

Web API Design with SpringBoot Week 1 Coding Assignment

Points possible: 70

Category	Criteria	% of Grade
Functionality	Does the code work?	25
Organization	Is the code clean and organized? Proper use of white space, syntax, and consistency are utilized. Names and comments are concise and clear.	25
Creativity	Student solved the problems presented in the assignment using creativity and out of the box thinking.	25
Completeness	All requirements of the assignment are complete.	25

Instructions: In Eclipse, or an IDE of your choice, write the code that accomplishes the objectives listed below. Ensure that the code compiles and runs as directed. Take screenshots of the code and of the running program (make sure to get screenshots of all required functionality) and paste them in this document where instructed below. Create a new repository on GitHub for this week's assignments and push this document, with your Java project code, to the repository. Add the URL for this week's repository to this document where instructed and submit this document to your instructor when complete.

Coding Steps:

Follow the tutorial in the homework section to build an API. Paste screenshots of your code and your postman requests and responses to show the API works. Push your project to GitHub and paste the link below.

Screenshots of Code:

App.java

```

1 package com.promineotech.socialMediaApi;
2
3 import org.springframework.boot.SpringApplication;
4
5 @ComponentScan("com.promineotech.socialMediaApi")
6 @SpringBootApplication
7 public class App {
8
9     public static void main( String[] args ) {
10        SpringApplication.run(App.class, args);
11    }
12 }
13 
```

Following.java

```

1 package com.promineotech.socialMediaApi.controller;
2
3 import org.springframework.beans.factory.annotation.Autowired;
4 import org.springframework.http.HttpStatus;
5 import org.springframework.web.bind.annotation.PathVariable;
6 import org.springframework.web.bind.annotation.PathVariable;
7 import org.springframework.web.bind.annotation.RequestMapping;
8 import org.springframework.web.bind.annotation.RequestMethod;
9 import org.springframework.web.bind.annotation.RequestParam;
10 import org.springframework.web.bind.annotation.RestController;
11 import com.promineotech.socialMediaApi.entity.Comment;
12 import com.promineotech.socialMediaApi.service.CommentService;
13
14 @RestController
15 @RequestMapping("/users/{userId}/posts/{postId}/comments")
16 public class CommentController {
17
18     @Autowired
19     private CommentService service;
20
21     // What about RequestMethod.GET OR RequestMethod.PUT???
22
23     @RequestMapping(method=RequestMethod.POST)
24     public ResponseEntity<Object> createComment(@RequestBody Comment comment, @PathVariable Long userId, @PathVariable Long postId) {
25         try {
26             return new ResponseEntity<Object>(service.createComment(comment, userId, postId), HttpStatus.OK);
27         } catch (Exception e) {
28             return new ResponseEntity<Object>(e.getMessage(), HttpStatus.BAD_REQUEST);
29         }
30     }
31
32     @RequestMapping(value="/commentId", method=RequestMethod.DELETE)
33     public ResponseEntity<Object> deleteComment(@PathVariable Long commentId) {
34         service.deleteComment(commentId);
35         return new ResponseEntity<Object>("Deleted comment with id:" + commentId, HttpStatus.OK);
36     }
37
38 }
39 
```

Following.java

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3   xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
4   <modelVersion>4.0.0</modelVersion>
5
6   <groupId>com.promineotech.socialMediaApi</groupId>
7   <artifactId>socialMediaApi</artifactId>
8   <version>0.0.1-SNAPSHOT</version>
9   <name>socialMediaApi</name>
10  <url>http://maven.apache.org/url</url>
11
12  <properties>
13      <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
14      <maven.compiler.source>1.7</maven.compiler.source>
15      <maven.compiler.target>1.7</maven.compiler.target>
16  </properties>
17
18  <parent>
19      <groupId>org.springframework.boot</groupId>
20      <artifactId>spring-boot-starter-parent</artifactId>
21      <version>1.2.6.RELEASE</version>
22  </parent>
23
24  <dependencies>
25      <dependency>
26          <groupId>org.springframework.boot</groupId>
27          <artifactId>spring-boot-starter-web</artifactId>
28          <exclusions>
29              <exclusion>
30                  <groupId>org.springframework.boot</groupId>
31                  <artifactId>spring-boot-starter-logging</artifactId>
32              </exclusion>
33          </exclusions>
34      </dependency>
35      <dependency>
36          <groupId>org.springframework.boot</groupId>
37          <artifactId>spring-boot-starter-log4j2</artifactId>
38      </dependency>
39      <dependency>
40          <groupId>org.springframework.boot</groupId>
41          <artifactId>spring-boot-starter-logstash</artifactId>
42      </dependency>
43      <dependency>
44          <groupId>mysql:mysql-connector-java</groupId>
45          <artifactId>mysql-connector-java</artifactId>
46          <version>8.0.13</version>
47          <scope>runtime</scope>
48      </dependency>
49      <dependency>
50          <groupId>com.jayway.jsonpath</groupId>
51          <artifactId>json-path</artifactId>
52          <scope>test</scope>
53      </dependency>
54      <dependency>
55          <groupId>org.springframework.boot</groupId>
56          <artifactId>spring-boot-starter-data-jpa</artifactId>
57      </dependency>
58      <dependency>
59          <groupId>org.springframework.security</groupId>
60          <artifactId>spring-security-crypto</artifactId>
61          <version>5.1.2.RELEASE</version>
62      </dependency>
63      <dependency>
64          <groupId>junit:junit</groupId>
65          <artifactId>junit</artifactId>
66          <version>4.13</version>
67          <scope>test</scope>
68      </dependency>
69  </dependencies>
70 
```

Following.java

```

1 package com.promineotech.socialMediaApi.controller;
2
3 import org.springframework.beans.factory.annotation.Autowired;
4 import org.springframework.http.HttpStatus;
5 import org.springframework.http.ResponseEntity;
6 import org.springframework.web.bind.annotation.PathVariable;
7 import org.springframework.web.bind.annotation.RequestBody;
8 import org.springframework.web.bind.annotation.RequestMapping;
9 import org.springframework.web.bind.annotation.RequestMethod;
10 import org.springframework.web.bind.annotation.RestController;
11
12 import com.promineotech.socialMediaApi.entity.Post;
13 import com.promineotech.socialMediaApi.service.PostService;
14
15 @RestController
16 @RequestMapping("/users/{userId}/posts")
17 public class PostController {
18
19     @Autowired
20     private PostService service;
21
22     // Get all posts corresponding to a user
23     @RequestMapping(method=RequestMethod.GET)
24     public ResponseEntity<Object> getAllPosts() {
25         return new ResponseEntity<Object>(service.getAllPosts(), HttpStatus.OK);
26     }
27
28     // Get one post by postId corresponding to a user
29     @RequestMapping(value="/{postId}", method=RequestMethod.GET)
30     public ResponseEntity<Object> getPost(@PathVariable Long postId) {
31         return new ResponseEntity<Object>(service.getPost(postId), HttpStatus.OK);
32     }
33
34     // Update a post by postId
35     @RequestMapping(value="/{postId}", method=RequestMethod.PUT)
36     public ResponseEntity<Object> updatePost(@RequestBody Post post, @PathVariable Long postId) {
37         try {
38             return new ResponseEntity<Object>(service.updatePost(post, postId), HttpStatus.OK);
39         } catch (Exception e) {
40             return new ResponseEntity<Object>(e.getMessage(), HttpStatus.BAD_REQUEST);
41         }
42     }
43
44     // Create a comment
45     @RequestMapping(method=RequestMethod.POST)
46     public ResponseEntity<Object> createPost(@RequestBody Post post, @PathVariable Long userId) {
47         try {
48             return new ResponseEntity<Object>(service.createPost(post, userId, HttpStatus.OK));
49         } catch (Exception e) {
50             return new ResponseEntity<Object>(e.getMessage(), HttpStatus.BAD_REQUEST);
51         }
52     }
53 }
54 
```

Following.java

```

1 package com.promineotech.socialMediaApi.controller;
2
3 import java.nio.file.File;
4 import java.nio.file.Path;
5 import java.nio.file.Paths;
6
7 import org.springframework.beans.factory.annotation.Autowired;
8 import org.springframework.http.HttpHeaders;
9 import org.springframework.http.HttpStatus;
10 import org.springframework.http.ResponseEntity;
11 import org.springframework.web.bind.annotation.PathVariable;
12 import org.springframework.web.bind.annotation.RequestBody;
13 import org.springframework.web.bind.annotation.RequestMapping;
14 import org.springframework.web.bind.annotation.RequestMethod;
15 import org.springframework.web.bind.annotation.RequestParam;
16 import org.springframework.web.bind.annotation.RestController;
17 import org.springframework.web.multipart.MultipartFile;
18
19 import com.promineotech.socialMediaApi.entity.User;
20 import com.promineotech.socialMediaApi.service.UserService;
21
22 @RestController
23 @RequestMapping("/users")
24 public class UserController {
25
26     private static String UPLOADED_FOLDER = ".\\pictures\\";
27
28     @Autowired
29     private UserService service;
30
31     // Register or Create a user
32     @RequestMapping(value = "register", method = RequestMethod.POST)
33     public ResponseEntity<Object> register(@RequestBody User user) {
34         try {
35             return new ResponseEntity<Object>(service.createUser(user), HttpStatus.CREATED);
36         } catch (Exception e) {
37             // Allow a user to login
38             if (e.getMessage().equals("login")) {
39                 public ResponseEntity<Object> login(@RequestBody User user) {
40                     try {
41                         return new ResponseEntity<Object>(service.login(user), HttpStatus.OK);
42                     } catch (Exception e) {
43                         return new ResponseEntity<Object>(e.getMessage(), HttpStatus.BAD_REQUEST);
44                     }
45                 }
46             }
47             @RequestMapping(value = "/{id}/follow", method = RequestMethod.GET)
48             public ResponseEntity<Object> showFollowedUsers(@PathVariable Long id) {
49                 try {
50                     return new ResponseEntity<Object>(service.showFollowedUsers(id), HttpStatus.CREATED);
51                 } catch (Exception e) {
52                     return new ResponseEntity<Object>(e.getMessage(), HttpStatus.BAD_REQUEST);
53                 }
54             }
55             @RequestMapping(value = "/{id}/follow/{followId}", method = RequestMethod.POST)
56             public ResponseEntity<Object> follow(@PathVariable Long id, @PathVariable Long followId) {
57                 try {
58                     return new ResponseEntity<Object>(service.followUser(id, followId), HttpStatus.CREATED);
59                 } catch (Exception e) {
60                     return new ResponseEntity<Object>(e.getMessage(), HttpStatus.BAD_REQUEST);
61                 }
62             }
63             @RequestMapping(value = "/{id}/profilePicture", method = RequestMethod.POST)
64             public ResponseEntity<Object> singleFileUpload(@PathVariable long id, @RequestParam("file") MultipartFile file) {
65                 if (file.isEmpty()) {
66                     return new ResponseEntity<Object>("Please upload a file.", HttpStatus.BAD_REQUEST);
67                 }
68                 try {
69                     String url = UPLOADED_FOLDER + file.getOriginalFilename();
70                     Path path = Paths.get(url);
71                     Files.write(path, bytes);
72                     return new ResponseEntity<Object>(service.updateProfilePicture(id, url), HttpStatus.CREATED);
73                 } catch (Exception e) {
74                     return new ResponseEntity<Object>(e.getMessage(), HttpStatus.INTERNAL_SERVER_ERROR);
75                 }
76             }
77         }
78     }
79
80 }
81
82
83
84
85 
```

```

1 package com.promineotech.socialMediaApi.entity;
2
3 import java.util.Date;
4
5 @Entity
6 public class Comment {
7
8     private long id;
9     private String content;
10    private Date date;
11    private User user;
12
13    @JsonIgnore
14    private Post post;
15
16    @Id
17    @GeneratedValue(strategy = GenerationType.AUTO)
18    public long getId() {
19        return id;
20    }
21
22    public void setId(long id) {
23        this.id = id;
24    }
25
26    public String getContent() {
27        return content;
28    }
29
30    public void setContent(String content) {
31        this.content = content;
32    }
33
34    public Date getDate() {
35        return date;
36    }
37
38    public void setDate(Date date) {
39        this.date = date;
40    }
41
42    @ManyToOne
43    @JoinColumn(name="userId")
44    public User getUser() {
45        return user;
46    }
47
48    public void setUser(User user) {
49        this.user = user;
50    }
51
52    @ManyToOne
53    @JoinColumn(name = "postId")
54    public Post getPost() {
55        return post;
56    }
57
58    public void setPost(Post post) {
59        this.post = post;
60    }
61
62
63
64
65
66
67
68
69 }

```

```

1 package com.promineotech.socialMediaApi.entity;
2
3 import java.util.Set;
4
5 import javax.persistence.CascadeType;
6 import javax.persistence.Entity;
7 import javax.persistence.GeneratedValue;
8 import javax.persistence.GenerationType;
9 import javax.persistence.Id;
10 import javax.persistence.JoinColumn;
11 import javax.persistence.ManyToOne;
12 import javax.persistence.OneToMany;
13
14 @Entity
15 public class User {
16
17     private Long id;
18     private String username;
19     private String profilePictureUrl;
20
21     @JsonIgnore
22     private Set<User> following;
23
24     private String password;
25
26     @JsonIgnore
27     private Set<Post> posts;
28
29     @JsonIgnore
30     private Set<Comment> comments;
31
32     @Id
33     @GeneratedValue(strategy = GenerationType.AUTO)
34     public Long getId() {
35         return id;
36     }
37
38     public void setId(Long id) {
39         this.id = id;
40     }
41
42     public String getUsername() {
43         return username;
44     }
45
46     public void setUsername(String username) {
47         this.username = username;
48     }
49
50     @JsonIgnore
51     public String getPassword() {
52         return password;
53     }
54
55     @JsonProperty
56     public void setPassword(String password) {
57         this.password = password;
58     }
59
60     @OneToMany(mappedBy = "user")
61     public Set<Post> getPosts() {
62         return posts;
63     }
64
65     public void setPosts(Set<Post> posts) {
66         this.posts = posts;
67     }
68
69     @OneToMany(mappedBy = "user")
70     public Set<Comment> getComments() {
71         return comments;
72     }
73
74     public void setComments(Set<Comment> comments) {
75         this.comments = comments;
76     }
77
78     @ManyToOne(cascade = CascadeType.ALL)
79     @JoinTable(name = "following",
80     joinColumns = @JoinColumn(name = "userId", referencedColumnName = "id"),
81     inverseJoinColumns = @JoinColumn(name = "followingId", referencedColumnName = "id"))
82     public Set<User> getFollowing() {
83         return following;
84     }
85
86     public void setFollowing(Set<User> following) {
87         this.following = following;
88     }
89
90     public String getProfilePictureUrl() {
91         return profilePictureUrl;
92     }
93
94     public void setProfilePictureUrl(String profilePictureUrl) {
95         this.profilePictureUrl = profilePictureUrl;
96     }
97 }

```

```

1 package com.promineotech.socialMediaApi.entity;
2
3 import java.util.Date;
4
5 import java.util.Set;
6
7 import javax.persistence.Entity;
8 import javax.persistence.GeneratedValue;
9 import javax.persistence.GenerationType;
10 import javax.persistence.Id;
11 import javax.persistence.JoinColumn;
12 import javax.persistence.ManyToOne;
13 import javax.persistence.OneToMany;
14
15 @Entity
16 public class Post {
17
18     private Long id;
19     private String content;
20     private Date date;
21     private User user;
22     private Set<Comment> comments;
23
24     @Id
25     @GeneratedValue(strategy = GenerationType.AUTO)
26     public Long getId() {
27         return id;
28     }
29
30     public void setId(Long id) {
31         this.id = id;
32     }
33
34     public String getContent() {
35         return content;
36     }
37
38     public void setContent(String content) {
39         this.content = content;
40     }
41
42     public Date getDate() {
43         return date;
44     }
45
46     public void setDate(Date date) {
47         this.date = date;
48     }
49
50     @ManyToOne
51     @JoinColumn(name = "userId")
52     public User getUser() {
53         return user;
54     }
55
56     public void setUser(User user) {
57         this.user = user;
58     }
59
60     @OneToMany(mappedBy = "post")
61     public Set<Comment> getComments() {
62         return comments;
63     }
64
65     public void setComments(Set<Comment> comments) {
66         this.comments = comments;
67     }
68 }

```

```
 UserRepository.java Following.java CommentService.java PostService.java
1 package com.promineotech.socialMediaApi.repository;
2
3 import org.springframework.data.repository.CrudRepository;
4
5 import com.promineotech.socialMediaApi.entity.User;
6
7 public interface UserRepository extends CrudRepository<User, Long> {
8
9     User findByUsername(String username);
10 }
11
12 }
```

```
 CommentRepository.java
1 package com.promineotech.socialMediaApi.repository;
2
3 import org.springframework.data.repository.CrudRepository;
4
5 import com.promineotech.socialMediaApi.entity.Comment;
6
7 public interface CommentRepository extends CrudRepository<Comment, Long> {
8
9 }
10
```

```
 PostRepository.java
1 package com.promineotech.socialMediaApi.repository;
2
3 import org.springframework.data.repository.CrudRepository;
4
5 import com.promineotech.socialMediaApi.entity.Post;
6
7 public interface PostRepository extends CrudRepository<Post, Long> {
8
9 }
10
```

```
 Following.java PostService.java UserService.java
1 package com.promineotech.socialMediaApi.view;
2
3 import java.util.Set;
4
5 import com.promineotech.socialMediaApi.entity.User;
6
7 public class Following {
8
9     private Set<User> following;
10
11     public Following(User user) {
12         following = user.getFollowing();
13     }
14
15     public Set<User> getFollowing() {
16         return following;
17     }
18
19     public void setFollowing(Set<User> following) {
20         this.following = following;
21     }
22
23
24 }
```

```
 Following.java PostService.java UserService.java
1 package com.promineotech.socialMediaApi.service;
2
3 import java.util.Date;
4
5 import org.springframework.beans.factory.annotation.Autowired;
6 import org.springframework.stereotype.Service;
7
8 import com.promineotech.socialMediaApi.entity.Post;
9 import com.promineotech.socialMediaApi.entity.User;
10 import com.promineotech.socialMediaApi.repository.PostRepository;
11 import com.promineotech.socialMediaApi.repository.UserRepository;
12
13 @Service
14 public class PostService {
15
16     @Autowired
17     private PostRepository repo;
18
19     @Autowired
20     private UserRepository userRepo;
21
22     // Find all posts
23     public Iterable<Post> getAllPosts() {
24         return repo.findAll();
25     }
26
27     //Find an Individual post
28     public Post getPost(Long id) {
29         return repo.findById(id);
30     }
31
32     // Update a Post
33     public Post updatePost(Post post, Long id) throws Exception {
34         Post foundPost = repo.findById(id);
35         if (foundPost == null) {
36             throw new Exception("Post not found.");
37         }
38         foundPost.setContent(post.getContent());
39         return repo.save(foundPost);
40     }
41
42     // Create a Post
43     public Post createPost(Post post, Long userId) throws Exception {
44         User user = userRepo.findOne(userId);
45         if (user == null) {
46             throw new Exception("User not found.");
47         }
48         post.setDate(new Date());
49         post.setUser(user);
50         return repo.save(post);
51     }
52
53     //Delete a post (Should we add this??)
54 }
```

```
 Following.java CommentService.java PostService.java UserService.java
1 package com.promineotech.socialMediaApi.service;
2
3 import java.util.Date;
4
5 import org.springframework.beans.factory.annotation.Autowired;
6 import org.springframework.stereotype.Service;
7
8 import com.promineotech.socialMediaApi.entity.Comment;
9 import com.promineotech.socialMediaApi.entity.Post;
10 import com.promineotech.socialMediaApi.entity.User;
11 import com.promineotech.socialMediaApi.repository.CommentRepository;
12 import com.promineotech.socialMediaApi.repository.PostRepository;
13 import com.promineotech.socialMediaApi.repository.UserRepository;
14
15 @Service
16 public class CommentService {
17
18     @Autowired
19     private CommentRepository repo;
20
21     @Autowired
22     private PostRepository postRepo;
23
24     @Autowired
25     private UserRepository userRepo;
26
27     public Comment createComment(Comment comment, Long userId, Long postId) throws Exception {
28         User user = userRepo.findOne(userId);
29         Post post = postRepo.findOne(postId);
30         if (user == null || post == null) {
31             throw new Exception("User or Post does not exist.");
32         }
33         comment.setDate(new Date());
34         comment.setUserId(user);
35         comment.setPost(post);
36         return repo.save(comment);
37     }
38
39     public void deleteComment(Long commentId) {
40         repo.delete(commentId);
41     }
42
43 }
```

```
 Following.java PostService.java UserService.java
1 package com.promineotech.socialMediaApi.service;
2
3 import org.springframework.beans.factory.annotation.Autowired;
4 import org.springframework.stereotype.Service;
5
6 import com.promineotech.socialMediaApi.entity.User;
7 import com.promineotech.socialMediaApi.repository.UserRepository;
8 import com.promineotech.socialMediaApi.view.Following;
9
10 @Service
11 public class UserService {
12
13     @Autowired
14     private UserRepository repo;
15
16     // register a new user
17     public User createNewUser(User user) {
18         return repo.save(user);
19     }
20
21     // allow users to log in
22     public User login(User user) throws Exception {
23         User foundUser = repo.findByUsername(user.getUsername());
24         if (foundUser != null && foundUser.getPassword().equals(user.getPassword())) {
25             return foundUser;
26         } else {
27             throw new Exception("Invalid username or password.");
28         }
29     }
30
31     // allow users to follow other users
32     public Following follow(Long userId, Long followId) throws Exception {
33         User user = repo.findById(userId);
34         User follow = repo.findById(followId);
35         if (user == null || follow == null) {
36             throw new Exception("User does not exist.");
37         }
38         user.getFollowing().add(follow);
39         repo.save(user);
40         return new Following();
41     }
42
43     // get a list of followed users for a given user
44     public Following getFollowedUsers(Long userId) throws Exception {
45         User user = repo.findOne(userId);
46         if (user == null) {
47             throw new Exception("User does not exist.");
48         }
49         return new Following();
50     }
51
52     // update a user's profile picture
53     public User updateProfilePicture(Long userId, String url) throws Exception {
54         User user = repo.findOne(userId);
55         if (user == null) {
56             throw new Exception("User does not exist.");
57         }
58         user.setProfilePictureUrl(url);
59         return repo.save(user);
60     }
61 }
```

Screenshots of Running Application:

The screenshots demonstrate the application's functionality using a REST client (Postman).

- POST /users/{id}/posts**: Adds a new post for user with id 1. Content: "Hey everyone, this is my first post!".
- GET /users/{id}/posts**: Returns all posts for user with id 1. One post is returned with id 1, content "Hey everyone, this is my first post!", date 161781342623, user id 1, username "Lisa", profile picture URL ./pictures/IMG_1339.HEIC, and no comments.
- PUT /users/{id}/posts/{post_id}**: Updates the post with id 1 for user with id 1. Content: "I am updating this post now!".
- POST /users/{id}/posts/{post_id}/comments**: Adds a comment to the post with id 1 for user with id 1. Content: "Commenting on this post".
- GET /users/{id}/posts/{post_id}**: Returns the updated post with id 1 for user with id 1. Content: "I am updating this post now!", date 161781342623, user id 1, username "Lisa", profile picture URL ./pictures/IMG_1339.HEIC, and one comment with id 2, content "Commenting on this post", date 161781342623, user id 2, username "Sally", profile picture URL null.
- DELETE /users/{id}/posts/{post_id}/comments/{comment_id}**: Deletes the comment with id 1 for user with id 1.
- PUT /users/{id}/posts/{post_id}**: Updates the post with id 1 for user with id 1. Content: "I am updating this post now!".

```
mysql> select * from user;
+----+-----+-----+-----+
| id | password | profile_picture_url | username |
+----+-----+-----+-----+
| 1  | 98765   | NULL           | Lisa      |
| 2  | 76543   | NULL           | Sally     |
+----+-----+-----+-----+
2 rows in set (0.00 sec)

mysql> select * from post;
Empty set (0.00 sec)

mysql> select * from following;
Empty set (0.00 sec)

mysql> select * from user;
+----+-----+-----+-----+
| id | password | profile_picture_url | username |
+----+-----+-----+-----+
| 1  | 98765   | ./pictures/IMG_1339.HEIC | Lisa      |
| 2  | 76543   | NULL           | Sally     |
+----+-----+-----+-----+
2 rows in set (0.01 sec)

mysql> select * from post;
+----+-----+-----+-----+
| id | content          | date        | user_id |
+----+-----+-----+-----+
| 1  | I am updating this post now! | 2021-01-27 16:02:22 | 1        |
+----+-----+-----+-----+
1 row in set (0.00 sec)

mysql> select * from comment;
Empty set (0.00 sec)

mysql> select * from following;
+----+-----+
| user_id | following_id |
+----+-----+
| 1       | 2            |
+----+-----+
1 row in set (0.01 sec)

mysql> █
```

URL to GitHub Repository: <https://github.com/sw-dev-lisa-s-nh/SpringBoot-week1.git>