

# vert.x Maven Plugin

Kamesh Sampath,Roland Huß

Version 1.0-SNAPSHOT, 2016-11-18

# Vert.X Maven Plugin

|  |    |
|--|----|
| 1. Introduction .....  | 2  |
| 1.1. Packaging .....   | 2  |
| 1.2. Running .....   | 2  |
| 1.2.1. In foreground using <b>vertx:run</b> .....                      | 2  |
| 1.2.2. As a daemon using <b>vertx:start</b> .....                      | 3  |
| 2. Common Configurations .....   | 4  |
| 3. Common Run Configurations .....                                     | 5  |
| 4. Maven Goals .....   | 6  |
| 4.1. <b>vertx.package</b> .....  | 6  |
| 4.1.1. Configuration .....   | 6  |
| 4.1.2. How to add this goal my maven project ? .....                   | 6  |
| 4.2. <b>vertx:run</b> .....  | 6  |
| 4.2.1. How to add this goal my maven project ? .....                   | 6  |
| 4.3. <b>vertx:start</b> .....  | 6  |
| 4.3.1. How to add this goal my maven project ? .....                   | 7  |
| 4.4. <b>vertx:stop</b> .....   | 7  |
| 4.4.1. How to add this goal my maven project ? .....                   | 8  |
| 5. Example Configurations .....  | 9  |
| 5.1. vert.x:package Examples .....                                     | 9  |
| 5.2. vert.x:run Examples .....   | 9  |
| 5.2.1. run goal with vert.x redeploy enabled .....                     | 9  |
| 5.2.2. run goal with vert.x redeploy enabled with custom pattern ..... | 10 |
| 5.3. vert.x:start Examples .....                                       | 11 |
| 5.3.1. start goal with defaults .....                                  | 11 |
| 5.3.2. start goal with custom application id .....                     | 11 |
| 5.3.3. start goal with custom java options .....                       | 11 |
| 5.4. vert.x:stop Examples .....  | 12 |
| 5.4.1. stop with no additional configuration .....                     | 12 |
| 5.4.2. stopping one or more application .....                          | 12 |
| 5.5. References .....  | 13 |



# Chapter 1. Introduction

This is a Maven plugin for packaging and running [vert.x](#) applications. The plugin tries to add some intelligence to build in adding some common options automatically based on conventions, for an example if you add file called *application.json* or *application.yml* or *application.yml* in your `src/main/conf` folder of the maven sources, the file gets automatically added as `vertx -conf` option during [vertx:package](#) or [vertx:start](#) goals.

## 1.1. Packaging

The `vert.x` application needs to be packaged as fat or uber jar, which can then be executed using `java -jar <my-app-fat.jar>`. The packaging also need to add certain **MANIFEST.MF** entries that controls the `vert.x` application launching. The plugin takes care of adding the required entries to the **MANIFEST.MF** with values configured using the [Common Configuration](#)

The following are the Manifest entries that will be added based on the configuration elements:

Table 1. Package configuration

| Property                    | Manifest Attribute | Remarks  |
|-----------------------------|--------------------|--|
| <code>vertx.verticle</code> | Main-Verticle      |  |
| <code>vertx.launcher</code> | Main-class         | If custom launcher is used the Main-Class is always defaulted to <code>io.vertx.core.Launcher</code> |

To package the project as `vert.x` application, execute the following maven command

```
mvn clean package vertx:package
```

For more information on packaging refer to [vertx:package](#)

## 1.2. Running

The plugin allows running `vert.x` applications in following ways:

### 1.2.1. In foreground using [vertx:run](#)

The `vert.x` application could also be run without building the fat jars, typically during the development mode. The [vertx:run](#) goal helps in running the application. The application will be run in as forked process with additional configurations based on the goals.

To run the application `vert.x` application in foreground, execute the following maven command

```
mvn clean install vertx:run
```

For more information on configuration options refer to [vertx:run](#)

### 1.2.2. As a daemon using [vertx:start](#)

This allows the application be run using the uber or fat jar as background process. This option also attaches a configurable id or an autogenerated id which could be used to stop the process using [vertx:stop](#)

To start the application vert.x application in background, execute the following maven command

```
mvn clean install vertx:start
```

For more information on configuration options refer to [vertx:start](#)

To stop the application vert.x application running in background, execute the following maven command

```
mvn vertx:stop
```

For more information on configuration options refer to [vertx:stop](#)

# Chapter 2. Common Configurations

All goals share the following configuration:

*Table 2. Package configuration*

| Element  | Description               | Default                | Property       |
|----------|---------------------------|------------------------|----------------|
| verticle | Main verticle to start up |                        | vertx.verticle |
| launcher | Vert.x launcher to use    | io.vertx.core.Launcher | vertx.launcher |

# Chapter 3. Common Run Configurations

These are the common configuration shared by the run based goals such as **run**, **start** and **stop**.

Table 3. Run configuration

| Element           | Description   | Property        | Default  |
|-------------------|---|-----------------|--|
| classes Directory | The project classes directory where the sources will be compiled in to.   |                 | <code>\${project.build.outputDirectory}</code>   |
| config            | the application configuration file path that will be passed to the vertx launcher as <b>-conf</b> . If a yaml file is configured then it will be converted to json by the plugin. The converted file will be saved in <code>\${project.outputDir}/conf</code> directory | vertx.config    | <code>\${basedir}/src/main/\${project.artifactId}.json</code> or <code>\${basedir}/src/main/\${project.artifactId}.yaml</code> or <code>\${basedir}/src/main/\${project.artifactId}.yml</code> |
| fork              | Controls whether the application will be run in forked mode or non-forked mode  | fork            | false  |
| redeploy          | controls whether vertx redeploy is enabled  |                 | false  |
| redeployPatterns  | The ant based pattern for scanning changes for redeployment. If redeploy is <b>true</b> and redeployPatterns is empty then a default value is will be applied. This is list of values following standard maven list/array configuration                                 |                 | <code>src/main/**/*.java</code>  |
| workDirectory     | The working directory of the running process of the application   | vertx.directory | <code>\${project.basedir}</code>   |



Right now the plugin supports only file based vert.x configuration

# Chapter 4. Maven Goals

This plugin supports the following goals which are explained in detail in the next sections.

Table 4. Plugin Goals

| Goal                          | Description  |
|-------------------------------|--|
| <a href="#">vertx.package</a> | Package Vert.x applications                              |
| <a href="#">vertx:run</a>     | Run a Vert.x application in foreground                   |
| <a href="#">vertx:start</a>   | Run a Vert.x application in daemon mode with specific id |
| <a href="#">vertx:stop</a>    | Stops the vert.x applicaiton running in daemon mode      |

## 4.1. vertx.package

This goal will package a vert.x application as fat or uber jar with its dependencies bundled as part of the jar

### 4.1.1. Configuration

The package goal does not have any additional configuration other the ones mentioned in [Common Configuration](#)

### 4.1.2. How to add this goal my maven project ?

You can see [Examples](#) on how to to add the start goal to your maven project

## 4.2. vertx:run

This goal allows to run the vert.x application as part of the maven build. The application is always run the vert.x application as a forked process.

The goal does not have any exclusive configuration, [Common Run Configuration](#) defines all the applicable configurations for the goal

### 4.2.1. How to add this goal my maven project ?

You can see [Examples](#) on how to to add the start goal to your maven project



Currently redeploy pattern is added, but redeploy compilation is work in progress

## 4.3. vertx:start

This goal allows to start the vert.x application as a background process from maven build. This goal triggers the vert.x `start` command, passing the configuration values as mentioned below.



Table 5. Run configuration

| Element   | Description   | Property            | Default   |
|-----------|---|---------------------|---|
| timeout   | The time in seconds that will be used to check if the application has started   | vertx.start.timeout | 10  |
| startMode | The property to decide how the vert.x application will be started in background. The application can be started in <b>jar</b> mode in which the application will be packaged as fat jar and started, or can be run in <b>exploded</b> mode where the application will be launched with exploded <i>classesDirectory</i> and maven dependencies to the classpath | vertx.start.mode    | jar   |
| appId     | The application id that will added as <b>-id</b> option to the vert.x start command   | vertx.app.id        | If this is not passed a default uuid will be generated and set as appId |
| jvmArgs   | The Java Options that will be used when starting the application, these are the values that are typically passed to vert.x applications using --java-opts   | vertx.jvmArguments  |   |

Apart from the above list of exclusive start configuration, the goal shares the common **Common Run Configuration** with the following configuration ignored by the goal,

- redeploy
- redeployPatterns
- classesDirectory - if run in jar mode
- fork - by default every start is a forked process

#### 4.3.1. How to add this goal my maven project ?

You can see **Examples** on how to to add the start goal to your maven project

## 4.4. vertx:stop

This goal allows to stop the vert.x application running as background process from maven build. This goal triggers the vert.x **stop** command, passing the configuration values as mentioned below.

Table 6. Run configuration

| Element | Description   | Property           | Default |
|---------|---|--------------------|---------|
| timeout | The time in seconds that will be used to check if the application has stopped | vertx.stop.timeout | 10      |

| Element | Description  | Property | Default  |
|---------|--|----------|--|
| appIds  | The application id's that will stopped using the vert.x stop command |          | If this is not passed, the vertx-start-proc.id file present workingDirectory will be read for the application id |

Apart from the above list of exclusive start configuration, the goal shares the common **Common Run Configuration** with the following configuration ignored by the goal,

- redeploy
- redeployPatterns
- fork - by default every stop is a forked process

#### 4.4.1. How to add this goal my maven project ?

You can see **Examples** on how to to add the start goal to your maven project

# Chapter 5. Example Configurations

The following sections shows example plugin snippets for the goals provided by the plugin.



please update the plugin version as needed

## 5.1. vert.x:package Examples

```
---
<plugin>
  <groupId>org.workspace7.maven.plugins</groupId>
  <artifactId>vertx-maven-plugin</artifactId>
  <version>1.0.0-SNAPSHOT</version>
  <executions>
    <execution>
      <phase>package</phase>
      <goals>
        <goal>package</goal>
      </goals>
    </execution>
  </executions>
</plugin>
---
```

## 5.2. vert.x:run Examples

```
---
<plugin>
  <groupId>org.workspace7.maven.plugins</groupId>
  <artifactId>vertx-maven-plugin</artifactId>
  <version>1.0.0-SNAPSHOT</version>
  <executions>
    <execution>
      <phase>package</phase>
      <goals>
        <goal>run</goal>
      </goals>
    </execution>
  </executions>
</plugin>
---
```

### 5.2.1. run goal with vert.x redeploy enabled

```

---
<plugin>
  <groupId>org.workspace7.maven.plugins</groupId>
  <artifactId>vertx-maven-plugin</artifactId>
  <version>1.0.0-SNAPSHOT</version>
  <executions>
    <execution>
      <phase>package</phase>
      <goals>
        <goal>run</goal>
      </goals>
      <configuration>
        <redeploy>true</redeploy> ①
      </configuration>
    </execution>
  </executions>
</plugin>
---

```

① enabling vert.x redeploy

### 5.2.2. run goal with vert.x redeploy enabled with custom pattern

```

---
<plugin>
  <groupId>org.workspace7.maven.plugins</groupId>
  <artifactId>vertx-maven-plugin</artifactId>
  <version>1.0.0-SNAPSHOT</version>
  <executions>
    <execution>
      <phase>package</phase>
      <goals>
        <goal>run</goal>
      </goals>
      <configuration>
        <redeploy>true</redeploy> ①
        <redeployPatterns> ②
          <redeployPattern>src/main/java/*.java</redeployPattern>
          <redeployPattern>src/resources/webroot/*.html</redeployPattern>
        </redeployPatterns>
      </configuration>
    </execution>
  </executions>
</plugin>
---

```

① enabling vert.x redeploy

② patterns that will be watched for changes and trigger redeploy

## 5.3. vert.x:start Examples

### 5.3.1. start goal with defaults

```
---
<plugin>
  <groupId>org.workspace7.maven.plugins</groupId>
  <artifactId>vertx-maven-plugin</artifactId>
  <version>1.0.0-SNAPSHOT</version>
  <executions>
    <execution>
      <phase>package</phase>
      <goals>
        <goal>start</goal>
      </goals>
    </execution>
  </executions>
</plugin>
---
```

### 5.3.2. start goal with custom application id

```
---
<plugin>
  <groupId>org.workspace7.maven.plugins</groupId>
  <artifactId>vertx-maven-plugin</artifactId>
  <version>1.0.0-SNAPSHOT</version>
  <executions>
    <execution>
      <phase>package</phase>
      <goals>
        <goal>start</goal>
      </goals>
      <configuration>
        <appId>my-app-id</appId> ①
      </configuration>
    </execution>
  </executions>
</plugin>
---
```

① Custom unique application id

### 5.3.3. start goal with custom java options

```

---
<plugin>
  <groupId>org.workspace7.maven.plugins</groupId>
  <artifactId>vertx-maven-plugin</artifactId>
  <version>1.0.0-SNAPSHOT</version>
  <executions>
    <execution>
      <phase>package</phase>
      <goals>
        <goal>start</goal>
      </goals>
      <configuration>
        <jvmArgs> ①
          <jvmArg>-Xms512m</jvmArg>
          <jvmArg>-Xmx1024m</jvmArg>
        </jvmArgs>
      </configuration>
    </execution>
  </executions>
</plugin>
---

```

① The jvm arguments that gets passed as `--java-opts` to the vert.x application

## 5.4. vert.x:stop Examples

### 5.4.1. stop with no additional configuration

```

---
<plugin>
  <groupId>org.workspace7.maven.plugins</groupId>
  <artifactId>vertx-maven-plugin</artifactId>
  <version>1.0.0-SNAPSHOT</version>
  <executions>
    <execution>
      <phase>package</phase>
      <goals>
        <goal>stop</goal>
      </goals>
    </execution>
  </executions>
</plugin>
---

```

### 5.4.2. stopping one or more application

When you have configured to [start goal with custom application id](#) or know the application ids,

then you can add list of application ids as shown below to trigger stop of the those applications

```
---
<plugin>
  <groupId>org.workspace7.maven.plugins</groupId>
  <artifactId>vertx-maven-plugin</artifactId>
  <version>1.0.0-SNAPSHOT</version>
  <executions>
    <execution>
      <phase>package</phase>
      <goals>
        <goal>start</goal>
      </goals>
      <configuration>
        <appIds> ①
          <appId>my-app-id-1</appId>
          <appId>my-app-id-2</appId>
        </appIds>
      </configuration>
    </execution>
  </executions>
</plugin>
---
```

① List of custom unique application ids

## 5.5. References

[Maven Properties](#)