

# Quarkus Practicing with a Blog Engine

It's just a simple application made of two microservices put together to behave correctly in order to serve a blog engine. The two microservices are : **Posts Microservice** and the **Authors microservice**. Each microservice is autonomous and independant, hence it has its own business logic scope.

- The *Authors microservice* has a REST endpoint that returns the requested author. It is used by the Posts microservice when it needs to return a new post.
- The *Posts microservice* has one endpoint that returns a post.

## Quakrus maven plugin

The plugin allows to create a Quarkus project that scaffolds a basic directory structure, Maven dependencies, some code and test classes. The Quarkus plugin is based on the following Maven coordinates: **io.quarkus:quarkus-maven-plugin**.

To check the available goals and the latest version of it with the following command:

```
mvn -Dplugin=io.quarkus:quarkus-maven-plugin help:describe
```

The first time, maven will download all the dependencies. And then provide a helpful and complete output about the plugin.

Make sure you're using the right platform :

```
mvn quarkus:list-platforms
```

To list the extensions :

```
mvn quarkus:list-extension
```

Then we can look for an extension in order to add it to our project. For example, we look for **jdbc** extension

```

mvn quarkus:list-extensions | grep jdbc
[INFO] Camel JDBC camel-quarkus-jdbc
[INFO] Elytron Security JDBC quarkus-elytron-security-jdbc
[INFO] JDBC Driver - DB2 quarkus-jdbc-db2
[INFO] JDBC Driver - Derby quarkus-jdbc-derby
[INFO] JDBC Driver - H2 quarkus-jdbc-h2
[INFO] JDBC Driver - MariaDB quarkus-jdbc-mariadb
[INFO] JDBC Driver - Microsoft SQL Server quarkus-jdbc-mssql
[INFO] JDBC Driver - MySQL quarkus-jdbc-mysql
[INFO] JDBC Driver - Oracle quarkus-jdbc-oracle
[INFO] JDBC Driver - PostgreSQL quarkus-jdbc-postgresql

```

## Developing the REST Author Microservice

To achieve this microservice, we :

- Implement a REST API using JAX-RS and Quarkus with **quarkus-resteasy**,
- Inject external configuration,
- Customise the JSON Output with JSON-B with **quarkus-resteasy-jsonb**,
- Enable OpenAPI and Swagger UI with **quarkus-smallrye-openapi**,
- Check the health of the REST endpoint with **quarkus-smallrye-health**,
- Creates, validate and CRUD entities with **quarkus-hibernate-orm-panache**, **hibernate-validation** and **quarkus-jdbc-h2**
- Configure Quarkus HTTP port listening.

```

mvn io.quarkus:quarkus-maven-plugin:2.3.1.Final:create \
  -DplatformVersion=2.3.1.Final \
  -DgroupId=org.pfsilga.blogengine \
  -DartifactId=rest-authors \
  -DprojectVersion=1.0-SNAPSHOT \
  -DclassName="org.pfsilga.blogengine.authors.AuthorResource" \
  -Dpath="/api/authors" \
  -Dextensions="resteasy, resteasy-jsonb, hibernate-orm-panache, jdbc-h2, smallrye-openapi, smallrye-health"

```

And we get this listing in the **pom.xml** :

```

<dependencies>
  <dependency>
    <groupId>io.quarkus</groupId>
    <artifactId>quarkus-resteasy-jsonb</artifactId>
  </dependency>
  <dependency>
    <groupId>io.quarkus</groupId>
    <artifactId>quarkus-smallrye-openapi</artifactId>
  </dependency>
  <dependency>
    <groupId>io.quarkus</groupId>
    <artifactId>quarkus-resteasy</artifactId>
  </dependency>
  <dependency>
    <groupId>io.quarkus</groupId>
    <artifactId>quarkus-hibernate-validator</artifactId>
  </dependency>
  <dependency>
    <groupId>io.quarkus</groupId>
    <artifactId>quarkus-jdbc-h2</artifactId>
  </dependency>
  <dependency>
    <groupId>io.quarkus</groupId>
    <artifactId>quarkus-smallrye-health</artifactId>
  </dependency>
</dependencies>

```

## Developing the REST Posts Microservice

To achieve this microservice, we :

- Implement a Posts REST API using JAX-RS and Quarkus **quarkus-resteasy**,
- Generate the JSON Output with JSON-P with **quarkus-jsonb**,
- Invoke the Authors microservice thanks to REST Client with **quarkus-rest-client**,
- Handle fault tolerance with **quarkus-smallrye-fault-tolerance**,
- Add metrics with **quarkus-smallrye-metrics**.

```
mvn io.quarkus:quarkus-maven-plugin:2.3.1.Final:create \
  -DplatformVersion=2.3.1.Final \
  -DprojectId=org.pfsilga.blogengine \
  -DprojectArtifactId=rest-posts \
  -DprojectVersion=1.0-SNAPSHOT \
  -DclassName="org.pfsilga.blogengine.posts.PostResource" \
  -Dpath="/api/posts" \
  -Dextensions="resteasy, resteasy-jsonb, rest-client, smallrye-fault-tolerance,
  smallrye-metrics"
```

And we get this listing in the **pom.xml** almost the same:

```
<dependencies>
  <dependency>
    <groupId>io.quarkus</groupId>
    <artifactId>quarkus-smallrye-fault-tolerance</artifactId>
  </dependency>
  <dependency>
    <groupId>io.quarkus</groupId>
    <artifactId>quarkus-rest-client</artifactId>
  </dependency>
  <dependency>
    <groupId>io.quarkus</groupId>
    <artifactId>quarkus-resteasy-jsonb</artifactId>
  </dependency>
  <dependency>
    <groupId>io.quarkus</groupId>
    <artifactId>quarkus-resteasy</artifactId>
  </dependency>
  <dependency>
    <groupId>io.quarkus</groupId>
    <artifactId>quarkus-smallrye-metrics</artifactId>
  </dependency>
  <dependency>
    <groupId>com.github.javafaker</groupId>
    <artifactId>javafaker</artifactId>
    <version>1.0.2</version>
  </dependency>
</dependencies>
```

We added manually **javafaker** dependency, so we can generate fake posts data.