## **Section-End Project - Lesson 4**

## **Create a Spring Cloud Eureka Server and Client**

****Part 1, create server****

1. Create a new Spring Boot Starter application .

* Name the project "lab-4-eureka-server”, and use this value for the Artifact.
* Use JAR packaging and the version 11 of Java.
* You can generally use the latest stable version of Spring Boot available.

1. Check the ”Dependencies Management” section is present.
2. Add a dependency for group "org.springframework.cloud" and artifact "spring-cloud-starter-netflix-eureka-server". You do not need to specify a version -- this is already defined in the parent project.
3. Save an application.yml (or properties) file in the root of your classpath (src/main/resources recommended). Add the following key / values (use correct YAML formatting):

* server.port: 8010

1. Save a bootstrap.yml (or properties) file in the root of your classpath. Add the following key / values (use correct YAML formatting):

* spring.application.name=lab-4-eureka-server

1. Add @EnableEurekaServer to the Application class. Save. Start the server. Temporarily ignore the warnings about running a single instance (i.e. connection refused, unable to refresh cache, backup registry not implemented, etc.). Open a browser to [http://localhost:8010](http://localhost:8010/) to see the server running.

****Part 2, create clients****

In this next section we will create several client applications that will work together to compose a sentence. The sentence will contain a subject, verb, article, adjective and noun such as “I saw a leaky boat” or “You have the reasonable book”. 5 services will randomly generate the word components, and a 6th service will assemble them into a sentence.

1. Create a new Spring Boot web application.

* Name the project "lab-4-subject”, and use this value for the Artifact.
* Use JAR packaging and the latest versions of Java.
* Use Boot version 2.5.6 or the latest stable version available.
* Add actuator and web as a dependencies.

1. Modify the POM (or Gradle) file.

* Add the same dependency management section you inserted into the server POM. (You could simply change the parent entries, but most clients will probably be ordinary applications with their own parents.)
* Add a dependency for group "org.springframework.cloud" and artifact "spring-cloud-starter-netflix-eureka-client".

1. Modify the Application class. Add @EnableDiscoveryClient.
2. Save an application.yml (or properties) file in the root of your classpath (src/main/resources recommended). Add the following key / values (use correct YAML formatting):

* eureka.client.serviceUrl.defaultZone=<http://localhost:8010/eureka/>
* words=I,You,He,She,It
* server.port=${PORT:${SERVER\_PORT:0}} (this will cause a random, unused port to be assigned if none is specified)

1. Save a bootstrap.yml (or properties) file in the root of your classpath. Add the following key / values (use correct YAML formatting):

* spring.application.name=lab-4-subject

1. Add a Controller class

* Place it in the 'demo' package or a subpackage of your choice.
* Name the class anything you like. Annotate it with @RestController.
* Add a String member variable named “words”. Annotate it with @Value("${words}”).
* Add the following method to serve the resource (optimize this code if you like):

@GetMapping("/")

public @ResponseBody String getWord() {

String[] wordArray = words.split(",");

int i = (int)Math.round(Math.random() \* (wordArray.length - 1));

return wordArray[i];

}

1. Repeat steps 7 thru 12 (copy the entire project if it is easier), except use the following values:

* Name of application: “lab-4-verb”
* spring.application.name: “lab-4-verb”
* words: “ran,knew,had,saw,bought”

1. Repeat steps 7 thru 12, except use the following values:

* Name of application: “lab-4-article”
* spring.application.name: “lab-4-article”
* words: “a,the”

1. Repeat steps 7 thru 12, except use the following values:

* Name of application: “lab-4-adjective”
* spring.application.name: “lab-4-adjective”
* words: “reasonable,leaky,suspicious,ordinary,unlikely”

1. Repeat steps 7 thru 12, except use the following values:

* Name of application: “lab-4-noun”
* spring.application.name: “lab-4-noun”
* words: “boat,book,vote,seat,backpack,partition,groundhog”

1. Create a new Spring Boot web application.

* Name the application “lab-4-sentence”, and use this value for the Artifact.
* Use JAR packaging and the latest versions of Java and Boot.
* Add actuator and web as a dependencies.
* Alter the POM (or Gradle) just as you did in step 8.

1. Add @EnableDiscoveryClient to the Application class.
2. Save an application.yml (or properties) file in the root of your classpath (src/main/resources recommended). Add the following key / values (use correct YAML formatting):

* eureka.client.serviceUrl.defaultZone=<http://localhost:8010/eureka/>
* server.port: 8020

1. Add a Controller class to assemble and return the sentence:

* Name the class anything you like. Annotate it with @RestController.
* Use @Autowired to obtain a DiscoveryClient (import from Spring Cloud).
* Add the following methods to serve the sentence based on the words obtained from the client services. (feel free to optimize / refactor this code as you like:

@GetMapping("/sentence")

public @ResponseBody String getSentence() {

return

getWord("LAB-4-SUBJECT") + " "

+ getWord("LAB-4-VERB") + " "

+ getWord("LAB-4-ARTICLE") + " "

+ getWord("LAB-4-ADJECTIVE") + " "

+ getWord("LAB-4-NOUN") + "."

;

}

public String getWord(String service) {

List<ServiceInstance> list = client.getInstances(service);

if (list != null && list.size() > 0 ) {

URI uri = list.get(0).getUri();

if (uri !=null ) {

return (new RestTemplate()).getForObject(uri,String.class);

}

}

return null;

}

1. Run all of the word services and sentence service. (Run within your IDE, or build JARs for each one (mvn clean package) and run from the command line (java -jar name-of-jar.jar), whichever you find easiest). (If running from STS, uncheck “Enable Live Bean support” in the run configurations). Since each service uses a separate, random port, they should be able to run side-by-side on the same computer. Open <http://localhost:8020/sentence> to see the completed sentence. Refresh the URL and watch the sentence change.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*