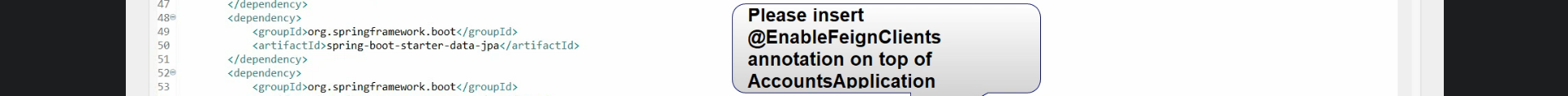
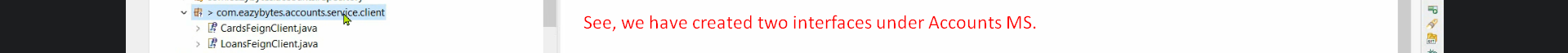
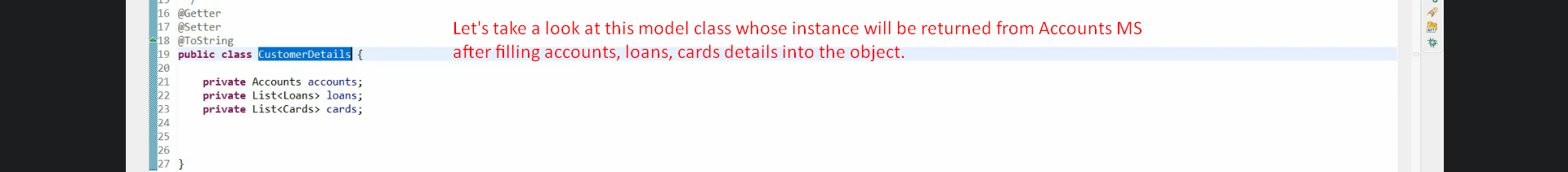
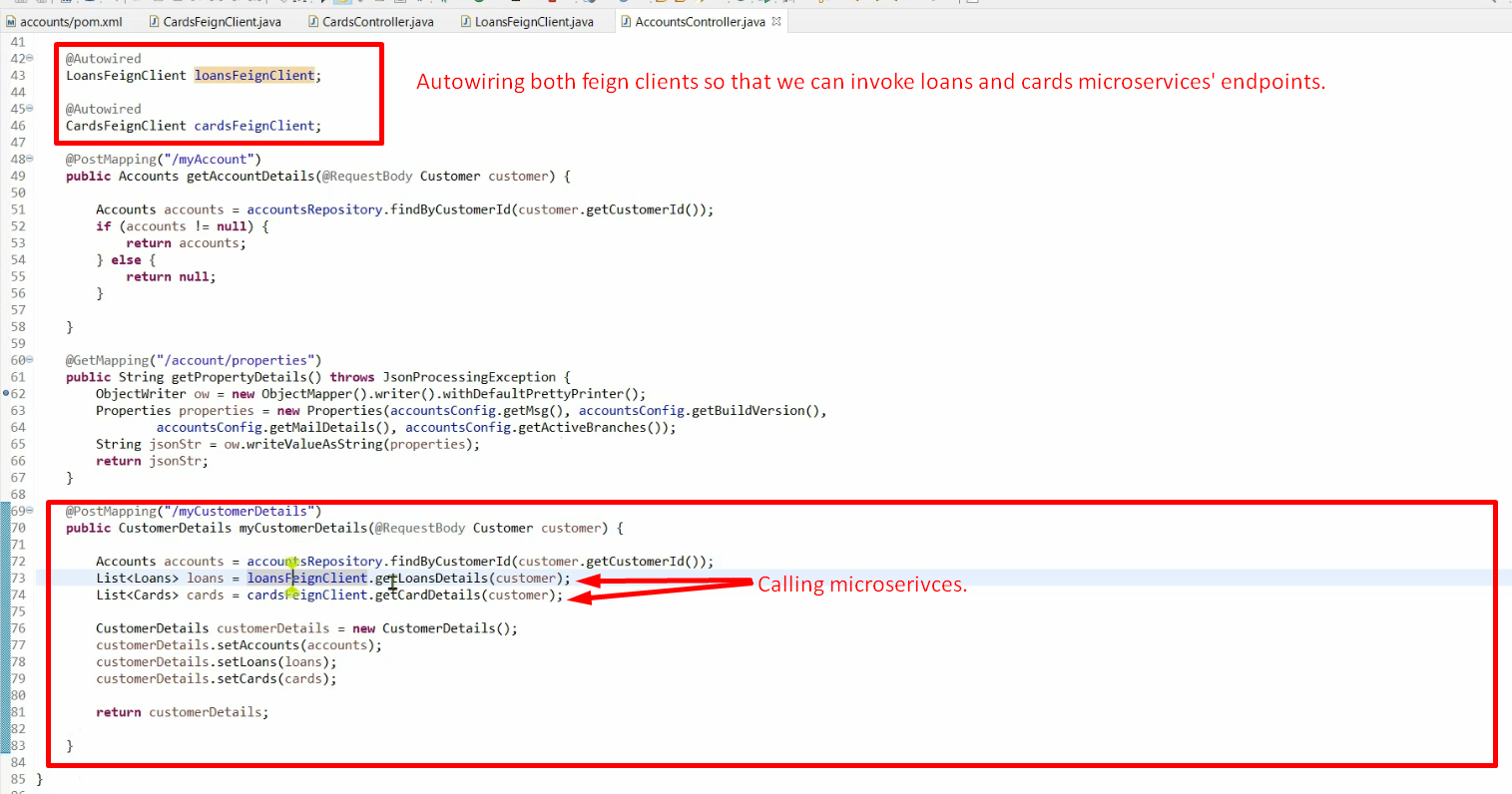
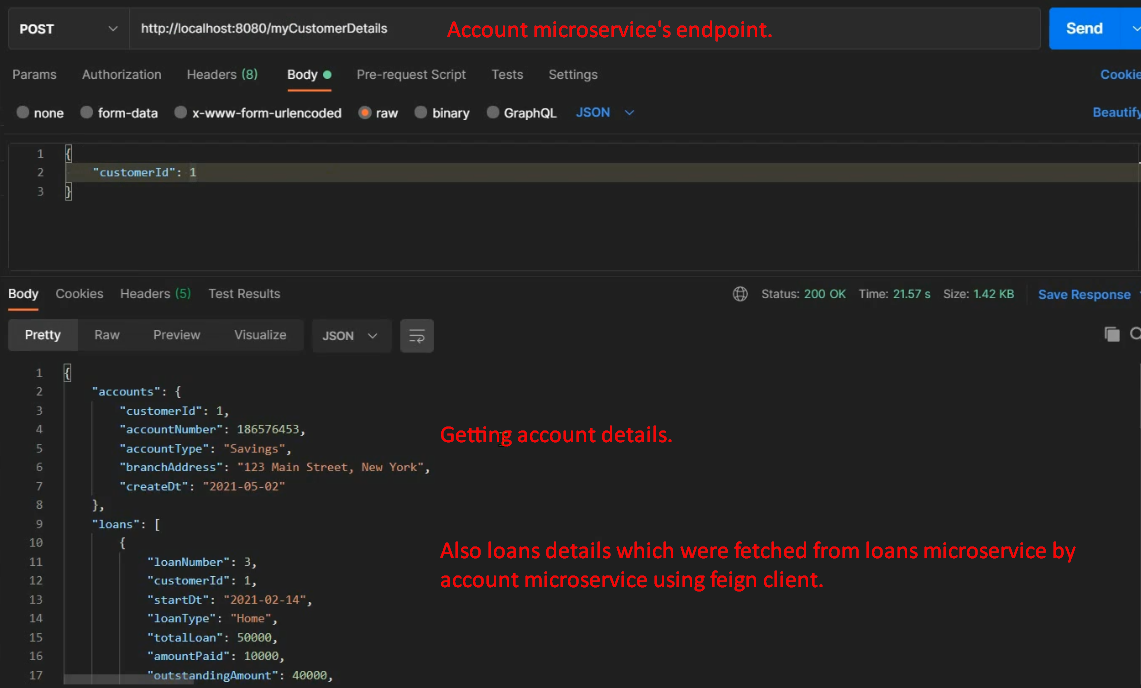
1. **Agenda:**
   1. How one microservice can connect with another microservice using **Eureka Registry Details** and **Netflix Feign client**.
2. With Feign client, we can achieve the client-side load balancing which will reduce the load on Eureka Server.
3. Now we will discuss the last component which is feign client which allows microservice inside a network to talk to each other and share info b/w each other without knowing their endpoints of each of the services and this is possible with the help of Eureka Server in b/w.
4. What we’re going to do?
   1. We will create one new REST API Path “/myCustomerDetails” inside Accounts MS which will be exposed to UI app which can be invoked by Bank Call Centre Executives.   
      This path invocation in Accounts MS will give us details about Accounts, Cards, Loans of a particular customer.   
      Accounts microservice would call other microservices (cards, loans) by passing only account id then the complete info (account, loan, card) would be sent to the caller from Accounts MS.
   2. So, account microservice will interact with other microservices using feign client.
5. **Step 01**:
6. **Step 02**: Adding **@EnableFeignClients**  
   
7. **Step 03**:  
   now we want to create interface inside Accounts MS which will act as proxy whenever we want to invoke other microservice from Accounts MS.  
   As we know in JPA Repository, we just create interface and method name like findByCustomerId(long id). Based on the method name, JPA Framework is smart enough to build runtime behaviour and execute all the business logic against my Database.   
   So is the case with interface for feign client.  
   Let’s see one by one   
   A picture containing text

   Description automatically generated  
   This interface can be used to invoke Cards Business Logic from Accounts MS.  
   @FeignClient(“<logical\_name\_for\_cards\_registered\_with\_Eureka\_Server>”)  
   @RequestMapping(value=”path”) => path = “/myCards”. See the below slide.  
   Graphical user interface, text, application, email

   Description automatically generated  
   **Let’s see the Loans Feign Client.**
8. 
9. 
10. **NOTE**: Bean for Feign client is created automatically.
11. 
12. Remember, when first time feign client goes to Eureka Service Discovery, it will fetch the complete details about all registered instances for a microservices and cache them locally and does the load balancing based on this cached details.
13. Let’s demo.  
    Start in this order 🡺 Config Server, Eureka Server, (Accounts, Loans, Cards)
14. First start Eureka Server then all the 3 microservices (accounts, loans, cards).
15. 
16. Let’s try to generate Docker image based on the changes we made and push onto the Docker Hub.  
    Post that we can update our Docker Compose File for various environment which can be tested using Docker Commands.