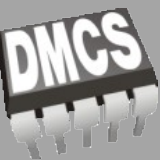




Interactive Web Applications



1. Client – Server architecture
2. Spring Framework – introduction
3. Tools
4. Spring Boot – .jsp example – „Hello world”
5. Spring Boot – .jsp example – model and simple form

1

Client – Server architecture

Definition

Mainframe / PC

Two-tier architecture

Three-tier architecture

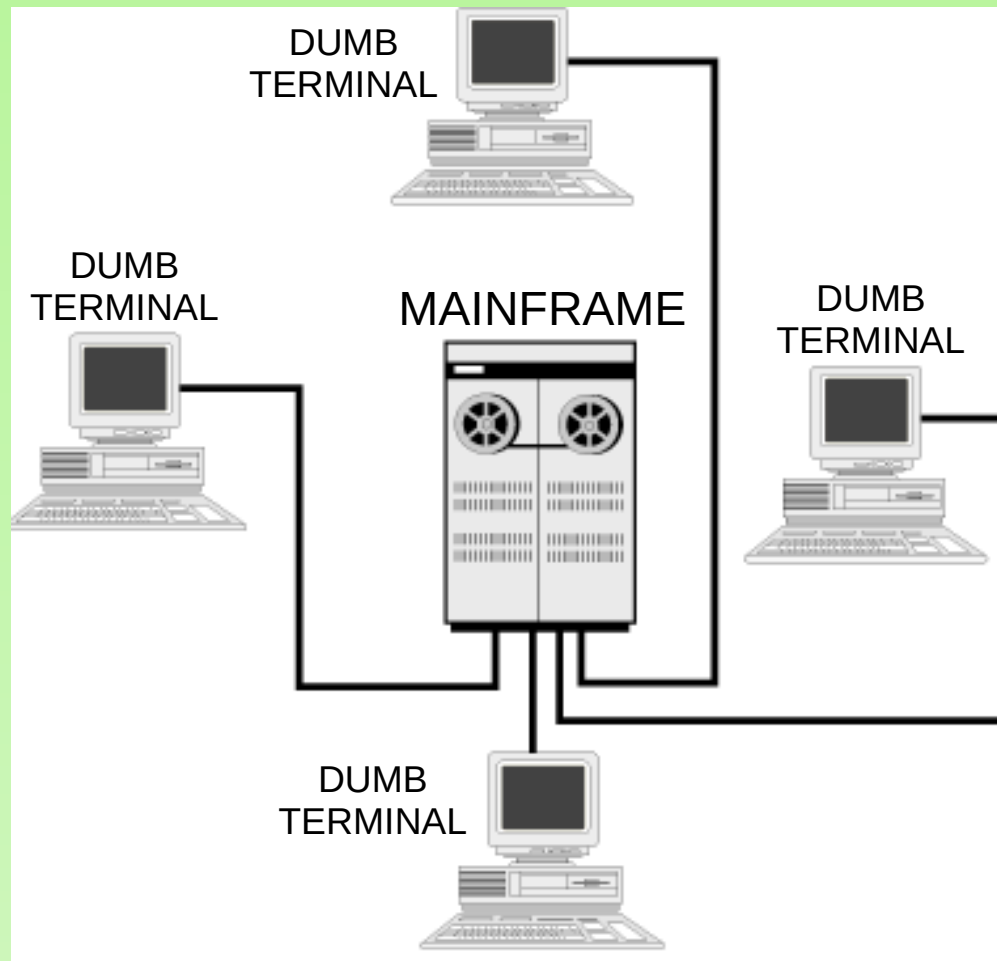
N-tier architecture

It is an architecture of a computer network in which many clients (remote processors) request and receive service from a centralized server (host computer). Client computers provide an interface to allow a computer user to request services of the server and to display the results the server returns. Servers wait for requests to arrive from clients and then respond to them. Ideally, a server provides a standardized transparent interface to clients so that clients need not be aware of the specifics of the system (i.e., the hardware and software) that is providing the service. Clients are often situated at workstations or on personal computers, while servers are located elsewhere on the network, usually on more powerful machines.

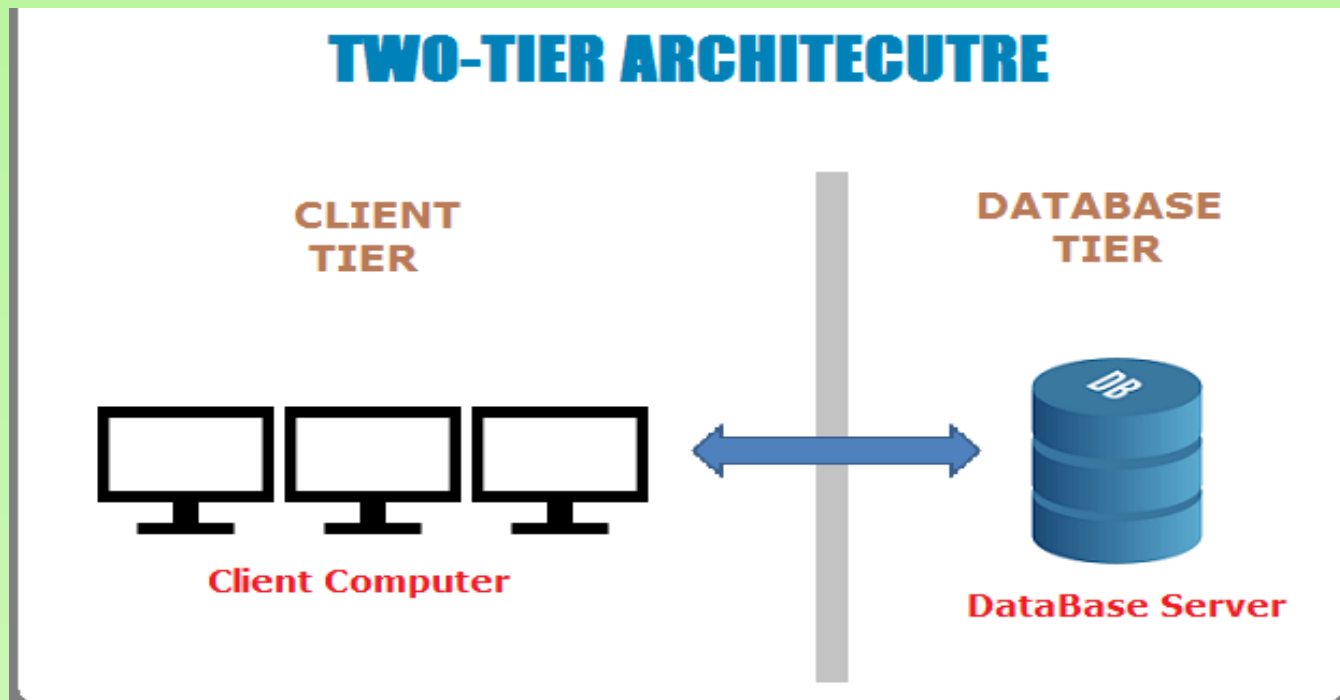
This computing model is especially effective when clients and the server each have distinct tasks that they routinely perform. Many clients can access the server's information simultaneously, and, at the same time, a client computer can perform other tasks, such as sending e-mail.

Once upon a time...

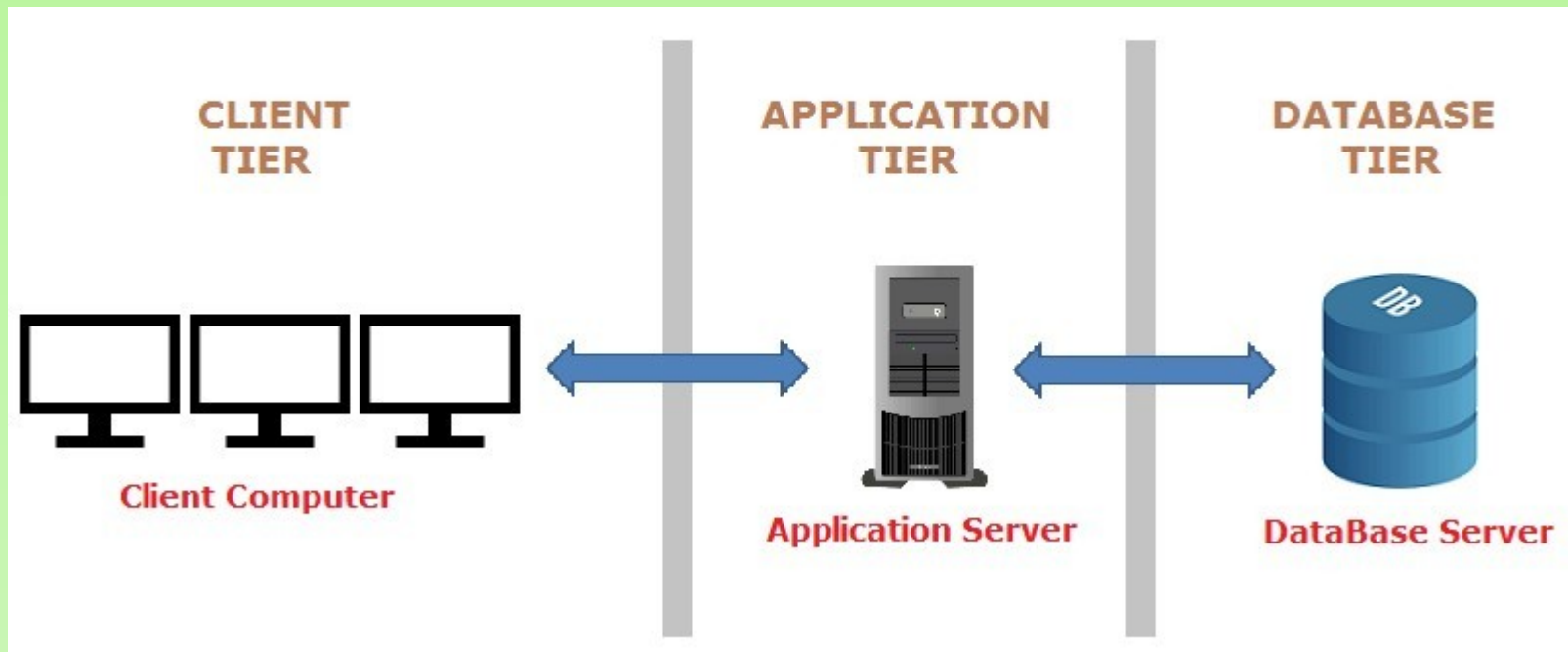
SINGLE-TIER (MAINFRAME-BASED) ARCHITECTURE



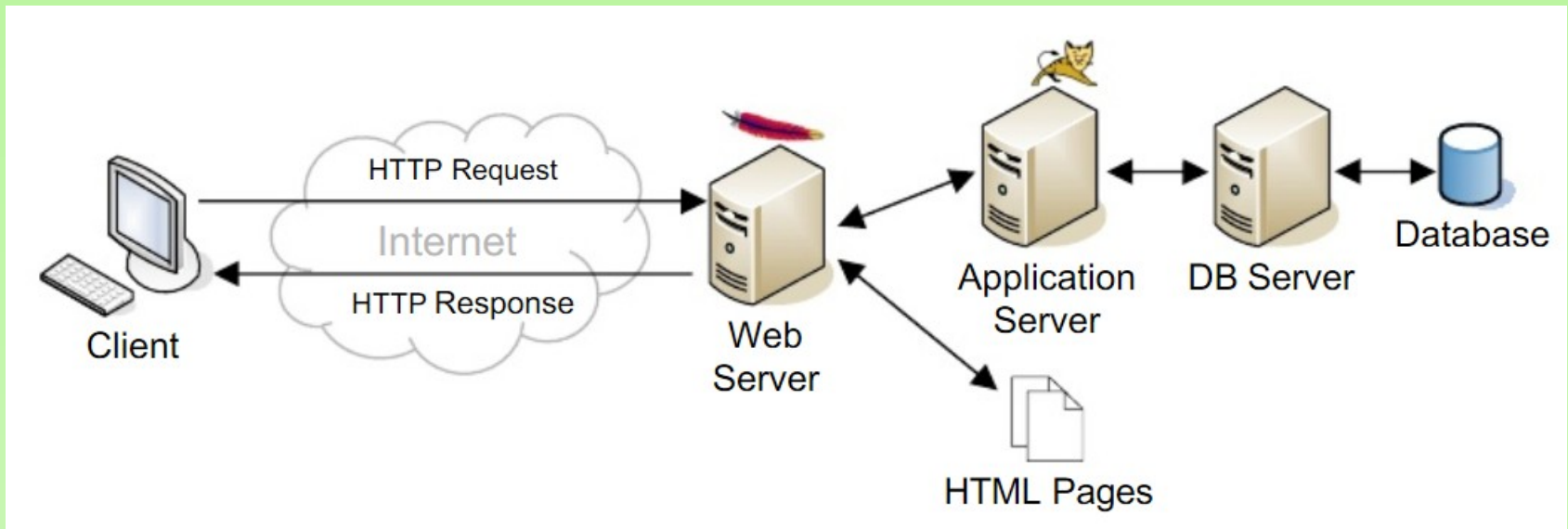
TWO-TIER ARCHITECTURE

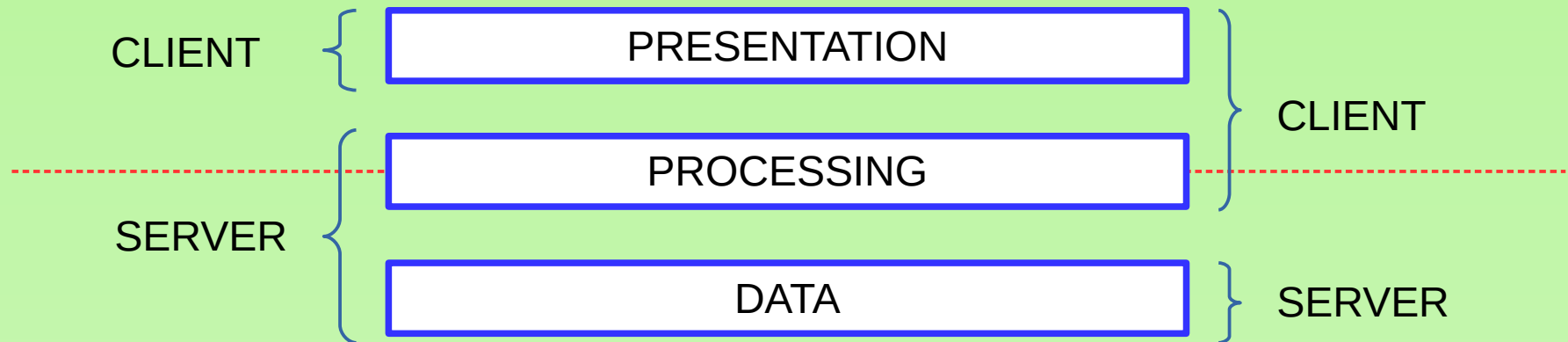


THREE-TIER ARCHITECTURE



N-TIER ARCHITECTURE





2

Spring Framework - introduction

Introduction

The Spring Framework provides a comprehensive programming and configuration model for modern Java-based enterprise applications - on any kind of deployment platform. A key element of Spring is infrastructural support at the application level: Spring focuses on the "plumbing" of enterprise applications so that teams can focus on application-level business logic, without unnecessary ties to specific deployment environments.



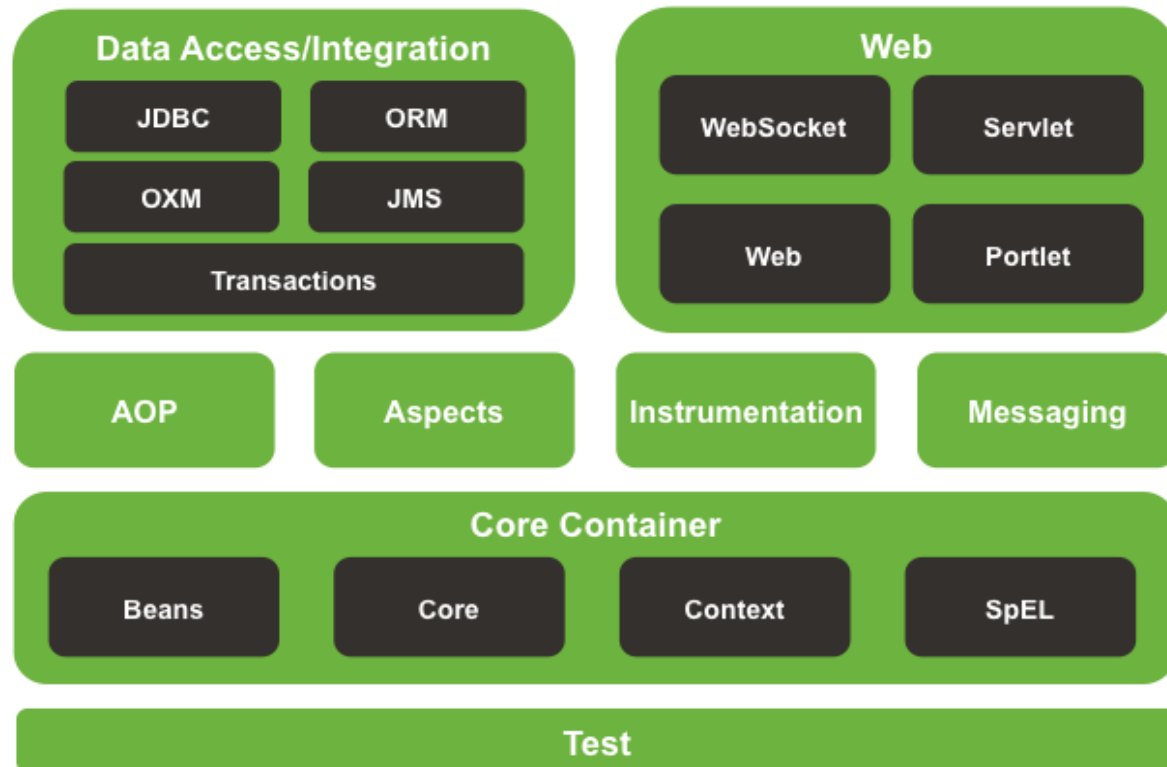
Features

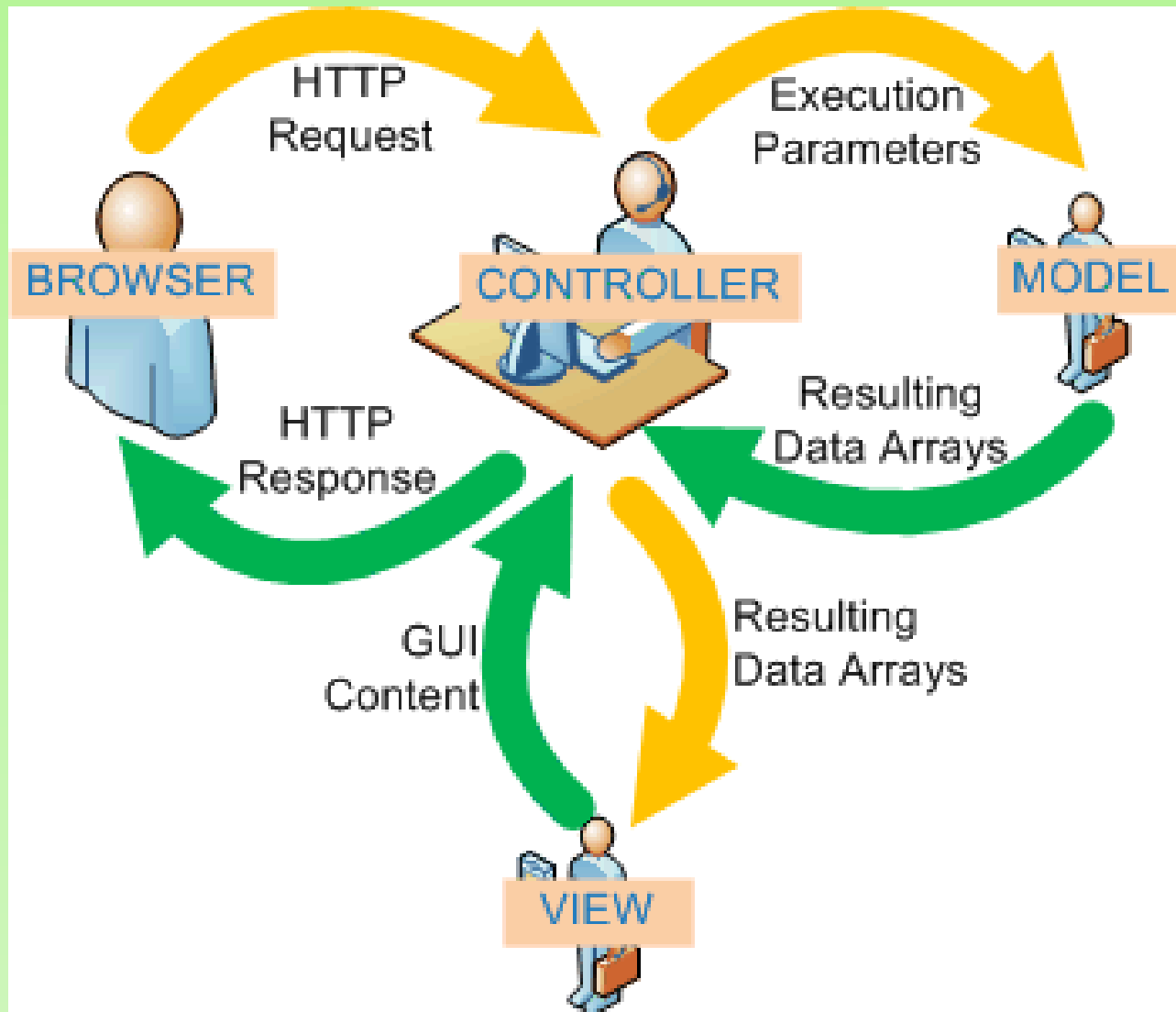
- Core technologies: dependency injection, events, resources, i18n, validation, data binding, type conversion, SpEL.
- Testing: mock objects, TestContext framework, Spring MVC Test, WebTestClient.
- Data Access: transactions, DAO support, JDBC, ORM, Marshalling XML.
- Spring MVC and Spring WebFlux web frameworks
- Integration: remoting, email, tasks, scheduling, cache.
- Languages: Kotlin, Groovy, dynamic languages.

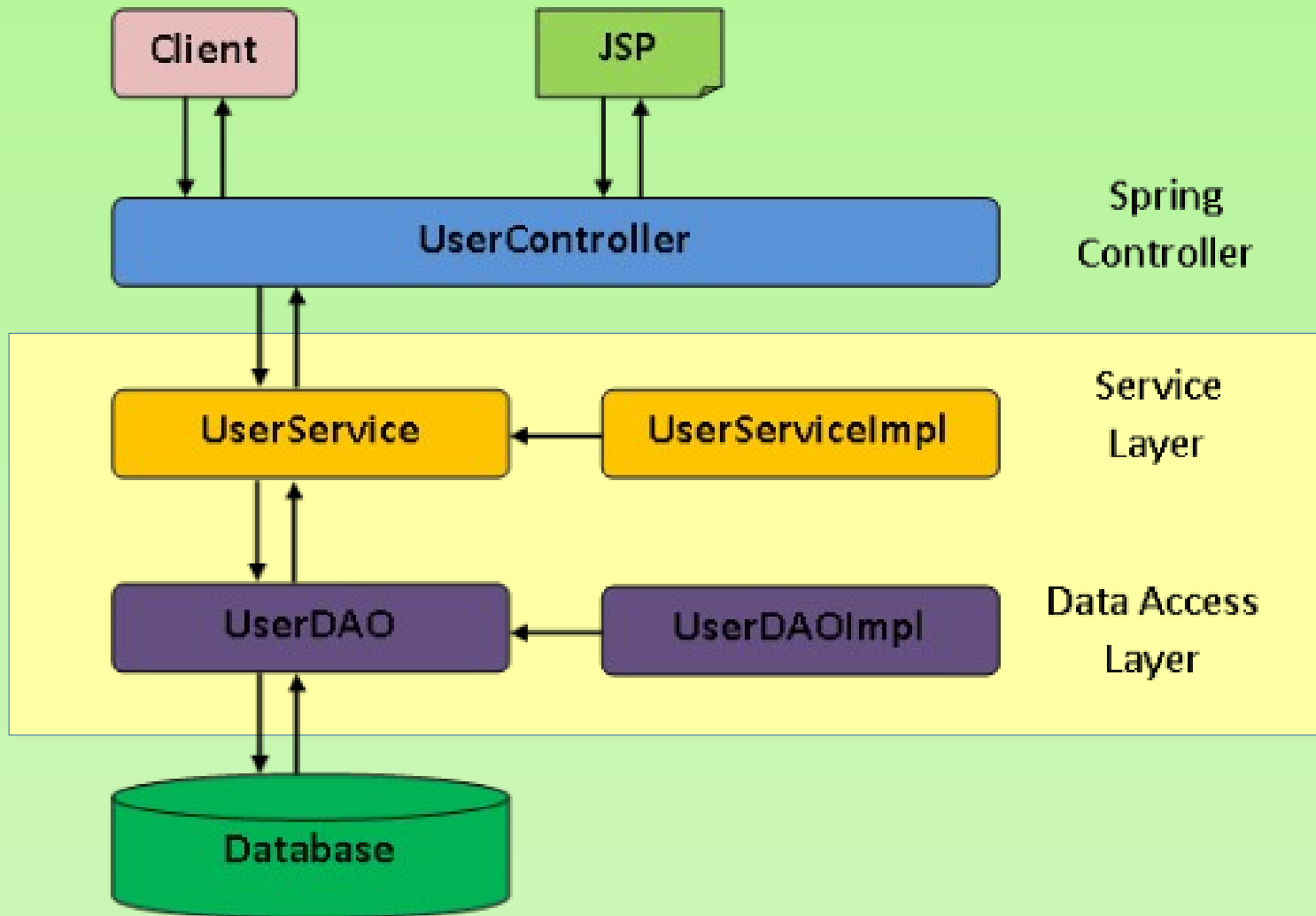




Spring Framework Runtime

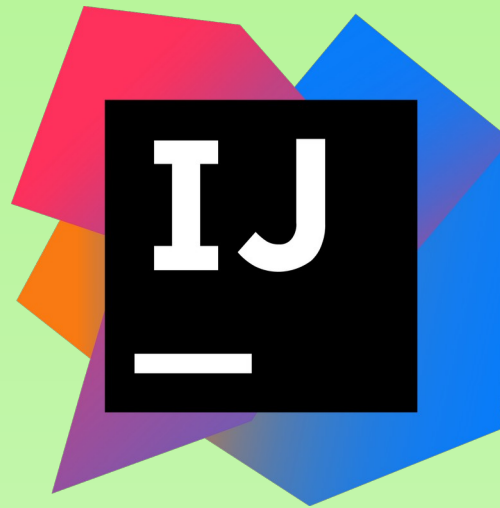






3

Tools



4

Spring Boot - .jsp example

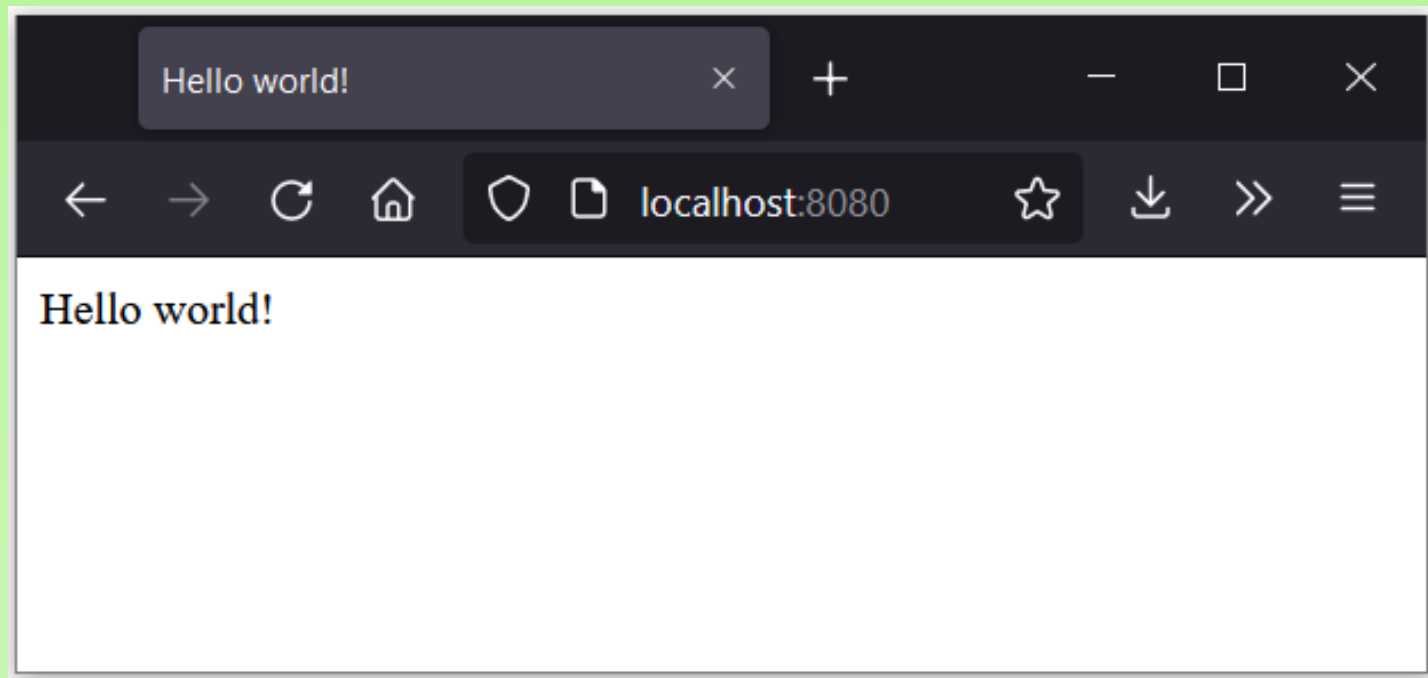
„Hello world”

Spring Boot – "Hello world!"



20

dr inż. Rafał Kotas, rkotas@dmcs.pl



Spring Boot makes it easy to create stand-alone, production-grade Spring-based applications that you can run. Most Spring Boot applications need very little Spring configuration.

You can use Spring Boot to create Java applications that can be started by using **java -jar** or more traditional **war** (**Web Application Resource** or **Web application ARchive**) deployments.

Primary goals:

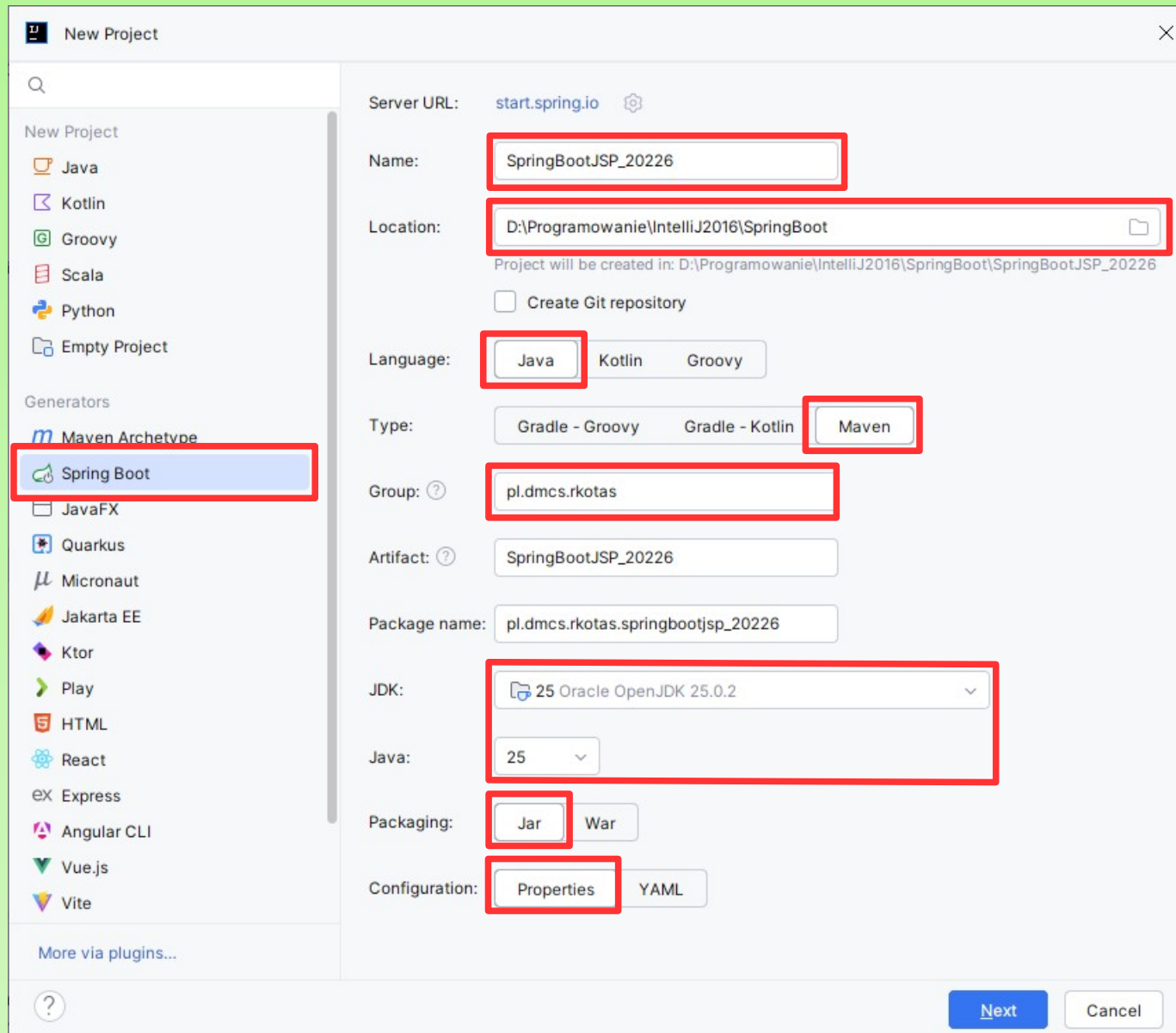
- Provide a radically faster and widely accessible getting-started experience for Spring development.
- Provide a range of non-functional features that are common to large classes of projects (such as embedded servers, security, metrics, health checks, and externalized configuration).
- Absolutely no code generation and no requirement for XML configuration.



Spring Boot – "Hello world!"

22

dr inż. Rafał Kotas, rkotas@dmcs.pl



The image shows the 'New Project' dialog in IntelliJ IDEA. The 'Spring Boot' generator is selected in the left sidebar. The main configuration area has several fields highlighted with red boxes: 'Name' is 'SpringBoot.JSP_20226', 'Location' is 'D:\Programowanie\IntelliJ2016\SpringBoot', 'Language' is 'Java', 'Type' is 'Maven', 'Group' is 'pl.dmcs.rkotas', 'Artifact' is 'SpringBoot.JSP_20226', 'Package name' is 'pl.dmcs.rkotas.springbootjsp_20226', 'JDK' is '25 Oracle OpenJDK 25.0.2', 'Java' is '25', 'Packaging' is 'Jar', and 'Configuration' is 'Properties'. The 'Server URL' is 'start.spring.io'. The 'Create Git repository' checkbox is unchecked. The 'Next' button is at the bottom right.

New Project

Search

New Project

- Java
- Kotlin
- Groovy
- Scala
- Python
- Empty Project

Generators

- Maven Archetype
- Spring Boot**
- JavaFX
- Quarkus
- Micronaut
- Jakarta EE
- Ktor
- Play
- HTML
- React
- Express
- Angular CLI
- Vue.js
- Vite

More via plugins...

Server URL: start.spring.io

Name: SpringBoot.JSP_20226

Location: D:\Programowanie\IntelliJ2016\SpringBoot

Project will be created in: D:\Programowanie\IntelliJ2016\SpringBoot\SpringBoot.JSP_20226

☐ Create Git repository

Language: Java Kotlin Groovy

Type: Gradle - Groovy Gradle - Kotlin **Maven**

Group: pl.dmcs.rkotas

Artifact: SpringBoot.JSP_20226

Package name: pl.dmcs.rkotas.springbootjsp_20226

JDK: 25 Oracle OpenJDK 25.0.2

Java: 25

Packaging: Jar War

Configuration: Properties YAML

Next Cancel

Spring Boot – "Hello world!"



23

dr inż. Rafał Kotas, rkotas@dmcs.pl

New Project

Spring Boot: 4.0.3

Dependencies:

Search

Web

☒ Spring Web

☐ Spring Reactive Web

☐ HTTP Client

☐ Reactive HTTP Client

☐ Spring for GraphQL

☐ Rest Repositories

☐ Spring Session for Spring Data MongoDB

☐ Spring Session for Spring Data Redis

☐ Spring Session for Hazelcast

☐ Spring Session for JDBC

☐ Rest Repositories HAL Explorer

☐ Spring HATEOAS

☐ Spring Web Services

☐ Jersey

☐ Vaadin

☐ Netflix DGS

☐ htmx

☐ SpringDoc OpenAPI

Spring Web

Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

[Building a RESTful Web Service ↗](#)

[Serving Web Content with Spring MVC ↗](#)

[Building REST services with Spring ↗](#)

Added dependencies:

Spring Web

?

Previous>CreateCancel

Spring Boot – "Hello world!"



24

dr inż. Rafał Kotas, rkotas@dmcs.pl

The screenshot displays the IntelliJ IDEA IDE interface for a Spring Boot project named 'SpringBootJSP_IWA2026'. The project structure on the left includes folders for '.idea', '.mvn', 'src', 'main', 'resources', 'static', 'templates', 'test', and files like '.gitattributes', '.gitignore', 'HELP.md', 'mvnw', 'mvnw.cmd', and 'pom.xml'. The 'HELP.md' file is open in the main editor, showing sections for 'Getting Started', 'Reference Documentation', 'Guides', and 'Maven Parent overrides'. The right sidebar shows a 'Getting Started' guide with links to 'Official Apache Maven documentation', 'Spring Boot Maven Plugin Reference Guide', 'Create an OCI image', and 'Spring Web'. The bottom status bar shows the project path and AWS credentials.

Project Structure:

- SpringBootJSP_IWA2026
- .idea
- .mvn
- src
 - main
 - java
 - pl.dmcs.rkotas.springbootjsp_iwa2026
 - SpringBootJspIwa2026Application
 - resources
 - static
 - templates
 - application.properties
 - test
- .gitattributes
- .gitignore
- HELP.md
- mvnw
- mvnw.cmd
- pom.xml
- External Libraries
- Scratches and Consoles

HELP.md Content:

```
1 # Getting Started
2
3 ### Reference Documentation
4
5 For further reference, please consider the following sections:
6
7 * [Official Apache Maven documentation](...)
8 * [Spring Boot Maven Plugin Reference Guide](...)
9 * [Create an OCI image](...)
10 * [Spring Web](...)
11
12 ### Guides
13
14 The following guides illustrate how to use some features concretely:
15
16 * [Building a RESTful Web Service](...)
17 * [Serving Web Content with Spring MVC](...)
18 * [Building REST services with Spring](...)
19
20 ### Maven Parent overrides
21
22 Due to Maven's design, elements are inherited from the parent POM to the
23 project POM. While most of the inheritance is fine, it also inherits unwanted
24 elements like <license> and <developers> from the parent. To prevent this,
25 the project POM contains empty overrides for these elements. If you manually
26 switch to a different parent and actually want the inheritance, you need to remove
27 those overrides.
```

Getting Started

Reference Documentation

For further reference, please consider the following sections:

- Official Apache Maven documentation
- Spring Boot Maven Plugin Reference Guide
- Create an OCI image
- Spring Web

Guides

The following guides illustrate how to use some features concretely:

- Building a RESTful Web Service
- Serving Web Content with Spring MVC
- Building REST services with Spring

Maven Parent overrides

Due to Maven's design, elements are inherited from the parent POM to the project POM. While most of the inheritance is fine, it also inherits unwanted elements like <license> and <developers> from the parent. To prevent this, the project POM contains empty overrides for these elements. If you manually switch to a different parent and actually want the inheritance, you need to remove those overrides.

Run: SpringBootJSP_IWA2026 [clean]

Status Bar: SpringBootJSP_IWA2026 > src > main > java > pl > dmcs > rkotas > springbootjsp_iwa2026 > SpringBootJspIwa2026Application

Footer: AWS: No credentials selected 1:1 LF UTF-8 4 spaces

Spring Boot – "Hello world!"



25

dr inż. Rafał Kotas, rkotas@dmcs.pl

m pom.xml (SpringBootJSP_IWA2026) x

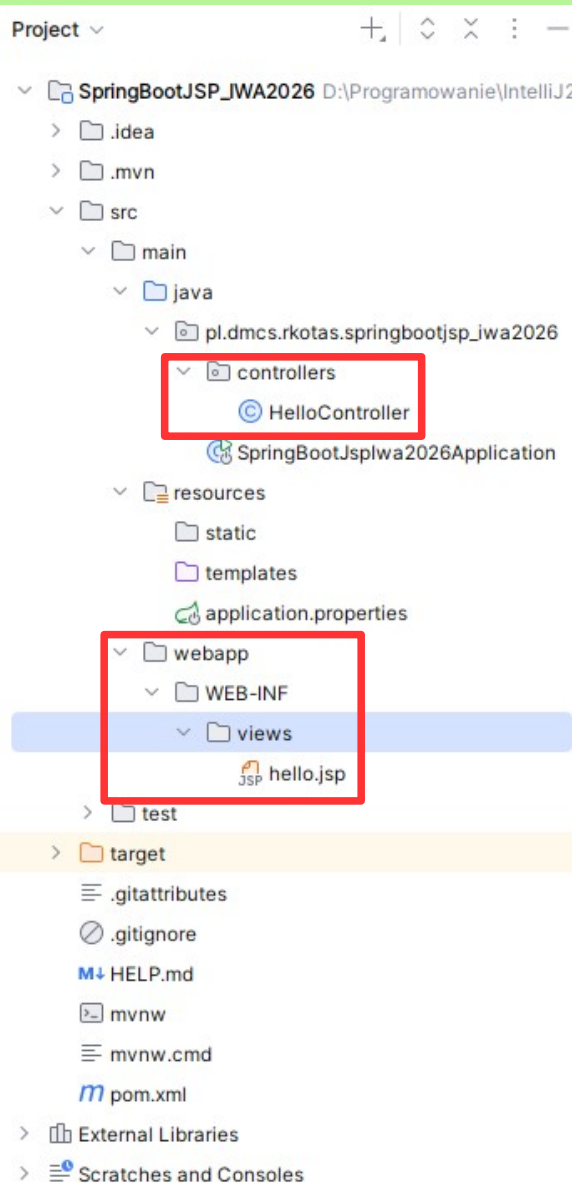
```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3     xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
4     <modelVersion>4.0.0</modelVersion> Compatible with Maven 3
5     <parent>
6         <groupId>org.springframework.boot</groupId>
7         <artifactId>spring-boot-starter-parent</artifactId>
8         <version>4.0.3</version>
9         <relativePath/> <!-- lookup parent from repository -->
10    </parent>
11    <groupId>pl.dmcs.rkotas</groupId>
12    <artifactId>SpringBootJSP_IWA2026</artifactId>
13    <version>0.0.1-SNAPSHOT</version>
14    <name>SpringBootJSP_IWA2026</name>
15    <description>SpringBootJSP_IWA2026</description>
16    <url/>
17    <licenses...>
20    <developers...>
23    <scm...>
29    <properties>
30        <java.version>25</java.version>
31    </properties>
32    <dependencies> Add Starters...
33        <dependency>
34            <groupId>org.springframework.boot</groupId>
35            <artifactId>spring-boot-starter-webmvc</artifactId>
36        </dependency>
37        <dependency>
38            <groupId>org.apache.tomcat.embed</groupId>
39            <artifactId>tomcat-embed-jasper</artifactId>
40        </dependency>
```

Spring Boot – "Hello world!"



26

dr inż. Rafał Kotas, rkotas@dmcs.pl



Where the Spring Boot app starts:

```
SpringBootJspIwa2026Application.java x
1 package pl.dmcs.rkotas.springbootjsp_iwa2026;
2
3 import ...
4
5
6 @SpringBootApplication
7 public class SpringBootJspIwa2026Application {
8
9     public static void main(String[] args) {
10         SpringApplication.run(SpringBootJspIwa2026Application.class, args);
11     }
12
13 }
```

Where to configure the Spring Boot app:

```
application.properties x
1 spring.application.name=SpringBootJSP_IWA2026
2 spring.mvc.view.prefix=/WEB-INF/views/
3 spring.mvc.view.suffix=.jsp
```

Spring Boot – "Hello world!"



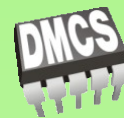
27

dr inż. Rafał Kotas, rkotas@dmcs.pl

```
© HelloController.java ×  
1 package pl.dmcs.rkotas.springbootjsp_iwa2026.controllers;  
2  
3 > import ...  
4  
5  
6 @Controller  
7 public class HelloController {  
8  
9     @RequestMapping("/")  
10    public String hello() {  
11        return "hello";  
12    }  
13  
14 }
```

```
JSP hello.jsp ×  
1 <html lang="en">  
2 <head>  
3     <meta charset="UTF-8">  
4     <title>Hello world!</title>  
5 </head>  
6 <body>  
7     Hello world!  
8 </body>  
9 </html>
```

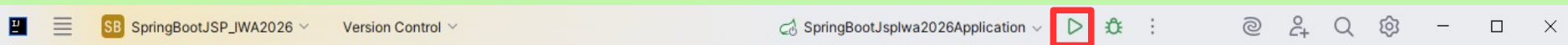
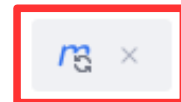
Spring Boot – "Hello world!"



28

dr inż. Rafał Kotas, rkotas@dmcs.pl

```
m pom.xml (SpringBootJSP_IWA2026) x
32 <dependencies> Add Starters...
33 <dependency>
34   <groupId>org.springframework.boot</groupId>
35   <artifactId>spring-boot-starter-webmvc</artifactId>
36 </dependency>
37 <dependency>
38   <groupId>org.apache.tomcat.embed</groupId>
39   <artifactId>tomcat-embed-jasper</artifactId>
40 </dependency>
```



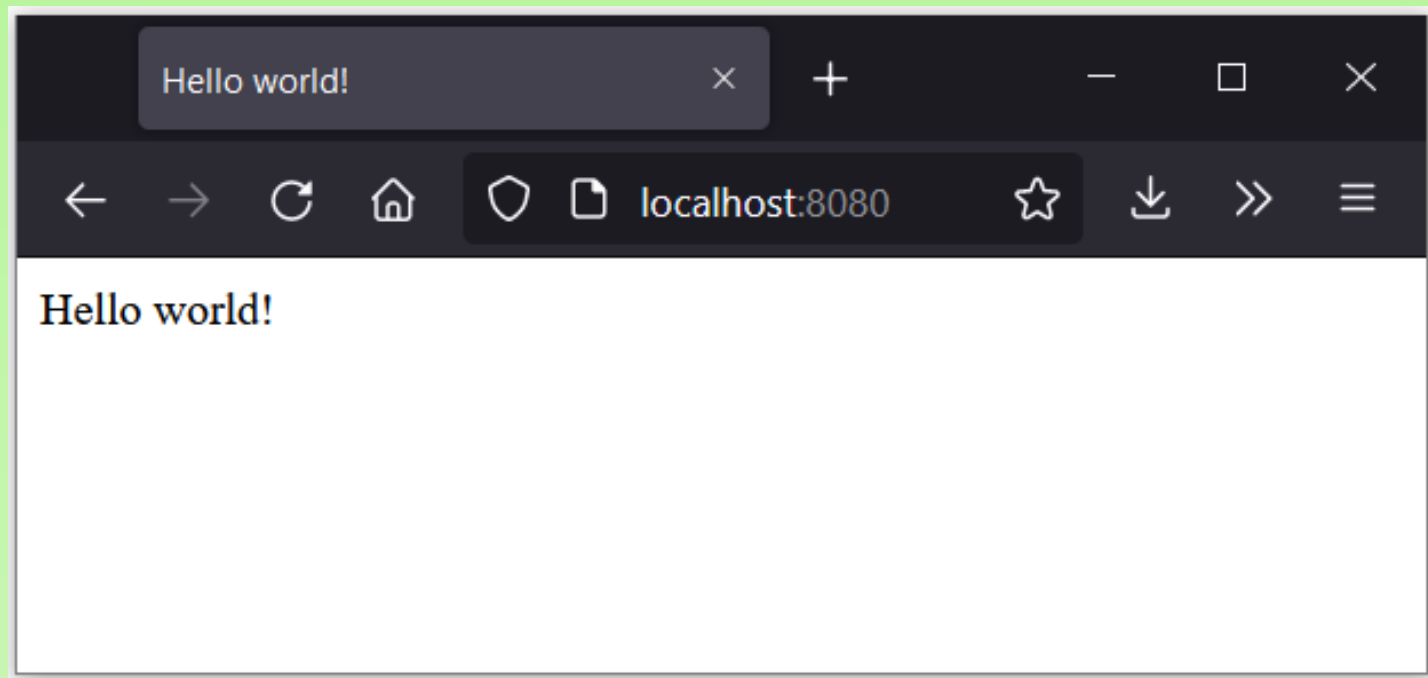
```
Run SpringBootJspIwa2026Application x
Console Beans Environment Health Mappings
2026-03-09T00:02:10.702+01:00 INFO 34636 --- [SpringBootJSP_IWA2026] [
2026-03-09T00:02:18.911+01:00 INFO 34636 --- [SpringBootJSP_IWA2026] [
2026-03-09T00:02:20.553+01:00 INFO 34636 --- [SpringBootJSP_IWA2026] [
2026-03-09T00:02:20.580+01:00 INFO 34636 --- [SpringBootJSP_IWA2026] [
2026-03-09T00:02:20.581+01:00 INFO 34636 --- [SpringBootJSP_IWA2026] [
2026-03-09T00:02:20.882+01:00 INFO 34636 --- [SpringBootJSP_IWA2026] [
2026-03-09T00:02:20.891+01:00 INFO 34636 --- [SpringBootJSP_IWA2026] [
2026-03-09T00:02:21.781+01:00 INFO 34636 --- [SpringBootJSP_IWA2026] [
2026-03-09T00:02:21.797+01:00 INFO 34636 --- [SpringBootJSP_IWA2026] [
main p.d.r.s.SpringBootJspIwa2026Application : Starting SpringBootJspIwa2026Application us
main p.d.r.s.SpringBootJspIwa2026Application : No active profile set, falling back to 1 de
main o.s.boot.tomcat.TomcatWebServer : Tomcat initialized with port 8080 (http)
main o.apache.catalina.core.StandardService : Starting service [Tomcat]
main o.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/11.
main org.apache.jasper.servlet.TldScanner : At least one JAR was scanned for TLDs yet c
main b.w.c.s.WebApplicationContextInitializer : Root WebApplicationContext: initialization
main o.s.boot.tomcat.TomcatWebServer : Tomcat started on port 8080 (http) with cor
main p.d.r.s.SpringBootJspIwa2026Application : Started SpringBootJspIwa2026Application in
```

Spring Boot – "Hello world!"



29

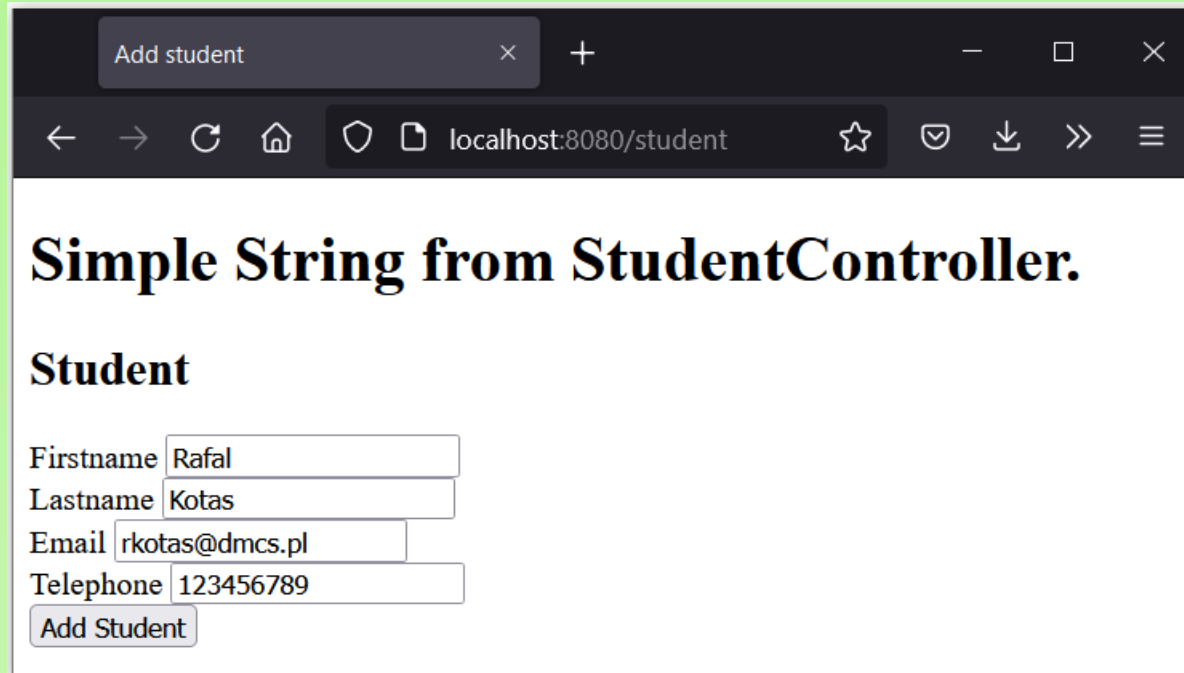
dr inż. Rafał Kotas, rkotas@dmcs.pl



5

Spring Boot - .jsp example

Model and simple form



Add student

localhost:8080/student

Simple String from StudentController.

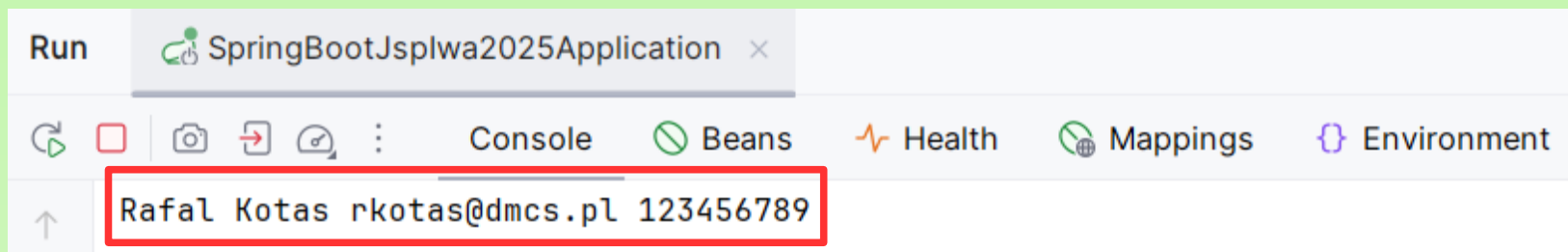
Student

Firstname

Lastname

Email

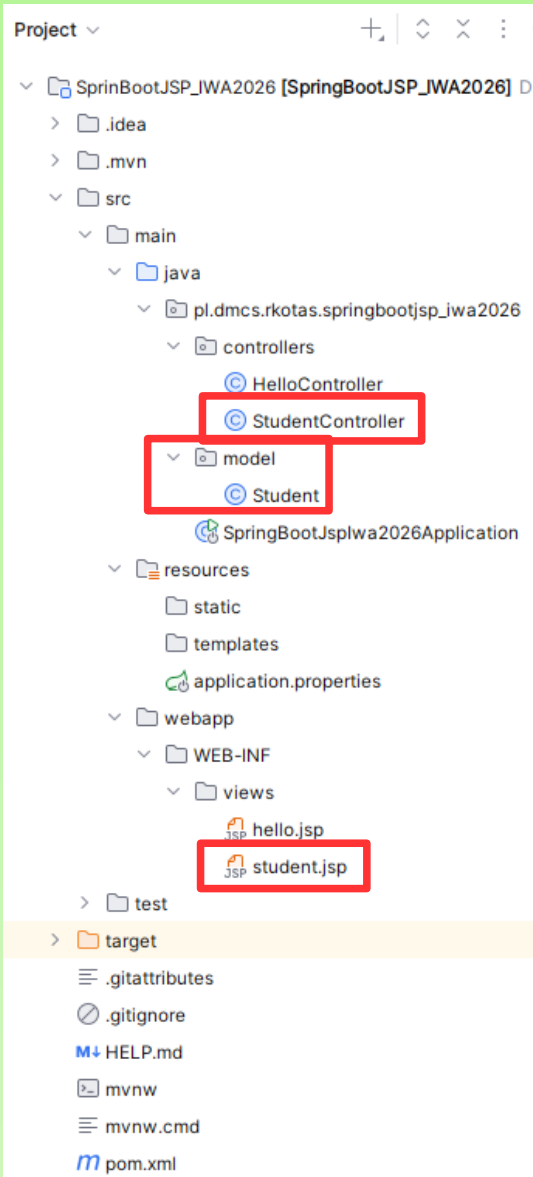
Telephone



Model and simple form

32

dr inż. Rafał Kotas, rkotas@dmcs.pl



```
Student.java x
1 package pl.dmcs.rkotas.springbootjsp_iwa2026.model;
2
3
4 public class Student {
5
6     private String firstname;
7     private String lastname;
8     private String email;
9     private String telephone;
10
11     public String getFirstname() { return firstname; }
12
13     public void setFirstname(String firstname) { this.firstname = firstname; }
14
15     public String getLastname() { return lastname; }
16
17     public void setLastname(String lastname) { this.lastname = lastname; }
18
19     public String getEmail() { return email; }
20
21     public void setEmail(String email) { this.email = email; }
22
23     public String getTelephone() { return telephone; }
24
25     public void setTelephone(String telephone) { this.telephone = telephone; }
26
27 }
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
```


Model and simple form

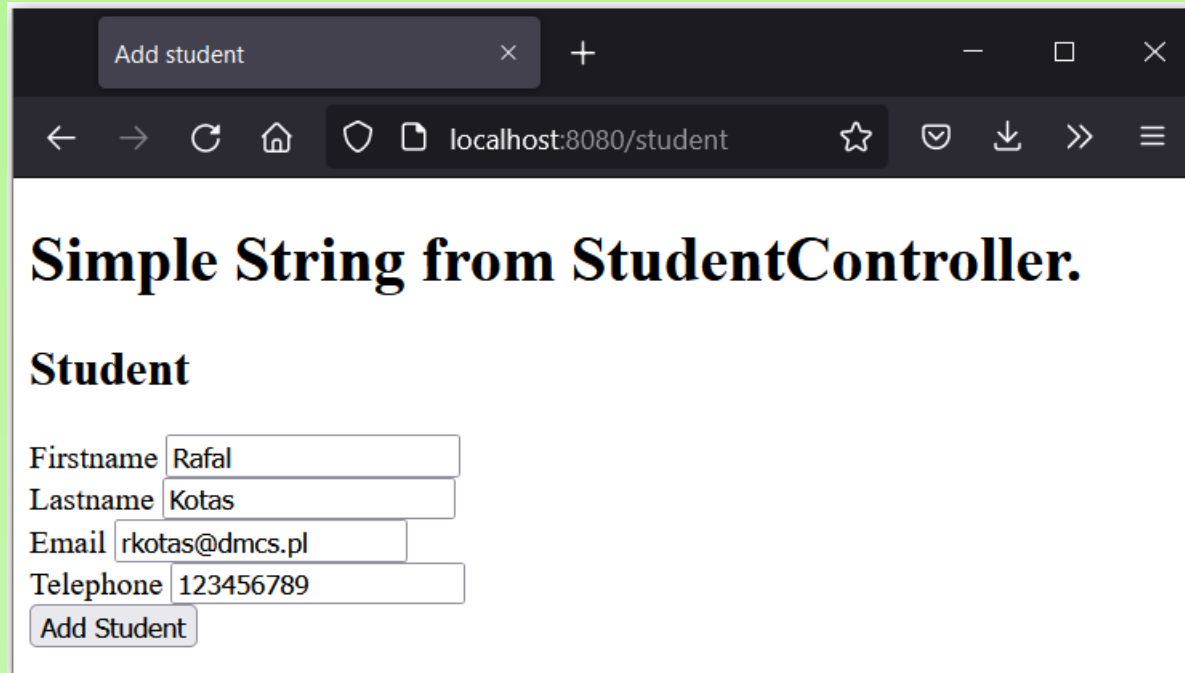
```
StudentController.java x
1 package pl.dmcs.rkotas.springbootjsp_iwa2026.controllers;
2
3 > import ...
9
10 @Controller
11 public class StudentController {
12
13     @RequestMapping("/student")
14     public String student(Model model) {
15         model.addAttribute("message", "Simple String from StudentController.");
16         Student newStudent = new Student();
17         model.addAttribute("student", newStudent);
18         return "student";
19     }
20
21     @RequestMapping(value = "/addStudent.html", method = RequestMethod.POST)
22     public String addStudent(@ModelAttribute("student") Student student) {
23
24         System.out.println(student.getFirstname() + " " + student.getLastname() +
25             " " + student.getEmail() + " " + student.getTelephone());
26
27         return "redirect:student";
28     }
29
30 }
```

Model and simple form

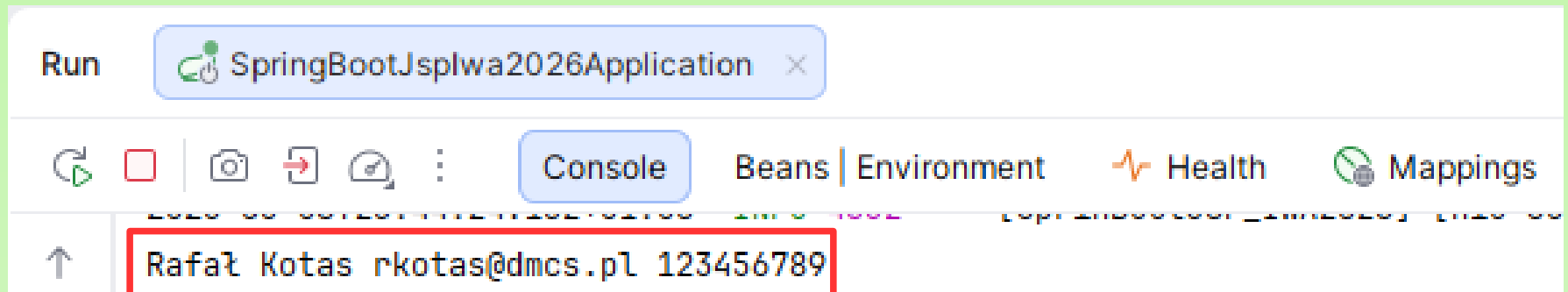
34

dr inż. Rafał Kotas, rkotas@dmcs.pl

```
student.jsp x
1 <%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
3 <%@taglib uri="http://www.springframework.org/tags/form" prefix="form"%>
4 <html>
5 <head>
6 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
7 <title>Add student</title>
8 </head>
9 <body>
10
11 <h1>${message}</h1>
12
13 <h2>Student</h2>
14 <form:form method="post" action="addStudent.html" modelAttribute="student">
15 <form:label path="firstname">Firstname</form:label>
16 <form:input path="firstname" />
17 <br>
18 <form:label path="lastname">Lastname</form:label>
19 <form:input path="lastname" />
20 <br>
21 <form:label path="email">Email</form:label>
22 <form:input path="email" />
23 <br>
24 <form:label path="telephone">Telephone</form:label>
25 <form:input path="telephone" />
26 <br>
27 <input type="submit" value="Add Student"/>
28 </form:form>
29 </body>
30 </html>
```



A screenshot of a web browser window with the title "Add student". The address bar shows "localhost:8080/student". The main content area displays the text "Simple String from StudentController." followed by the heading "Student". Below the heading is a form with four input fields: "Firstname" containing "Rafal", "Lastname" containing "Kotas", "Email" containing "rkotas@dmcs.pl", and "Telephone" containing "123456789". At the bottom of the form is a button labeled "Add Student".



1. Create new project and implement HelloWorld in Spring Boot
2. Implement Student form
3. Implement Fibonacci sequence and quadratic equation forms



THE END

WIKAMP → IWA_2.zip

