Spring Cloud Task Lab

Let's Get Started!

1) First let's goto lab 2 by going to our lab directory i.e. `cd <..>/DNDataflow/labs/lab2` directory
2) Now let's build the project and run it
   a) `mvnw clean package`
   b) `java -jar target/tasklab-0.0.1-SNAPSHOT.jar`
3) Following the execution we will see the our "Hello World" show up along with the typical logging information from a basic boot app. Something like…

```
  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::        (v1.5.1.RELEASE)

2017-02-03 11:08:09.573  INFO 60708 --- [           main] io.spring.TasklabApplication  ...
2017-02-03 11:08:09.575  INFO 60708 --- [           main] io.spring.TasklabApplication  ...
: No active profile set, falling back to default profiles: default
2017-02-03 11:08:09.618  INFO 60708 --- [           main] s.c.a.AnnotationConfigApplicat...
2017-02-03 11:08:10.289  INFO 60708 --- [           main] o.s.j.e.a.AnnotationMBeanExporter
: Registering beans for JMX exposure on startup
Hello World
2017-02-03 11:08:10.299  INFO 60708 --- [           main] io.spring.TasklabApplication
: Started TasklabApplication in 0.963 seconds (JVM running for 1.287)
2017-02-03 11:08:10.299  INFO 60708 --- [       Thread-2] s.c.a.AnnotationConfigApplicat...
[Fri Feb 03 11:08:09 EST 2017]; root of context hierarchy
2017-02-03 11:08:10.301  INFO 60708 --- [       Thread-2] o.s.j.e.a.AnnotationMBeanExpor...
```

Now Let's Taskify our Boot App!

1. First lets add our dependencies to the pom.xml
2. Using your favorite editor or IDE open the pom.xml
3. We want to add the starter-task dependency along with H2 DataSource dependency.
   a. Copy the dependencies below and paste them on Line 34 of your pom.xml .

```
<dependency>
  <groupId>org.springframework.cloud</groupId>
  <artifactId>spring-cloud-starter-task</artifactId>
</dependency>
```

```
<dependency>
  <groupId>com.h2database</groupId>
  <artifactId>h2</artifactId>
</dependency>
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-jdbc</artifactId>
</dependency>
```

4. Now let's enable the task
    a. Open the `<..>/DNDataflow/labs/lab2/src/main/java/io/spring/TasklabApplication.java`
    b. On Line 13 add the *@EnableTask* annotation .
    c. If you're not using an IDE uncomment lines 6-9, so it can be imported.
5. From your shell rebuild the application and rerun the app
    a. `mvnw clean package`
    b. `java -jar target/tasklab-0.0.1-SNAPSHOT.jar`
6. Hmmm… It looks the same, the log messages you demoed are not present. Well let's update our configuration so that we can see the log messages.
    a. Open the `<..>/DNDataflow/labs/lab2/src/main/resources/application.properties`
    b. Now let's add the following properties:

    ```
    logging.level.org.springframework.cloud.task=DEBUG
    spring.application.name=lab2-task
    ```

    c. We just set the log level for spring cloud task so that our log messages will now show up.
    d. Notice that we also set the name of the application so that when it is stored in the database, the task_name entry in the TASK_EXECUTION table has something meaningful.
7. From your shell rebuild the application and rerun the app:
    a. `mvnw clean install`
    b. `java -jar target/tasklab-0.0.1-SNAPSHOT.jar`
8. Now we see our log messages, like what is shown below:

```
  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::        (v1.5.1.RELEASE)
```

```
...
2017-02-03 14:29:49.198 DEBUG 63357 --- [           main] o.s.c.t.r.support.SimpleTaskRepository
: Creating: TaskExecution{executionId=0, parentExecutionId=null, exitCode=null,
taskName='lab2-task', startTime=Fri Feb 03 14:29:49 EST 2017, endTime=null, exitMessage='null',
externalExecutionId='null', errorMessage='null', arguments=[]}
Hello World
2017-02-03 14:29:49.225 DEBUG 63357 --- [           main] o.s.c.t.r.support.SimpleTaskRepository
: Updating: TaskExecution with executionId=1 with the following {exitCode=0, endTime=Fri Feb 03
14:29:49 EST 2017, exitMessage='null', errorMessage='null'}
...
2017-02-03 14:29:49.232  INFO 63357 --- [           main] io.spring.TasklabApplication
: Started TasklabApplication in 1.37 seconds (JVM running for 1.716)
```

Now Let's Fail our Task to see if it records the error!

1. Now let's force an exception to be thrown to test how Task handles Exceptions
    a. Open the `<..>/DNDataflow/labs/lab2/src/main/java/io/spring/TasklabApplication.java`
    b. Copy the code below and paste it on line 26 add the following Exception:

    ```
    throw new IllegalStateException("No Task For You!!");
    ```

    c. From your shell rebuild the application and rerun the app:
        i. NOTE: that we are skipping tests for now so we can include the exception
        ii. `mvnw clean package -DskipTests`
        iii. `java -jar target/tasklab-0.0.1-SNAPSHOT.jar`
    d. Notice below that the "Updating" log message is a bit different. The exitCode for the application is 1 instead of 0 (because it failed) and also that the errorMessage is not empty but rather contains the stack trace.

```
  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::        (v1.5.1.RELEASE)

...
2017-02-03 14:40:26.987 DEBUG 63799 --- [           main] o.s.c.t.r.support.SimpleTaskRepository
: Creating: TaskExecution{executionId=0, parentExecutionId=null, exitCode=null,
taskName='lab2-task', startTime=Fri Feb 03 14:40:26 EST 2017, endTime=null, exitMessage='null',
externalExecutionId='null', errorMessage='null', arguments=[]}
```

```
Hello World
2017-02-03 14:40:26.995  INFO 63799 --- [           main]
utoConfigurationReportLoggingInitializer :

Error starting ApplicationContext. To display the auto-configuration report re-run your
application with 'debug' enabled.
2017-02-03 14:40:27.007 DEBUG 63799 --- [           main] o.s.c.t.r.support.SimpleTaskRepository
: Updating: TaskExecution with executionId=1 with the following {exitCode=1, endTime=Fri Feb 03
14:40:26 EST 2017, exitMessage='null', errorMessage='java.lang.IllegalStateException: Failed to
execute CommandLineRunner
      at org.springframework.boot.SpringApplication.callRunner(SpringApplication.java:779)
      at org.springframework.boot.SpringApplication.callRunners(SpringApplication.java:760)
      at org.springframework.boot.SpringApplication.afterRefresh(SpringApplication.java:747)
      at org.springframework.boot.SpringApplication.run(SpringApplication.java:315)
      at org.springframework.boot.SpringApplication.run(SpringApplication.java:1162)
      at org.springframework.boot.SpringApplication.run(SpringApplication.java:1151)
      at io.spring.TasklabApplication.main(TasklabApplication.java:14)
      at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
      at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
      at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
      at java.lang.reflect.Method.invoke(Method.java:498)
      at org.springframework.boot.loader.MainMethodRunner.run(MainMethodRunner.java:48)
      at org.springframework.boot.loader.Launcher.launch(Launcher.java:87)
      at org.springframework.boot.loader.Launcher.launch(Launcher.java:50)
      at org.springframework.boot.loader.JarLauncher.main(JarLauncher.java:51)
Caused by: java.lang.IllegalStateException: No Task For You!!
      at io.spring.TasklabApplication$1.run(TasklabApplication.java:23)
      at org.springframework.boot.SpringApplication.callRunner(SpringApplication.java:776)
      ... 14 more
'}

...
```

9. Now let's comment out our `throw new IllegalStateException.`

Now Let's do some pre and post processing!

1. Let's test the before and after task processing capabilities for Task.
    a. Open the `<..>/DNDataflow/labs/lab2/src/main/java/io/spring/TasklabApplication.java`
    b. Copy the code below and paste it on line 30 add the following Exception:

```
@BeforeTask
public void beforeTask(TaskExecution taskExecution) {
  System.out.println("Before TASK");
}
```

```
@AfterTask
public void afterTask(TaskExecution taskExecution) {
  System.out.println("After TASK");
}
```

c. From your shell rebuild the application and rerun the app:
   i. `mvnw clean install`
   ii. `java -jar target/tasklab-0.0.1-SNAPSHOT.jar`

d. Notice below that the after the “Creating:” log message we see that the method annotated with *@BeforeTask* fired printing “Before TASK”. And before the “Updating:” log message we see that the method annotated with *@AfterTask* fired printing “After TASK”.

```
  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::        (v1.5.1.RELEASE)

...
2017-02-03 15:05:26.087 DEBUG 64067 --- [           main] o.s.c.t.r.support.SimpleTaskRepository
: Creating: TaskExecution{executionId=0, parentExecutionId=null, exitCode=null,
taskName='lab2-task', startTime=Fri Feb 03 15:05:26 EST 2017, endTime=null, exitMessage='null',
externalExecutionId='null', errorMessage='null', arguments=[]}
Before TASK
Hello World
After TASK
2017-02-03 15:05:26.111 DEBUG 64067 --- [           main] o.s.c.t.r.support.SimpleTaskRepository
: Updating: TaskExecution with executionId=1 with the following {exitCode=0, endTime=Fri Feb 03
15:05:26 EST 2017, exitMessage='null', errorMessage='null'}

...
```

Extra Credit!

If you are running MySQL or other database on your laptop locally, we can test Tasks ability to create its tables in that repository and update with a Task Execution.

1. Using your favorite editor or IDE open the pom.xml
2. In this example we are going to add the MySQL (mariadb) DataSource dependency.
   a. Note: if you are using another database you can use that database dependencies.
   b. Copy the dependencies below and paste them on Line 46 of your pom.xml.

```
<dependency>
  <groupId>org.mariadb.jdbc</groupId>
  <artifactId>mariadb-java-client</artifactId>
</dependency>
```

3. Now let’s rebuild your project
    a. `mvnw clean install`
4. And now let’s setup the environment variables for example

```
export spring_datasource_url=jdbc:mariadb://localhost:3306/practice
export spring_datasource_username=root
export spring_datasource_password=password
export spring_datasource_driverClassName=org.mariadb.jdbc.Driver
```

5. Now rerun the app
    a. `java -jar target/tasklab-0.0.1-SNAPSHOT.jar`