UNIVERSITATEA DIN BUCUREȘTI

VIRTUTE ET SAPIENTIA

WEB TECHNOLOGIES USING JAVA

COURSE 4 – SPRING BOOT, SPRING MVC

AGENDA

- SPRING BOOT CORE FEATURES
- GETTING STARTED
- SPRING MVC INTRO
- WEB AWARE BEAN SCOPES

SPRING BOOT CORE FEATURES

1. Simplified dependency management
   - avoids mismatch between different versions of dependencies
   - Spring Boot starters: provide a group of related functionalities in a single application dependency (spring-boot-starter-web)
   - dramatically diminishes the overhead of testing, maintaining, and upgrading them

. 2. Executable JARs for Simplified Deployment
   - single Spring Boot JAR with all dependencies makes the deployment easy.

. 3. Autoconfiguration
   - developer's superpower
   - convention over configuration: if you follow simple, well-established and documented conventions to do something, the configuration code you must write is minimal

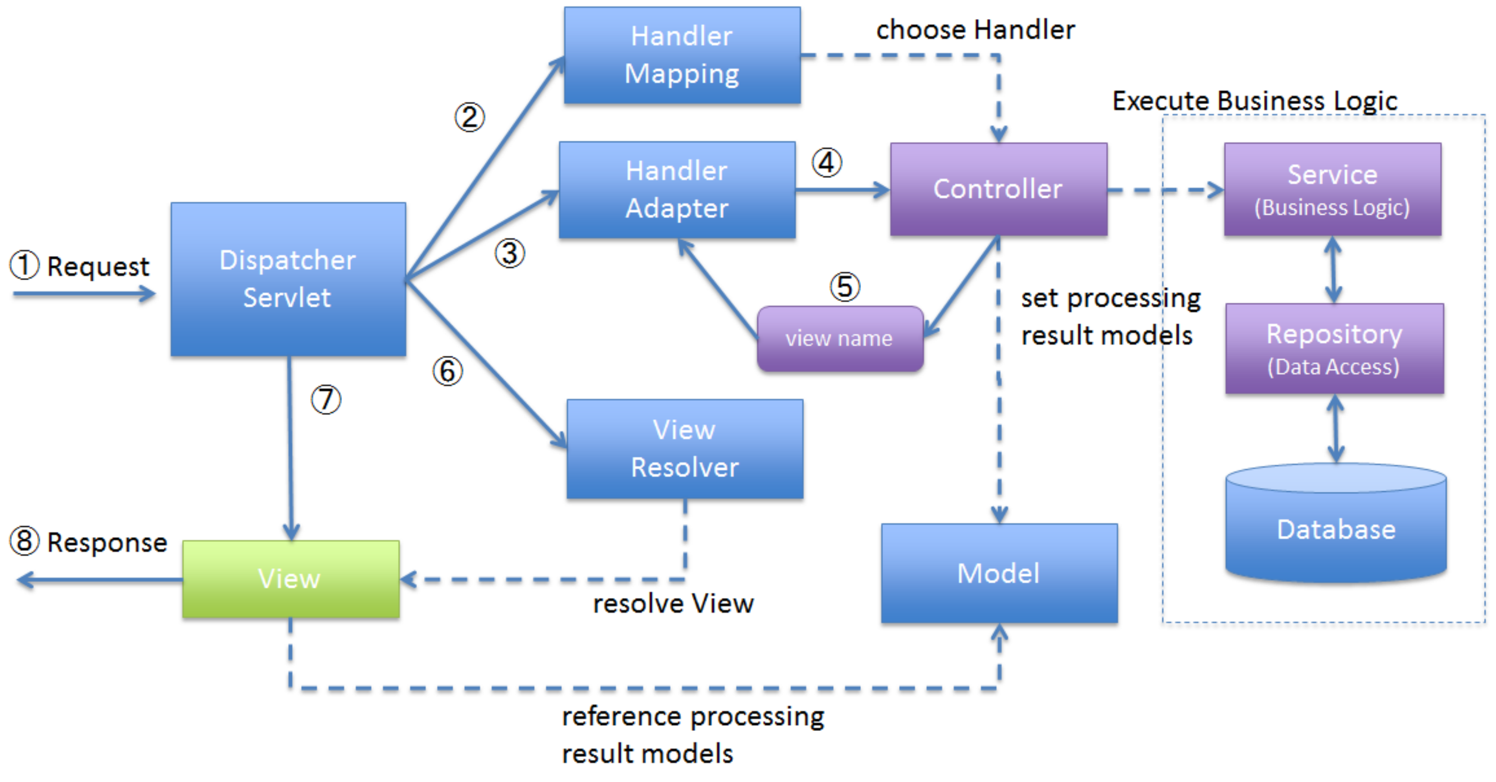
GETTING STARTED

- Maven or Gradle project
- The Spring Initializr: https://start.spring.io
- @SpringBootApplication: upon startup, a Spring Boot app checks the environment, configures the application, creates the initial context, and launches the Spring Boot application
  - @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 settings, other beans, and various property settings.
  - @ComponentScan: tells Spring to look for other components in the package from the same level as the application class.

SPRING MVC INTRO

- MVC: Model-View-Controller
- a pattern and an implementation
- Spring MVC framework is request-driven, designed around a central Servlet that dispatches requests to controllers
- User interaction can be done in multiple ways (server-side rendering of the HTML)
  - JSP (Java Server Pages)
  - template engines: Thymeleaf

SPRING MVC INTRO



🟪	... implemented by developers
🟦	... provided by Spring Source
🟩	... provided by Spring Source sometimes implemented by developers

SPRING MVC INTRO

- Spring MVC project with Spring Boot and Thymeleaf

  - **spring-boot-starter-web** dependency, which auto-configures:
    - Dispatcher Servlet
    - Error Page
    - Web Jars to manage your static dependencies
    - Embedded Servlet Container - Tomcat is the default

  - **spring-boot-starter-thymeleaf**
    - integration between Spring MVC and Thymeleaf template engine

SPRING MVC INTRO

- General structure of a Controller method
  - annotated with @Controller
  - annotated with @RequestMapping
    - ensures that the HTTP requests made at the specified url are mapped to the method
  - @Model:
    - automatically autowired parameter in the controller methods
    - used to expose an object to the view template
  - @ModelAttribute:
    - automatically autowired parameter in the controller methods
    - used to receive an object populated in the view template
  - returns a String, which is the name of the template (templates are placed under src/main/resources/templates)

WEB AWARE BEAN SCOPES

- **Request**:
  - bean is scoped to the lifecycle of an HTTP request level
  - a new instance per each HTTP request
  - when the request completes processing, the bean that is scoped to the request is discarded
  - real world use cases: keeping the result of a search, keeping the confirmation details of an order

- **Session**
  - bean is scoped to the lifecycle of an HTTP Session
  - a new instance for the lifetime of a single HTTP Session
  - when the HTTP Session is eventually discarded, the bean that is scoped to that HTTP Session is also discarded
  - real world use cases: keeping authentication information, user preferences

- **Application**
  - bean is scoped to the lifecycle of a ServletContext
  - a new instance for the entire web application
  - real world use cases: application preferences

WEB AWARE BEAN SCOPES

- **Websocket**
  - bean is scoped to the lifecycle of a WebSocket
  - the WebSocket protocol provides a standardized way to establish a full-duplex, two-way communication channel between client and server over a single TCP connection
  - it is the combination of low latency, high frequency, and high volume that make the best case for the use of WebSocket
  - real world use cases: real-time feeds, real-time collaborative editing, real-time events updates

BIBLIOGRAPHY

- Spring in Action, by Craig Walls
- https://docs.spring.io/spring-framework/reference/web/webmvc.html
- https://www.thymeleaf.org/documentation.html
- https://docs.spring.io/spring-framework/reference/core/beans/factory-scopes.html

Q&A

THANK YOU

DANIELA SPILCĂ