##Lab 6 - Using Feign Declarative REST clients

**Part 1, Start existing services.**

1. Stop all of the services that you may have running from previous exercises. If using an IDE you may also wish to close all of the projects that are not related to "lab-6" or "common".
2. Start the common-config-server and the common-eureka-server. These are versions of what you created and used in the last few chapters.
3. Lab 6 has copies of the word server and sentence server that have been slightly refactored from what we worked on previously. Start 5 separate copies of the lab-6-word-server, using the profiles "subject", "verb", "article", "adjective", and "noun". There are several ways to do this, depending on your preference:

* If you wish to use Maven, open separate command prompts in the target directory and run these commands:
  + mvn spring-boot:run -Drun.jvmArguments="-Dspring.profiles.active=subject"
  + mvn spring-boot:run -Drun.jvmArguments="-Dspring.profiles.active=verb"
  + mvn spring-boot:run -Drun.jvmArguments="-Dspring.profiles.active=article"
  + mvn spring-boot:run -Drun.jvmArguments="-Dspring.profiles.active=adjective"
  + mvn spring-boot:run -Drun.jvmArguments="-Dspring.profiles.active=noun"
* Or if you wish to run from directly within STS, right click on the project, Run As... / Run Configurations... . From the Spring Boot tab specify a Profile of "subject", UNCHECK live bean support, and Run. Repeat this process (or copy the run configuration) for the profiles "verb", "article", "adjective", "noun".

1. Check the Eureka server running at<http://localhost:8010>. Ignore any warnings about running a single instance; this is expected. Ensure that each of your 5 applications are eventually listed in the "Application" section, bearing in mind it may take a few moments for the registration process to be 100% complete.
2. Optional - If you wish, you can click on the link to the right of any of these servers. Replace the "/info" with "/" and refresh several times. You can observe the randomly generated words within a JSON structure.
3. Run the lab-6-sentence-server project. Refresh Eureka to see it appear in the list. Test to make sure it works by opening<http://localhost:8020/sentence>. You should see several random sentences appear. We will refactor this code to make use of Feign.

**Part 2 - Refactor**

1. First, take a look at the lab-6-sentence-server project. It has been refactored a bit from previous examples. The controller has been simplified to do only web work, the task of assembling the sentence is now in the service layer. The SentenceService uses @Autowire to reference individual DAO components which have been created to obtain the words from the remote resources. Since all of the remote resources are structurally the same, there is a fair bit of inheritance in the dao package to make things easy. But each uses the same Ribbon client technology and RestTemplate used previously.
2. Open the POM. Add another dependency for **spring-cloud-starter-feign.**
3. Edit the main Application configuration class and @EnableFeignClients.
4. Refactor the “Noun” service to use Feign. Create a new interface in the dao package called NounClient. Annotate it with @FeignClient. What value should you use for the service ID? The existing code should tell you.
5. Next, provide the method signature to be implemented by Feign. To think this through, take a fresh look at the lab-6-word-server WordController. Note the annotation used and return type. You can actually copy/paste this signature as-is, except 1) remove the method implementation, and 2) there is no need for @ResponseBody as this is implied, and 3) it will be necessary to add method=RequestMethod.GET to clarify that this is a GET request.
6. Edit the SentenceService. Replace the private WordDao nounService and its associated setter with the NounClient you just made. Depending on how you built your NounClient, you may need to refactor the buildSentence method slightly.
7. Notice that you have probably introduced an error on the SentenceServiceImplTest test class. Make adjustments to accomodate the NounClient.
8. Stop any previously running sentence server and launch your new one. Test it to make sure it works by opening<http://localhost:8020/sentence>. The application should work the same as it did before, though now it is using a declarative Feign client to make the noun call.

**BONUS - Additional Refactoring**

1. If you like, you can also refactor the subject, verb, article, and adjective clients. This can be done reasonably easy by copy paste. Remove the old DAO implementations and note that you will no longer have any executable code to maintain or unit test.

**Reflection:**

1. While we have no dao code that requires UNIT testing, we still need to perform INTEGRATION testing. Still, it is nice to be freed of the need to do as much coding.
2. Our application will still fail if we can’t find at least one of each kind of word server. We will improve this later when we discuss circuit breakers with Hystrix.
3. To improve performance, can we run each of the calls in parallel? We will improve this later when discussing Hystrix.