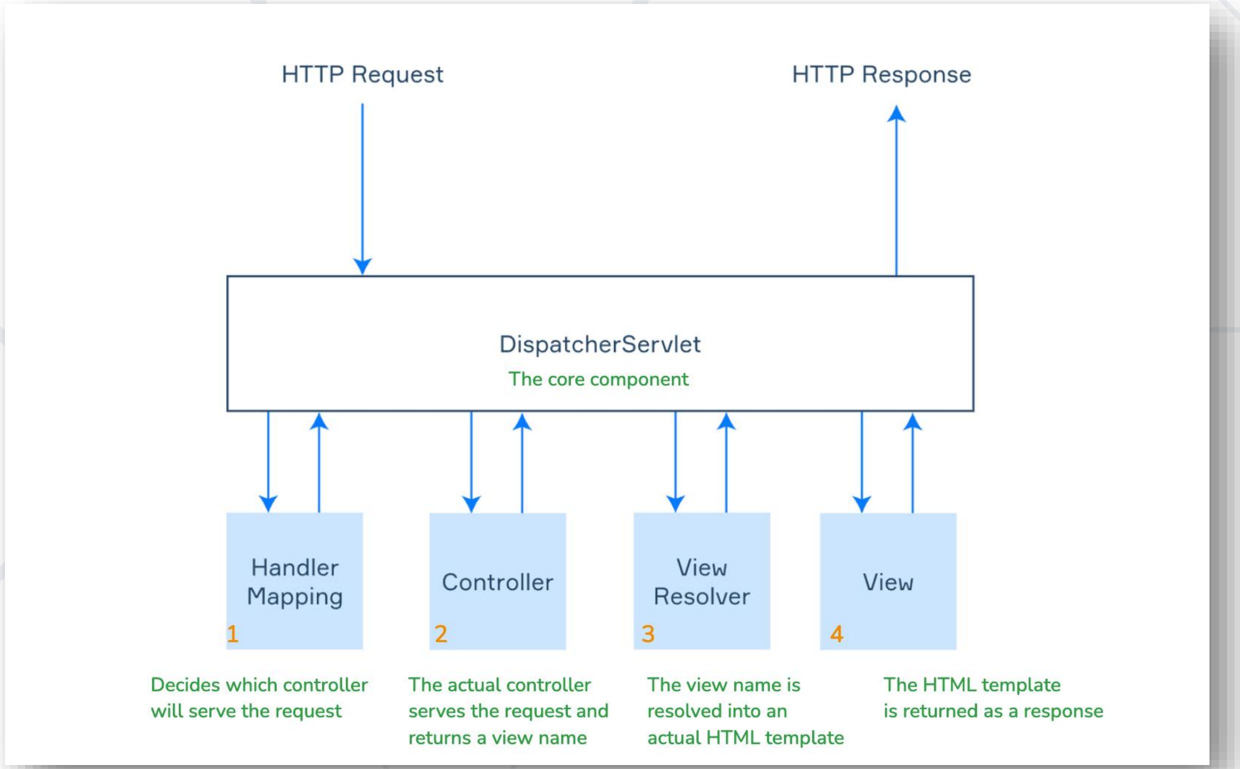
- **Other Utility Objects**:
  - **#numbers**: Formats numeric values
  - **#bools**: Handles boolean operations
  - **#lists**: Provides utilities for working with lists
  - **#maps**: Helps with map operations
  - **#arrays**: Works with arrays
  - **#messages**: Fetches messages from i18n files
  - **#uris**: Manages URI and query parameters
  - **#objects**: Provides general object utilities

Examples here

DispatcherServlet

DispatcherServlet

- The **DispatcherServlet** is the **core component** of Spring MVC, responsible for handling and processing **all** incoming HTTP requests
- Acts as the **front controller** in the MVC architecture



HTTP Request
HTTP Response
DispatcherServlet
The core component
Handler Mapping
1
Controller
2
View Resolver
3
View
4
Decides which controller will serve the request
The actual controller serves the request and returns a view name
The view name is resolved into an actual HTML template
The HTML template is returned as a response

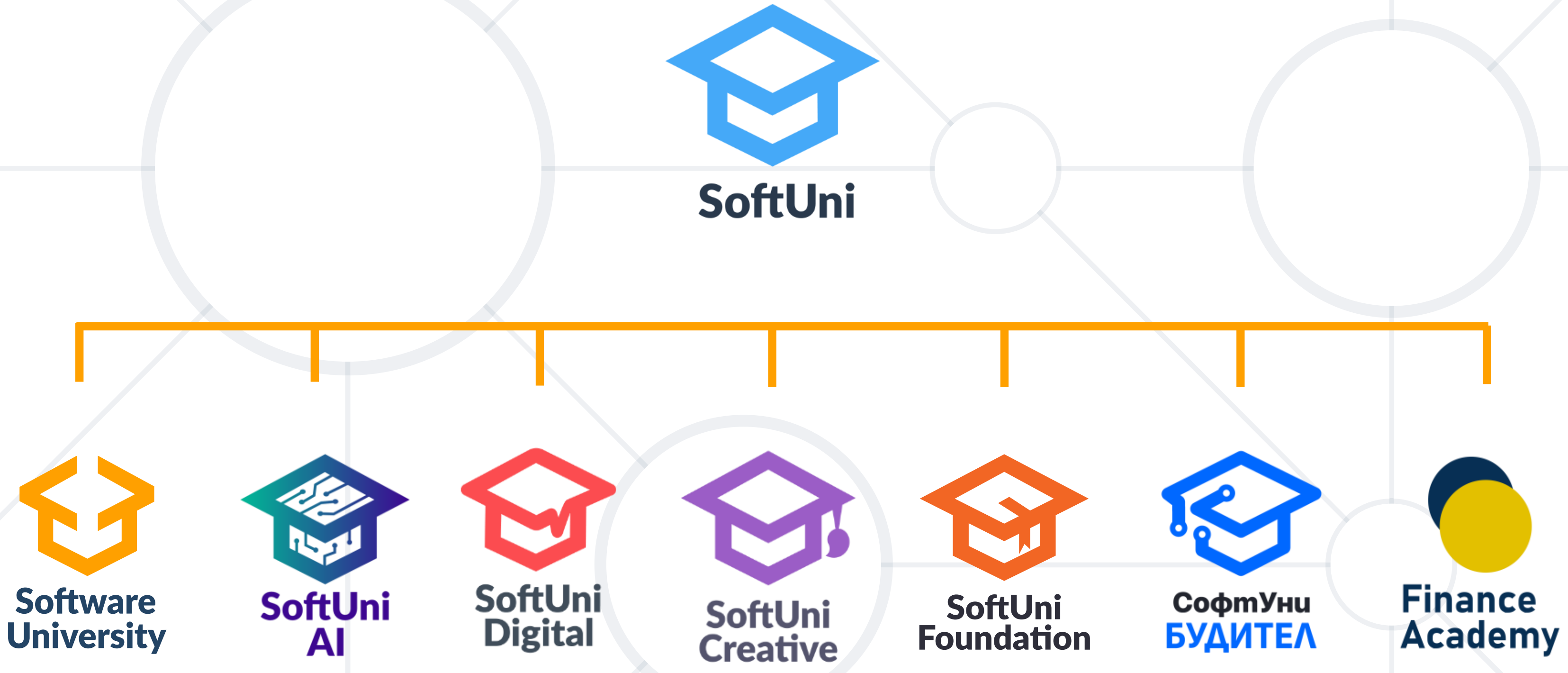
Request Processing Flow

- **HTTP Request Sent**:
  - The client sends an HTTP request (e.g., GET /users)
- **Request Hits DispatcherServlet**
- **Routing to Controller**:
  - DispatcherServlet routes the request to the appropriate controller
- **Controller Processes Request**:
  - Returns raw data (e.g. JSON, XML) if it's **@RestController** or a view data if it's **@Controller**
- **View Resolution**:
  - DispatcherServlet consults the ViewResolver for the view template
- **HTTP Response**:
- **The final response** (view or raw data) is sent back to the client

Summary

- What is Servlet?
- Spring Controllers
- Spring MVC
- View Technology
- Thymeleaf
- Key Thymeleaf Functions
- Thymeleaf Utility Objects
- DispatcherServlet

Questions?


SoftUni
Software University
SoftUni AI
SoftUni Digital
SoftUni Creative
SoftUni Foundation
СофтУни БУДИТЕЛ
Finance Academy

SoftUni Diamond Partners

Software University

Coca-Cola HBC Bulgaria

Postbank

top EMPLOYER България Bulgaria 2025 FOR A BETTER WORLD OF WORK

SUPER HOSTING .BG

encorp.ai

DRAFT KINGS THE CROWN IS YOURS

createX

INDEAVR Serving the high achievers

VIVACOM

Trainings @ Software University (SoftUni)

- Software University – High-Quality Education, Profession and Job for Software Developers
  - softuni.bg, about.softuni.bg
- Software University Foundation
  - softuni.foundation
- Software University @ Facebook
  - facebook.com/SoftwareUniversity

License