

A large, semi-transparent red arrow points from left to right, covering the left half of the slide.

**WELCOME BACK
LESSON 8**



**Spring/Spring Boot
Crash Course**
For TD Bank

MEET YOUR CRASH COURSE TEAM



TANGY F.
CEO



HAL M.
DEVELOPER



WILLIAM D
DEVELOPER



LESSON 8

AGENDA

- 1 Recap Lesson of 7
 >>Rapid review of Lesson seven
- 2 Current Lesson- Eureka
- 3 Q&A

SETTING UP YOUR ENVIRONMENT



The screenshot shows a GitHub repository page for 'SWEUpskilling Cre8tivePilot'. The 'Source' branch is selected. The repository contains a '.idea' folder and two 'README.md' files. One file describes a longer assessment involving Spring Boot, with instructions to set a timer for 30 minutes, download a ZIP file, and follow a series of steps to create a ProfileService application. The second 'README.md' file is a 'Longer Pre-assessment' prompt for a 10-minute take-home assessment on Spring Boot, avoiding Google or AI tools.

Prompt:
The purpose of this pre-assessment, is to get an understanding of your knowledge in Spring Boot. The pre-assessment is 30 minutes. We ask that you avoid using Google or AI tools for this part. The instructions to send the code are below.

You will be creating a profile application using your current knowledge of Spring SpringBoot.

1. Set your time for 30 minutes.
2. Download the ZIP file with the project.
3. Create a new repository (using any service you want), and add the downloaded folders into the repository.
4. Open the "ProfileService" project in your IDE.
5. Add an optional dependency for lombok, from the group projectlombok
6. Add a dependency for hibernate-validator version 8.0.0.Final, from the group hibernate-validator
7. Fill out the Profile entity class and AuthRecord entity class using the comments in the shell, showing best practices for efficient programming
8. Fill out the controller class using the comments in the shell, following best practices for 12-factor app
9. Create a MySQL database. Create databases named ProfileData and SongData. Connect those databases to the ProfileService application.
10. Create a test that verifies the REST API endpoint that you created.
11. Run the application.
12. When time stops at 30 minutes, push to the repository.

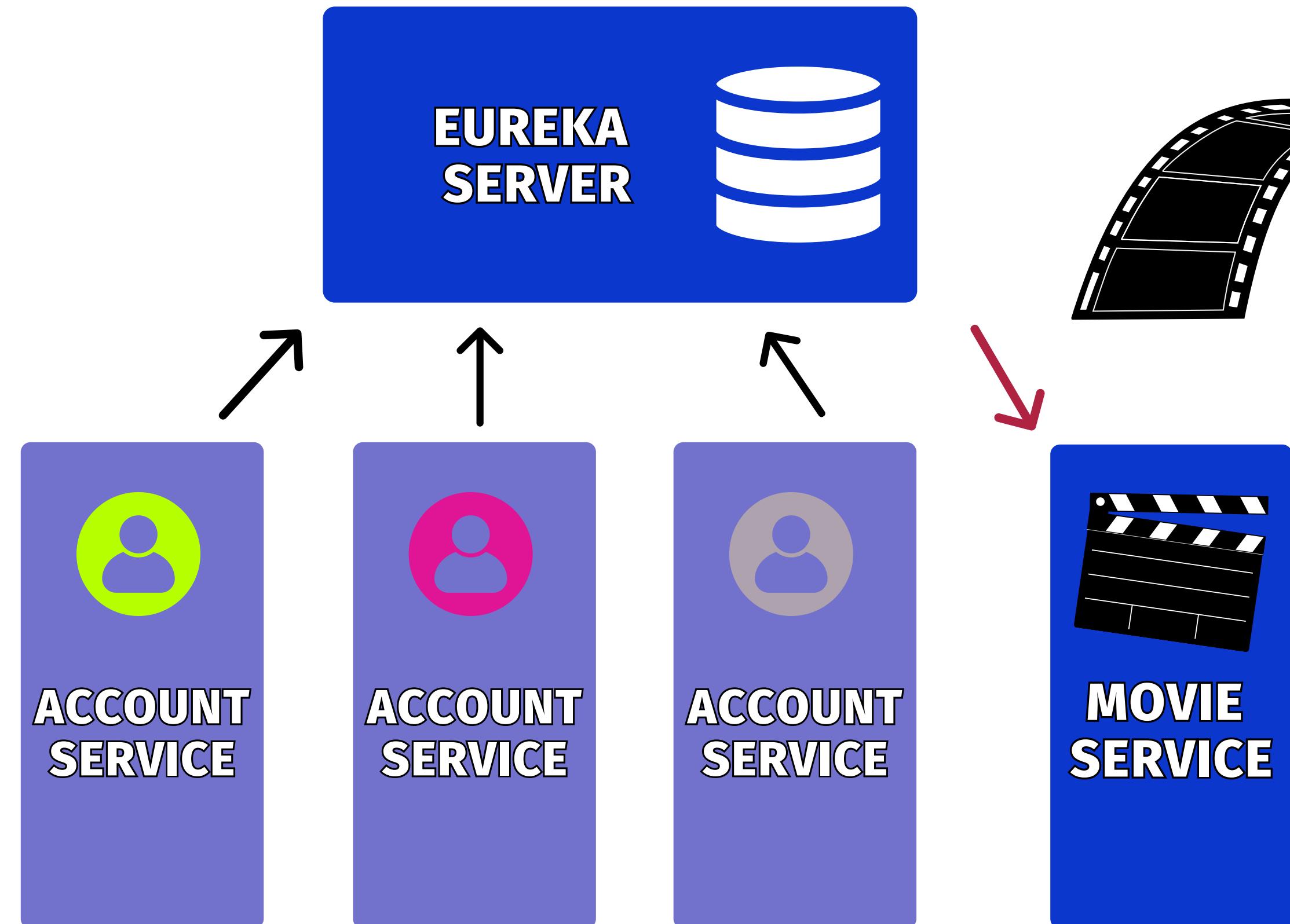
Longer Pre-assessment

During the course you had 10 mins to do a pre assessment. This is a take home assessment that will also help. We look forward to finding you sharing your results

Prompt:
The purpose of this pre-assessment, is to get an understanding of your knowledge in Spring Boot. The pre-assessment is 30 minutes. We ask that you avoid using Google or AI tools for this part. The instructions to send the code are below.

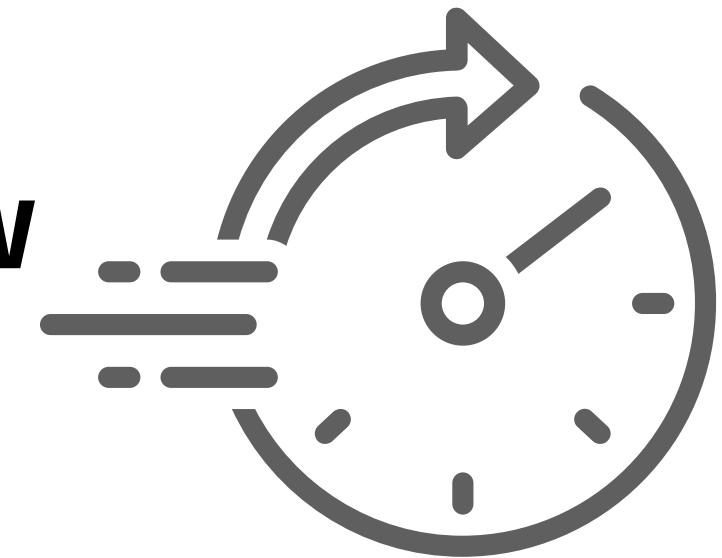
WE ARE BUILDING

BITE SIZE
MOVIE REVIEW APP



<Cre8tive
Software/>
Devs.

WELCOME TO LESSON 8



Rapid Review Trivia

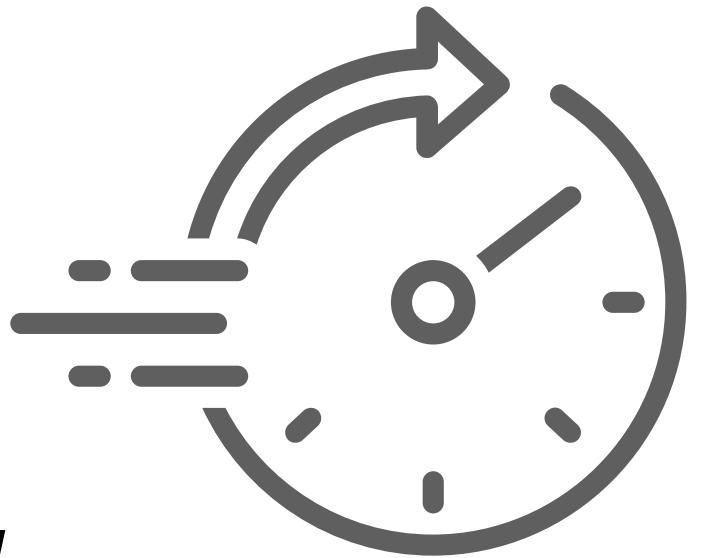


1. Name all seven Registered Claims defined by the JWT specification.

2. What is Mockito used for?

**Rapid Review
of
LESSON
6&7**

WELCOME TO
LESSON 8



Rapid Review
of
LESSON 7

Rapid Review
Unit Testing
&
Hibernate

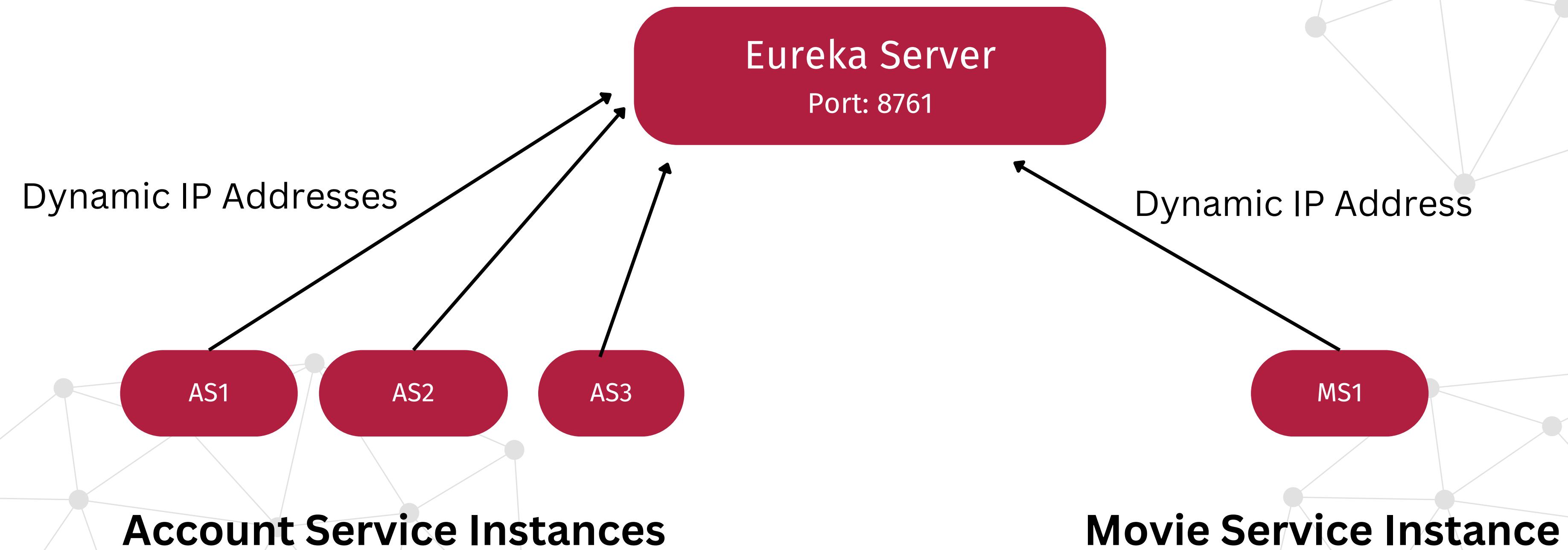
LESSON 8

Eureka Server

LESSON 8

EUREKA SERVER

- 1 What is Eureka Server and what are its components
- 2 Setting up the server and registering the Profile & Song services as Clients
- 3 Advantages/ Necessity of using Eureka Server



Reference
AS1 = Account Service 1
SS1 = Song Service 1

Components of Eureka

- Eureka Server
- Eureka Client
- Load Balancing

Advantages

- Dynamic Scaling
- Load Balancing
- Health Metrics
- Self healing

Alternatives

- Apache ZooKeeper
- Kubernetes

12-FACTOR APP

</> Factor 3: Config

Watch for how we declare the ports for our services in the application.properties files



Factor 8: Concurrency

Take note of how we scale horizontally, how we run multiple instances of Account Service at once, how we use a distributed process manager to keep track of which processes are running at any given time.

CODING EXERCISE #1



Below is the beginning of the application.properties file inside our Eureka server. Write code that prevents our Eureka server from acting as its own client.

```
server.port=8761
//Code Goes Here
```

CODING EXERCISE #1



Below is the beginning of the application.properties file inside our Eureka server. Write code that prevents our Eureka server from acting as its own client.

```
server.port=8761
//Code Goes Here
```

eureka.client.registerWithEureka=false
eureka.client.fetchRegistry=false

CODING EXERCISE #2



Our Movie Service is intended to use our Eureka Server as a discovery service in order to connect with our Account Service clients, automatically choosing an Account Service client for each call to the discovery service. What is wrong with this configuration class in Movie Service?

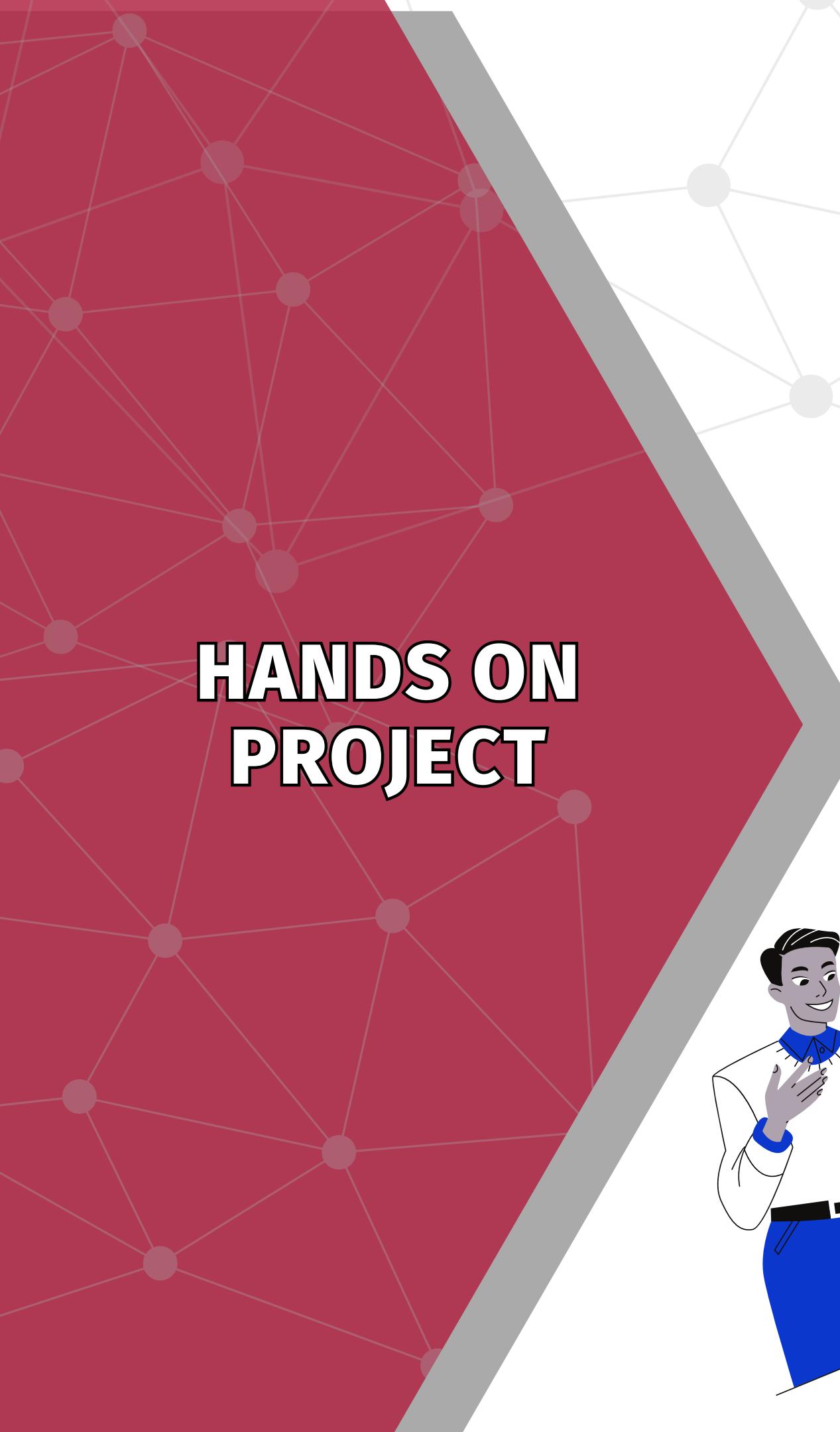
```
@Configuration  
public class BeanConfig {  
  
    @Bean  
    public ModelMapper modelMapper() { return new ModelMapper(); }  
  
    @Bean  
    public RestTemplate restTemplate() { return new RestTemplate(); }  
}
```

CODING EXERCISE #2



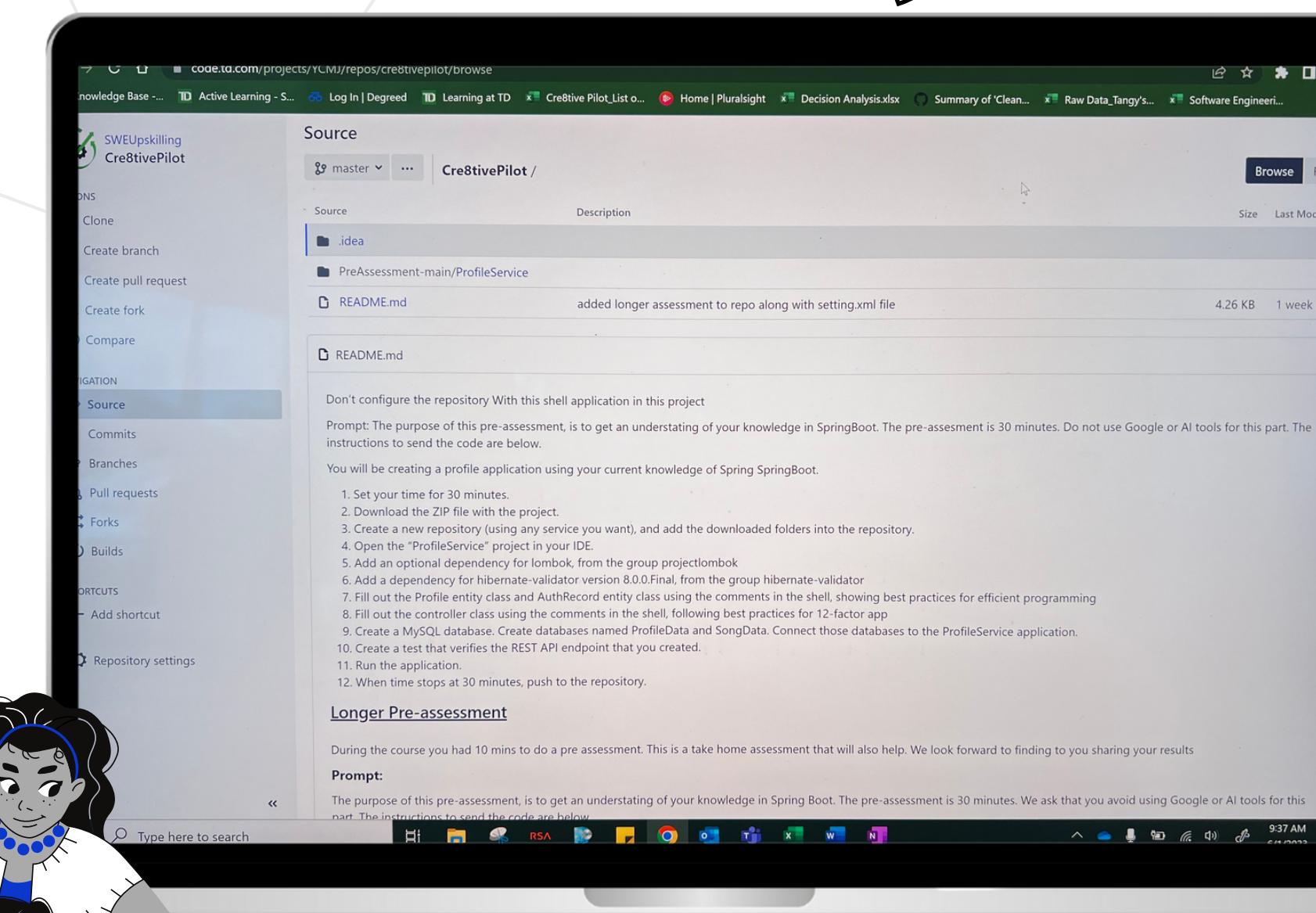
The RestTemplate is missing our load balancing, so we won't cycle between Account Service clients properly. The correct code is as follows.

```
@Bean  
@LoadBalanced  
public RestTemplate restTemplate() {  
    return new RestTemplate();  
}
```



HANDS ON PROJECT

YOU CAN WORK ON YOUR HANDS ON PROJECT



<Cre8tive Devs.
Software />



THANK YOU

<Creative Software/>

Crash Course

We will see you on Monday