Red Hat build of Quarkus 3.8

Configuring your Red Hat build of Quarkus applications by using a YAML file

Last Updated: 2024-05-09

Legal Notice

Abstract

This guide describes how to configure Red Hat build of Quarkus applications by using a YAML file.

Table of Contents

PROVIDING FEEDBACK ON RED HAT BUILD OF QUARKUS DOCUMENTATION

To report an error or to improve our documentation, log in to your Red Hat Jira account and submit an issue. If you do not have a Red Hat Jira account, then you will be prompted to create an account.

Procedure

1. Click the following link to create a ticket.
2. Enter a brief description of the issue in the **Summary**.
3. Provide a detailed description of the issue or enhancement in the **Description**. Include a URL to where the issue occurs in the documentation.
4. Clicking **Submit** creates and routes the issue to the appropriate documentation team.

MAKING OPEN SOURCE MORE INCLUSIVE

Red Hat is committed to replacing problematic language in our code, documentation, and web properties. We are beginning with these four terms: master, slave, blacklist, and whitelist. Because of the enormity of this endeavor, these changes will be implemented gradually over several upcoming releases. For more details, see our CTO Chris Wright's message .

CHAPTER 1. CONFIGURING YOUR RED HAT BUILD OF QUARKUS APPLICATIONS BY USING A YAML FILE

As an application developer, you can use Red Hat build of Quarkus to create microservices-based applications written in Java that run on OpenShift Container Platform and serverless environments. Applications compiled to native executables have small memory footprints and fast startup times.

Apply structured configuration by updating the **application.yaml** file to configure your Quarkus application.

NOTE

Alternatively, you can configure your Quarkus application by setting properties in the **application.properties** file. For more information, see Setting configuration properties.

The procedures include configuration examples created using the Quarkus **config-quickstart** exercise.

NOTE

For a completed example of the getting started exercise, download the Quarkus Quickstarts archive or clone the Quarkus Quickstarts Git repository and go to the **getting-started** directory.

Prerequisites

- You have installed OpenJDK 17 or 21 and set the **JAVA_HOME** environment variable to specify the location of the Java SDK.
  - To download Red Hat build of OpenJDK, log in to the Red Hat Customer Portal and go to Software Downloads.
- You installed Apache Maven 3.8.6 or later.
  - Download Maven from the Apache Maven Project website.
- You have configured Apache Maven to use artifacts from the Quarkus Maven repository.
  - To learn how to configure Apache Maven settings, see Getting started with Quarkus.

1.1. RED HAT CONFIGURATION OPTIONS

You can use configuration options to change the settings of your application in a single configuration file. Red Hat build of Quarkus supports configuration profiles that you can use to group related properties and switch between profiles as required.

By default, Quarkus reads properties from the **application.properties** file located in the **src/main/resources** directory. You can also configure Quarkus to read properties from a YAML file instead.

When you add the **quarkus-config-yaml** dependency to your project **pom.xml** file, you can configure and manage your application properties in the **application.yaml** file. For more information, see Adding YAML configuration support.

Red Hat build of Quarkus also supports MicroProfile Config, which you can use to load your application's configuration from other sources.

You can use the MicroProfile Config specification from the Eclipse MicroProfile project to inject configuration properties into your application and access them by using a method defined in your code.

Quarkus can also read application properties from different origins, including the following sources:

- The file system
- A database
- A Kubernetes or OpenShift Container Platform **ConfigMap** or Secret object
- Any source that a Java application can load

1.2. ADDING YAML CONFIGURATION SUPPORT

Red Hat build of Quarkus supports YAML configuration files through the **SmallRye Config** implementation of Eclipse MicroProfile Config. You can add the **Quarkus Config YAML** extension and use the YAML configuration file over the properties file for configuration. Quarkus supports the use of **application.yml** and **application.yaml** as the name of the YAML file.

The YAML configuration file takes precedence over the **application.properties** file. To avoid errors, you can delete the **application.properties** file and use only one type of configuration file.

Procedure

1. Use one of the following methods to add the YAML extension to your project:
   - Open the **pom.xml** file and add the **quarkus-config-yaml** extension as a dependency:

     Example pom.xml file

     ```
     <dependency>
         <groupId>io.quarkus</groupId>
         <artifactId>quarkus-config-yaml</artifactId>
     </dependency>
     ```

   - To add the **quarkus-config-yaml** extension from the command line, enter the following command from your project directory:

     Add quarkus-config-yaml extension

     ```
     ./mvnw quarkus:add-extension -Dextensions="quarkus-config-yaml"
     ```

1.2.1. Using nested object configuration with YAML

You can define nested configuration properties within the existing ones for your Red Hat build of Quarkus application by using the **application.yaml** configuration file.

Prerequisites

- You have a Quarkus Maven project.

- You have a PostgreSQL data source.
- You have the following extensions as dependencies in the **pom.xml** file of your project:
  - **quarkus-resteasy-client**
  - **quarkus-jdbc-postgresql**
  - **quarkus-config-yaml**

Procedure

1. Open the **src/main/resources/application.yaml** configuration file.
2. Add the nested class configuration properties to your **application.yaml** file, as shown in the following example:

   Example application.yaml file

```
# Properties that configure the JDBC data source driver of your PostgreSQL data source
quarkus:
  datasource:
    db-kind: postgresql
    jdbc:
      url: jdbc:postgresql://localhost:5432/quarkus_test
    username: quarkus_test
    password: quarkus_test

# Property that configures the URL of the endpoint to which the REST client sends requests
quarkus:
  rest-client:
    org.acme.rest.client.ExtensionsService:
      url: https://stage.code.quarkus.io/api

# Property that configures the log message level for your application
# For configuration property names that use quotes, do not split the string inside the quotes
quarkus:
  log:
    category:
      "io.quarkus.category":
        level: INFO
```

WARNING

For production, do not set the username and password in the configuration file, as shown in the preceding example. This was only for illustration purposes. Instead set them in your environmental variables. For more information, see Setting configuration properties section of the "Configuring your Red Hat build of Quarkus applications by using a properties file" guide.

Similar to the **application.properties** file, you can use comments to describe your configuration properties in YAML format.

NOTE

Always use spaces to indent the properties in your YAML configuration file. YAML does not support using tabs for indentation.

1.2.2. Setting custom configuration profiles with YAML

With Quarkus, you can set configuration properties and values that are specific to different configuration profiles of your application. You can start your application with a specific profile to access a particular configuration. This procedure shows how you can provide a configuration for a specific profile in YAML format.

Prerequisites

- You have a Quarkus Maven project configured to use a PostgreSQL data source with a JDBC data source driver.
- You have the **quarkus-jdbc-postgresql** and **quarkus-config-yaml** extensions as dependencies in your project's **pom.xml** file.

Procedure

1. Open your project's configuration file, **src/main/resources/application.yaml**.
2. To set a profile-dependent configuration, add the profile name before defining the key-value pairs by using the **"%<profile_name>"** syntax. Ensure that you place the profile name inside quotation marks.

   TIP

   In YAML, you must place all strings that begin with a special character inside quotation marks.

   In the following example, the PostgreSQL database is configured to be available at the **jdbc:postgresql://localhost:5432/some-database** URL when you start your Quarkus application in development mode:

   src/main/resources/application.yaml

```
"%dev":
  quarkus:
    datasource:
      db-kind: postgresql
        jdbc:
          url: jdbc:postgresql://localhost:5432/quarkus_test
        username: quarkus_test
        password: quarkus_test
```

WARNING

For production, do not set the username and password in the configuration file, as shown in the preceding example. This was only for illustration purposes. Instead set them in your environmental variables. For more information, see Setting configuration properties section of the "Configuring your Red Hat build of Quarkus applications by using a properties file" guide.

1.3. PROPERTY EXPRESSIONS

You can combine property references and text strings into property expressions and use these expressions as values in your Quarkus configuration.

Like variables, property expressions substitute configuration property values dynamically, avoiding hard-coded values.

You can expand an expression in one configuration source with a value defined in another.

The application resolves a property expression when **java.util.Properties** reads the property value from a configuration source: at compile time if read then, and at runtime if overridden at that point.

If the application cannot resolve the value of a property in an expression, and the property does not have a default value, your application throws a **NoSuchElementException** error.

1.3.1. Example: Property expressions in a YAML file

The following example shows how to use property expressions for flexible configuration of your Quarkus application.

Example application.yaml file

```yaml
mach: 3
x:
  factor: 2.23694

display:
  mach: ${mach}
  unit:
    name: "mph"
    factor: ${x.factor}
```

NOTE

To reference nested properties, use the **.** (dot) separator, as in **{x.factor}**.

Additional resources

- For more information about property expressions, see Property expressions.

- For an example usage of property expressions in a properties file, see Example usage of property expressions.

1.4. EXTERNAL APPLICATION.YAML FILE FOR CONFIGURING PROPERTIES AT RUNTIME

To configure your application properties at runtime, add your **application.yaml** file to the **config** directory.

When **config/application.yaml** and **src/main/resources/application.yaml** share properties, values from **config/application.yaml** override those in **src/main/resources/application.yaml**.

Ensure that the **config/application.yaml** file is in the root of the working directory relative to the Quarkus application runner, as outlined in the following example:

```
├── config
│   └── application.yaml
├── my-app-runner
```

Additional resources

- For more information about adding YAML configuration support, see Adding YAML configuration support.

1.5. MANAGING CONFIGURATION PROPERTY CONFLICTS

Structured formats such as YAML only support a subset of the possible configuration namespace. The following procedure shows how to resolve a conflict between two configuration properties, **quarkus.http.cors** and **quarkus.http.cors.methods**, where one property is the prefix of another.

Prerequisites

- You have a Quarkus project that is configured to read YAML configuration files.

Procedure

1. Open your YAML configuration file.
2. To define a YAML property as a prefix of another property, add a tilde (~) in the scope of the property as shown in the following example:

   Example of defining a YAML property as a prefix

   ```
   quarkus:
     http:
       cors:
         ~: true
         methods: GET,PUT,POST
   ```

3. To compile your Quarkus application in development mode, enter the following command from the project directory:

   Compile your application

```
./mvnw quarkus:dev
```

NOTE

You can use YAML keys for conflicting configuration keys at any level because they are not included in the assembly of the configuration property name.

1.6. ADDITIONAL RESOURCES

- Configuring your Quarkus applications by using a properties file