**Bodha SERvice KT Document**

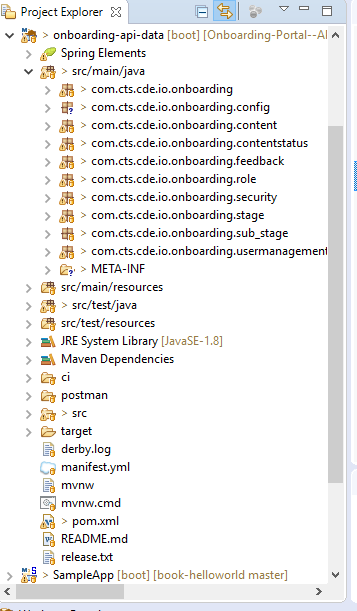
List of Content

1. Technology Details
2. Code Walkthrough
   * Package description
   * Application Introduction
   * Maven Dependencies
   * Entity description
   * login using spring security
   * Controller, Service and Repository
   * Database connection configure description using yml
   * Testing
   * Dev and Prod environment details
   * Run and build the application
   * Application Dev and Prod environment (pcf) details and url description
   * GitHub Details
   * CI-CD (Blue Green Deployment)
3. Technology Details.

|  |
| --- |
| Technology |
| Spring Boot |
| Java |
| MySQL |
|  |
| Maven |
| JUnit |
| Concourse |
| Fly |
| GitHub |

1. Code Walkthrough

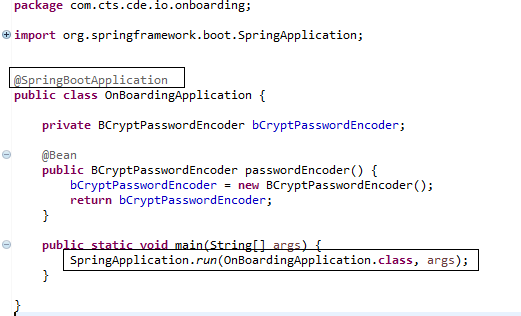
* Package description



* Application Introduction

Bodha Server api is written as SpringBootApplication and main class named as OnBoardingApplication.java

*Spring Boot* provides a number of “Starters” that make easy to add jars to your classpath. Our *application* has already used *spring*-*boot*-starter-parent in the parent section of the POM. The *spring*-*boot*-starter-parent is a special starter that provides useful Maven defaults. It also provides a dependency-management.



* Maven Dependencies.

Declared all related maven dependency to pom.xml. Below are the few of them:

<dependencies>

<!-- Starter for using Spring Data JPA with Hibernate -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-jpa</artifactId>

</dependency>

<!-- Starter for exposing Spring Data repositories over REST using Spring Data REST -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-rest</artifactId>

</dependency>

<!-- Starter for building web, including RESTful, applications using Spring MVC. Uses Tomcat as the default embedded container -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<!-- Starter for testing Spring Boot applications with libraries including JUnit, Hamcrest and Mockito -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

</dependency>

<!-- JDBC Type 4 driver for MySQL -->

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

<scope>runtime</scope>

</dependency>

<!-- Starter for using Spring Security -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-security</artifactId>

</dependency>

<!-- JSON Web Token Support For The JVM -->

<dependency>

<groupId>io.jsonwebtoken</groupId>

<artifactId>jjwt</artifactId>

<version>0.7.0</version>

</dependency>

<!-- Gson -->

<dependency>

<groupId>com.google.code.gson</groupId>

<artifactId>gson</artifactId>

<version>2.8.2</version>

</dependency>

<!-- H2 Database Engine -->

<dependency>

<groupId>com.h2database</groupId>

<artifactId>h2</artifactId>

<scope>test</scope>

</dependency>

<!-- Apache HttpComponents Client -->

<dependency>

<groupId>org.apache.httpcomponents</groupId>

<artifactId>httpclient</artifactId>

<version>4.5.3</version>

<scope>test</scope>

</dependency>

<!-- json -->

<dependency>

<groupId>org.json</groupId>

<artifactId>json</artifactId>

<version>20080701</version>

</dependency>

<!-- The Apache Commons IO library contains utility classes, stream implementations, file filters, file comparators, endian transformation classes, and much more -->

<dependency>

<groupId>org.apache.commons</groupId>

<artifactId>commons-io</artifactId>

<version>1.3.2</version>

</dependency>

<!-- Spring Context Support -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context-support</artifactId>

<version>3.2.0.RELEASE</version>

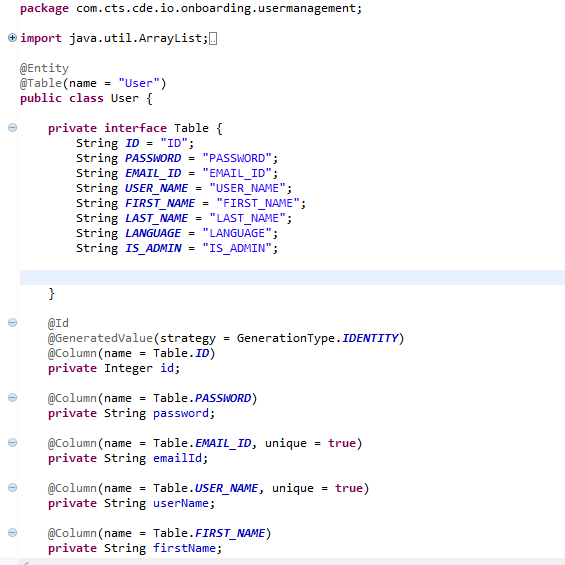
</dependency>

</dependencies>

* Entity description

We have entities namely User, Role, Content, Content Status, Feedback, Stage and Substage.

Eg:



Here in above screenshot we can see the User Entity with table name **User** corresponding column declarations. Same as like we have for other entities.

* login using spring security

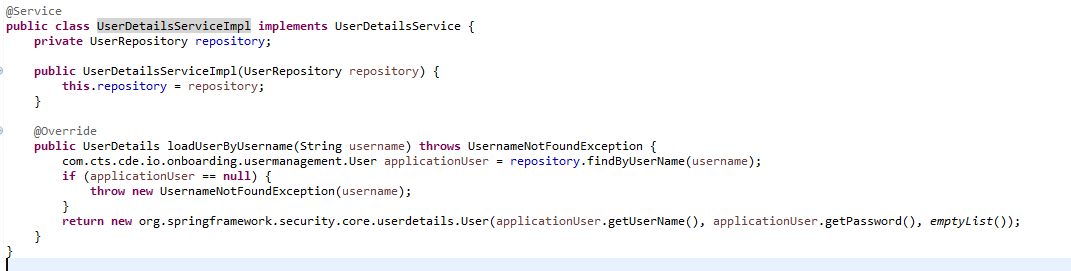
We have used Spring security for login to Bodha portal. Spring Security is a framework which provides various security features like: authentication, authorization to create secure Java Enterprise Applications.

JSON Web Tokens, commonly known as JWTs, are tokens that are used to authenticate users on applications. During the authentication process, when a user successfully logs in using their credentials, a JSON Web Token is returned and must be saved locally (typically in local storage). Whenever the user wants to access a protected route or resource (an endpoint), the user agent must send the JWT, usually in the Authorization header using the [Bearer schema](http://self-issued.info/docs/draft-ietf-oauth-v2-bearer.html), along with the request.



**UserDetailsService:**

If we want to use any DAO class for authentication, we need to implement UserDetailsService interface. Once the DAO is configured, it’s loadUserByUsername() is used to validate the user.



* Controller, Service and Repository

Corresponding to every module such as User, Role, Content, Content Status, Feedback, Stage and Substage we have separate controllers, Service and Repositories.

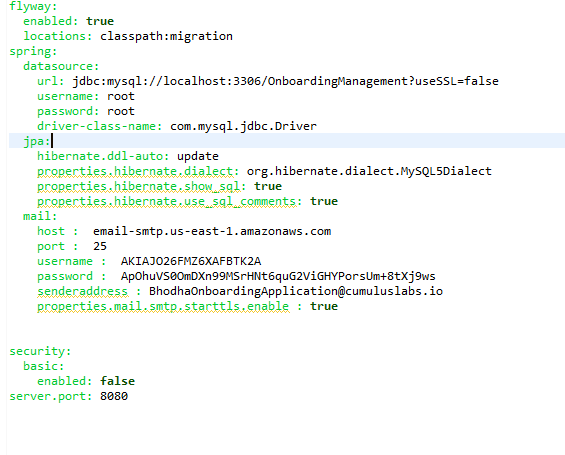
Presentation Layer: Controller (@Controller)

Application Service Layer: Application Service (@Service)

Domain Layer: Model, Domain Service (@Service), Repository (@Repository)

* Database connection configure description using yml

Spring Boot has excellent support for YAML configuration. In our YML we have database connection properties and server port declaration and flyway db migration. Below is the yml screenshot.



* Testing

We have written Junit test cases for each class and integration test for the application. Have used Mockito. Mockito is a popular mock framework which can be used in conjunction with JUnit.





* Run and build the application

Run: mvn spring-boot:run

Build: mvn clean package

* Application Dev and Prod Urls

**DEV API URL:** [**https://bodha-apijsoningnore.app.dev.digifabricpcf.com/**](https://bodha-apijsoningnore.app.dev.digifabricpcf.com/)

**PROD API URL :** [**https://bodhaapi.app.dev.digifabricpcf.com/**](https://bodhaapi.app.dev.digifabricpcf.com/)

* GitHub Details

<https://github.com/TheCognizantFoundry/Onboarding-Portal--API.git>