Adavanced Java

Agenda

- Spring Web MVC

Spring Web MVC

- MVC is a design-pattern / architeture-style.
- Divide application code into multiple relevant components to make application maintainable and extendable.
    - M: Model: Data of the application.
    - V: View: Appearance of data.
    - C: Controller: Interaction between business logic & views.
- Spring MVC components
    - Model (DTO): POJO classes holding data between view & controller.
    - View: JSP** or Thymeleaf or Freemarker pages
    - Controller: Spring Front Controller i.e. DispatcherServlet
    - User defined controller: Interact with front controller to collect/send data to appropriate view, process with service layer.

Spring Web MVC Flow

- DispatcherServlet receives the request.
- DispatcherServlet dispatches the task of selecting an appropriate controller to HandlerMapping. HandlerMapping selects the controller which is mapped to the incoming request URL and returns the (selected Handler) and Controller to DispatcherServlet.
- DispatcherServlet dispatches the task of executing of business logic of Controller to HandlerAdapter.
- HandlerAdapter calls the request handler method of @Controller.
- Controller executes the business logic, sets the processing result in Model and returns the logical name of view to HandlerAdapter.
- DispatcherServlet dispatches the task of resolving the View corresponding to the View name to ViewResolver. ViewResolver returns the View Path mapped to View name.
- DispatcherServlet dispatches the rendering process to View class / View Engine.
- It renders view with Model data and returns the response.

Request handler Methods

- @RequestMapping attributes
  - value/path = "url-pattern"
  - method = GET | POST | PUT | DELETE
  - params/header = ... (map request only if given param or header is present).
  - consumes = ... (map request only if given request body type is available).
  - produces = ... (produce given response type from handler method)
- One request handler method can be mapped to multiple HTTP request methods (get, post, ...).
- One request handler method can be restrict to set of HTTP request methods.
- To restrict handler method to single request method, shorthand annotations available.
  - @GetMapping --> @RequestMapping(method="GET")
  - @PostMapping --> @RequestMapping(method="POST")
  - @PutMapping --> @RequestMapping(method="PUT")
  - @DeleteMapping --> @RequestMapping(method="DELETE")

Flexible signatures

- https://docs.spring.io/spring-framework/docs/3.2.x/spring-framework-reference/html/mvc.html#mvc-ann-arguments
- An @RequestMapping handler method can have a very flexible signatures.
- Supported method argument types
  - HttpServletRequest, HttpServletResponse
  - @RequestParam, @RequestHeader, @PathVariable
  - Map or Model or ModelMap
  - Command (model) object, @ModelAttribute, Errors or BindingResult
  - InputStream, OutputStream
  - HttpSession, @CookieValue("cookie-name") String value, Locale
  - HttpEntity<>, @RequestBody -- for REST services.
- Supported method return types
  - String (viewName), View, Model or Map, ModelAndView
  - Command (Model) object
  - HttpHeaders, void - OutputStream

- HttpEntity<>, ResponseEntity<>, @ResponseBody -- for REST services.

Spring Web MVC -- Video Tutorial

- https://youtu.be/NzhEXRDojzA

Spring Web MVC (sp03mvc)

- Create New "Maven Project" -> Select WebApp Archetype 1.4 -> Give groupId and artifactId.
- In pom.xml add following settings and Update Maven project.

```
<properties>
    <maven.compiler.source>11</maven.compiler.source>
    <maven.compiler.target>11</maven.compiler.target>
</properties>
<dependencies>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-context</artifactId>
        <version>5.3.30</version>
    </dependency>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-webmvc</artifactId>
        <version>5.3.30</version>
    </dependency>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-jdbc</artifactId>
        <version>5.3.30</version>
    </dependency>
    <dependency>
        <groupId>javax.servlet</groupId>
        <artifactId>javax.servlet-api</artifactId>
        <version>4.0.1</version>
```

```
        <scope>provided</scope>
    </dependency>
    <dependency>
        <groupId>javax.servlet.jsp</groupId>
        <artifactId>javax.servlet.jsp-api</artifactId>
        <version>2.3.3</version>
        <scope>provided</scope>
    </dependency>
    <dependency>
        <groupId>javax.servlet</groupId>
        <artifactId>jstl</artifactId>
        <version>1.2</version>
    </dependency>
    <dependency>
        <groupId>com.mysql</groupId>
        <artifactId>mysql-connector-j</artifactId>
        <version>8.1.0</version>
    </dependency>
</dependencies>
```

- Create WEB-INF/web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<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">
<display-name>sp03mvc</display-name>
    <servlet>
        <servlet-name>spring</servlet-name>
        <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
        <load-on-startup>1</load-on-startup>
    </servlet>
```

```
    <servlet-mapping>
        <servlet-name>spring</servlet-name>
        <url-pattern>/</url-pattern>
    </servlet-mapping>
</web-app>
```

- Create WEB-INF/spring-servlet.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:context="http://www.springframework.org/schema/context"
    xmlns:mvc="http://www.springframework.org/schema/mvc"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
https://www.springframework.org/schema/beans/spring-beans-4.3.xsd
        http://www.springframework.org/schema/context https://www.springframework.org/schema/context/spring-context-
4.3.xsd
        http://www.springframework.org/schema/mvc http://www.springframework.org/schema/mvc/spring-mvc-4.3.xsd">

    <context:component-scan base-package="com.sunbeam"/>

    <bean id="viewResolver" class="org.springframework.web.servlet.view.UrlBasedViewResolver">
        <property name="viewClass" value="org.springframework.web.servlet.view.JstlView"/>
        <property name="prefix" value="/"/>
        <property name="suffix" value=".jsp"/>
    </bean>

    <mvc:annotation-driven/>

    <import resource="classpath:beans.xml"/>
</beans>
```

- Create src/main/resources/beans.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:context="http://www.springframework.org/schema/context"
    xmlns:tx="http://www.springframework.org/schema/tx"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
https://www.springframework.org/schema/beans/spring-beans-4.3.xsd
        http://www.springframework.org/schema/context https://www.springframework.org/schema/context/spring-context-
4.3.xsd
        http://www.springframework.org/schema/tx https://www.springframework.org/schema/tx/spring-tx-4.3.xsd">

    <bean id="dataSrc" class="org.springframework.jdbc.datasource.DriverManagerDataSource">
        <property name="driverClassName" value="com.mysql.cj.jdbc.Driver"/>
        <property name="url" value="jdbc:mysql://localhost:3306/classwork"/>
        <property name="username" value="dmc"/>
        <property name="password" value="dmc"/>
    </bean>

    <bean id="jdbcTemplate" class="org.springframework.jdbc.core.JdbcTemplate">
        <property name="dataSource" ref="dataSrc"></property>
    </bean>
</beans>
```

- Create Customer POJO class.
- Create CustomerRowMapper @Component class.

```
@Component
public class CustomerRowMapper implements RowMapper<Customer> {
    @Override
    public Customer mapRow(ResultSet rs, int rowNum) throws SQLException {
        int id = rs.getInt("id");
        String name = rs.getString("name");
        String password = rs.getString("password");
```

```
        String email = rs.getString("email");
        // ...
        Customer u = new Customer(id, name, password, email, ...);
        return u;
    }
}
```

- Create CustomerDaoImpl @Repository class inherited from CustomerDao interface.

```
@Repository
public class CustomerDaoImpl implements CustomerDao {
    @Autowired
    private JdbcTemplate jdbcTemplate;
    @Autowired
    private CustomerDaoRowMapper rowMapper;

    public Customer findByEmail(String email) throws Exception {
        String sql = "SELECT * FROM users WHERE email=?";
        List<Customer> list = jdbcTemplate.query(sql, rowMapper, email);
        return list.isEmpty() ? null : list.get(0);
    }
}
```

- Create Credentials POJO class (Model).
- Create CustomerServiceImpl class inherited from CustomerService interface.

```
@Service
public class CustomerServiceImpl implements CustomerService {
    @Autowired
    private CustomerDao userDao;

    public Customer authenticate(Credentials cr) {
```

```
        Customer user = userDao.findByEmail(cr.getEmail());
        if(user != null && user.getPassword().equals(cr.getPasswd()))
            return user;
        return null;
    }
}
```

- Create webapp/index.jsp.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Login</title>
</head>
<body>
    <form action="login" method="post">
        Email: <input type="text" name="email"/> <br/><br/>
        Password: <input type="password" name="password"/> <br/><br/>
        <input type="submit" value="Sign In"/>
    </form>
</body>
</html>
```

- Create LoginController @Controller class.

```
@Controller
public class LoginController {
    @Autowired
    private CustomerService userService;
```

```
    @RequestMapping("/login")
    public String authenticateUser(Credentials cr) {
        Customer user = userService.authenticate(cr);
        if(user == null)
            return "failed";
        return "welcome";
    }
}
```

- Create webapp/welcome.jsp

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Welcome</title>
</head>
<body>
    Welcome, User!!
</body>
</html>
```

- Create webapp/failed.jsp

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Failed</title>
</head>
<body>
    Invalid email or password.
```

```
</body>
</html>
```

- Deploy in Tomcat web server and run it.

Spring Boot Web MVC with JSP (boot_bookshop)

- Spring Initialer: Create new Maven project.
  - groupId: com.sunbeam
  - artifactId and projectName: boot_bookshop
  - package: com.sunbeam
  - Select dependencies
    - JDBC APIs
    - MySQL Driver
    - Spring Web
    - Spring Boot Dev Tools
- In pom.xml,
  - Change java.version to "11".
  - Change parent -- version to "2.7.18".
  - Add dependencies (Spring Boot doesn't have JSP support by default -- It is deprecated).

```
<dependency>
    <groupId>org.apache.tomcat.embed</groupId>
    <artifactId>tomcat-embed-jasper</artifactId>
</dependency>
<dependency>
    <groupId>javax.servlet</groupId>
    <artifactId>jstl</artifactId>
</dependency>
```

- Update Maven Project.

- In application.properties

```
spring.datasource.url=jdbc:mysql://localhost:3306/dmcdb
spring.datasource.username=dmc
spring.datasource.password=dmc

spring.mvc.view.prefix=/
spring.mvc.view.suffix=.jsp
```

- Refer GIT commits for further steps.