Lab 6 : Read Custom Property Files

Add the following property to the `application.properties` file:

```
bootrest.customproperty=hello
```

Then, edit the `GreetingController` class as follows:

```
@Autowired
Environment env;

Greet greet(){
    logger.info("bootrest.customproperty "+
env.getProperty("bootrest.customproperty"));
    return new Greet("Hello World!");
}
```

Rerun the application. The log statement prints the custom variable in the console, as follows:

```
org.rvslab.chapter2.GreetingController    : bootrest.customproperty hello
```

Using a .yaml file for configuration

As an alternate to `application.properties`, one may use a `.yaml` file. YAML provides a JSON-like structured configuration compared to the flat properties file.

To see this in action, simply replace `application.properties` with `application.yaml` and add the following property:

```
server
  port: 9080
```

Rerun the application to see the port printed in the console.

Using multiple configuration profiles

Furthermore, it is possible to have different profiles such as development, testing, staging, production, and so on. These are logical names. Using these, one can configure different values for the same properties for different environments. This is quite handy when running the Spring Boot application against different environments. In such cases, there is no rebuild required when moving from one environment to another.

Update the `.yaml` file as follows. The Spring Boot group profiles properties based on the dotted separator:

```
spring:
    profiles: development
server:
      port: 9090
---

spring:
    profiles: production
server:
      port: 8080
```

Run the Spring Boot application as follows to see the use of profiles:

```
mvn -Dspring.profiles.active=production install
mvn -Dspring.profiles.active=development install
```

Active profiles can be specified programmatically using the `@ActiveProfiles` annotation, which is especially useful when running test cases, as follows:

```
@ActiveProfiles("test")
```

Other options to read properties

The properties can be loaded in a number of ways, such as the following:

- Command-line parameters `(-Dhost.port =9090`)
- Operating system environment variables
- JNDI (`java:comp/env`)