

# Quarkus - Command Mode Applications

This reference covers how to write applications that run and then exit.

## Writing Command Mode Applications

There are two different approaches that can be used to implement applications that exit.

1. Implement `QuarkusApplication` and have Quarkus run this method automatically
2. Implement `QuarkusApplication` and a Java main method, and use the Java main method to launch Quarkus

In this document the `QuarkusApplication` instance is referred to as the application main, and a class with a Java main method is the Java main.

The simplest possible command mode application might appear as follows:

```
import io.quarkus.runtime.QuarkusApplication;
import io.quarkus.runtime.annotations.QuarkusMain;

@QuarkusMain ①
public class HelloWorldMain implements QuarkusApplication {
    @Override
    public int run(String... args) throws Exception { ②
        System.out.println("Hello World");
        return 10;
    }
}
```

① The `@QuarkusMain` annotation tells Quarkus that this is the main entry point.

② The `run` method is invoked once Quarkus starts, and the application stops when it finishes.

If we want to use a Java main to run the application main it would look like:

```
import io.quarkus.runtime.Quarkus;
import io.quarkus.runtime.annotations.QuarkusMain;

@QuarkusMain
public class JavaMain {

    public static void main(String... args) {
        Quarkus.run>HelloWorldMain.class, args);
    }
}
```

This is effectively the same as running the `HelloWorldMain` application main directly, but has the advantage it can be run from the IDE.



It is recommended that a Java main perform very little logic, and just launch the application main. In development mode the Java main will run in a different `ClassLoader` to the main application, so may not behave as you would expect.

## Multiple Main Methods

It is possible to have multiple main methods in an application, and select between them at build time. The `@QuarkusMain` annotation takes an optional 'name' parameter, and this can be used to select the main to run using the `quarkus.package.main-class` build time configuration option. If you don't want to use annotations this can also be used to specify the fully qualified name of a main class.

By default the `@QuarkusMain` with no name (i.e. the empty string) will be used, and if it is not present and `quarkus.package.main-class` is not specified then Quarkus will automatically generate a main class that just runs the application.



The `name` of `@QuarkusMain` must be unique (including the default of the empty string). If you have multiple `@QuarkusMain` annotations in your application the build will fail if the names are not unique.

## The command mode lifecycle

When running a command mode application the basic lifecycle is as follows:

1. Start Quarkus
2. Run the `QuarkusApplication` main method
3. Shut down Quarkus and exit the JVM after the main method returns

Shutdown is always initiated by the application main thread returning. If you want to run some logic on startup, and then run like a normal application (i.e. not exit) then you should call `Quarkus.waitForExit` from the main thread (A non-command mode application is essentially just running an application that just calls `waitForExit`).

If you want to shut down a running application and you are not in the main thread then you should call `Quarkus.asyncExit` in order to unblock the main thread and initiate the shutdown process.

## Dev mode

Also for command mode applications the dev mode is supported. When running `mvn compile quarkus:dev`, the command mode application is executed and on press of the Enter key, is restarted.

As command mode applications will often require arguments to be passed on the commandline, this is also possible in dev mode via:

```
mvn compile quarkus:dev -Dquarkus.args='--help'
```

The same can be achieved with Gradle:

```
./gradlew quarkusDev --quarkus-args='--help'
```