Meme Generator Exercise

We will be creating a meme generator using an API.

We will be creating our own API which will use that imgflp meme generator API to create the memes as well as a client which will consume the API we create.

STEP 1

We will first create the server code which will use the imgflip meme generator API to create the memes.

Before you begin, review the API docs at https://imgflip.com/api

STEP 2

The following classes are provided as DTOs for consuming the imgflip meme generator API's `get_memes` endpoint:

```
Meme
MemeApiGetResponse
MemeData
```

The `Meme` class will represent a single meme returned by the `get_memes` endpoint.

The `MemeData` class will hold the data portion of the API response.

The `MemeApiGetResponse` will hold the entire API response for the `get_memes` endpoint.

Your task in this step is to add properties and getters/setters to these classes so that you can consume the API's `get_memes` endpoint

STEP 3

Now that you have the needed DTOs to consume the `get_memes` endpoint, write the code to use the endpoint in the `getMemeList()` method of the `MemeGeneratorService` and use it to create and return a List of `MemeListItem`s The `MemeListItem` class has been provided..

STEP 4

A `MemeController` class has been provided for you. You will need to add a private member to it which will hold an instance of `MemeGeneratorService` and then use dependency injection to inject it.

HINT: You may need to make adjustments to the controller and the service to allow this.

STEP 5

In the `MemeController` create a method which will call the `getMemeList()` method in the `MemeGeneratorService` to return a List of `MemeListItem`s and then wire up this method to be a `GET`

endpoint at `/memes` .

STEP 6

Test your `GET` endpoint using Postman.

STEP 7

The following classes are provided as DTOs for consuming the imgflip meme generator API's `caption_image` endpoint:

```
CaptionedMeme
MemeApiCreateResponse
UrlInfo
```

The `CaptionedMeme` class will be used for POST to the `caption_image` endpoint.

The `UrlInfo` class will hold the data portion of the API response.

The `MemeApiCreateResponse` will hold the entire API response for the `caption_image` endpoint.

Your task in this step is to add properties and getters/setters to these classes so that you can use the API's `caption_image` endpoint (the properties and getters/setters for `CaptionedMeme` have already been provided).

Step 8

We will use the `CreateMemeInfo` class to receive info about the meme to be created from the client consuming OUR API.

Create a method `createMeme` in `MemeGeneratorService` which will take a `CreateMemeInfo` object as a parameter and use it to POST to the `caption_image` endpoint and receive a response. The method will return a `String` .

The imgflip API DOES NOT USE JSON in the POST method.

1. Create a `CaptionedMeme` object using the `CreateMemeInfo` object passed in.
2. Use the `formDataFromCaptionedMeme` in `MemeUtils` to convert the `CaptionedMeme` object you created into A `MultiValueMap<String, String>` which will be used for the POST to the `caption_image` endpoint. You will need to uncomment some code in the `formDataFromCaptionedMeme` method once you have completed the `CaptionedMeme` class.
3. You will need to create headers and an entity to use for the POST. When you create the headers, use `MediaType.APPLICATION_FORM_URLENCODED` rather than `MediaType.APPLICATION_JSON` for the headers content type.
4. POST to the `caption_image` endpoiont using the entity you created.
5. If a meme is returned succesfully return its URL from the method. Otherwise, return `null` .

Step 9

In the `MemeController` create a method which will call the `createMeme()` method in the `MemeGeneratorService` and return a String representing the meme URL.

Wire up this method to be a `POST` endpoint at `/memes` which will receive a `CreateMemeInfo` object as a parameter.

Step 10

Test your POST endpoint using Postman

Step 11

In the client project, wire up the `getMemes()` method in the `MemeGeneratorApiService` class like this:

- Make a `GET` request to `/memes` endpoint of the API you created.
- If successful, have the method return the array of `Meme` objects.
- If not successful, return `null`

Step 12

Run the CLI to see if it shows memes

Step 13

In the client project, wire up the `createMeme(String memeId, String caption)` method in the `MemeGeneratorApiService` class like this:

- Create a `CreateMemeInfo` object and populate it using the params of the method.
- Make a `POST` request to `/memes` endpoint of the API you created using the `CreateMemeInfo` object you created. (NOTE that this API DOES use JSON.)
- If successful, have the method return the String returned by the endppoint
- If not successful, return `null`

STEP 14

Test to see if you can create a captioned meme using the CLI