open business software solutions

Assignments

1. Standards

Standards that you should follow are listed below.

1. Use Maven.
   It will help you on management of projects on the following topics:
   - Building deployable artifacts
   - Dependency management
   - Project properties

   You can use maven either as a standalone application or eclipse plugin.

2. You are free to use any IDE to implement your solutions.

3. Create a maven project and implement your solutions as modules of this project. (see Figure 1)

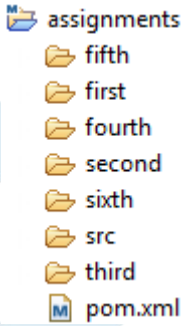

Figure 1 – Maven project structure

4. Use MySQL as database server. Configure Hibernate in a way that it will drop/create all database tables in each run.

5. Use "MainClass" as runnable class name (the class that contains the main method) in all questions.

6. Use Hibernate 5.3 or above version in questions.

7. Use JDK 1.8+ and JavaEE 8 (EJB 3.2, JPA 2.2) where appropriate.

2. Questions

1. Implement a java application which consists of three classes: Class1, Class2, and Class3.

   Class3 will call a method of Class2. This method will call a method of Class1. The method in Class1 will generate an exception. The method in Class2 will not handle this exception. The method in Class3 will catch this exception, wrap it as a CustomException and log its trace details to a file under the project folder using SLF4j. File name must be output/error.log (see Figure 2). Write a main class to call Class3.

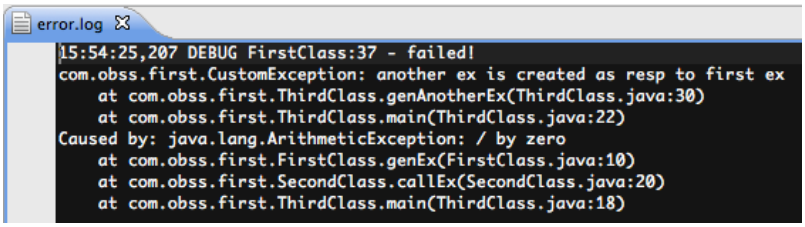

Figure 2 – Sample log file content

2. Create custom annotations and detect these annotations if they are used on any java classes.

   Create two custom annotations described as below. (see Table 1.)

Annotation	Retention	Target	Methods
Food	RUNTIME	TYPE	double price()
Time	RUNTIME	METHOD	int takes()

Table 1

There will be different classes for three different kinds of foods: Pizza, Kebab and Sushi. Food classes and their annotation values are listed below. (see Table 2)

Menu Item	Type Annotation	Annotation Value
Pizza	Food	price = 22.5
Kebab	Food	price = 12.5
Sushi	Food	price = 30

Table 2

Methods of the food classes and their annotation values are listed below. (see Table 3)

Menu Item	Method	Method Annotation	Annotation Value
Pizza	Prepare	Time	Takes = 10
Pizza	Cook	Time	Takes = 20
Pizza	Send	Time	Takes = 15
Kebab	Prepare	Time	Takes = 12
Kebab	Cook	Time	Takes = 20
Kebab	Send	Time	Takes = 10
Sushi	Prepare	Time	Takes = 30
Sushi	Cook	-	-
Sushi	Send	Time	Takes = 20

Table 3

After creating and using your custom annotations you have to detect the usage of the annotations. So, create a main class, which detects classes that use Food annotation. Do not use any third-party library on detection of the annotations! Load the classes from the classpath and try to get the annotations by reflection.
Finally, create a bean, which contains information about menu items (see Figure 3).

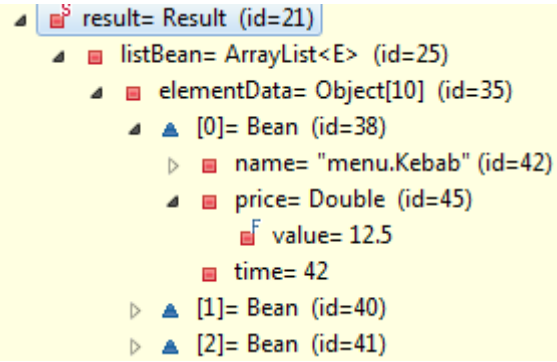

Figure 3

3. Implement a small web application using Spring MVC framework, Spring Security 5, JQuery, Json and Jasper Reports. Use Spring MVC 5.2.0+ framework while building your application. Use Hibernate with JPA annotations in database operations. Use Spring Security 5.2.0+ while building security parts of the application. Divide the project into parts explicitly such as DAO, Model, Service, Controller etc.

   There will be two types of users in the system: Admin, User. The roles and users will be stored in database. For creating database tables you can use the following DDL statements (see Figure 4 and 5).

```
CREATE TABLE users (
        username character varying(50) NOT NULL,
        password character varying(50) NOT NULL,
        email character varying(50) NOT NULL,
        birthday date,
        sexsmallint,
        enabledboolean,
        CONSTRAINT users_pkey PRIMARY KEY (username)
)
```

Figure 4

```
CREATE TABLE authorities (
        username character varying(50) NOT NULL,
        authority character varying(50) NOT NULL,
        CONSTRAINT fk_authorities_users FOREIGN KEY (username)
        REFERENCES users (username)
        MATCH SIMPLE ON UPDATE NO ACTION ON DELETE NO ACTION
)
```

Figure 5

There will be four pages on the client side of the application:

1. Login.jsp

Login page is one of the two (see Registration page for the other one) pages which is open for anonymous request. Users must login by entering their username and password to access home page. Login page will look like the below figure:

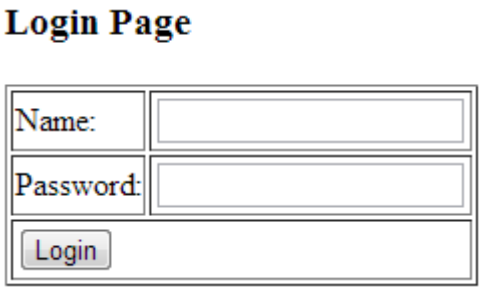

Figure 6

2. Registration.jsp

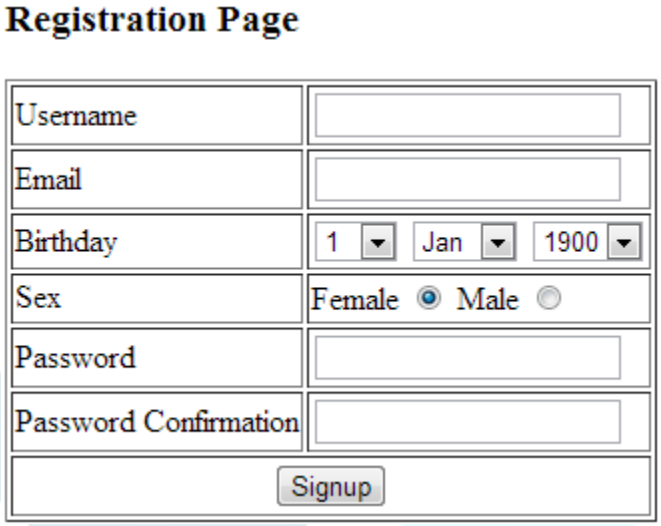

Figure 7

Registration page is open for anonymous access. There will be a form with several inputs on the page (see Figure 7). By using the registration page, users will be able to register themselves on the system. Users will be redirected to the login page after they complete the registration successfully.

Apply the following validations during form submit using JQuery and/or regex:

a. Username, Email, Password and Password confirmation inputs will not be null. If one of these inputs is blank, a notification will appear at the bottom of the form (see Figure 8).

Figure 4

b. User must supply a valid email address. A research about email address validation rules might be useful before starting to apply validation.
c. Password and Password Confirmation inputs have to match each other.

Finally, values of the username and email inputs will have to be checked if they are unique or not. You have to use JQuery Ajax requests (upon onBlur events of the corresponding fields) in order to check these values are unique or not. If they are not unique, notifications will appear at the bottom of the page as shown below (see Figure 9).

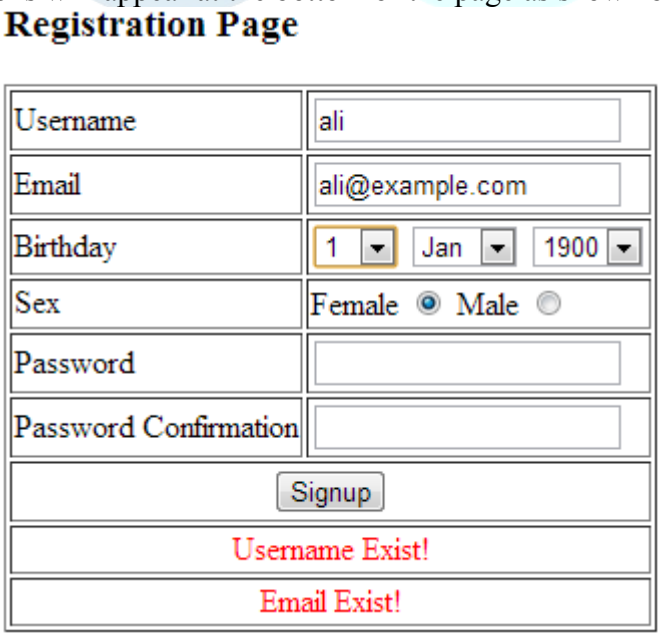

Figure 5

3. Home.jsp

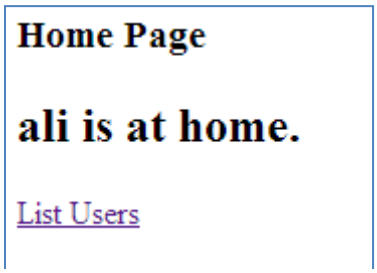

Figure 10

Home page will look like figure 10. It will contain a link (list users), which is only available for admin users. The other users will not see the link.

4. List-Users.jsp

Figure 6

List user page will contain a table, which lists the users in the system. There will be username, email, birthday and sex columns (see Figure 12).
User list will be sent from server to the client in the json format. There will be also "print table" button, which is visible only for admin users. This button will generate a pdf report. You have to use iReport and Jasper Reports for designing and printing respectively.

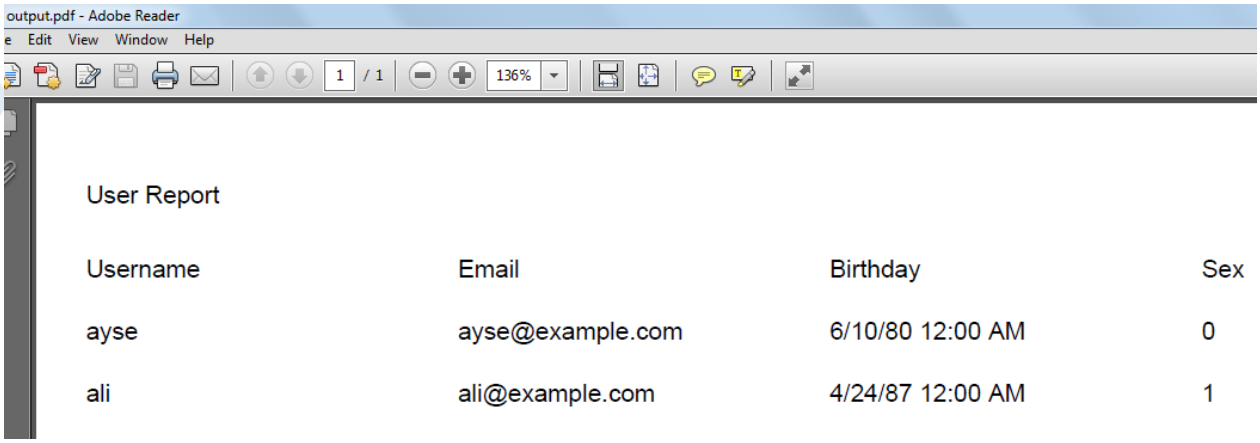

Figure 7

4. Implement a web service and a web service client by using JAX-WS 2.2 API. For database operations do not use JPA, only pure Hibernate. Your project must consist of a maven project with sub-modules (service and client). Use users table that was created in second question on your database.

Your web service
Your web service must be deployed on tomcat.

It must provide basic operations: select, insert, update and delete.

Your web service client
Web service client must be created by consuming the WSDL of your web service.
It will be able to read, update, delete a specific user by username and add a new user.
So, design your service according to your client needs.

5. Implement a java enterprise application which contains enterprise java beans and a remote client java application which invokes these beans by using JNDI API. Use WildFly 18+ as an application server. If you cannot succeed to deploy your J2EE project to the WildFly, try to deploy your application to different application servers which have EJB container. (Such as Glassfish 5)

   In Server Side

   Create and deploy a java enterprise application which contains an enterprise java bean. So, create an EJB application and place it inside of the enterprise application. In other words, you will have a J2EE project (.ear) which contains an EJB application (.jar)

   You must have two beans that will be placed inside of the EJB application. They will be invoked from a remote client and they must implement same "remote" interface. One of the session beans will be stateless session bean and the other one will be a stateful.

   Interface that will be implemented by the beans will have two methods: “getUserByName” and “updateUser”. You will be using "getUserByName" method to retrieve users from database and "updateUser" to update users on the database.

   Please use users table that you have created in third question. Also use Hibernate (annotations or configuration file) for database operations.

   In Client Side

   There will be two methods in remote client application. In first method:

   1. A user must be retrieved from database by username.
   2. Then, the stateless session bean must be invoked via JNDI to change birth date of the user.
   3. Update the user info on the database by remote invoking.
   4. After that, get another instance of the remote bean from JNDI. Retrieve the same user; update its sex this time via this new remote bean instance.
   5.

   In second method

   All steps that are explained above must be done with invoking stateful session bean.

6. Implement a client-server application and their communication over network. I/O operations between client and server applications must be implemented by using Non-blocking I/O (java.nio package).
   Create a maven project and develop the client and server as the modules of this project.

   Server Side

   In the server side you must build a simple NIO server which accepts connection request from the client side.

   After the connection request accepted, a number between 0-9 must be generated randomly. This number later will be compared with the number which is sent by the client. So, you have to able to read data over the connection established with the client. If the values are equal, write a message such as "Congratulations!", otherwise write a message "Auto generated value was X. Try less/bigger than Y." to the channel. The client will read this message. So, you have to write data over the connection.

   Client Side

   You must establish a connection with server. After connection is opened, you will send a number between 0-9. Send three numbers to the server successively. Client will get the response for the numbers respectively.

   You will need functions to send and receive data over connection with the server.

   Please do not use any application framework to implement your application.

7. Implement a small web application using Spring MVC framework (Spring Boot is also acceptable), Spring Security 5 and ReactJS. Use Spring MVC 5.2.0+ framework while building your application. Use Hibernate with JPA annotations in database operations. Use Spring Security 5.2.0+ while building security parts of the application. Divide the project into parts explicitly such as DAO, Model, Service, Controller etc.

   There will be two types of users in the system: Admin, User. The roles and users will be stored in database. In addition to users and roles, there will be book, authors, read_list and favorite_list(additional tables can be added in accordance with your design) tables in DB. Main requirements of the project(Book Portal) are stated below:

   Book Portal:

   - ✓ Requirement:
     - Functional:
       - Admins/users can login to system
       - Admin should be able to:
         - Add/delete/update books, users, authors etc. to/from system
         - Search for book and users

- Users should be able to:
  - See list of books
  - Add books to read list
  - Add books to favorite list
  - Search book by name (Optional: you can search by other information too)

- Technical:
  - There should be two different users: Admin & End-user
  - Use Spring Security for authentication/authorization
  - Use Hibernate with JPA annotations in database operations
  - DB information should be read from properties file
  - Use ReactJS for frontend

Flow:

1) Draw ER diagrams
2) Create your schema and DB tables
2) Create Spring MVC project with Spring Security integration
3) Create models corresponding to your DB tables in Spring project
4) Create Service and DAO components
5) Create RESTful services
6) Perform operations via Postman to test and verify your services
7) Develop frontend using ReactJS and its components
8) Perform end-to-end tests to increase your applications stability