**Báo cáo tích hợp Spring Cloud vào dự án Microservice**

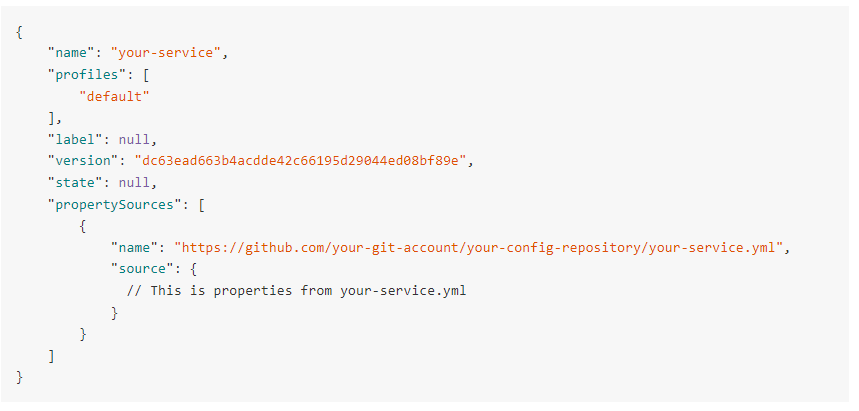
1. **Khái niệm về Spring Cloud:**

* Spring Cloud là một dự án con (sub-project) nằm trong Spring IO.
* Spring Cloud là một công nghệ phần mềm sử dụng để phát triển các ứng dụng phân tán. Một ứng dụng được gọi là phân tán (Distributed application) khi các phần của nó có thể được phát triển trên các ngôn ngữ khác nhau, và được triển khai trên các máy chủ khác nhau.
* Mục tiêu của Spring Cloud là làm sao để các thành phần của ứng dụng có thể giao tiếp với nhau.
* Dưới đây là danh sách các sub-project và các khuôn mẫu (pattern) trong Spring Cloud:
  + Spring Cloud Config
  + Spring Cloud Netflix
  + Spring Cloud Bus
  + Spring Cloud Sleuth
  + Spring Cloud Strem
  + Spring Cloud Stream Modules
  + Spring Cloud Connectors
  + Spring Cloud Starters

**…**

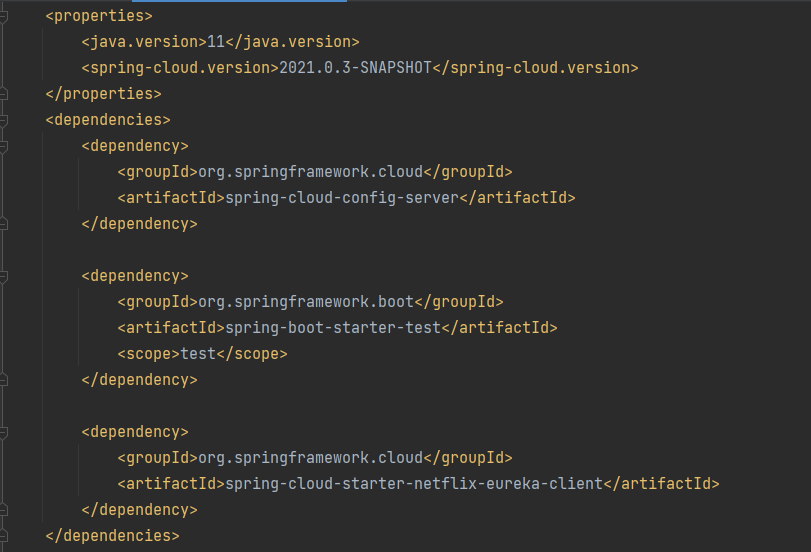
1. **Tích hợp Spring Cloud Config vào dự án microservice:**

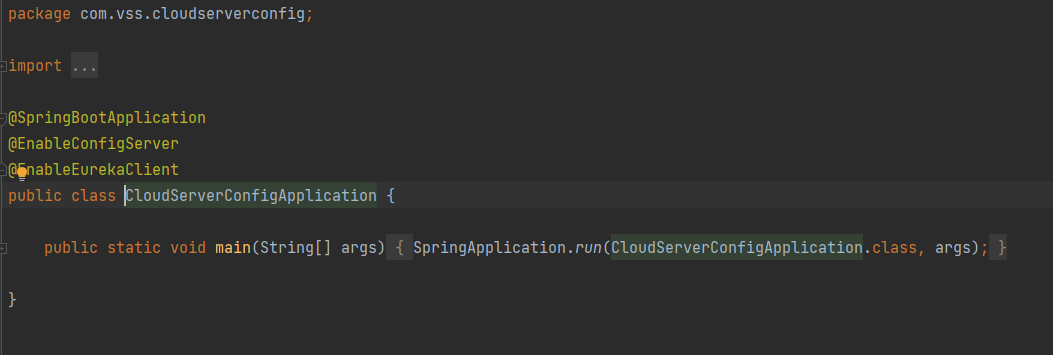
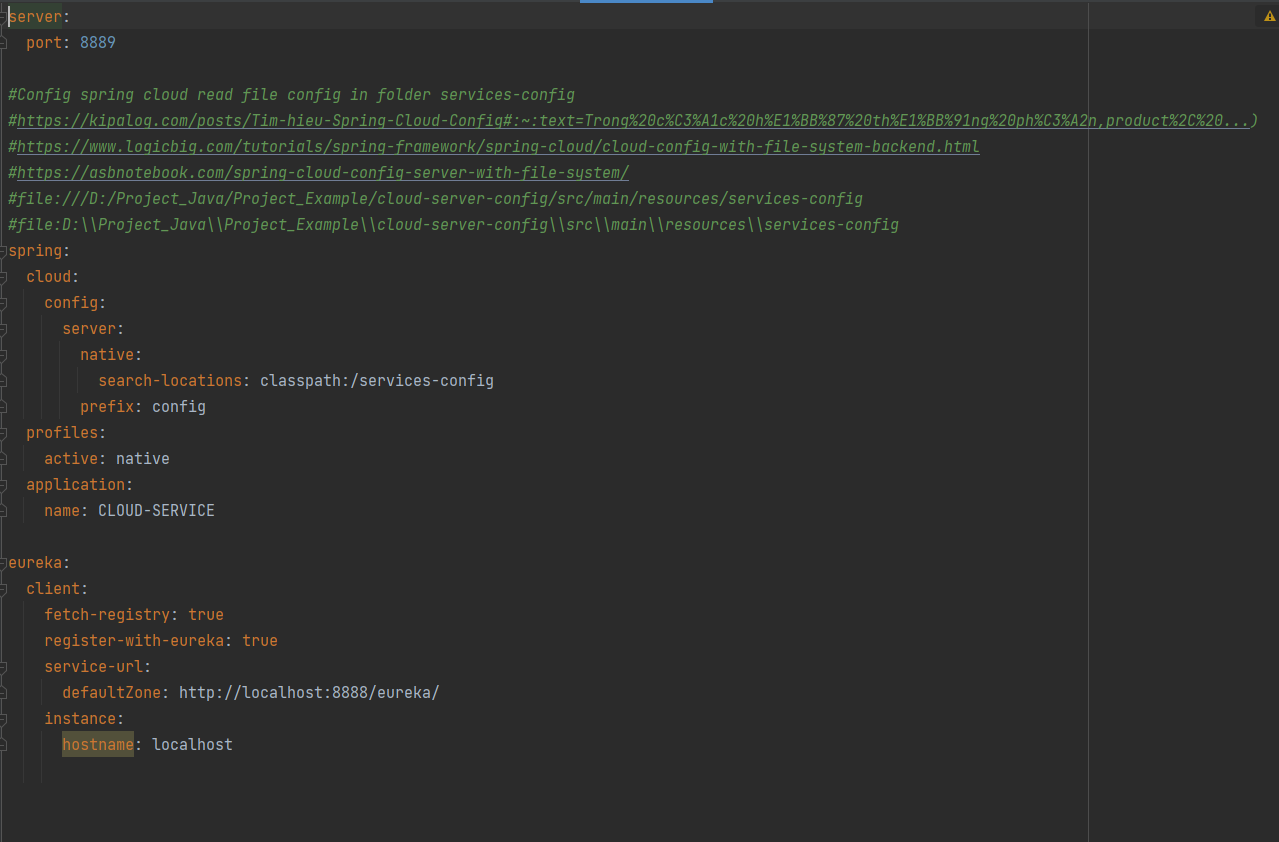
* Trong các hệ thống phân tán, Spring Cloud Config được dùng cho mục đích *externalized configuration* và *centralized configuration* – tách biệt và tập trung các property của các ứng dụng Spring tại một nơi. Đây là các property có giá trị khác nhau trên các môi trường phát triển (dev, staging, product,…)
* Spring Cloud Config hoạt động theo mô hình kiến trúc client-server. Bao gồm:
  + Spring Cloud Config Server
  + Spring Cloud Config Client
* Các property của các ứng dụng Spring (được lưu trữ trong các property source như file properties hoặc file YAML) được tập trung trên một hệ thống backend: GIT repository (mặc định), File System, Vault, JDBC, Redis,…
* **Spring Config Server:**
  + Nhiệm vụ của Spring Config Server là pull các property này về và dùng EnvironmentRepository để lưu trữ. EnvironmentRepository cung cấp các đối tượng Spring Environment.
  + Sau đó cung cấp các property cho Config Client thông qua các HTTP resource-base API (HTTP method là GET).
* **Spring Config Client:**
  + Đây chính là các ứng dụng Spring đã tách biệt các property. Khi startup, Config Client sẽ đọc các property từ API của Config Server và khởi tạo đối tượng Environment với property source phù hợp
  + Các API này có các PATH sau:
    - /{application}-{profile}.yml
    - /{application}-{profile}.properties
    - Trong đó:
      * {application}: map với giá trị spring.application.name trong file bootstrap.yml của Config Client.
      * {profile}: map với giá trị spring.profiles.active trong file bootstrap.yml của Config Client. Nếu có nhiều active profile thì các active profile sẽ được ngăn cách bởi dấu phẩy. Giá trị mặc định là default.
    - VD: Git repository có property source là your-service.yml và your-service-dev.yml. Như vậy:
      * your-service.yml: có {application} là your và {profile} là service. Config Server sẽ cung cấp API <http://localhost:8889/your/service> để Config Client đọc được các property từ property source your-service.yml đặt trên hệ thống backend.

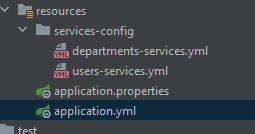


1. **Tích hợp Spring Cloud Config vào dự án microservice:**

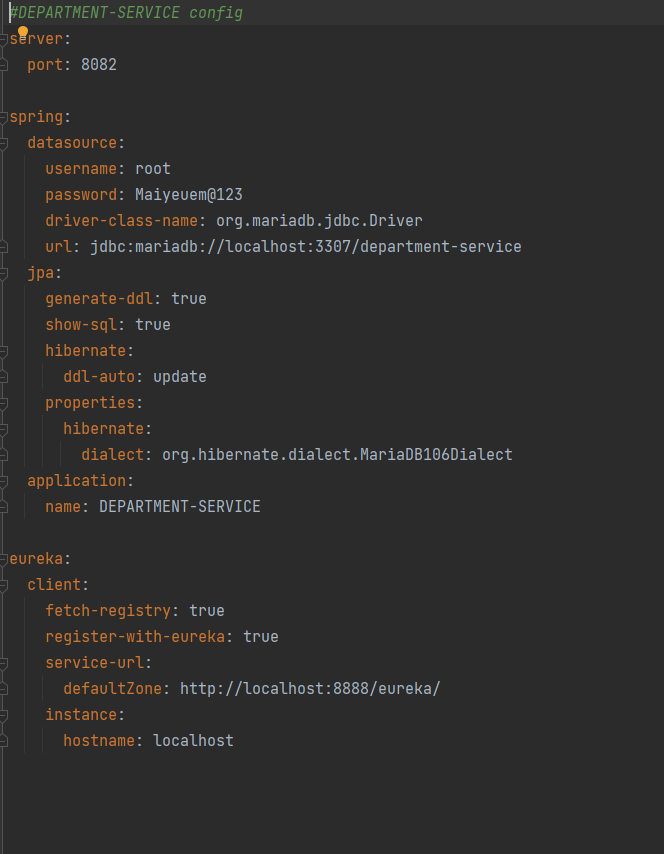
* Tạo dự án registry-eureka-service
* Tạo dự án cloud-gateway service
* Tạo dự án user-service
* Tạo dự án department-service
* Tạo dự án
* Trong ví dụ này ta sẽ để các file config tới database của user-service và department-service lên Config Server.
* Tạo dự án cloud-server-config
  + Thêm các dependency:



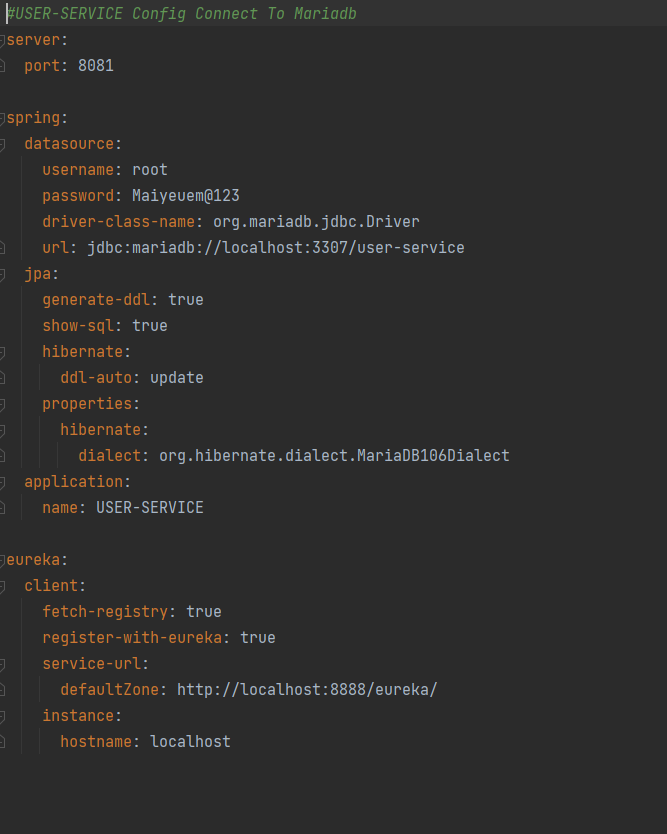
* Tại class main chúng ta thêm 2 annotation @EnableConfigServer và @EnableEurekaClient 
* Config ile application.yml: 
* Trong đó:
  + spring.cloud.config.server.native.search-locations là đường dẫn đến thư mục chứa các config tại Config Server
  + Để tránh conflic với context path hoặc server path của ứng dụng, chúng ta sẽ bổ sung giá trị spring.cloud.config.server.prefix=config (mặc định là rỗng). Đây là prefix của các API của Config Server. VD nếu spring.cloud.config.server.prefix=config thì API <http://localhost:8889/users/services> sẽ được chuyển thành <http://localhsot:8889/config/users/services>. Trong đó localhost:8889 là host và port của Client Server
* Trong folder resource tạo 2 file departments-services.yml và users-services.yml

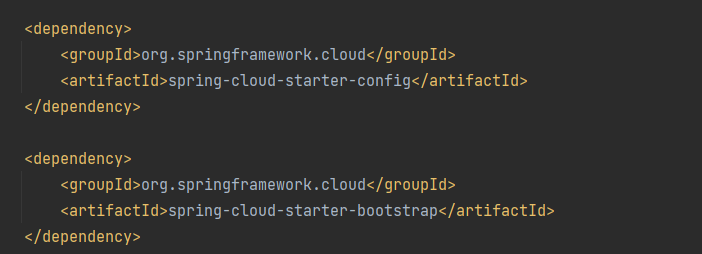
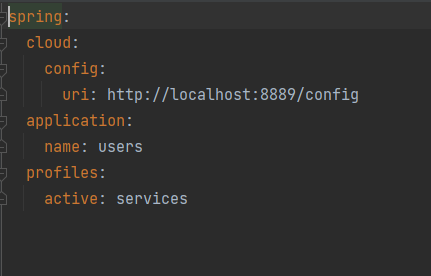
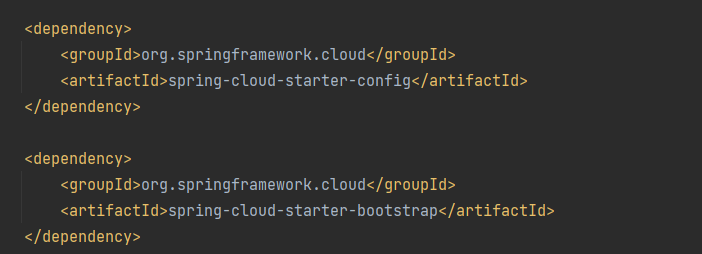
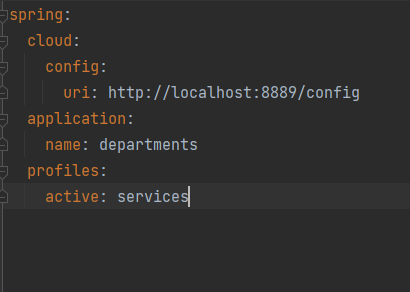


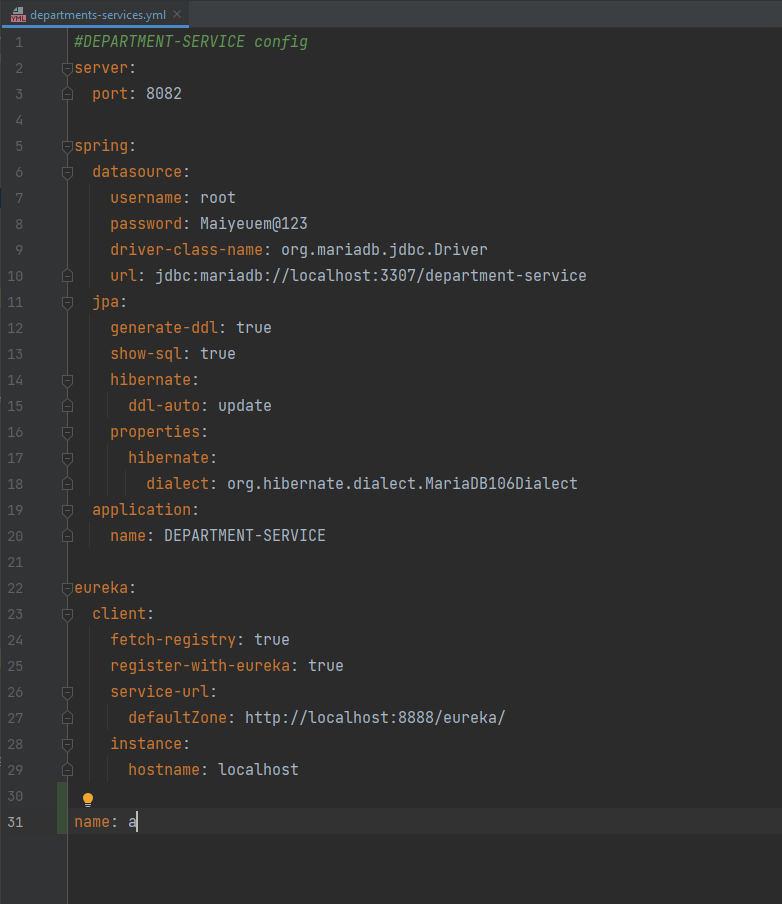
* + File departments-service.yml



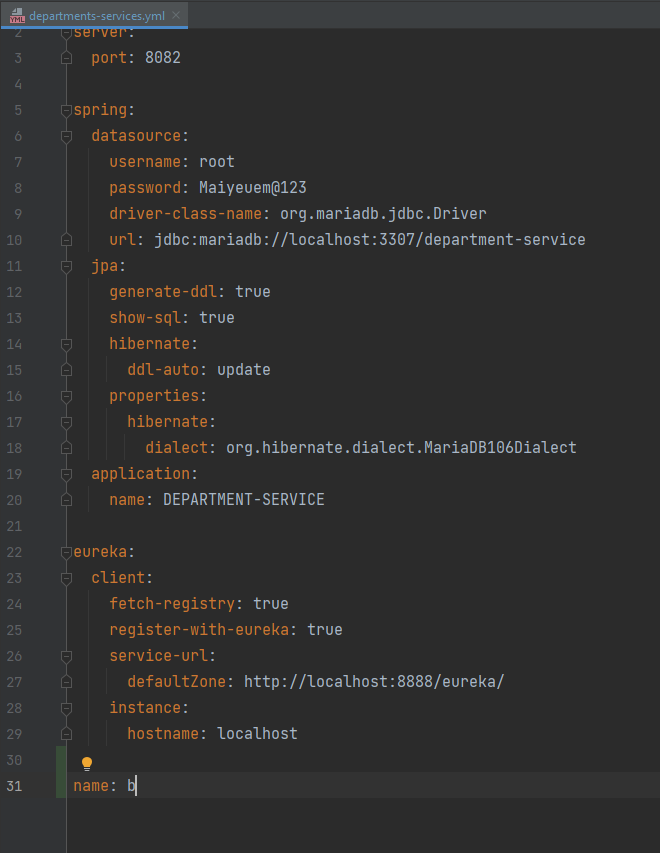
* + File users-services.yml



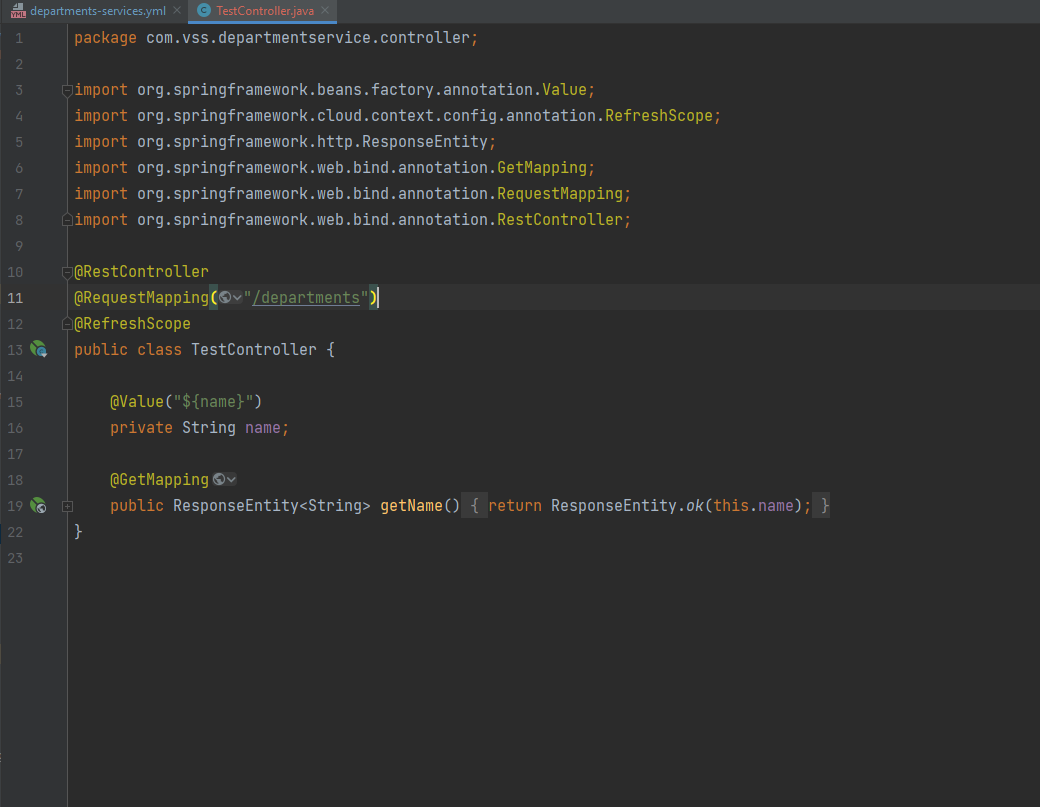
* **Tại project user-service ta thêm hai thư viện sau:** 
* Tạo file bootstrap.yml ta config như sau: 
  + Trong đó:
    - spring.cloud.config.uri: URL của Config Server
    - spring.application.name=users
    - spring.profiles.active=services
    - (file users-services ở Config Server thì spring.application.name=users và spring.profiles.active=services)
* **Tại project departments-service ta thêm hai thư viện sau:** 
* Tạo file bootstrap.yml ta config như sau: 
* **Test sự thay đổi dữ liệu với Spring Cloud Config**
  + Tại file departments-services.yml trong folder resource/services-config tại project cloud-server-config ta thêm name:a



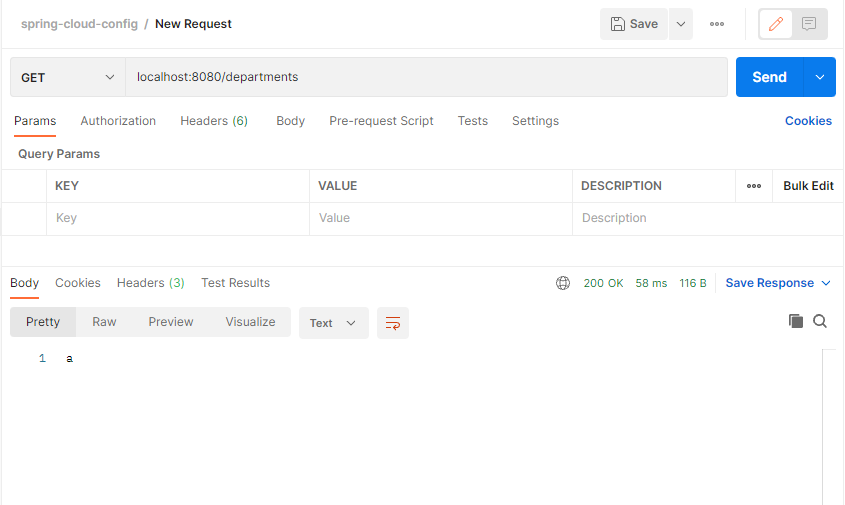
* + Sửa name: b



* + Tại project department-service ta tạo một controller để test như sau:



* + Restart run lại project cloud-server-config và department-service
  + Test trên postman trước khi thay đổi:



* + Test trên postman sau khi thay đổi:

