

Web API Design with SpringBoot Week 1 Coding Assignment

Points possible: 70

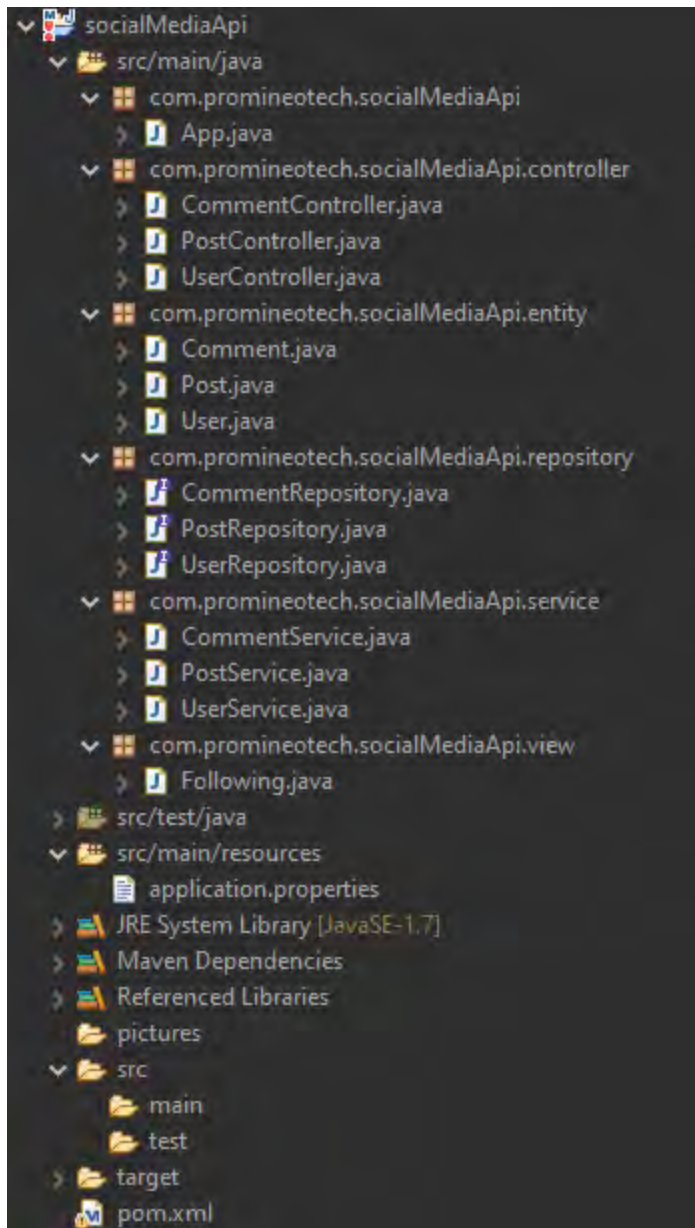
Category	Criteria	% of Grade
Functionality	Does the code work?	25
Organization	Is the code clean and organized? Proper use of white space, syntax, and consistency are utilized. Names and comments are concise and clear.	25
Creativity	Student solved the problems presented in the assignment using creativity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

```
1 package com.promineotech.socialMediaApi;
2
3 import org.springframework.boot.SpringApplication;
4
5
6
7 @ComponentScan("com.promineotech.socialMediaApi")
8 @SpringBootApplication
9
10 public class App
11 {
12     public static void main( String[] args )
13     {
14         SpringApplication.run(App.class, args);
15     }
16 }
17
```

CommentController.java x

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

```
PostController.java X
1 package com.promineotech.socialMediaApi.controller;
2
3 import org.springframework.beans.factory.annotation.Autowired;
14
15 @RestController
16 @RequestMapping("/users/{userId}/posts")
17 public class PostController {
18
19     @Autowired
20     private PostService service;
21
22     @RequestMapping(method=RequestMethod.GET)
23     public ResponseEntity<Object> getAllPosts() {
24         return new ResponseEntity<Object>(service.getAllPosts(), HttpStatus.OK);
25     }
26
27     @RequestMapping(value="/{postId}", method=RequestMethod.GET)
28     public ResponseEntity<Object> getPost(@PathVariable Long postId) {
29         return new ResponseEntity<Object>(service.getPost(postId), HttpStatus.OK);
30     }
31
32     @RequestMapping(value="/{postId}", method=RequestMethod.PUT)
33     public ResponseEntity<Object> updatePost(@RequestBody Post post, @PathVariable Long postId) {
34         try {
35             return new ResponseEntity<Object>(service.updatePost(post, postId), HttpStatus.OK);
36         } catch (Exception e) {
37             return new ResponseEntity<Object>(e.getMessage(), HttpStatus.BAD_REQUEST);
38         }
39     }
40
41     @RequestMapping(method=RequestMethod.POST)
42     public ResponseEntity<Object> createPost(@RequestBody Post post, @PathVariable Long userId) {
43         try {
44             return new ResponseEntity<Object>(service.createPost(post, userId), HttpStatus.OK);
45         } catch (Exception e) {
46             return new ResponseEntity<Object>(e.getMessage(), HttpStatus.BAD_REQUEST);
47         }
48     }
49
50 }
51
```

```

UserController.java x
1 package com.promineotech.socialMediaApi.controller;
2
3 import java.nio.file.Files;
20
21 @RestController
22 @RequestMapping("/users")
23 public class UserController {
24
25     private static String UPLOADED_FOLDER = "./pictures/";
26
27     @Autowired
28     private UserService service;
29
30     @RequestMapping(value = "/register", method = RequestMethod.POST)
31     public ResponseEntity<Object> register(@RequestBody User user) {
32         return new ResponseEntity<Object>(service.createUser(user), HttpStatus.CREATED);
33     }
34
35     @RequestMapping(value = "/login", method = RequestMethod.POST)
36     public ResponseEntity<Object> login(@RequestBody User user) {
37         try {
38             return new ResponseEntity<Object>(service.login(user), HttpStatus.OK);
39         } catch (Exception e) {
40             return new ResponseEntity<Object>(e.getMessage(), HttpStatus.BAD_REQUEST);
41         }
42     }
43
44     @RequestMapping(value =("/{id}/follows")
45     public ResponseEntity<Object> showFollowedUsers(@PathVariable Long id) {
46         try {
47             return new ResponseEntity<Object>(service.getFollowedUsers(id), HttpStatus.CREATED);
48         } catch (Exception e) {
49             return new ResponseEntity<Object>(e.getMessage(), HttpStatus.BAD_REQUEST);
50         }
51     }
52
53     @RequestMapping(value =("/{id}/follows/{followId}", method = RequestMethod.POST)
54     public ResponseEntity<Object> follow(@PathVariable Long id, @PathVariable Long followId) {
55         try {
56             return new ResponseEntity<Object>(service.follow(id, followId), HttpStatus.CREATED);
57         } catch (Exception e) {
58             return new ResponseEntity<Object>(e.getMessage(), HttpStatus.BAD_REQUEST);
59         }
60     }
61
62     @RequestMapping(value =("/{id}/profilePicture", method = RequestMethod.POST)
63     public ResponseEntity<Object> singleFileUpload(@PathVariable Long id, @RequestParam("file") MultipartFile file) {
64         if (file.isEmpty()) {
65             return new ResponseEntity<Object>("Please upload a file.", HttpStatus.BAD_REQUEST);
66         }
67
68         try {
69             String url = UPLOADED_FOLDER + file.getOriginalFilename();
70             byte[] bytes = file.getBytes();
71             Path path = (Path) Paths.get(url);
72             Files.write(path, bytes);
73             return new ResponseEntity<Object>(service.updateProfilePicture(id, url), HttpStatus.CREATED);
74         } catch (Exception e) {
75             return new ResponseEntity<Object>(e.getMessage(), HttpStatus.INTERNAL_SERVER_ERROR);
76         }
77     }
78
79
80 }
81

```

Comment.java X

```
1 package com.promineotech.socialMediaApi.entity;
2
3 import java.util.Date;
13
14 @Entity
15 public class Comment {
16
17     private Long id;
18     private String content;
19     private Date date;
20     private User user;
21
22     @JsonIgnore
23     private Post post;
24
25     @Id
26     @GeneratedValue(strategy = GenerationType.AUTO)
27     public Long getId() {
28         return id;
29     }
30
31     public void setId(Long id) {
32         this.id = id;
33     }
34
35     public String getContent() {
36         return content;
37     }
38
39     public void setContent(String content) {
40         this.content = content;
41     }
42
43     public Date getDate() {
44         return date;
45     }
46
47     public void setDate(Date date) {
48         this.date = date;
49     }
50
```

```
51     @ManyToOne
52     @JoinColumn(name = "postID")
53     public Post getPost() {
54         return post;
55     }
56
57     public void setPost(Post post) {
58         this.post = post;
59     }
60
61     @ManyToOne
62     @JoinColumn(name = "userId")
63     public User getUser() {
64         return user;
65     }
66
67     public void setUser(User user ) {
68         this.user = user;
69     }
70 }
71
```

```

Postjava X
1 package com.promineotech.socialMediaApi.entity;
2
3 import java.util.Date;
13
14 @Entity
15 public class Post {
16
17     private Long id;
18     private String content;
19     private Date date;
20     private User user;
21     private Set<Comment> comments;
22
23     @Id
24     @GeneratedValue(strategy = GenerationType.AUTO)
25     public Long getId() {
26         return id;
27     }
28
29     public void setId(Long id) {
30         this.id = id;
31     }
32
33     public String getContent() {
34         return content;
35     }
36
37     public void setContent(String content) {
38         this.content = content;
39     }
40
41     public Date getDate() {
42         return date;
43     }
44
45     public void setDate(Date date) {
46         this.date = date;
47     }
48
49     @ManyToOne
50     @JoinColumn(name = "userId")
51     public User getUser() {
52         return user;
53     }
54
55     public void setUser(User user) {
56         this.user = user;
57     }
58
59     @OneToMany(mappedBy = "post")
60     public Set<Comment> getComments() {
61         return comments;
62     }
63
64     public void setComments(Set<Comment> comments) {
65         this.comments = comments;
66     }
67
68
69 }
70

```

```
User.java x
1 package com.promineotech.socialMediaApi.entity;
2
3 import java.util.Set;
17
18 @Entity
19 public class User {
20
21     private Long id;
22     private String username;
23     private String profilePictureUrl;
24
25     @JsonIgnore
26     private Set<User> following;
27     private String password;
28
29     @JsonIgnore
30     private Set<Post> posts;
31
32     @JsonIgnore
33     private Set<Comment> comments;
34
35     @Id
36     @GeneratedValue(strategy = GenerationType.AUTO)
37     public Long getID() {
38         return id;
39     }
40
41     public void setID(Long id) {
42         this.id = id;
43     }
44
45     public String getUsername() {
46         return username;
47     }
48
49     public void setUsername(String username) {
50         this.username = username;
51     }
52
```



```

53  @JsonIgnore
54  public String getPassword() {
55      return password;
56  }
57
58  @JsonProperty
59  public void setPassword(String password) {
60      this.password = password;
61  }
62
63  @OneToMany(mappedBy = "user")
64  public Set<Post> getPosts() {
65      return posts;
66  }
67
68  public void setPosts(Set<Post> posts) {
69      this.posts = posts;
70  }
71
72  @OneToMany (mappedBy = "user")
73  public Set<Comment> getComments() {
74      return comments;
75  }
76
77  public void setComments(Set<Comment> comments) {
78      this.comments = comments;
79  }
80
81  @ManyToMany(cascade = CascadeType.ALL)
82  @JoinTable(name = "following",
83      joinColumns = @JoinColumn(name = "userId", referencedColumnName = "id"),
84      inverseJoinColumns = @JoinColumn(name = "followingId", referencedColumnName = "id"))
85  public Set<User> getFollowing() {
86      return following;
87  }
88
89  public void setFollowing(Set<User> following) {
90      this.following = following;
91  }
92
93  public String getProfilePictureUrl() {
94      return profilePictureUrl;
95  }
96
97  public void setProfilePictureUrl(String profilePictureUrl) {
98      this.profilePictureUrl = profilePictureUrl;
99  }
100 }
101

```

```
CommentRepository.java X
1 package com.promineotech.socialMediaApi.repository;
2
3+ import org.springframework.data.repository.CrudRepository;
6
7 public interface CommentRepository extends CrudRepository<Comment, Long> {
8
9 }
10
11
```

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

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

```
CommentService.java X
1 package com.promineotech.socialMediaApi.service;
2
3 import java.util.Date;
4
5 @Service
6 public class CommentService {
7
8     @Autowired
9     private CommentRepository repo;
10
11     @Autowired
12     private PostRepository postRepo;
13
14     @Autowired
15     private UserRepository userRepo;
16
17     public Comment createComment(Comment comment, Long userId, Long postId) throws Exception {
18         User user = userRepo.findOne(userId);
19         Post post = postRepo.findOne(postId);
20         if (user == null || post == null) {
21             throw new Exception("User or Post does not exist.");
22         }
23         comment.setDate(new Date());
24         comment.setUser(user);
25         comment.setPost(post);
26         return repo.save(comment);
27     }
28
29     public void deleteComment(Long commentId) {
30         repo.delete(commentId);
31     }
32 }
33
34
35
36
37
38
39
40
41
42
43
44
```

```

PostService.java X
1 package com.promineotech.socialMediaApi.service;
2
3 import java.util.Date;
4
12
13 @Service
14 public class PostService {
15
16     @Autowired
17     private PostRepository repo;
18
19     @Autowired
20     private UserRepository userRepo;
21
22     public Iterable<Post> getAllPosts() {
23         return repo.findAll();
24     }
25
26     public Post getPost(Long id) {
27         return repo.findOne(id);
28     }
29
30     public Post updatePost(Post post, Long id) throws Exception {
31         Post foundPost = repo.findOne(id);
32         if (foundPost == null) {
33             throw new Exception("Post not found.");
34         }
35         foundPost.setContent(post.getContent());
36         return repo.save(foundPost);
37     }
38
39     public Post PostService(Post post, Long id) throws Exception {
40         Post foundPost = repo.findOne(id);
41         if (foundPost == null) {
42             throw new Exception("Post not found.");
43         }
44         foundPost.setContent(post.getContent());
45         return repo.save(foundPost);
46     }
47
48     public Post createPost(Post post, Long userId) throws Exception {
49         User user = userRepo.findOne(userId);
50         if (user == null) {
51             throw new Exception("User not found.");
52         }
53         post.setDate(new Date());
54         post.setUser(user);
55         return repo.save(post);
56     }
57
58 }
59

```

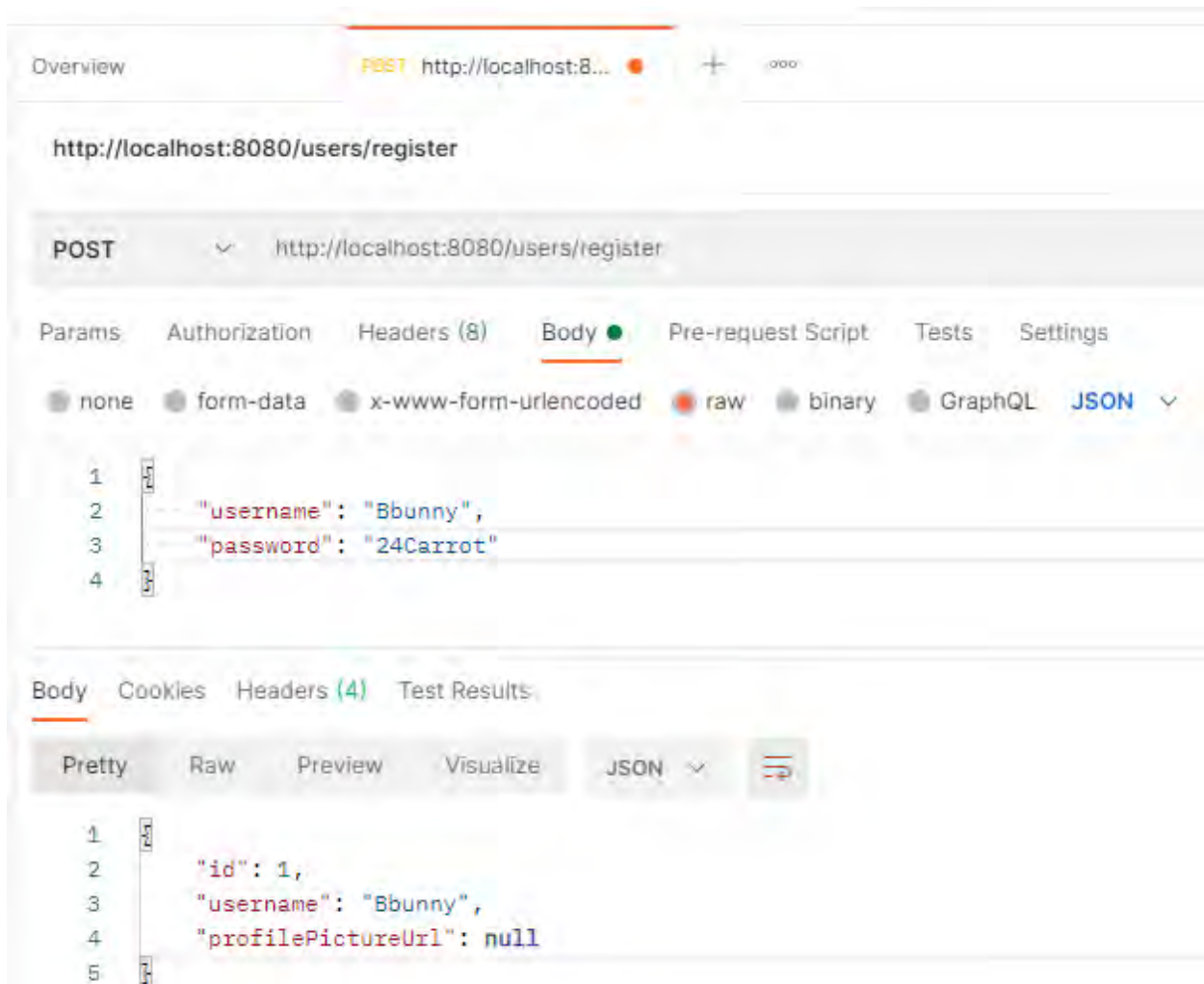
UserService.java X

```
1 package com.promineotech.socialMediaApi.service;
2
3
4 import org.springframework.beans.factory.annotation.Autowired;
10
11 @Service
12 public class UserService {
13
14     @Autowired
15     private UserRepository repo;
16
17     public User createUser(User user) {
18         return repo.save(user);
19     }
20
21     public User login(User user) throws Exception {
22         User foundUser = (User) repo.findByUsername(user.getUsername());
23         if (foundUser != null && foundUser.getPassword().equals(user.getPassword())) {
24             return foundUser;
25         } else {
26             throw new Exception("Invalid username or password.");
27         }
28     }
29
30     public Following follow(Long userId, Long followId) throws Exception {
31         User user = repo.findOne(userId);
32         User follow = repo.findOne(followId);
33         if (user == null || follow == null) {
34             throw new Exception("User does not exist.");
35         }
36         user.getFollowing().add(follow);
37         repo.save(user);
38         return new Following(user);
39     }
40
41     public Following getFollowedUsers(Long userId) throws Exception {
42         User user = (User) repo.findOne(userId);
43         if (user == null) {
44             throw new Exception("User does not exist.");
45         }
46         return new Following(user);
47     }
48
49     public User updateProfilePicture(Long userId, String url) throws Exception {
50         User user = (User) repo.findOne(userId);
51         if (user == null) {
52             throw new Exception("User does not exist.");
53         }
54         user.setProfilePictureUrl(url);
55         return repo.save(user);
56     }
57 }
58
```

```
Following.java X
1 package com.promineotech.socialMediaApi.view;
2
3 import java.util.Set;
4
5
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
```

```
application.properties X
1 spring.datasource.url=jdbc:mysql://localhost/social_api
2 spring.datasource.username=root
3 spring.datasource.password=xxxxxxxxx1
4 spring.datasource.driver-class-name=com.mysql.jdbc.Driver
5
6 spring.jpa.hibernate.ddl-auto=create
7 spring.jpa.show-sql=true
8 spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL5Dialect
9
10 multipart.max-file-size=10MB
11 multipart.max-request-size=10MB
```

Screenshots of Running Application:



Overview POST http://localhost:8... + ...

http://localhost:8080/users/register

POST http://localhost:8080/users/register

Params Authorization Headers (8) Body Pre-request Script Tests Settings

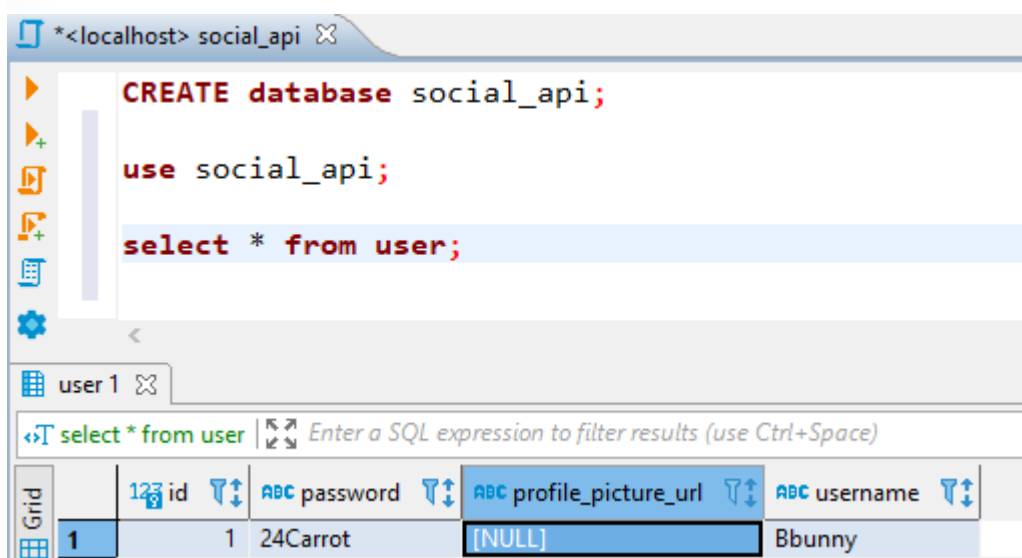
none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {
2   "username": "Bbunny",
3   "password": "24Carrot"
4 }
```

Body Cookies Headers (4) Test Results

Pretty Raw Preview Visualize JSON

```
1 {
2   "id": 1,
3   "username": "Bbunny",
4   "profilePictureUrl": null
5 }
```



*<localhost> social_api

```
CREATE database social_api;
use social_api;
select * from user;
```

user 1

select * from user Enter a SQL expression to filter results (use Ctrl+Space)

Grid	id	password	profile_picture_url	username
1	1	24Carrot	[NULL]	Bbunny

http://localhost:8080/users/register

POST http://localhost:8080/users/register

Params Authorization Headers (8) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {
2   "username": "Efudd",
3   "password": "Getwabbits"
4 }
```

Body Cookies Headers (4) Test Results

Pretty Raw Preview Visualize JSON

```
1 {
2   "id": 2,
3   "username": "Efudd",
4   "profilePictureUrl": null
5 }
```

*localhost social_api <none> SocialMediaDB

CREATE database social_api;

use social_api;

select * from user;

user 1

select * from user Enter a SQL expression to filter results (use Ctrl+Space)

	id	password	profile_picture_url	username
1	1	24Carrot	[NULL]	Bbunny
2	2	Getwabbits	[NULL]	Efudd

Overview

POST http://localhost:8080/users/1/follows/2

http://localhost:8080/users/1/follows/2

POST http://localhost:8080/users/1/follows/2

Params Authorization Headers (8) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {
2   "username": "Efudd",
3   "password": "Getwabbits"
4 }
```

Body Cookies Headers (4) Test Results

Pretty Raw Preview Visualize JSON

```
1 {
2   "following": [
3     {
4       "id": 2,
5       "username": "Efudd",
6       "profilePictureUrl": null
7     }
8   ]
9 }
```

http://localhost:8080/users/1/follows/


GET http://localhost:8080/users/1/follows/

Params Authorization Headers (6) Body Pre-request Script Tests Settings

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL **JSON** ▾

1

Body Cookies Headers (4) Test Results

Pretty Raw Preview Visualize **JSON** ▾ 

```
1  [
2    "following": [
3      {
4        "id": 2,
5        "username": "Efudd",
6        "profilePictureUrl": null
7      }
8    ]
9  ]
```

http://localhost:8080/users/1/profilePicture

POST http://localhost:8080/users/1/profilePicture

Params Authorization Headers (8) **Body** Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL

KEY

VALUE



file

BuggsBunny_ProfilePic.jpg X

Key

Value

Body Cookies Headers (4) Test Results

Pretty

Raw

Preview

Visualize

JSON



```
1  {
2    "id": 1,
3    "username": "Bbunny",
4    "profilePictureUrl": "./pictures/BuggsBunny_ProfilePic.jpg"
5  }
```

http://localhost:8080/users/1/posts

POST http://localhost:8080/users/1/posts

Params Authorization Headers (8) **Body** Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1  {
2    "content": "Here I go with the timid little woodland creature bit again. It's shameful, but...ehhh, it's a living."
3  }
```

Body Cookies Headers (4) Test Results

Pretty

Raw

Preview

Visualize

JSON



```
1  {
2    "id": 4,
3    "content": "Here I go with the timid little woodland creature bit again. It's shameful, but...ehhh, it's a living.",
4    "date": 1616110787544,
5    "user": {
6      "id": 1,
7      "username": "Bbunny",
8      "profilePictureUrl": "./pictures/BuggsBunny_ProfilePic.jpg"
9    },
10   "comments": null
11 }
```

http://localhost:8080/users/1/posts

GET http://localhost:8080/users/1/posts

Params Authorization Headers (8) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1
2 "content": "Here I go with the timid little woodland creature bit again. It's shameful, but...ehhh, it's a living."
3
```

Body Cookies Headers (4) Test Results

Pretty Raw Preview Visualize JSON

```
1
2 {
3   "id": 4,
4   "content": "Here I go with the timid little woodland creature bit again. It's shameful, but...ehhh, it's a living.",
5   "date": 1616110788000,
6   "user": {
7     "id": 1,
8     "username": "Bbunny",
9     "profilePictureUrl": "./pictures/BuggsBunny_ProfilePic.jpg"
10  },
11   "comments": []
12 }
13
```

http://localhost:8080/users/1/posts/4

GET http://localhost:8080/users/1/posts/4

Params Authorization Headers (6) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

1

Body Cookies Headers (4) Test Results

Pretty Raw Preview Visualize JSON

```
1
2 "id": 4,
3 "content": "Here I go with the timid little woodland creature bit again. It's shameful, but...ehhh, it's a living.",
4 "date": 1616110788000,
5 "user": {
6   "id": 1,
7   "username": "Bbunny",
8   "profilePictureUrl": "./pictures/BuggsBunny_ProfilePic.jpg"
9 },
10 "comments": []
11
```

http://localhost:8080/users/1/posts/4

PUT http://localhost:8080/users/1/posts/4

Params Authorization Headers (8) **Body** Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL **JSON**

```
1 {
2   "content": "Post Update... 'The way I run this thing you'd think I knew something about it.'"
3 }
```

Body Cookies Headers (4) Test Results

Pretty Raw Preview Visualize JSON

```
1 {
2   "id": 4,
3   "content": "Post Update... 'The way I run this thing you'd think I knew something about it.'",
4   "date": 1616110788000,
5   "user": {
6     "id": 1,
7     "username": "Bbunny",
8     "profilePictureUrl": "/pictures/BuggsBunny_ProfilePic.jpg"
9   },
10  "comments": []
11 }
```

http://localhost:8080/users/2/posts/4/comments

POST http://localhost:8080/users/2/posts/4/comments

Params Authorization Headers (8) **Body** Pre-request Script Tests Settings

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL **JSON** ▾

```
1 {
2   "content": "'Hewwo! Acme Pest Contwol? Weww I have a pest I want contwollled.'"
3 }
```

Body Cookies Headers (4) Test Results

Pretty Raw Preview Visualize **JSON** ▾ 

```
1 {
2   "id": 1,
3   "content": "'Hewwo! Acme Pest Contwol? Weww I have a pest I want contwollled.'",
4   "date": 1616111639959,
5   "user": {
6     "id": 2,
7     "username": "Efudd",
8     "profilePictureUrl": null
9   }
10 }
```

http://localhost:8080/users/1/posts

GET http://localhost:8080/users/1/posts

Params Authorization Headers (6) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

1

Body Cookies Headers (4) Test Results

Pretty Raw Preview Visualize JSON

```
1 {
2   {
3     "id": 4,
4     "content": "Post Update... 'The way I run this thing you'd think I knew something about it.'",
5     "date": 1616110788000,
6     "user": {
7       "id": 1,
8       "username": "Bbunny",
9       "profilePictureUrl": "/pictures/BuggsBunny_ProfilePic.jpg"
10    },
11    "comments": [
12      {
13        "id": 1,
14        "content": "'Hewwo! Acme Pest Contwol? Weww I have a pest I want contwolled.'",
15        "date": 1616111640000,
16        "user": {
17          "id": 2,
18          "username": "Efudd",
19          "profilePictureUrl": null
20        }
21      }
22    ]
23  }
24 }
```

http://localhost:8080/users/1/posts/4/comments/1

DELETE

http://localhost:8080/users/1/posts/4/comments/1

Params

Authorization


Headers (6)

Body

Pre-request Script

Tests

Settings

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL **JSON** 

1

Body

Cookies

Headers (4)

Test Results

Pretty

Raw

Preview

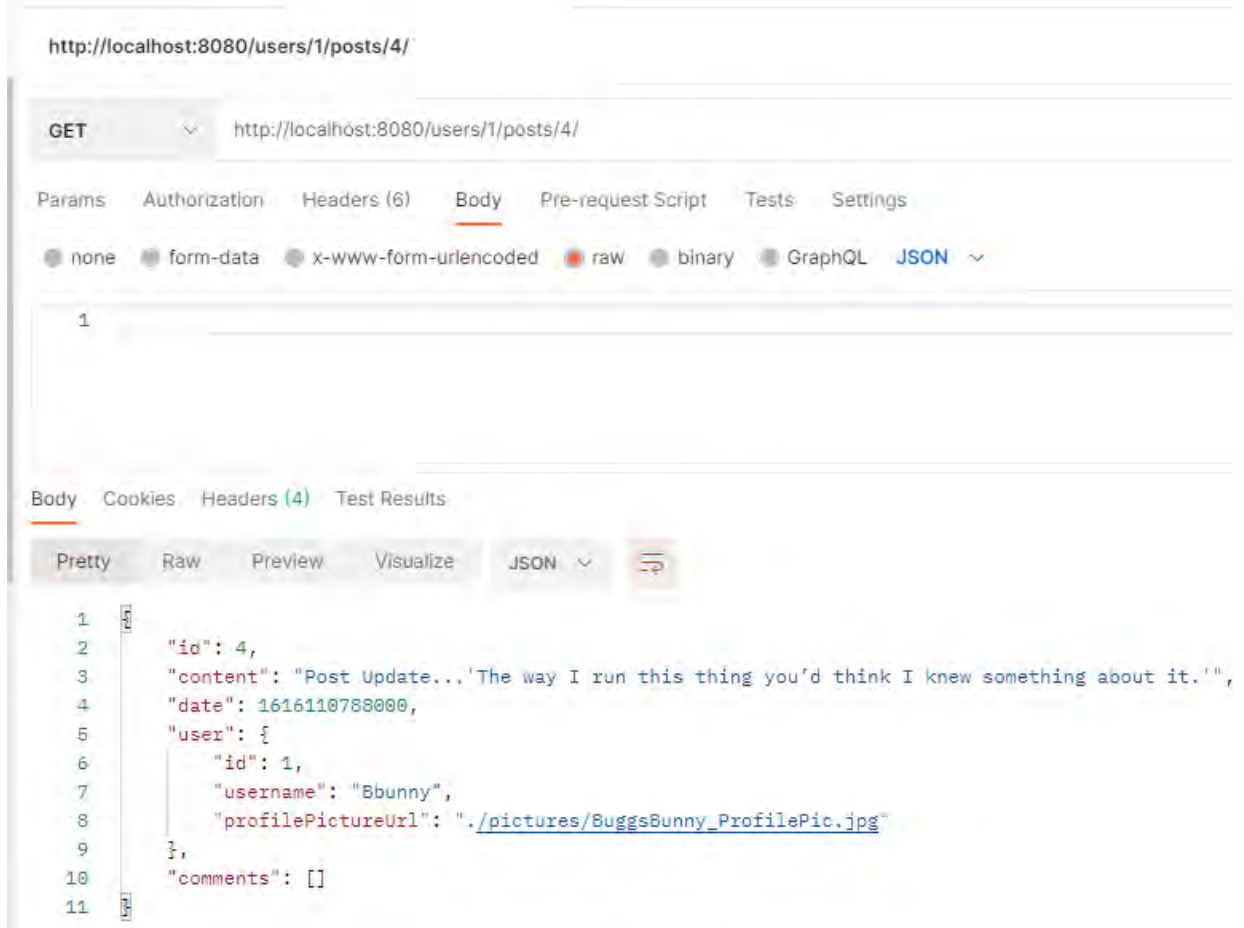
Visualize

Text





1 Deleted comment with id:1



URL to GitHub Repository: https://github.com/becje/Springboot_Coding_Assignments