LAB 21: QUARKUS MONITORING LOGGING

Autor: José Díaz

Github Repo: https://github.com/joedayz/quarkus-bcp-2025.git

Abre el proyecto **monitor-logging**

Instructions

In this exercise, you must configure logging and produce log messages in the expenses application.

1. Run the expenses application in development mode.

1.1. Navigate to the `~/D0378/monitor-logging` directory.

```
[student@workstation ~]$ cd ~/D0378/monitor-logging
```

1.2. Open the project with an editor, such as VSCodium or vim.

1.3. Run the expenses application in development mode.

```
[student@workstation monitor-logging]$ mvn quarkus:dev
...output omitted...
INFO  [io.quarkus] (Quarkus Main Thread) expenses ... Listening on: http://
localhost:8080
...output omitted...
```

2. Log an error message when a request to the GET `/expenses/{name}` endpoint tries to get an expense that does not exist.

2.1. Open the `ExpensesResource` class and modify the `getByName` method to log `ExpenseNotFoundException` error messages.

```
@GET
@Path( "/{name}" )
public Expense getByName( @PathParam( "name" ) String name ) {
```

```
    try {
        return expenses.getByName( name );
    } catch ( ExpenseNotFoundException e ) {
        var message = e.getMessage();
        Log.error( message );
        throw new NotFoundException( message );
    }
}
```

2.2. Open a new terminal and make a request to get the expense called **none**. This expense does not exist.

```
[student@workstation monitor-logging]$ curl -v \
http://localhost:8080/expenses/none
...output omitted...
>
< HTTP/1.1 404 Not Found
< Content-Type: application/json
< content-length: 0
<
...output omitted...
```

2.3. Return to the terminal where the application is running and verify that the output displays an error message.

```
...output omitted...
ERROR [com.red.tra.exp.ExpensesResource] (executor-thread-0) Expense not found:
 none
```

3. Log a debug message and adjust the application log level to **DEBUG**.

3.1. In the `ExpensesResource` class, modify the `getByName` method to log the `Getting expense {name}` debug message.

```
@GET
@Path("/{name}")
public Expense getByName( @PathParam( "name" ) String name ) {
    Log.debug( "Getting expense " + name );

    try {
        return expenses.getByName( name );
    } catch ( ExpenseNotFoundException e ) {
        var message = e.getMessage();
        Log.error( message );
        throw new NotFoundException( message );
    }
}
```

3.2. Make a request to get the expense called `joel-2`.

```
[student@workstation monitor-logging]$ curl -s \
http://localhost:8080/expenses/joel-2 | jq
{
  "uuid": "6df0b95d-afec-4171-a980-7c915a309f69",
  "name": "joel-2",
  "creationDate": "2023-01-23T10:22:32.005245452",
  "paymentMethod": "CASH",
  "amount": 10.0,
  "username": "joel@example.com"
}
```

3.3. Return to the terminal where the application is running. Verify that the debug log message is not displayed. The console does not show debug messages because the default log level is **INFO**.

3.4. Add the following line to the **src/main/resources/application.properties** file.

```
quarkus.log.level = DEBUG
```

3.5. Rerun the request to the same endpoint.

```
[student@workstation monitor-logging]$ curl -s \
http://localhost:8080/expenses/joel-2 | jq
...output omitted...
```

3.6. Verify that the **DEBUG** log message is displayed in the application console.

```
...output omitted...
DEBUG [io.qua.arc.run.BeanContainerImpl] (Quarkus Main Thread) No matching
 bean ...
DEBUG [io.qua.arc.run.BeanContainerImpl] (Quarkus Main Thread) No matching
 bean ...
...output omitted...
DEBUG [com.red.tra.exp.ExpensesResource] (executor-thread-0) Getting expense
 joel-2
```

Note that the application might also log other debug messages that are not relevant for this exercise.

```
...output omitted...
DEBUG [io.qua.arc.run.BeanContainerImpl] (Quarkus Main Thread) ...
DEBUG [io.qua.arc.run.BeanContainerImpl] (Quarkus Main Thread) ...
...output omitted...
DEBUG [com.red.tra.exp.ExpensesResource] (executor-thread-0) Getting expense
 joel-2
```

4. Configure the **DEBUG** log level only for the **com.redhat.training.expense** package.

4.1. In the **application.properties** file, change the root log level from **DEBUG** to **INFO**:

```
quarkus.log.level = INFO
```

4.2. In the same file, add the following line to set the log level of the `com.redhat.training.expense` category to DEBUG.

```
quarkus.log.category."com.redhat.training.expense".level = DEBUG
```

4.3. Rerun the request to the same endpoint.

```
[student@workstation monitor-logging]$ curl -s \
http://localhost:8080/expenses/joel-2 | jq
...output omitted...
```

4.4. Verify that the application logs display only the debug messages generated in the `com.redhat.training.expense` package.

```
...output omitted...
INFO  [io.quarkus] (Quarkus Main Thread) Profile dev activated. Live Coding
 activated.
INFO  [io.quarkus] (Quarkus Main Thread) Installed features: [...]
INFO  [io.qua.dep.dev.RuntimeUpdatesProcessor] (vert.x-worker-thread-0) Live
 reload total time ...
DEBUG [com.red.tra.exp.ExpensesResource] (executor-thread-0) Getting expense
 joel-2
```

5. Customize logging in development mode. Send the logs to a file, define a specific logging format, and deactivate the log rotation when the application restarts.

5.1. Add the following lines to the `application.properties` file.

```
%dev.quarkus.log.file.enable=true
%dev.quarkus.log.file.path=/home/student/D0378/monitor-logging/dev.logs
%dev.quarkus.log.file.format=%d %-5p [%F] %m%n
%dev.quarkus.log.file.rotation.rotate-on-boot=false
```

5.2. Rerun the request to the same endpoint.

```
[student@workstation monitor-logging]$ curl -s \
http://localhost:8080/expenses/joel-2 | jq
...output omitted...
```

5.3. Verify that the `/home/student/D0378/monitor-logging/dev.logs` file contains the logs in the specified format.

```
...output omitted...
2023-01-23 09:11:39,451 DEBUG [ExpensesResource.java] Getting expense joel-2
```

5.4. Return to the terminal where the application is running and press q to stop the application.

Finish

On the **workstation** machine, use the **lab** command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish monitor-logging
```

This concludes the section.

enjoy!

Jose