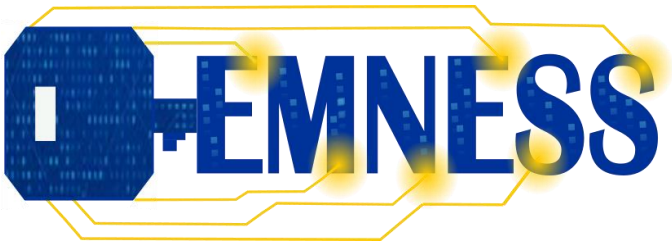


2020-1-FR01-KA203-080184

Development of secure applications based on IoT

WP2.1

Deliverable #2: First App with Quarkus

Tutorial

2022/10/07

Author: Stéphanie CHOLLET

Table of contents

1. Introduction

The purpose of the application is to track sheep in a sheepfold. All the sheep are stored in a database and they are manipulate (create, find, update and delete) via REST services. The application is developed with Quarkus.

In the first part of this tutorial, the project creation and configuration is explained. In the following sections, the tutorial is divided into two parts: JPA persistence to interact with the database and then the development of REST services.

2. Project creation and configuration

2.1. Project creation

From a console in an appropriate directory corresponding to your workspace (*e.g.*, $HOME/workspace):

```
mvn io.quarkus.platform:quarkus-maven-plugin:create
-DprojectGroupId=fr.esisar.sheepfold -DprojectArtifactId=sheepfold
```

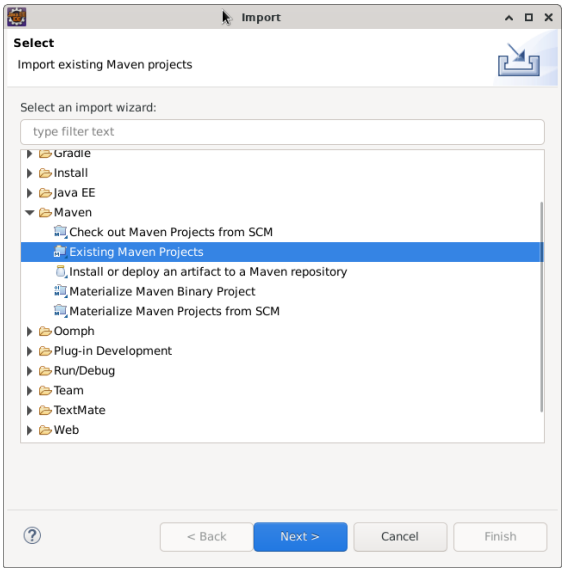
It is possible to run your first app from the directory sheepfold, created in your workspace:

```
mvn quarkus:dev
```

The application is available, by default, at: http://localhost:8080

2.2. Import project into Eclipse

To import the created project into Eclipse, from the File menu > Import, then in the Maven directory > Existing Maven Projects



The project tree for the sheepfold application is illustrated by Figure 1. It is a classical Maven project with dedicated repositories for source and test files. The application.properties file contains the project configuration.

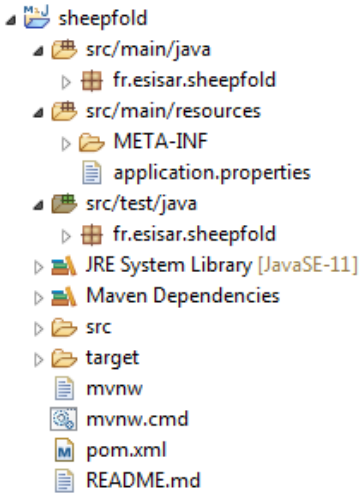


Figure 1: Project tree.

3. Data persistence

3.1. Project configuration for data persistence

For this application, we use a memory database, named H2. It is required to add an H2 extension for Quarkus to use this database. In addition, we add extensions to use Hibernate with Panache and Hibernate Validator:

```
mvn quarkus:add-extension -Dextensions="quarkus-hibernate-orm-panache, quarkus-hibernate-validator, quarkus-jdbc-h2"
```

In the src/main/resources/application.properties file, add the database configuration:

```
# H2 database configuration
quarkus.datasource.db-kind=h2
quarkus.datasource.jdbc.url=jdbc:h2:mem:default;DB_CLOSE_DELAY=-1
quarkus.hibernate-orm.database.generation=drop-and-create
quarkus.hibernate-orm.log.sql=true
```

3.2. Data model

The data model for the sheepfold application is presented in Figure 2. In this model, we find four entities (Bell, Sheep, Dog and Enclosure). They are bidirectional associations between these different

entities: a sheep can have a bell, a sheep can be in an enclosure and in an enclosure it could have many sheep, a sheep is guarded by dogs and dogs can guard many sheep.

Each entity is a PanacheEntity from Quarkus and has a set of public attributes including an id (coming from the PanacheEntity superclass).

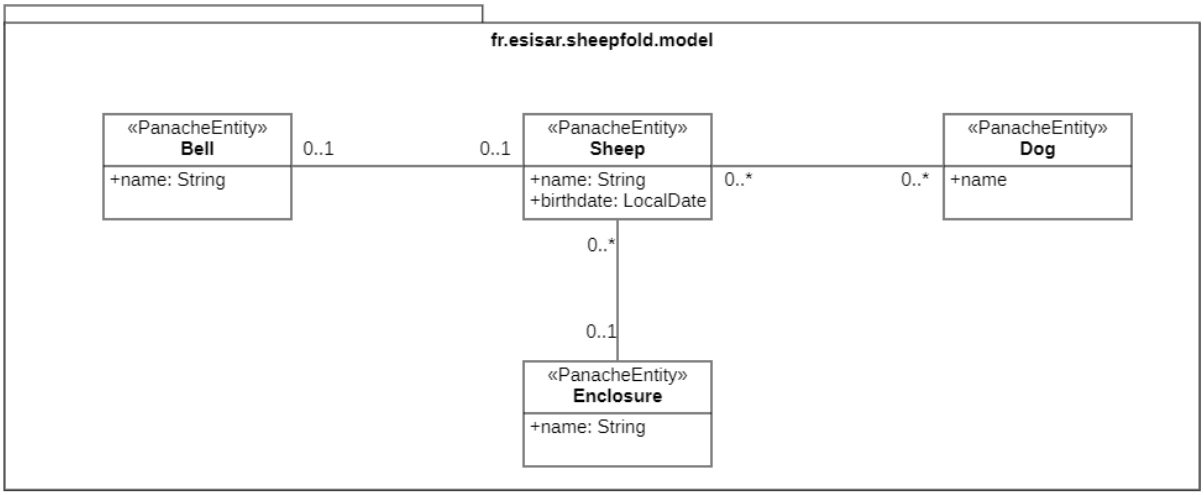


Figure 2: Data model.

In practice, these entities are Java classes containing only attributes and extending the PanacheEntity class. The mapping between Java classes and tables in database is made with JPA. The expected database schema is illustrated in Figure 3.

$$Sheep(\underline{id}, name, birthdate, \#bell\_id, \#enclosure\_id)$$

$$Bell(\underline{id}, name)$$

$$Enclosure(\underline{id}, name)$$

$$Dog(\underline{id}, name)$$

$$Sheep\_Dog(\#sheep\_id, \#dogs\_id)$$

$$Sheep[bell\_id] \subseteq Bell[id]$$

$$Sheep[enclosure\_id] \subseteq Enclosure[id]$$

$$Sheep\_Dog[sheep\_id] \subseteq Sheep[id]$$

$$Sheep\_Dog[dogs\_id] \subseteq Dog[id]$$

Figure 3 : Relational database schema for H2 database.

3.2.1. First entity: Sheep

To define the Sheep entity, extended PanacheEntity, annotate it with @Entity and add your columns as public fields (no need to have getters and setters). The Sheep entity should look like this:

```java
package fr.esisar.sheepfold.model;

import java.time.LocalDate;

import javax.persistence.Column;
import javax.persistence.Entity;

import io.quarkus.hibernate.orm.panache.PanacheEntity;

@Entity
public class Sheep extends PanacheEntity {

	@Column(unique = true, nullable = false)
	public String name;

	public LocalDate birthdate;
}
```

The attribute *name* is annotated with @Column indicating that the value must be unique (unique = true) and not null (nullable = false).

With Quarkus, every change to your application, such as database schema, will be recreated and your data (in import.sql file) will be used to repopulate the database without restarting the application. You can verify the creation of Sheep table in the Quarkus Web administration (http://localhost:8080) in