Table of Contents

1. Quarkus Sampler Documentation

[Compile status] [Maven central version of Parent pom project] [License of Parent pom project] [Use JakartaEE project] [Commits] [Supported jvms] [GitHub Repo Stars]

1.1. General

The purpose of the project is to provide a working example for the general structure of a Quarkus project.

NOTE	Documentation: `https://i-cell-mobilsoft-open-source.github.io/quarkus-sampler/`

1.2. Used technologies

- Coffee 2.0.0+
  "Java EE solution set aimed at gathering common algorithms used in the enterprise world, providing fundamental solutions that can be customized to meet specific needs." (https://github.com/i-Cell-Mobilsoft-Open-Source/coffee)
- Java 17+
- Maven 3.8.2+
- Jakarta EE 10+
- CDI 4.0+
- Microprofile 6.0+

- Quarkus 3.16.4+
- Tracing with grafana

1.3. Projekt structure

- `/.github` - GitHub settings
- `/docs` - Repository of documentation
- `/etc` - Configurations, development environment scripts, etc.
  - `/etc/config` - Configurations
    - `/etc/config/grafana/**` - Grafana dashboards, datasources.
    - `/etc/config/prometheus/**` - Prometheus configuration for localhost
  - `/etc/docker-compose` - Docker Compose files and their corresponding Dockerfiles.
  - `/etc/release` - Docker release management assistance in GitHub CI.

2. Useful commands and accesses

Commands used for development purposes, which are used to build and start development environments.

The application can be started in several ways:

- Starting Quarkus dev with Maven
- Creating a Quarkus uber-jar and running this jar file using `java -jar`
  - Placing the same jar into a Java Docker image and running it (using local.Dockerfile)
- Certain IDEs natively include a Quarkus project recognition module that automatically creates a run configuration, which can also be used to start the project.

Docker-compose is used for creating and running Docker images.

NOTE	The project includes a sampler-rest service that demonstrates the usage of the module. This example can run entirely on a local development machine. Therefore, it has no external dependencies.

2.1. Starting the qs-rest-service server in various ways

IDE included Quarkus run config

```
Several browsers natively support Quarkus similarly to Spring Boot projects,
recognizing it and creating their own run configurations.
```

Maven quarkus:dev

```
mvn clean compile quarkus:dev
```

IMPORTANT	The project does not consist of a single module as Quarkus would expect, hence requiring compilation.

NOTE	Using the Quarkus Maven plugin, the project can be started in dev mode, activating several development tools. For more information: https://quarkus.io/guides/dev-mode-differences.

Running Quarkus uber-jar in docker

```
mvn clean install ①
docker-compose -f <PROJECT_PATH>/quarkus-sampler/etc/docker-compose/docker-
compose.local.qs-rest-service.yml up --build --force-recreate ②
```

① Therefore, it is necessary for the jar that goes into the Docker image to be built.

② Running the docker-compose command in the project root initiates the Docker Compose build (forcing recreation with the --force-recreate parameter if necessary) and starts up the services (up).

2.2. qs-rest-service native

Quarkus native

```
mvn install -Pnative -Dnative -Dquarkus.native.additional-build-args="--initialize-at
-run-time=hu.icellmobilsoft.coffee.tool.utils.string.RandomUtil","--initialize-at-run
-time=io.grpc.internal.RetriableStream","--initialize-at-run
-time=hu.icellmobilsoft.coffee.se.util.string.RandomUtil","--add-opens
java.base/java.net=ALL-UNNAMED"
```

Quarkus native in docker

```
docker compose -f <PROJECT_PATH>/quarkus-sampler/etc/docker-compose/docker-
compose.local.native.qs-rest-service.yml up --build --force-recreate
```

3. Possibilities for project configuration settings

3.1. Quarkus-based configurations

Since the application is based on Quarkus, it utilizes Quarkus' foundational settings.

The description can be found here: https://quarkus.io/guides/all-config

NOTE	Only those elements from the configuration list are active which are included in the project at the dependency level.

Key elements that are already defined by default in the project:

Quarkus config key	Description	Env variable	Default value
quarkus.arc.remove-unused-beans	Arc setting - remove unused beans: Link	-	false
quarkus.log.category."hu.icellmobilsoft".level	hu.icellmobilsoft category log level	SAMPLER_LOG_HU_ICELLMOBILSOFT_LEVEL	INFO
quarkus.log.console.json	Json logging enable	SAMPLER_LOG_CONSOLE_JSON_ENABLED	false
quarkus.log.console.format	Console log format	-	`%d{yyyy-MM-dd HH:mm:ss.SSS} %-5p [thread:%t] [%c{10}] [sid:%X{extSessionId}] - %s%E%n`
quarkus.log.handler.gelf.additional-field."moduleVersion".value	Gelf log - **moduleVersion** additional-field value	SAMPLER_LOGSTASH_MODULE_VERSION	unknown
quarkus.log.handler.gelf.additional-field."moduleId".value	Gelf log - **moduleId** additional-field value	SAMPLER_LOGSTASH_MODULE_ID	unknown
quarkus.log.handler.gelf.additional-field."K8S_NAMESPACE".value	Gelf log - **K8S_NAMESPACE** additional-field value	SAMPLER_LOGSTASH_K8S_NAMESPACE	unknown
quarkus.handler.gelf.include-full-mdc	Gelf log - Whether to include all fields from the MDC.	SAMPLER_LOGSTASH_K8S_NAMESPACE	false
quarkus.handler.gelf.enabled	Gelf log - Enable it	SAMPLER_LOGSTASH_ENABLED	false
quarkus.handler.gelf.host	Gelf log - tcp host link of logstasth	SAMPLER_LOGSTASTH_HOST	tcp:localhost
quarkus.handler.gelf.port	Gelf log - port of logstasth	SAMPLER_LOGSTASTH_PORT	12201
quarkus.handler.gelf.version	Gelf log - version of logstasth's communication	SAMPLER_LOGSTASTH_VERSION	1.1
quarkus.log.level	Quarkus log level: Link	SAMPLER_LOG_LEVEL	INFO

Quarkus config key	Description	Env variable	Default value
quarkus.log.min-level	Quarkus min log level: Link	SAMPLER_LOG_MIN_LEVEL	ALL
quarkus.otel.exporter.otlp.endpoint	Otel endpoint: Link	-	http://opentelemetry-collector:4317
quarkus.package.add-runner-suffix	Quarkus package add runner suffix: Link	-	false
quarkus.package.type	Quarkus package type: Link	-	fast-jar
quarkus.smallrye-openapi.info-title	Openapi - info title : Link	-	Quarkus sampler service
quarkus.smallrye-openapi.info-version	Quartz - info version : Link	-	${quarkus.application.version}
quarkus.smallrye-openapi.info-description	Quartz - info version : Link	-	REST endpoints for operations. General responses in case of error: * __400__ - Bad Request * __401__ - Unauthorized * __404__ - Not found * __418__ - Database object not found * __500__ - Internal Server Error
quarkus.swagger-ui.enable	Enable swagger ui: Link	-	false

3.2. Coffee-based configurations

In addition, due to the use of the Coffee Toolset, the application contains additional configuration options.

Coffee config key	Description	Env variable	Default value
coffee.app.name	Coffee app name in logs	-	${quarkus.application.name}

Coffee config key	Description	Env variable	Default value
coffee.config.resource.bundles	Resource bundles' config for i18n	-	i18n.common-messages,i18n.messages
coffee.config.xml.catalog.path	Catalog path of Super catalog.xml	-	xsd/hu/icellmobilsoft/quarkus/sampler/dto/super.catalog.xml

3.3. Microprofile Openapi configuration

Furthermore, related to Coffee, it also includes a MicroProfile OpenAPI filter configuration.

MP Openapi config key	Description	Env variable	Default value
mp.openapi.filter	Microprofile openapi filter class with package	-	hu.icellmobilsoft.quarkus.sampler.common.rest.filter.OpenAPIFilter

3.4. Agroal configurations

The database connection management is similar to the pool managed by WildFly IronJacamar. Configuration and tuning options:

- QUARKUS_DATASOURCE_ICELLMOBILSOFTDS_JDBC_MIN_SIZE: The minimum number of connections. It is worth setting this if we need continuously initialized database connections.
- QUARKUS_DATASOURCE_ICELLMOBILSOFTDS_JDBC_MAX_SIZE: The maximum number of connections. It is essential to set an upper limit for the pool. Its size should be tuned according to the database-side maximum connection limits and the number of application instances.
- QUARKUS_DATASOURCE_ICELLMOBILSOFTDS_JDBC_INITIAL_SIZE: Specifies the number of pre-initialized connections at service startup. This is useful when preparing for burst loads or peak periods.
- QUARKUS_DATASOURCE_ICELLMOBILSOFTDS_JDBC_ACQUISITION_TIMEOUT: This operates similarly to the parameter in this script. This defines the time the pool has to provide a database connection. If the pool is full and the time expires, an exception will be thrown.
- QUARKUS_DATASOURCE_ICELLMOBILSOFTDS_JDBC_IDLE_REMOVAL_INTERVAL: The duration for which an initialized but idle connection remains in the pool. This parameter can be tuned to ensure that unnecessary active connections do not burden the database.
- QUARKUS_DATASOURCE_ICELLMOBILSOFTDS_JDBC_NEW_CONNECTION_SQL: For example, setting the schema or character encoding.
- QUARKUS.TRANSACTION-MANAGER.DEFAULT-TRANSACTION-TIMEOUT: The transaction timeout. This differs significantly from WildFly's 5-minute default, being set to 1 minute by default. This is the time a transaction has to complete.

Useful metrics have been implemented for a sample dashboard. TODO: Define and interpret the metrics.

4. Release notes

4.1. Quarkus Sampler 0.1.0

4.1.1. Release note

- Base project with Quarkus 3.16.x
- Documentations
- GitHub workflows for docker build and release build