Zuul

Requirements

Lab Requirements

Purpose of this lab

- How to set up a routing/proxy server using Zuul
- How to leverage Pivotal Cloud Foundry's service discovery with Zuul
- Estimated Time: 25 minutes

Start the `config-server`, `service-registry`, `fortune-service`, and `greeting-frontend`

Start the `config-server` in a terminal window. You may have terminal windows still open from previous labs. They may be reused for this lab.

```
cd $SPRING_CLOUD_SERVICES_LABS_HOME/config-server
mvn clean spring-boot:run
```

Note: If you are doing this lab stand-alone make sure to have the `app-config` repo needed linked in the `config-server` `application.yml`. This is covered in the `config-server` lab.

Start the `service-registry`

```
cd $SPRING_CLOUD_SERVICES_LABS_HOME/service-registry
mvn clean spring-boot:run
```

Start the `fortune-service`

```
cd $SPRING_CLOUD_SERVICES_LABS_HOME/fortune-service
mvn clean spring-boot:run
```

Explore the `greeting-frontend` application

In this lab, we have created a javascript frontend application to display our fortune. Look at the controller in this application: it makes a request to the fortune service to display a fortune.

```
@GetMapping("/")
public ModelAndView index() {
    ModelAndView modelAndView = new ModelAndView("index");

    Map<String, Object> model = modelAndView.getModel();
    model.put("apiServerUrl", getFortuneServiceUrl());

    return modelAndView;
}


private String getFortuneServiceUrl() {
    return discoveryClient
        .getNextServerFromEureka("fortune-service", false)
        .getHomePageUrl();
}
```

Before starting the `greeting-frontend` change the `HomeController` to return `fortune-service` instead of `gateway-application` in the `getGatewayUrl` method.

Start the `greeting-frontend`

```
cd $SPRING_CLOUD_SERVICES_LABS_HOME/greeting-frontend
mvn clean spring-boot:run
```

Check if the frontend works

Open the application in your browser: http://localhost:8791

It should render an error message.

If you look at your browser's console, you will see an error similar to the following:

```
XMLHttpRequest cannot load http://192.168.137.19:8787/. No 'Access-Control-Allow-Origin' header is present on the requested resource. Origin 'http://localhost:8791' is therefore not allowed access.
```

Let's fix this by introducing a Zuul gateway that will set the headers correctly.

Set up the Zuul gateway

Have a look at the configuration and code of the `gateway-app`.

Notice the annotations on the application.

Check the CorsFilter class.

See that the `pom.xml` has the `spring-cloud-starter-zuul` dependency as well as `spring-cloud-services-starter-config-client` and `spring-cloud-services-starter-service-registry`

Have a look at `application.yml` and `bootstrap.yml`

Now let's start the application:

```
cd $SPRING_CLOUD_SERVICES_LABS_HOME/gateway-app
mvn clean spring-boot:run
```

Update the frontend to go through the Zuul gateway

Apply the following changes to the original controller at `$SPRING_CLOUD_SERVICES_LABS_HOME/greeting-frontend/src/main/java/io/pivotal/greetingfrontend/HomeController.java`

```
    @GetMapping("/")
    public ModelAndView index() {
        ModelAndView modelAndView = new ModelAndView("index");

        Map<String, Object> model = modelAndView.getModel();
-       model.put("apiServerUrl", getFortuneServiceUrl());
+       model.put("apiServerUrl", getGatewayAppUrl() + "fortune/"
);

        return modelAndView;
    }


-   private String getFortuneServiceUrl() {
+   private String getGatewayAppUrl() {
        return discoveryClient
-           .getNextServerFromEureka("fortune-service", false)
+           .getNextServerFromEureka("gateway-application", false
)

            .getHomePageUrl();
    }
```

Note the usage of a trailing `/` in the `fortune/` String. It actually is important, the app will break without it. See this stack overflow thread for some more details.

Now run the application:

```
cd $SPRING_CLOUD_SERVICES_LABS_HOME/greeting-frontend
mvn clean spring-boot:run
```

Open the page in your browser once more, and check that it now works. It is at: http://localhost:8791

Deploy the frontend and the gateway to cloud foundry

We expect that you already have set up the following services: `service-registry`, `config-server`

We expect that you already have pushed the following application: `fortune-service`

Deploy the gateway application:

```
cd $SPRING_CLOUD_SERVICES_LABS_HOME/gateway-app
./mvnw clean package
cf push gateway-app -p target/gateway-application-0.0.1-SNAPSHOT.
jar --random-route --no-start
cf bind-service gateway-app service-registry
cf bind-service gateway-app config-server
cf start gateway-app
```

Deploy the frontend application:

```
cd $SPRING_CLOUD_SERVICES_LABS_HOME/greeting-frontend
./mvnw clean package
cf push greeting-frontend -p target/greeting-frontend-0.0.1-SNAPS
HOT.jar --random-route --no-start
cf bind-service greeting-frontend service-registry
cf bind-service greeting-frontend config-server
cf start greeting-frontend
```

Set the `TRUST_CERTS` environment variable for the `greeting-frontend` and `gateway-app` applications (our PCF instance is using self-signed SSL certificates).

```
$ cf set-env greeting-frontend TRUST_CERTS <your api endpoint>
$ cf set-env gateway-app TRUST_CERTS <your api endpoint>
```

Visit the `greeting-frontend` application, and check that everything is going according to plan.