Otto-Friedrich-University Bamberg

Distributed Systems Group

Distributed Systems Architecture and Middleware (DSAM)

Assignment 1 & 2

Introduction and Description
(Winter term 2020 - V1.0)

Preface

A former colleague, now working for a consulting company doing a lot of implementation projects, said once: ”The only decision nobody will criticize you in a software project nowadays is using SpringBoot for implementing your backend!”.

Spring and especially SpringBoot do and did a great job in getting enterprise application developers efficiently in production. It supports you at points where other frameworks are hard and nitpicky to configure and to handle. This first assignment is a starting point, where we will implement a full stack app in SpringBoot.

The following list is a recommendation which books and online resources are worth reading and where you find some conceptual help when designing your Spring application. Also the power point slides contain a lot of further information when following the various links:

- **Craig Walls**: Spring in Action, 5th Edition, Manning Publications, 2019.
- **Spring Boot Documentation**: Current SpringBoot version at time of writing this was 2.3.3.
  `https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle`.
- **JPA 2.2 spec**: As Thorben Janssen said there are a lot of ”easy ways to fix problems and exception but unfortunately you find a lot of bad advice online. The proclaimed fixes often replace the exception with a hidden problem in production.” When having problems with JPA it makes sense to read the spec first before reading some stackoverflow posts. Otherwise you will not understand the implications your decisions will have.
- **Baeldung**: Various posts from `https://www.baeldung.com/`. Here you will often find good advice. What's brilliant is that he concentrates on the problem, isolates it and then explains possible fixes.

We also recommend our sample project at GitHub:
`https://github.com/lion5/workshop-spring-boot`.

1 Assignment 1 - Dev

Winter is coming and we are all looking forward to having a barbecue with friends. For a nice barbecue, we also need some refreshing drinks. Therefore, we want to start a new business where we sell beverages and deliver them to people. The easiest way is to build some kind of online shop, where we show the beverages and offer customers an order option.

We will implement a part of this online shop. A user can look at all beverages and crates, add them to a basket and order them by checking out. We will not implement a purchase service or some aspects related to the warehouse management.

Technically we will realize our web interface with **Thymeleaf** (an HTML templating engine). For development purposes, we will use an **H2 database**. As enterprise framework we will use **Spring Boot** to ease our life.

The domain model of our application is designed as follows:

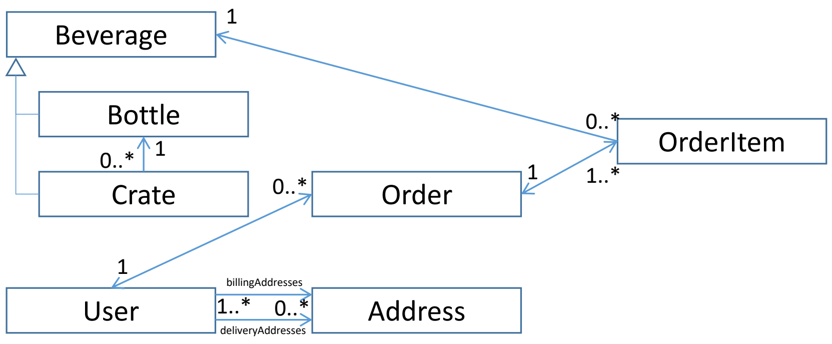

Figure 1: Domain Model of our Application

To keep the figure short, here are the attributes of the classes:

- **Bottle**
  - id: Long (autogenerated)
  - name: String (not null, not empty, only containing letters and digits)
  - bottlePic: String (url validation)
  - volume : double (> 0)
  - isAlcoholic: boolean
  - volumePercent: double (when value > 0.0, than *isAlcoholic* true)
  - price: int (>0)
  - supplier: String (not null, not empty)
  - inStock: int (≥ 0)
- **Crate**
  - id: Long (autogenerated)

  - name: String (not null, not empty, only containing letters and digits)
  - cratePic: String (url validation)
  - noOfBottles: int (>0)
  - price: double (>0.0)
  - cratesInStock: int (≥ 0)
- **User**
  - username: String (not null, not empty, unique)
  - password: String (not null, not empty)
  - birthday: LocalDate(>1.1.1900, <today)
- **Order**
  - id: Long (autogenerated)
  - price: double (>0.0)
- **Address**
  - id: Long (autogenerated)
  - street: String (not null, not empty)
  - number: String (not null, not empty)
  - postalCode: String (not null, not empty, only five digits)
- **OrderItem**
  - id: Long (autogenerated)
  - position: String (only digits)
  - price: double (>0.0)

We will use Java Bean Validation Specification (JSR-380) to validate the data used in our application. The mandatory constraints are written in braces. If you have further ideas how to enrich the data schema and/or validate the data attributes, feel free to add them.

DEADLINE: 23rd of December, 11.55PM

Hopefully you are finished earlier.
I'm looking forward to a lot of Spring Boot christmas presents.

Minimal System Requirements:

- Implement a web page to show all beverages and crates with the option to add them to a shopping basket. Add some demo data to your application as shown in the tutorial and store them in an H2 database.
- Implement another page to show all the elements within the basket and an option to submit the order. This stores the order in the H2 database.
- Implement a third page where you see all your orders.
- Implement a suitable navigation from one page to the other(s). I do not want to change the endpoint for every of these three pages manually.

Implementation Hints:

- Use SaveAction Plugin in IntelliJ or a similar mechanism to keep your source code clean. I will also assess this in your submission.
- Use a session scoped shopping basket, some helpful ideas might be in the spring documentation (`https://docs.spring.io/spring-session/docs/current/reference/html5/#introduction`) or in another web post(`https://www.baeldung.com/spring-bean-scopes`).
- In our GitHub repository under *solutions/task3*, we implemented a possible user management and also headers and footers for the web pages. So maybe you can profit from the code there. You do NOT have to implement the shown user handling, we will do this in the next step :)

Marking Hints:

- I will import your project in IntelliJ and run the application. If the application does not start properly, I will use the Gradle Wrapper to execute the bootRun task. If both solutions did not work, I will mark the submission with 0 points.
- Your mark is provided on a team basis. So work together to get the best out of it.