

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 Running Application: (see below)

(github url at far bottom)

Registering a user:

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": "Tim",
3   ... "password": "12345"
4 }
```

Body Cookies Headers (4) Test Results

Pretty Raw Preview Visualize JSON

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

Proof: (I accidentally made two....)

```
mysql> select * from user;
+----+-----+-----+-----+
| id | password | profile_picture_url | username |
+----+-----+-----+-----+
| 1  | 12345   | NULL                | Tim     |
| 2  | 12345   | NULL                | Tim     |
+----+-----+-----+-----+
2 rows in set (0.00 sec)

mysql>
```

Registering a 2nd User:

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

JSO

```
1 {  
2   ... "username": "Sally",  
3   ... "password": "12345"  
4 }
```

Body

Cookies

Headers (4)

Test Results

Pretty

Raw

Preview

Visualize

JSON

▼



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

Sally follows Tim:

http://localhost:8080/users/1/follows/3

POST

▼

http://localhost:8080/users/1/follows/3

Params

Authorization

Headers (8)

Body ●

Pre-request Script

Tests

Settings

☐ none

☐ form-data

☐ x-www-form-urlencoded

☒ raw

☐ binary

☐ GraphQL

JSC

```
1 {
2   ... "username": "Sally",
3   ... "password": "12345"
4 }
```

Body

Cookies

Headers (4)

Test Results

Pretty

Raw

Preview

Visualize

JSON ▼

```
1 {
2   "following": [
3     {
4       "id": 3,
5       "username": "Sally",
6       "profilePictureUrl": null

```

Find and Replace

Console

Proof of following:

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

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

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

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

```
1 {
2   ... "username": "Sally",
3   ... "password": "12345"
4 }
```

Body Cookies Headers (4) Test Results

Pretty Raw Preview Visualize **JSON**

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

Find and Replace Console

Uploading a profile picture:

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
<input checked="" type="checkbox"/>	file •	WP_20130922_001.jpg × •
	Key	Value



Body Cookies Headers (4) Test Results


Pretty Raw Preview Visualize JSON •


```
1 {
2   "id": 1,
3   "username": "Tim",
4   "profilePictureUrl": "./pictures/WP_20130922_001.jpg" •
5 }
```

Tim's first post:

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": "Hey everyone, this if my first post!"
3  }
```

Body Cookies Headers (4) Test Results

Pretty Raw Preview Visualize JSON  

```
1  {}
2  {
3    "id": 1,
4    "content": "Hey everyone, this if my first post!",
5    "date": 1615502163977,
6    "user": {
7      "id": 1,
```

Get Tim's first post by user:

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": "Hey everyone, this is my first post!"  
3 }
```

Body

Cookies

Headers (4)

Test Results

Pretty

Raw

Preview

Visualize

JSON

```
1 [  
2   {  
3     "id": 1,  
4     "content": "Hey everyone, this is my first post!",  
5     "date": 1615502164000,  
6     "user": {  
7       "id": 1.
```


Get Tim's first post by post:

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

The screenshot shows a REST client interface with a GET request to `http://localhost:8080/users/1/posts/1`. The request is configured with the method `GET` and the URL. The response is displayed in the `Body` tab, showing a JSON object with the following structure:

```
1 {  
2   "id": 1,  
3   "content": "Hey everyone, this if my first post!",  
4   "date": 1615502164000,  
5   "user": {  
6     "id": 1,  
7     "username": "Tim".  
}
```

Updating Tim's first post:

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

PUT

▼

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

Params

Authorization

Headers (8)

Body ●

Pre-request Script

Tests

Settings

☐ none

☐ form-data

☐ x-www-form-urlencoded

☒ raw

☐ binary

☐ GraphQL

JSON ▼

1

2

3

...

"content": "I am updating my first post now!"

Body

Cookies

Headers (4)

Test Results

Pretty

Raw

Preview

Visualize

JSON ▼

↻

1

2

3

4

5

6

7

...

"id": 1,

"content": "I am updating my first post now!",

"date": 1615502164000,

"user": {

"id": 1,

"username": "Tim".

Sally comments on Tim's post:

http://localhost:8080/users/3/posts/1/comments

POST

http://localhost:8080/users/3/posts/1/comments

ParamsAuthorizationHeaders (8)BodyPre-request ScriptTestsSettings

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

JSON

1

2

3

Body

Cookies

Headers (4)

Test Results

Pretty

Raw

Preview

Visualize

JSON

1

2

3

4

5

6

7

"id": 1,

"content": "Commenting on this post!",

"date": 1615502447772,

"user": {

"id": 3,

"username": "Sally".

Proof of comment:

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

The screenshot displays a REST client interface. At the top, the method is set to **GET** and the URL is `http://localhost:8080/users/1/posts`. Below the URL bar, tabs for **Params**, **Authorization**, **Headers (8)**, **Body**, **Pre-request Script**, **Tests**, and **Settings** are visible. The **Body** tab is selected, and the format is set to **JSON**. The body content is a JSON object with a `comments` array containing one comment object. The interface includes a line number column on the left and a 'Pretty' button to format the JSON.

```
1 {
2   ... "content": "Commenting on this post!"
3 }
```

body Cookies Headers (4) Test Results

Pretty Raw Preview Visualize JSON

```
7   "id": 1,
8   "username": "Tim",
9   "profilePictureUrl": "./pictures/WP_20130922_001.jpg"
10 },
11 "comments": [
12   {
13     "id": 1,
14     "content": "Commenting on this post!",
15     "date": 1615502448000,
16     "user": {
```

Deleting comment:

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

DELETE

▼

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

Params

Authorization

Headers (8)

Body ●

Pre-request Script

Tests

Settings

☐ none

☐ form-data

☐ x-www-form-urlencoded

☒ raw

☐ binary

☐ GraphQL

JSON ▼

1

2

3

```
1 {}
2 {
3   ... "content": "Commenting on this post!"
4 }
```

Body

Cookies

Headers (4)

Test Results

Pretty

Raw

Preview

Visualize

Text ▼

● 1 Deleted comment with id:1 ●

Proof of deletion:

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

GET

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

JSON

▼

2

id : 1,

3

"content": "Hey everyone, this is my first post!",

4

"date": 1615574852000,

5

"user": {

6

id: 1,

7

username: "Tim",

8

profilePictureUrl: "./pictures/WP_20130922_001.jpg"

9

},

10

• "comments": [] •

11

}

Attempt to login:

The screenshot shows a REST client interface with the following details:

- URL:** `http://localhost:8080/users/login`
- Method:** `POST`
- Body Type:** `raw` (selected)
- Request Body (JSON):**

```
1 {  
2   "username": "Tim",  
3   "password": "12345"  
4 }
```
- Response Body (JSON):**

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

URL to GitHub Repository:

<https://github.com/wlindstrom55/Spring-wk1-assignments>