CS308

Software Engineering Lab

Midterm Review

05-Apr-2018

Spring Boot and JSP

The following questions are regarding Spring Boot, JSP and development in general, both theoretical and practical. Find them as a practice for the upcoming midterm. Good luck!

1. **Q: What does JSP stand for?**
   A: JSP stands for Java Server Pages and it is a technology for creating dynamically generated web pages based on XML and HTML.

2. **Q: How does a general JSP model look like?**
   A: General JSP architecture starts with a Database (or a data source) that interacts with a Server which is comprised of JavaBeans (Java Model) that communicates with JSP (view) which receives instructions from Servlet (Controller) that interacts with a Web Browser. Illustrated: Database ← → Java Model ← → JSP ← Controller ← → Web Browser → JSP

3. **Q: How do you integrate JSP into basic HTML?**
   A: We write it as scripting functions that are specified by <% code %>

4. **Q: Write a sample code in JSP page**
   A:
```
<h1> Here is a sample loop </h1>
     <% for (int i=0; i<=10; i++) { %>
     <span> The output is: <%= i %> </span>
<% } %>
```

5. **Q: What is Spring Boot?**
   A: It is a Java framework for creating standalone Spring applications.

6. **Q: Where do we write dependencies? Give one example of a dependency.**
   A: We write dependencies in pom.xml file. An example:

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

7. **Q: Write a sample controller in Spring boot. No need for writing imports.**
   A: MidtermReview.java

```
package midtermreview;
import *;

@Controller
@EnableAutoConfiguration
public class MidtermReview {

    @RequestMapping("/")
    @ResponseBody
    String home() {
        return "Hello World!";
    }

    public static void main(String[] args) throws Exception {
        SpringApplication.run(SampleController.class, args);
    }
}
```

8. **Q: What dependency do we integrate into Spring Boot project so we do not need to restart server on every change? With which dependency is server restart automated?**
   A: Dependency called: DevTools

9. **Q: What do you write to specify mapping or routing?**
   A: We write: @RequestMapping("/midtermReview")

10. Q: What is a Domain Driven Design? What layers do exist and what do they consist of?
    A: DDD is way of connecting the implementation to an evolving model. It focuses on logical distribution of layers. These layers include:

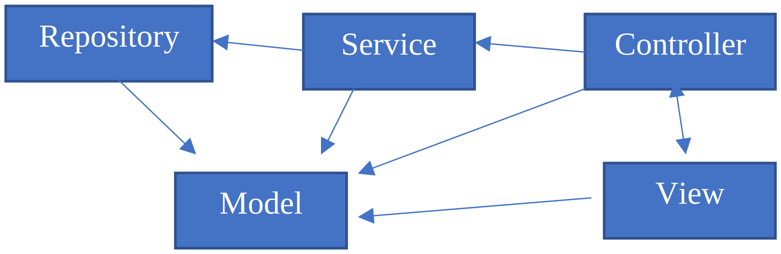
Presentational layer (Controller), Application Service Layer (Service), Domain Layer (Model, Service, Repository) and Infrastructure Layer (depends on project's domain artifacts).

11. **Q: Explain each layer.**
    A:
    Presentation layer is responsible to present the business function that is provided by Application Service Layer. It does presentation-related functionalities and delegates it to App Service.
    Application Service Layer represents business use cases. It deals with transactions set on methods and makes use of data transformations.
    Domain Layer is responsible for business logic that exists in models. Repositories are artifacts to retrieve the domain models.
    Infrastructure Layer is dependent of project's specifications and domain artifacts.

12. **Q: What is Dependency Injection and how can it be simply introduced in Spring? Where do we use it? Give an example.**
    A: Dependency Injection is a simple technique, a design pattern, of one object supplying the dependencies of another object. The dependency is an object that can be used and injection is passing this dependency to a dependent object that will use it. In simple words, it is a process where objects define their dependencies.
    In Spring, it is driven by @Autowired annotation. This annotation is used on properties, setters and constructors. With this, we simply skip configurations of what to inject, since @Autowired in Spring does the DI automatically.
    Example:

```
@Autowired
      public UserController (UserService userService) {
            this.userService = userService;
      }
```

13. **Q: Make a diagram of how Repository, Model, Controller, View and Service communicate.**
    A: Model makes the entity (i.e. specifies the db table). Repository communicates with model so that it serves as a central place where data is maintained. Service uses corresponding model and does actions on corresponding repository, that is 'serves' the repository (i.e. find User, calls on userRepository). Finally, controller communicates with service and uses it's 'services' to define the final step ready for View.



14. Q: Write a method that deletes a user when it's ID is specified in the path.

A:

```
@RequestMapping
public String deleteUser (@PathVariable Long id) {
        return userService.deleteUser(id);
}
```

15. Q: What is REST? What are it's most common operations and what is their use?

A: REST stands for Representational State Transfer and it defines constraints based on HTTP. The most common operations are GET, POST, PUT, DELETE. Get is used for reading/retrieving, POST for creating, PUT for updating/replacing and DELETE for deleting.

16. Q: What does CRUD stand for?

A: Create, Read/Retrieve, Update, Delete

17. Q: How would you read a username from the database in a JSP?

A: By simply putting: "${user.username}"

18. Q: How do you specify modes in JSP? Why would you use them?

A: We would specify mode like: <c:when test="${mode=='MODE_DEMO'}">
We use them as separate code snippets if we need to jump from one page block to another.

19. How would you write a ResourceNotFoundException when we want to go to a non-existing URL?

```
@ResponseStatus(value = HttpStatus.NOT_FOUND)
public class ResourceNotFoundException extends RuntimeException {

    private String resourceName;
    private String fieldName;
    private Object fieldValue;

    public ResourceNotFoundException( String resourceName, String fieldName, Object fieldValue) {
        super(String.format("%s not found with %s : '%s'", resourceName, fieldName, fieldValue));
        this.resourceName = resourceName;
        this.fieldName = fieldName;
        this.fieldValue = fieldValue;
    }

    public String getResourceName() {
        return resourceName;
    }

    public String getFieldName() {
        return fieldName;
    }

    public Object getFieldValue() {
        return fieldValue;
    }
}
```