PROGRAMMING TECHIE

Build a Full Stack Reddit Clone with – Spring boot and Angular – Part 1

2 Comments

BY SAI UPADHYAYULA | September 30, 2019

Share Tweet Pin

What's up, everybody!! In this series of blog posts, we will see how to build a Reddit Clone with Spring Boot and Angular. We will be using different Spring Technologies like Spring Data JPA, Spring Security with JWT Authentication and MySQL as the database. On the frontend side, we will be using Angular 8 as our frontend framework and Bootstrap as the CSS framework.

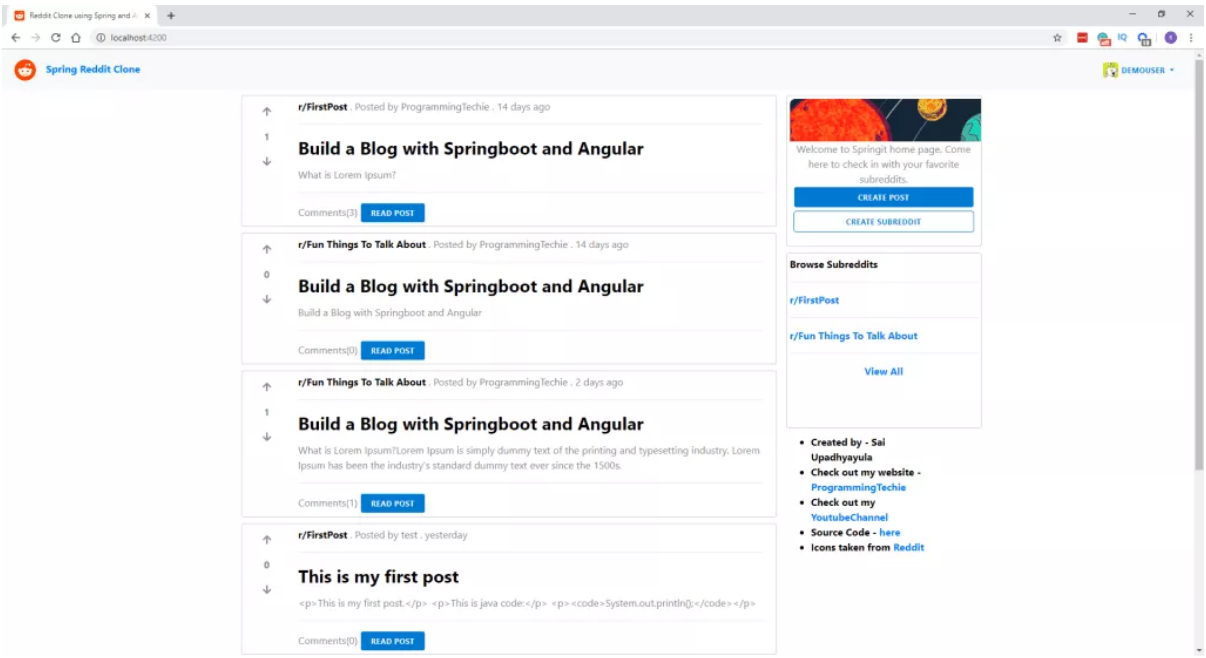
If you are a visual learner like me, you can have a look at the youtube video I have created as a companion to this post.

> The source code of this project, is hosted on Github –
> Backend – https://github.com/SaiUpadhyayula/spring-reddit-clone
> Frontend – https://github.com/SaiUpadhyayula/angular-reddit-clone

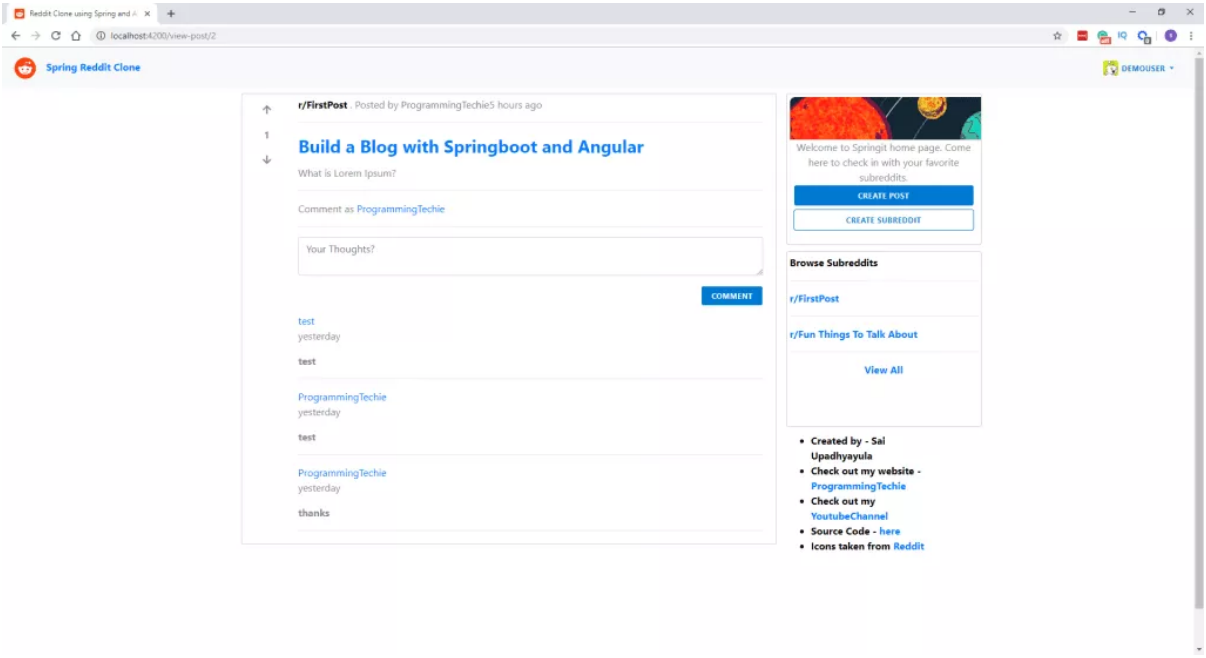
> If you like this kind of tutorials, then besure to check my other tutorial series on how to build a blog using Spring boot and Angular here:
> https://programmingtechie.com/2019/03/17/how-to-build-a-simple-blog-application-using-spring-boot-and-angular/

Here are some screenshots of the application:

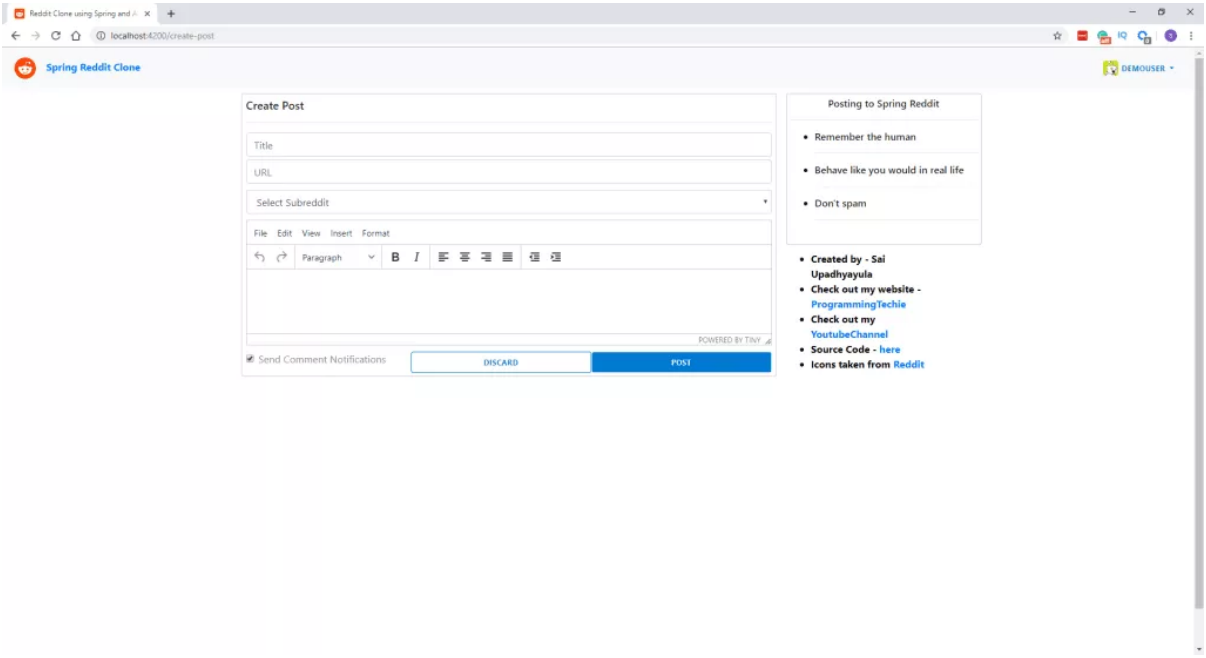
Home Page



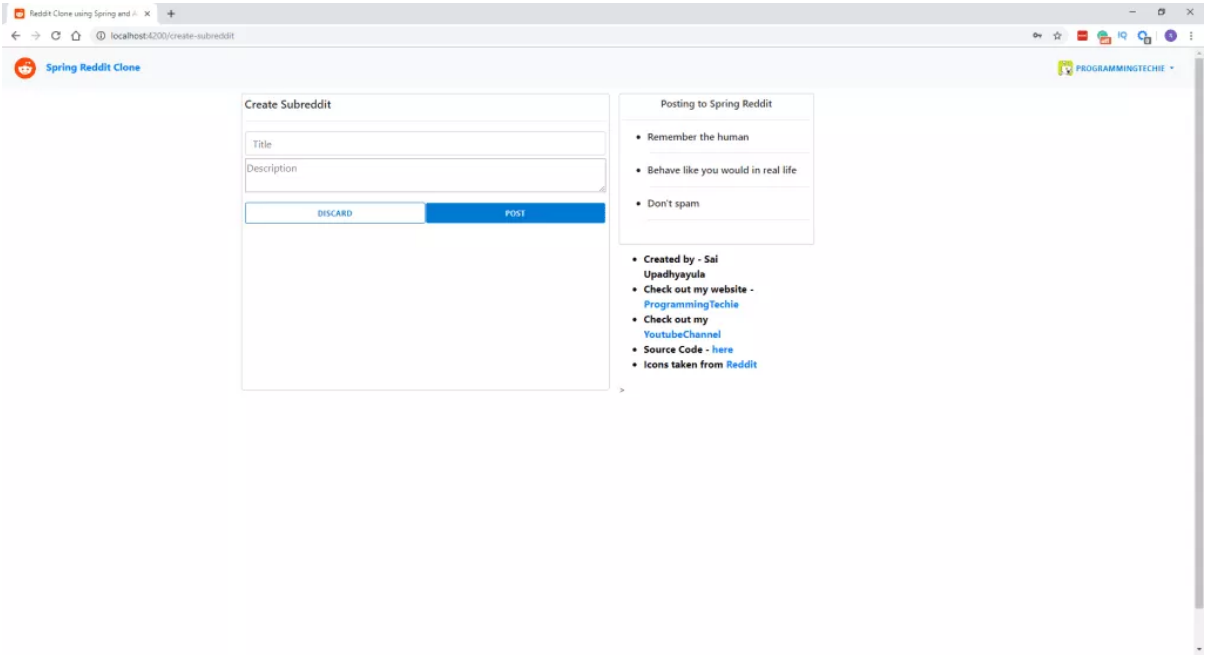
Post Page



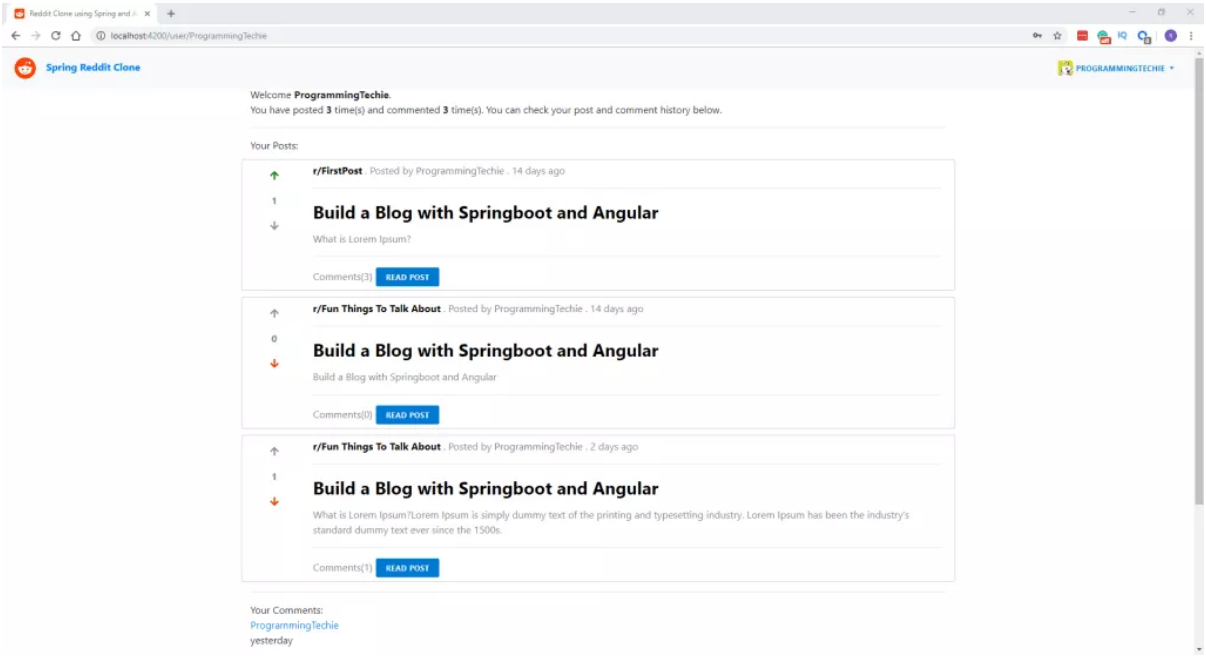
Create Post Page



Create Subreddit

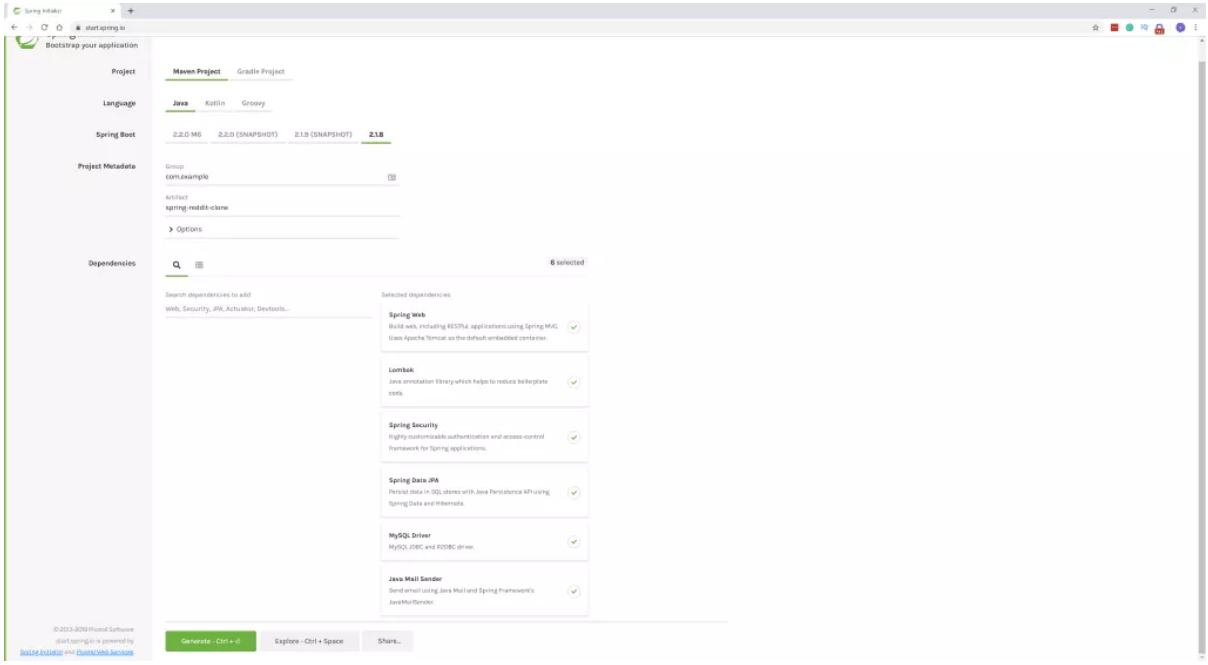


User Profile

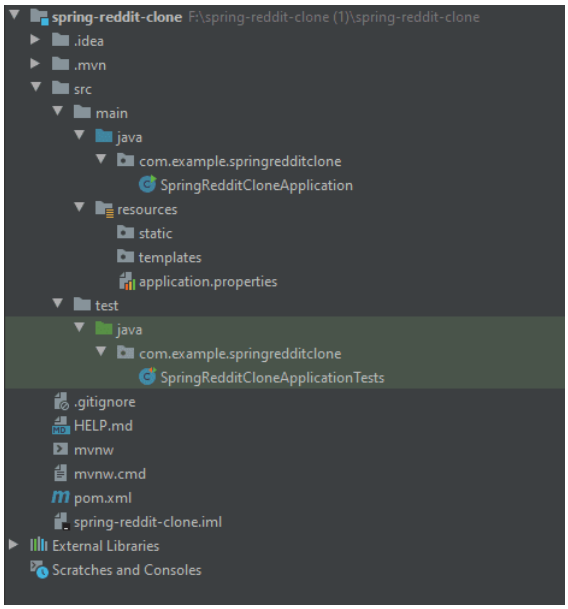


So let's start building the backend of this application, the first thing we have to do, go to Spring Initializr website .

- Enter spring-reddit-clone in **Artifact** field
- Add these dependencies – **Lombok**, **Spring Web**, **Spring Security**, **Spring Data JPA**, **MySQL Java Driver**, **Java Mail Sender**
- Click on the Generate button, and the project will be downloaded.



Open the project in your favourite IDE, the project structure looks something like below:



Additional Dependencies

Open the **pom.xml** file and add the below additional dependencies which are needed to implement the JWT authentication and other functionalities like dynamically displaying the relative duration (like “Posted 1 day ago”).

```
<!-- JWT related dependencies-->
        <dependency>
            <groupId>io.jsonwebtoken</groupId>
            <artifactId>jjwt-api</artifactId>
            <version>0.10.5</version>
        </dependency>
        <dependency>
            <groupId>io.jsonwebtoken</groupId>
            <artifactId>jjwt-impl</artifactId>
            <scope>runtime</scope>
            <version>0.10.5</version>
        </dependency>
        <dependency>
            <groupId>io.jsonwebtoken</groupId>
            <artifactId>jjwt-jackson</artifactId>
            <scope>runtime</scope>
```

```
        <version>0.10.5</version>
    </dependency>
    <!-- For Displaying time as Relative Time Ago ("Posted 1 Day ago"),
     as this is a Kotlin library, we also need Kotlin runtime libraries-->
    <dependency>
        <groupId>com.github.marlonlom</groupId>
        <artifactId>timeago</artifactId>
        <version>4.0.1</version>
    </dependency>
    <dependency>
        <groupId>org.jetbrains.kotlin</groupId>
        <artifactId>kotlin-stdlib-jdk8</artifactId>
        <version>${kotlin.version}</version>
    </dependency>
    <dependency>
        <groupId>org.jetbrains.kotlin</groupId>
        <artifactId>kotlin-test</artifactId>
        <version>${kotlin.version}</version>
        <scope>test</scope>
    </dependency>
```

Configure Database, Hibernate and Java Mail Properties

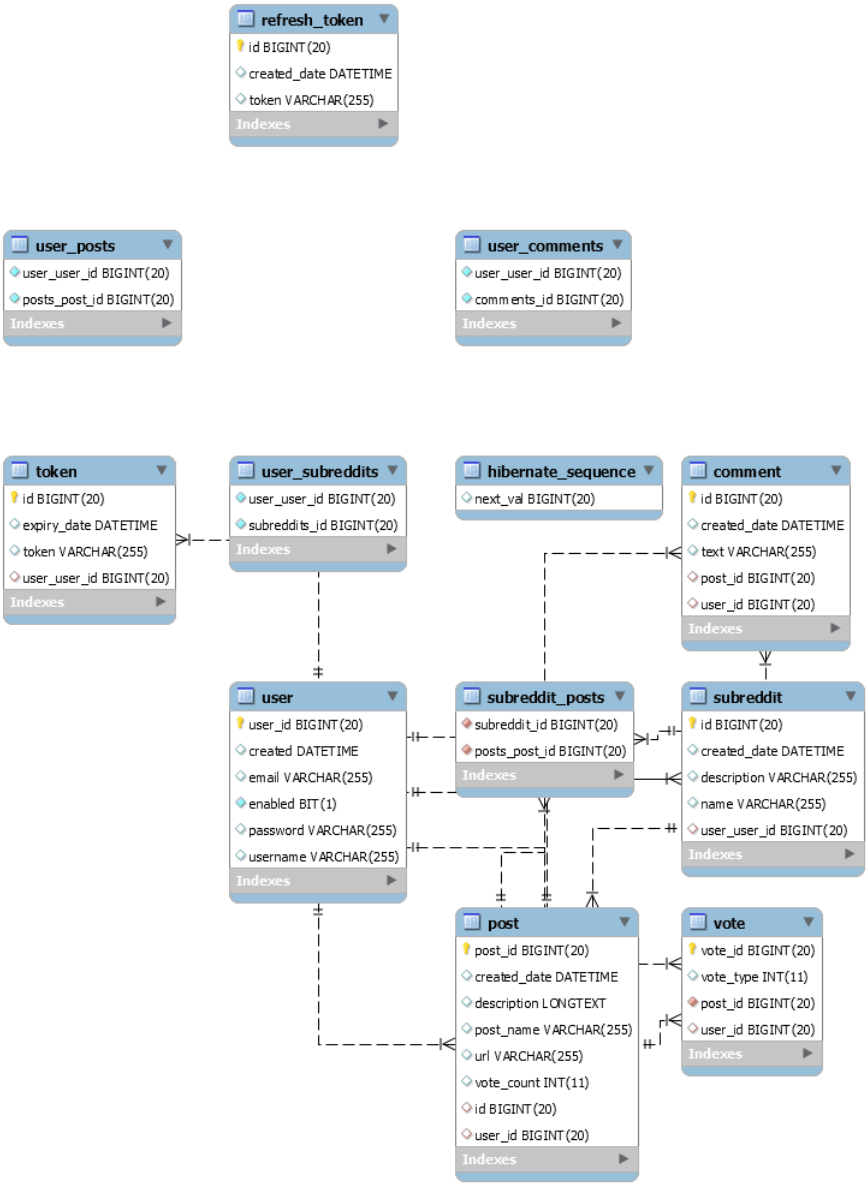
Let's now configure the MySQL Database, Hibernate JPA and Java Mail functionality in our application by adding the following properties to **src/main/resources/application.properties** file

```
###### Database Properties  ##########################################
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.url=jdbc:mysql://localhost:3306/spring-reddit-clone?
useSSL=false&serverTimezone=UTC&useLegacyDatetimeCode=false
spring.datasource.username=<your-db-username>
spring.datasource.password=<your-db-password>
spring.jpa.hibernate.ddl-auto=update
spring.datasource.initialize=true
spring.jpa.show-sql=true
############ Mail Properties ##########################################
spring.mail.host=smtp.mailtrap.io
spring.mail.port=25
spring.mail.username=<your-username>
spring.mail.password=<your-password>
spring.mail.protocol=smtp
```

In our application, we will be sending Account Activation Emails and Comment Notification Emails to the users, for that reason we need an SMTP server to send the emails, we can use a Fake SMTP Server called as MailTrap for this application, you can signup to Mailtrap using your Github or Google account.

Database Schema Diagram

Here is the DB Schema Diagram



Creating Domain Entities

Now let's create domain entities for our Reddit Clone Application. In our application, we have Users, who can create Subreddits and Posts, other users can add Comments on the Posts, and can Vote.

> Note that, we are using Lombok Annotations like @Data, @AllArgsConstructor, and @NoArgsConstructor. These annotations will generate the corresponding Getters/Setters/Equals and HashCode/toString methods and Constructors at compile time. To be able to use these annotations, you have to enable Annotation Processing in your IDE. For more details check this link on how to enable Lombok in your favorite IDE – https://www.baeldung.com/lombok-ide

- **User** Entity

```
package com.programming.techie.springredditclone.model;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;
import javax.validation.constraints.Email;
import javax.validation.constraints.NotBlank;
import javax.validation.constraints.NotEmpty;
```

```
import java.time.Instant;

import static javax.persistence.GenerationType.SEQUENCE;

@Data
@AllArgsConstructor
@NoArgsConstructor
@Entity
public class User {
    @Id
    @GeneratedValue(strategy = SEQUENCE)
    private Long userId;
    @NotBlank(message = "Username is required")
    private String username;
    @NotBlank(message = "Password is required")
    private String password;
    @Email
    @NotEmpty(message = "Email is required")
    private String email;
    private Instant created;
    private boolean enabled;
}
```

- **Post** Entity

```
package com.programming.techie.springredditclone.model;

import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;
import org.springframework.lang.Nullable;

import javax.persistence.*;
import javax.validation.constraints.NotBlank;
import java.time.Instant;

import static javax.persistence.FetchType.LAZY;
import static javax.persistence.GenerationType.SEQUENCE;

@Data
@Entity
@Builder
@AllArgsConstructor
@NoArgsConstructor
public class Post {
    @Id
    @GeneratedValue(strategy = SEQUENCE)
    private Long postId;
    @NotBlank(message = "Post Name cannot be empty or Null")
    private String postName;
    @Nullable
    private String url;
    @Nullable
    @Lob
    private String description;
    private Integer voteCount;
    @ManyToOne(fetch = LAZY)
    @JoinColumn(name = "userId", referencedColumnName = "userId")
    private User user;
    private Instant createdDate;
    @ManyToOne(fetch = LAZY)
    @JoinColumn(name = "id", referencedColumnName = "id")
    private Subreddit subreddit;
}
```

- **Subreddit** Entity

```
package com.example.springredditclone.model;

import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;

import javax.persistence.*;
import javax.validation.constraints.NotBlank;
import java.time.Instant;
import java.util.List;

import static javax.persistence.FetchType.LAZY;
import static javax.persistence.GenerationType.SEQUENCE;

@Data
@AllArgsConstructor
@NoArgsConstructor
@Entity
@Builder
public class Subreddit {
    @Id
    @GeneratedValue(strategy = SEQUENCE)
    private Long id;
    @NotBlank(message = "Community name is required")
    private String name;
    @NotBlank(message = "Description is required")
    private String description;
    @OneToMany(fetch = LAZY)
    private List<Post> posts;
    private Instant createdDate;
    @ManyToOne(fetch = LAZY)
    private User user;
}
```

- **Vote** Entity

```
package com.example.springredditclone.model;

import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;

import javax.persistence.*;
import javax.validation.constraints.NotNull;

import static javax.persistence.FetchType.LAZY;
import static javax.persistence.GenerationType.SEQUENCE;

@Data
@AllArgsConstructor
@NoArgsConstructor
@Entity
@Builder
public class Vote {
    @Id
    @GeneratedValue(strategy = SEQUENCE)
    private Long voteId;
    private VoteType voteType;
    @NotNull
    @ManyToOne(fetch = LAZY)
    @JoinColumn(name = "postId", referencedColumnName = "postId")
```

```
    private Post post;
    @ManyToOne(fetch = LAZY)
    @JoinColumn(name = "userId", referencedColumnName = "userId")
    private User user;
}
```

- **Comment** Entity

```
package com.example.springredditclone.model;

import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;

import javax.persistence.*;
import javax.validation.constraints.NotEmpty;
import java.time.Instant;

import static javax.persistence.FetchType.LAZY;
import static javax.persistence.GenerationType.SEQUENCE;

@Data
@AllArgsConstructor
@NoArgsConstructor
@Builder
@Entity
public class Comment {
    @Id
    @GeneratedValue(strategy = SEQUENCE)
    private Long id;
    @NotEmpty
    private String text;
    @ManyToOne(fetch = LAZY)
    @JoinColumn(name = "postId", referencedColumnName = "postId")
    private Post post;
    private Instant createdDate;
    @ManyToOne(fetch = LAZY)
    @JoinColumn(name = "userId", referencedColumnName = "userId")
    private User user;
}
```

- **VoteType** enum

```
package com.example.springredditclone.model;

public enum VoteType {
    UPVOTE(1), DOWNVOTE(-1),
    ;

    VoteType(int direction) {
    }
}
```

- **VerificationToken.java**

```
package com.example.springredditclone.model;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;
```

```
import javax.persistence.*;
import java.time.Instant;

import static javax.persistence.FetchType.LAZY;
import static javax.persistence.GenerationType.IDENTITY;
import static javax.persistence.GenerationType.SEQUENCE;

@Data
@AllArgsConstructor
@NoArgsConstructor
@Entity
@Table(name = "token")
public class VerificationToken {

    @Id
    @GeneratedValue(strategy = IDENTITY)
    private Long id;
    private String token;
    @OneToOne(fetch = LAZY)
    private User user;
    private Instant expiryDate;
}
```

- **NotificationEmail.java**

```
package com.example.springredditclone.model;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data
@AllArgsConstructor
@NoArgsConstructor
public class NotificationEmail {
    private String subject;
    private String recipient;
    private String body;
}
```

Configure Repositories

So these are our domain entities, all of the fields are pretty self-explanatory. Now let's create the repositories which are responsible to store these entities in the database. Create the package **repository** and create below repositories.

- **User Repository**

```
package com.example.springredditclone.repository;

import com.example.springredditclone.model.User;
import org.springframework.data.jpa.repository.JpaRepository;

import java.util.Optional;

public interface UserRepository extends JpaRepository<User, Long> {
    Optional<User> findByUsername(String username);
}
```

- **Post Repository**

```
package com.example.springredditclone.repository;

import com.example.springredditclone.model.Post;
import com.example.springredditclone.model.Subreddit;
import com.example.springredditclone.model.User;
import org.springframework.data.jpa.repository.JpaRepository;

import java.util.List;

public interface PostRepository extends JpaRepository<Post, Long> {
    List<Post> findAllBySubreddit(Subreddit subreddit);

    List<Post> findByUser(User user);
}
```

- **Comment Repository**

```
package com.example.springredditclone.repository;

import com.example.springredditclone.model.Comment;
import com.example.springredditclone.model.Post;
import com.example.springredditclone.model.User;
import org.springframework.data.jpa.repository.JpaRepository;

import java.util.List;

public interface CommentRepository extends JpaRepository<Comment, Long> {
    List<Comment> findByPost(Post post);

    List<Comment> findAllByUser(User user);
}
```

- **Subreddit Repository**

```
package com.example.springredditclone.repository;

import com.example.springredditclone.model.Subreddit;
import org.springframework.data.jpa.repository.JpaRepository;

import java.util.Optional;

public interface SubredditRepository extends JpaRepository<Subreddit, Long> {
    Optional<Subreddit> findByName(String subredditName);
}
```

- **Vote Repository**

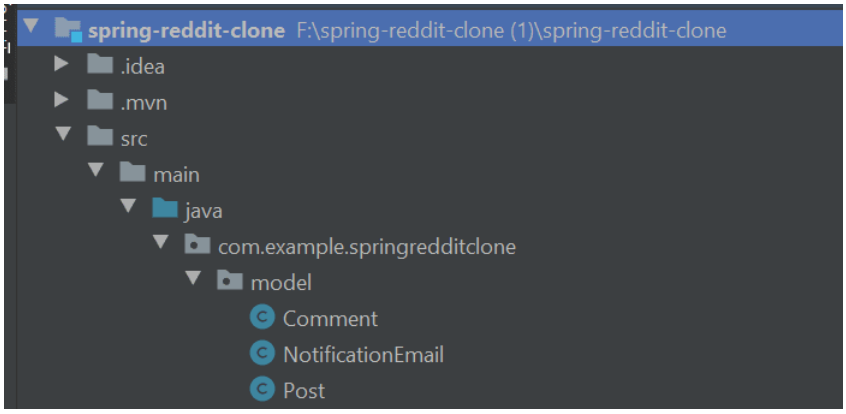
```
package com.example.springredditclone.repository;

import com.example.springredditclone.model.Post;
import com.example.springredditclone.model.User;
import com.example.springredditclone.model.Vote;
import org.springframework.data.jpa.repository.JpaRepository;

import java.util.Optional;

public interface VoteRepository extends JpaRepository<Vote, Long> {
    Optional<Vote> findTopByPostAndUserOrderByVoteIdDesc(Post post, User currentUser);
}
```

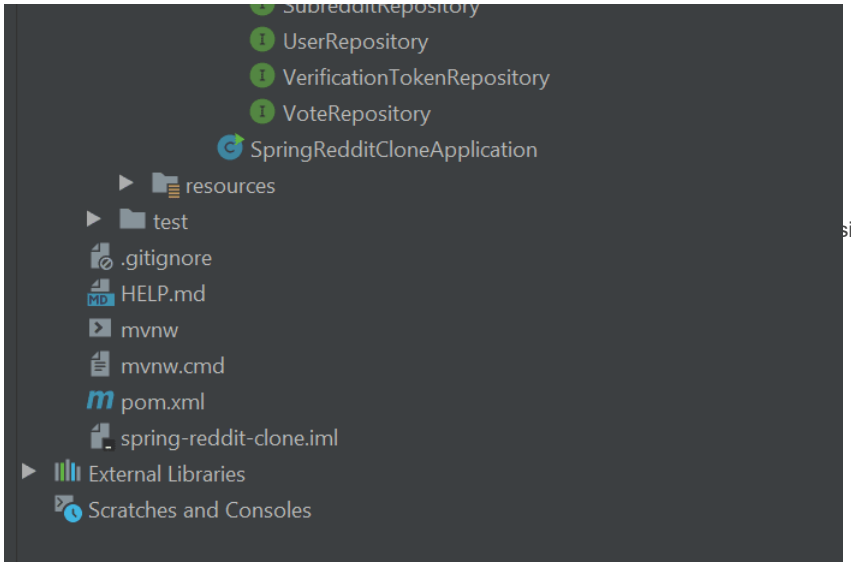
- VerificationToken Repository

```
package com.example.springredditclone.repository;

import com.example.springredditclone.model.VerificationToken;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import java.util.Optional;

@Repository
public interface VerificationTokenRepository extends JpaRepository<VerificationToken, Long> {
    Optional<VerificationToken> findByToken(String token);
}
```

Explore the project structure and Running the application

After creating all the models and repositories, the project structure looks like below





Now let's run the application by typing the below command by using the maven wrapper in the project.

```
./mvnw spring-boot:run
```

Now you can check the logs and you should see something like below at the end of the logs:

```
2019-09-30 21:45:56.819  INFO 23180 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat
started on port(s): 8080 (http) with context path ''
 2019-09-30 21:45:56.820  INFO 23180 --- [           main] c.e.s.SpringRedditCloneApplication       : Started
SpringRedditCloneApplication in 4.702 seconds (JVM running for 5.044)
```

In the next part, we will see how to configure Spring Security, implement API for Registration. We will also see how to send activation emails to activate the account.

About the author

Sai Upadhyayula

Max Kwan

October 3, 2019 at 9:00 am

Good tutorial.

Sai

October 4, 2019 at 7:41 pm

Thank you Max, make sure to subscribe to the blog, to get updates when I post the next parts.

Comments are closed.