Haaga-Helia

Server Programming

Spring Boot introduction

Juha Hinkula and Jukka Juslin

Updated Minna Pellikka 09.01.2025

Haaga-Helia

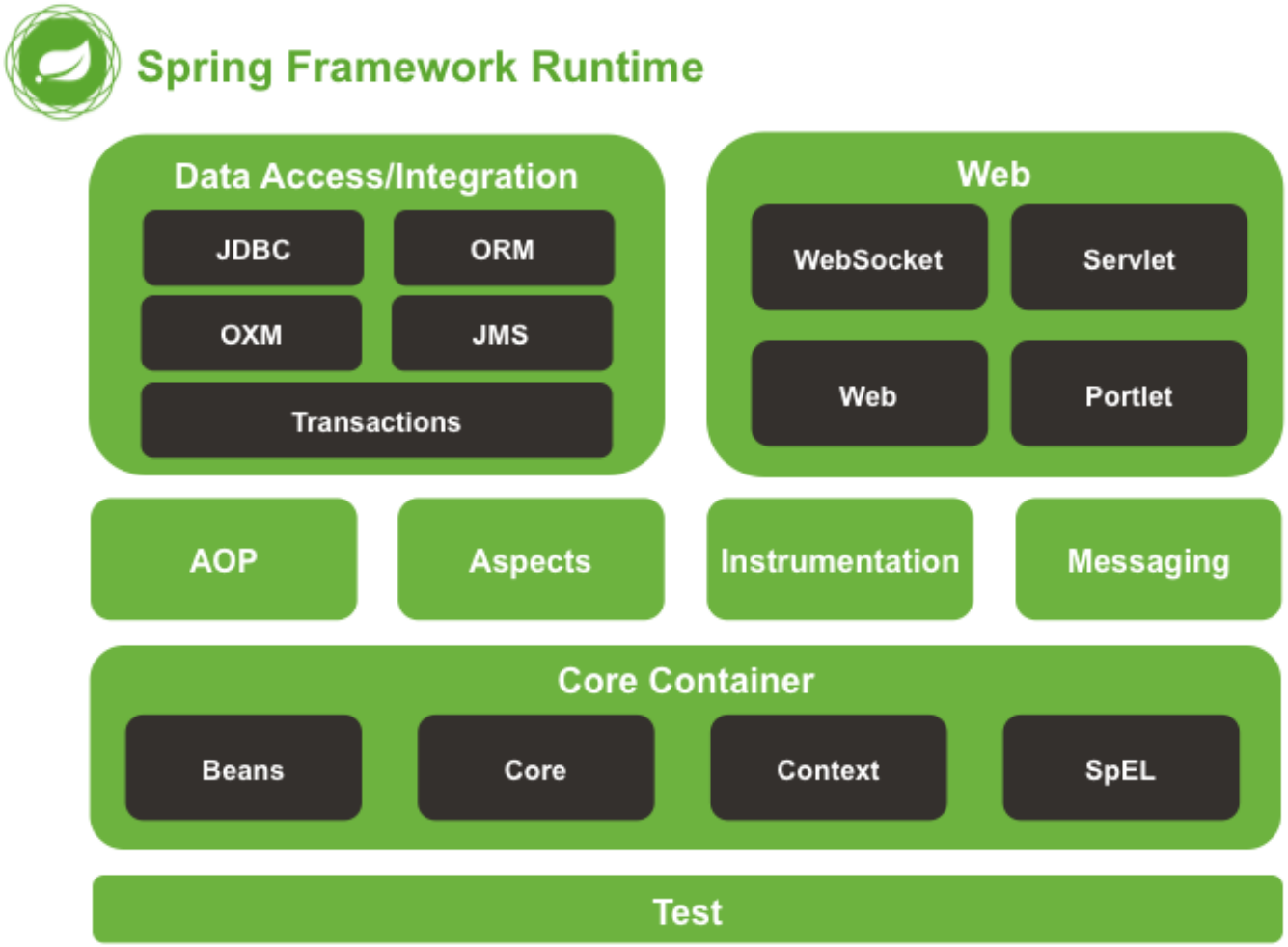
Spring

- Spring framework: https://spring.io/
- Spring framework is an open source application framework for Java platform
- The most popular framework for Java EE
- First version was released under Apache 2.0 license in 2003
- Spring is modular and it contains lot of extensions for different purposes

Haaga-Helia

Spring

- Material
  - Getting Started Guides: https://spring.io/guides
  - Tutorials: https://spring.io/guides#tutorials
  - Tutorialspoint: http://www.tutorialspoint.com/spring/


Spring Framework Runtime
Data Access/Integration
JDBC
ORM
OXM
JMS
Transactions
Web
WebSocket
Servlet
Web
Portlet
AOP
Aspects
Instrumentation
Messaging
Core Container
Beans
Core
Context
SpEL
Test

Haaga-Helia

Dependency Injection (DI)

- Web appliction typically contains object that have dependencies to each others
- Dependency injection helps interaction between classes but same time keeping them independent

Dependency Injection (DI) example

- Without DI

```
public class Vehicle {
  private Owner owner;

  public Vehicle() {
    owner = new Owner();
  }
}
```

- With DI

```
public class Vehicle {
  private Owner owner;

  public Vehicle(Owner owner) {
    this.owner = owner;
  }
}
```


Haaga-Helia

Spring Boot

- Spring Boot helps to easily setup new Spring project
- Create stand-alone Spring applications
  - Embedded application server
- Starter POM's for Maven configuration
- Minimized the need of configuration

Haaga-Helia

Maven

- Apache Maven is a software project management and comprehension tool
- https://maven.apache.org/
- Maven objectives
  - Easy build process
  - Uniform build system
  - Transparent migration of new features
  - Guidelines for best practice development

Haaga-Helia

Maven

- Maven in 5 minutes: https://maven.apache.org/guides/getting-started/maven-in-five-minutes.html
- The pom.xml file is the core of the project configuration in Maven
- Single configuration file to build a project

Haaga-Helia

POM.XML example

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>fi.haagahelia.course</groupId>
    <artifactId>helloFormValidation</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <packaging>jar</packaging>

    <name>helloFormValidation</name>
    <description>Demo project for Spring Boot</description>

    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>3.4.1</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>
    <url />
    <licenses>
        <license />
    </licenses>
    <developers>
        <developer />
    </developers>
    <scm>
        <connection />
        <developerConnection />
        <tag />
        <url />
    </scm>

    <properties>
        <java.version>17</java.version>
    </properties>

    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter</artifactId>
        </dependency>

        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency> …..
```

Haaga-Helia

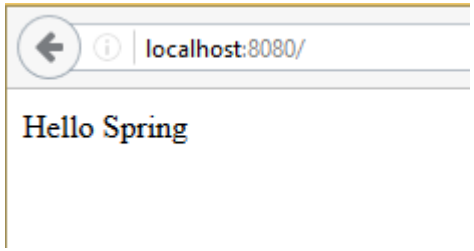
Spring Boot

- Recommended project structure
  - **NOTE**! locate the main application class in a root package above other classes. Otherwise your application won't work properly

Haaga-Helia

Spring Boot

- First Spring Boot project
  - Create a new Spring Boot project according to instructions in course site
  - Add new controller class HelloController.java (code in next slide)
  - Run your project and browse to localhost:8080



Haaga-Helia

Spring Boot

```
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.ResponseBody;

@Controller
@ResponseBody
public class HelloController {

    @RequestMapping("*")
    public String hello() {
        return "Hello Spring";
    }

}
```

Haaga-Helia

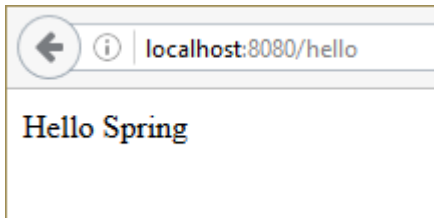
Spring Boot & Spring web MVC

- Spring uses annotations (starts with ’@’-sign) for configuration
- `@Controller` annotation
  - Marks class as a Spring web MVC controller
- `@ReponseBody` annotation
  - Converts the return value and write it to http reponse
- `@RequestMapping` annotation
  - Mapping URLs to controller methods

Haaga-Helia

Spring Boot

- `@RequestMapping(”*”)` means for any path in the application will excecute hello() method.
- Change the code `@RequestMapping(”/hello”)` means that path `address:port/hello` excecutes hello() method.



Haaga-Helia

Request parameters

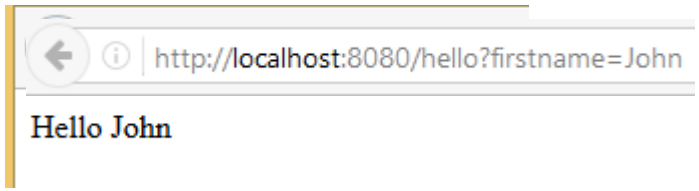
- HTTP GET request can contain parameters
- Example
  - `http://localhost:8080/hello?firstname=John&lastname=Doe`
- In HTTP POST request parameters are sent in the request body

Haaga-Helia

Spring Boot

- @RequestParam annotation binds the value of request parameter into name variable

```
@Controller
@ResponseBody
public class HelloController {
    @RequestMapping("/hello")
    public String hello(@RequestParam(name="firstname") String firstName) {
        return "Hello " + firstName;
    }
}
```



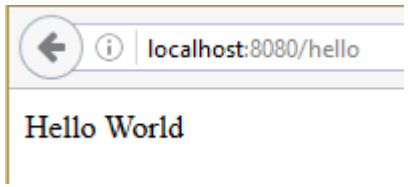
Haaga-Helia

Spring Boot

- In the pevious example the parameter is mandatory. If it is not required you can use following parameters in `@RequestParam` annotation

```
public String hello(@RequestParam(name="firstname", required=false,
defaultValue="World") String firstName)
```

- If the request does not contains parameter the default value is used



Haaga-Helia

Spring Boot

- Spring Boot packs everything to one executable JAR package
- Excecution starts from the main application class (main() method)

```
package hello;

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

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

Haaga-Helia

Spring Boot

- `@SpringBootApplication` annotation adds following annotations
  - `@Configuration` tags the class as a source of bean definitions for the application context.
  - `@EnableAutoConfiguration` tells Spring Boot to start adding beans based on classpath .
  - `@ComponentScan` tells Spring to look for other components, configurations, and services in the *package*
  - `@EnableWebMvc` annotation flags the application as a web application and activates key behaviors such as setting up a DispatcherServlet