CSC 301 Assignment 2

1 Objective

- Practice Enterprise Design Patterns, specifically dependency injection
- Practice NoSQL databases, specifically Mongo DB
- Experience with Git Flow

2 How to submit your work

- Your work will be submitted through Github Classrooms
- You may work in teams of two.
- First team member: please accept invitation by clicking here, then invite your team mate.
- Next team member: wait for your other team mate to invite you. Once invited, join the team.
- Ensure your final draft is on your master branch before the assignment deadline.
- Specifically, your `src` folder and `pom.xml` must be directly in your main folder of your repository.
- In the same place, pease add your `team.md` file, listing all team members utorid's.

3 Description

For this assignment, you will implement a simple blog post api. You will be required to use a specific dependency injection framework (Google Dagger2) and Mongo DB.

4 Starter Files

When you first run the starter code, you will encounter a NullPointerException. This is caused by the lack of dependency injection which you are required to implement. Start by taking a look at `DaggerModule.java` to see what you need to implement. Also you might use the dagger 2 class example, available on the course web site as a reference.

5 Project/IDE Setup

5.1 Command Line

- Install maven
- To compile your code simply run `mvn compile` in the root directory (the folder that has `pom.xml`)
- To run your code run `mvn exec:java`

5.2 Eclipse

- File → Import → Maven → Existing Maven Projects
- Navigate to the project location
- Right click the project → Maven Build...
- Input `compile exec:java` in the Goals input box
- Click apply then run/close
- You can now run the project by pressing the green play button

5.3 Intellij

- Import project
- Navigate to the project location
- Import project from external model → Maven
- Next → Next → Next → Finish
- Add Configuration → + Button → Maven
- Name the configuration and input `exec:java` in the Command Line box
- Apply → Ok
- You can now run the project by pressing the green play button

6 API Requirements

You are required to implement the following REST API endpoints. You should be familiar with REST from A1 and the REST tutorial. Feel free to refer to the following for reference: link.

6.1 `PUT /api/v1/post`

Add a post to the blog (stored in mongodb). Returns the `_id` from mongodb. If any extra attributes are provided in the request body, they should not be stored in the database. However, the request should still succeed provided all the required attrbutes are present (200 OK).

Request Body:

```
{
    "title": "My First Post",
    "author": "Jake Peralta",
    "content": "This is my first post. I’m very excited to create this blog. YAY!",
    "tags": [
        "first post",
        "new blog",
        "excitement"
    ]
}
```

Response:

- `200 OK` for a successful add
- `400 BAD REQUEST` if the request body is improperly formatted or missing required information
- `500 INTERNAL SERVER ERROR` if save or add was unsuccessful (if user has done everything as per requirements but request is unable to respond correctly)
- `405 METHOD NOT ALLOWED` if called with something other than GET, PUT, DELETE

Response Body:

```
{
    "_id": "3m4jk393jneke93h3k"
}
```

6.2 GET /api/v1/post

Get blog post. Returns list of corresponding posts. If `_id` is specified, return the exact post that has that `_id`. If `title` is specified, return all posts that contain the specified words in the given title, in order.

6.2.1 Acceptable Request Bodies

Request Body:

```
{
    "title": "First"
}
```

Request Body:

```
{
    "_id": "3m4jk393jneke93h3k"
}
```

Request Body:

```
{
    "title": "First",
    "_id": "3m4jk393jneke93h3k"
}
```

Note: In the above case, `_id` takes priority.

6.2.2 Corresponding Response Bodies

Response Body:

```
[
    {
        "_id": {
            "$oid": "5c705827ae58ed40863c8584"
        },
        "title": "First Day of School",
        "tags": [
            "school",
            "excitement",
            "fun"
        ],
        "author": "Jake Peralta",
        "content": "Fun!"
    },
    {
        "_id": {
            "$oid": "3m4jk393jneke93h3k"
        },
        "title": "My First Post",
        "author": "Jake Peralta",
        "content": "This is my first post. I’m very excited to create this blog. YAY!",
        "tags": [
            "first post",
            "new blog",
            "excitement"
        ]
    }
]
```

Response Body:

```
    [
        {
            "_id": {
                "$oid": "3m4jk393jneke93h3k"
            },
            "title": "My First Post",
            "author": "Jake Peralta",
            "content": "This is my first post. I’m very excited to create this blog. YAY!",
            "tags": [
                "first post",
                "new blog",
                "excitement"
            ]
        }
    ]
```

Response Body:

```
    [
        {
            "_id": {
                "$oid": "3m4jk393jneke93h3k"
            },
            "title": "My First Post",
            "author": "Jake Peralta",
            "content": "This is my first post. I’m very excited to create this blog. YAY!",
            "tags": [
                "first post",
                "new blog",
                "excitement"
            ]
        }
    ]
```

Response:

- `200 OK` if the post exists
- `400 BAD REQUEST` if the request body is improperly formatted or missing required information
- `500 INTERNAL SERVER ERROR` if save or add was unsuccessful (if user has done everything as per requirements but request is unable to respond correctly)
- `404 NOT FOUND` if post does not exist
- `405 METHOD NOT ALLOWED` if called with something other than `GET, PUT, DELETE`

6.3 `DELETE /api/v1/post`

Delete a blog post by `_id`

Request Body:

```
{
   "_id": "3m4jk393jneke93h3k"
}
```

Response: (no response body)

- `200 OK` for successful delete
- `400 BAD REQUEST` if the request body is improperly formatted or missing required information
- `500 INTERNAL SERVER ERROR` if save or add was unsuccessful (if user has done everything as per requirements but request is unable to respond correctly)
- `404 NOT FOUND` if post does not exist
- `405 METHOD NOT ALLOWED` if called with something other than GET, PUT, DELETE

7 Dependency Injection Requirements

You are required to use dependency injection for this assignment. You may not pass in any classes as arguments to a constructor for any of your classes, you should be using dependency injection instead. This does not mean you should be writing all your code in one file to avoid using Dagger. Good coding practices are still expected.

You will be using Dagger2. You can use the following resources for more information.

- Dependency Injection: link.
- Dagger2: link.

8 Database Requirements

- You are required to use MongoDB for this assignment
- We will be using mongo version 4.0 as a major stable release. Link here.
- You may also consider instaling compass as a means of visulaizing your db transactions, although it is not required for this assignment.
- Please do not change the default port and the default mongo credentials.

- The mongodb URL must be `localhost`.
- Your database should be called `csc301a2`.
- Your collection should be called `posts`.
- The records mirror the format of the `PUT` body.

9 Git Requirements

You are required to use git flow for this assignment. You will be evaluated on the correct usage of the types of branches. Use your tutorial handout as reference.

10 Testing Your Code

10.1 Run Your Code

Refer to Section 5 to run your code in your IDE of preference.

10.2 Hitting your Endpoints

10.2.1 CLI

From the command line the best way to test your endpoints is using curl.

```
curl -X POST http://localhost:8080/api/v1/endpoint/ --data \
     '{ "key": "value", "other": "thing" }'
```

- The `-X` flag specifies the REST action you wish to use.
- And the `--data` flag specifies the body of the request

10.2.2 Python

You can use python to programatically call your endpoints if you so desire.

```
import requests
d = { "key": "value" }
requests.put('http://localhost:8080/api/v1/endpoint', json=d)
```

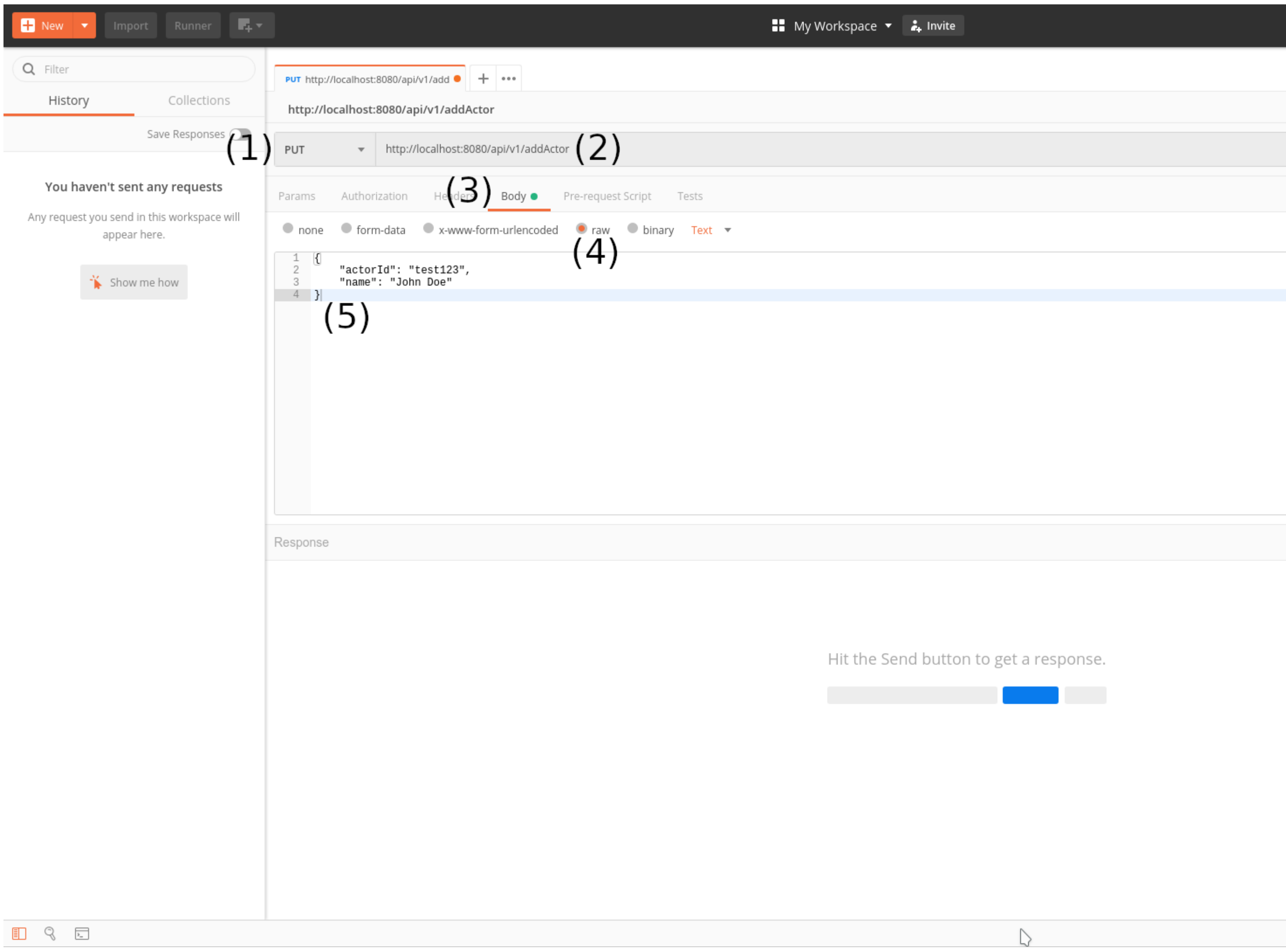
- The dictionary is the json body you want to send.
- There are also methods for `requests.get(...)` and `requests.post(...)`.

10.2.3 Postman

Postman is a nice tool to use to test your endpoints. It can be downloaded from this link.

1. Sending a Request
   - Open up Postman and select the type of request you would like to send (1)
   - Enter in your request url (2)
   - Open up the request body tab (3) and select `raw` (4)

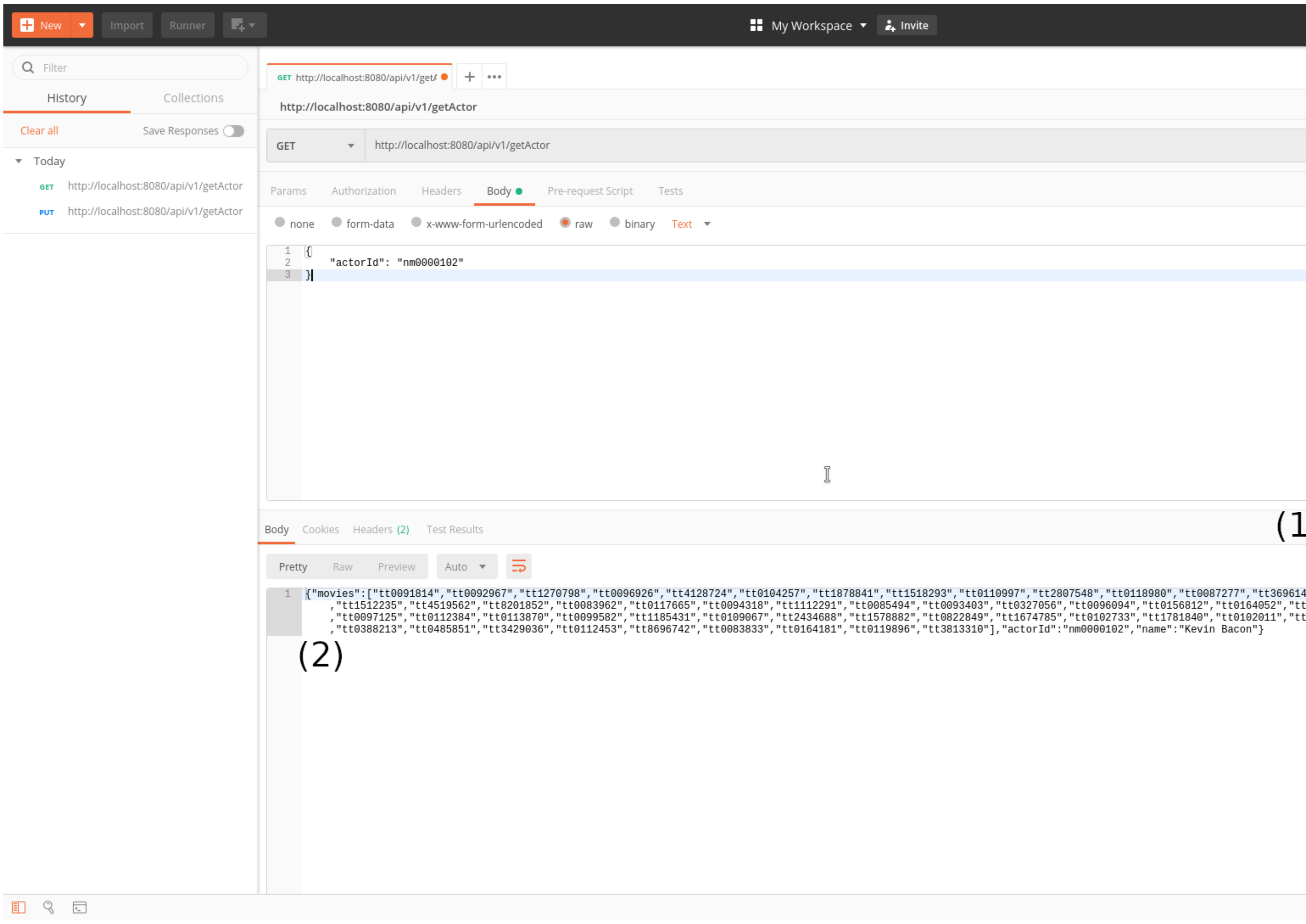
- Enter in your request body data into the test box (5)
- Click `Send` (6)



New
Import
Runner
My Workspace
Invite
Filter
History
Collections
Save Responses
You haven't sent any requests
Any request you send in this workspace will appear here.
Show me how
PUT http://localhost:8080/api/v1/add
http://localhost:8080/api/v1/addActor
(1)
PUT
http://localhost:8080/api/v1/addActor
(2)
Params
Authorization
Headers
(3)
Body
Pre-request Script
Tests
none
form-data
x-www-form-urlencoded
raw
binary
Text
(4)
(5)
Response
Hit the Send button to get a response.

```
{
    "actorId": "test123",
    "name": "John Doe"
}
```

2. Reading the Response
   - You can see the status of your response (1)
   - You can see the response body (2)
   - Ensure that responses that should not have a body, do not have a body



http://localhost:8080/api/v1/getActor

GET http://localhost:8080/api/v1/getActor

Body: raw [✓] Text

```
{
    "actorId": "nm0000102"
}
```

(1

Body

```
{"movies":["tt0091814","tt0092967","tt1270798","tt0096926","tt4128724","tt0104257","tt1878841","tt1518293","tt0110997","tt2807548","tt0118980","tt0087277","tt369614
    ,"tt1512235","tt4519562","tt8201852","tt0083962","tt0117665","tt0094318","tt1112291","tt0085494","tt0093403","tt0327056","tt0096094","tt0156812","tt0164052","tt
    ,"tt0097125","tt0112384","tt0113870","tt0099582","tt1185431","tt0109067","tt2434688","tt1578882","tt0822849","tt1674785","tt0102733","tt1781840","tt0102011","tt
    ,"tt0388213","tt0485851","tt3429036","tt0112453","tt8696742","tt0083833","tt0164181","tt0119896","tt3813310"],"actorId":"nm0000102","name":"Kevin Bacon"}
```

(2)