

JavaFX Deployment Tutorial – Made by Lorenz Geyser

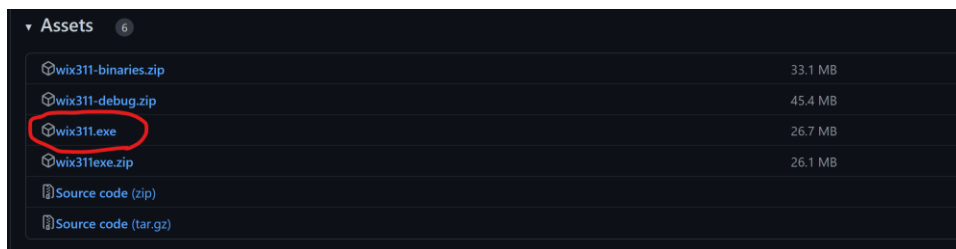
How to build a JavaFX Installer with Maven for Windows, Linux and MacOS

Requirements:

You need a working JavaFX Application with Maven

If you want to make an .msi Installer you need the [WiX Toolset](#) installed (Download .exe from [GitHub](#))

Only works with WiX v3!



Step 1:

You need to add the org.panteleyev Maven Build Plugin into your pom.xml (Documentation: <https://github.panteleyev.org/jpackage-maven-plugin/>)

It acts as a plugin for [jPackage](#) which supports the following packaging types:

--type or -t type

The type of package to create

Valid values are: {"app-image", "exe", "msi", "rpm", "deb", "pkg", "dmg"}

If this option is not specified a platform dependent default type will be created.

Add the <plugin>...</plugin> tag in the <build><plugins> tag in your pom.xml. You have to change the module path with your module name (at.htlhl.testfx) and to set the path to your main class (at.htlhl.testfx.App). You can use your name as the vendor.

```

    </plugin>
    <plugin>
      <groupId>org.panteleyev</groupId>
      <artifactId>jpackage-maven-plugin</artifactId>
      <version>1.6.0</version>
      <configuration>
        <name>${project.artifactId}</name>
        <appVersion>${project.version}</appVersion>
        <vendor>HTL Hollabrunn</vendor>
        <module>at.htlhl.testfx/at.htlhl.testfx.App</module>
        <runtimeImage>target/${project.artifactId}</runtimeImage>
      </configuration>
      <executions>
        <execution>
          <id>app_image</id>
          <configuration>
            <type>APP_IMAGE</type>
            <destination>target/dist/app_image</destination>
          </configuration>
        </execution>
      </executions>
    </plugin>
  </plugins>
</build>
</project>

```

Make sure to add the configuration tag into your org.openjfx plugin, to add the option in the execution configuration tag and to remove the tags I commented.

```

<plugin>
  <groupId>org.openjfx</groupId>
  <artifactId>javafx-maven-plugin</artifactId>
  <version>0.0.8</version>
  <configuration>
    <jlinkImageName>${project.artifactId}</jlinkImageName>
    <launcher>${project.artifactId}</launcher>
    <mainClass>at.htlhl.testfx/at.htlhl.testfx.App</mainClass>
  </configuration>
  <executions>
    <execution>
      <!-- Default configuration for running with: mvn clean javafx:run -->
      <id>default-cli</id>
      <configuration>
        <mainClass>at.htlhl.testfx/at.htlhl.testfx.App</mainClass>
        <!--<launcher>app</launcher>
        <jlinkZipName>app</jlinkZipName>
        <jlinkImageName>app</jlinkImageName>
        <noManPages>true</noManPages>
        <stripDebug>true</stripDebug>
        <noHeaderFiles>true</noHeaderFiles>
        <options>
          <option>--add-opens=javafx.graphics/javafx.scene=org.controlsfx.controls</option>
        </options>
      </configuration>
    </execution>
  </executions>
</plugin>

```

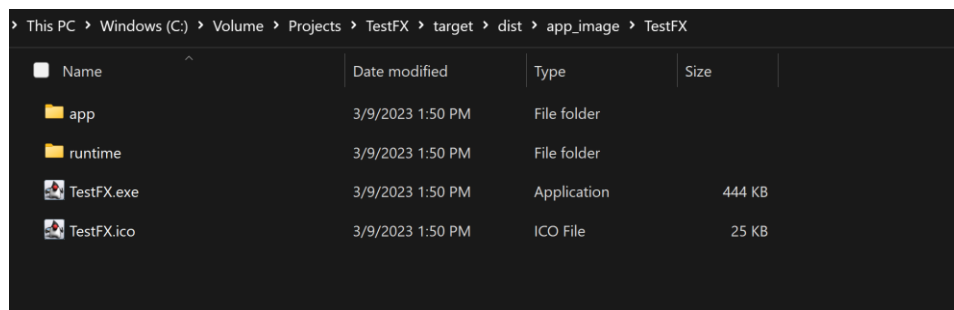
Before you are able to build the project, you have to change the version to a major, minor and patch value. This can be done at the top of your pom.xml:

```
<groupId>at.htlhl.testfx</groupId>
<artifactId>TestFX</artifactId>
<!--<version>1.0-SNAPSHOT</version>-->
<version>1.0.0</version>
<name>TestFX</name>
```

After you've done that, you should be able to execute that command:

```
mvn clean compile javafx:jlink jpackage:jpackage@app_image
```

This will remove previous compiled files, compile the most recent version and execute the `app_image` configuration we added to the `panteleyev` plugin before. `APP_IMAGE` will always build the project depending on your operating system. In my case, I will find an executable in `target/dist/app_image/TestFX`:



Name	Date modified	Type	Size
app	3/9/2023 1:50 PM	File folder	
runtime	3/9/2023 1:50 PM	File folder	
TestFX.exe	3/9/2023 1:50 PM	Application	444 KB
TestFX.ico	3/9/2023 1:50 PM	ICO File	25 KB

Java bundles your project with a necessary runtime. That's the reason why your application uses around 100MB.

If you want to add an icon to your application, just add the `<icon>` tag in your `panteleyev` configuration. Be sure to use the `.ico` format, otherwise it is going to throw an error when building the project. Maven will look in the project root, the location where your `src` directory and the `pom.xml` is located.

Now, we want to make an `.msi` installer. It works the same way as we configured the `app_image`:

```

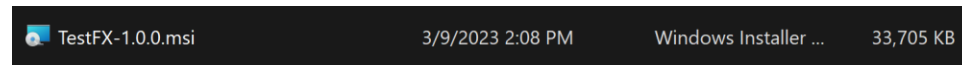
<executions>
  <execution>
    <id>app_image</id>
    <configuration>
      <type>APP_IMAGE</type>
      <destination>target/dist/app_image</destination>
    </configuration>
  </execution>
  <execution>
    <id>msi</id>
    <configuration>
      <type>MSI</type>
      <destination>target/dist/msi</destination>
      <winMenu>true</winMenu>
      <winShortcut>true</winShortcut>
      <winDirChooser>true</winDirChooser>
      <winMenuGroup>HTL Hollabrunn</winMenuGroup>
    </configuration>
  </execution>
</executions>

```

There are a few more configuration possibilities than the app_image has but with this you should be on the safe side.

If you execute the same command than before, just with msi instead of app_image you will find the .msi installer located in target/dist/msi.

Here the command: `mvn clean compile javafx:jlink jpackage:jpackage@msi`



This installer acts like a common installer you may have encountered before. The size will be reduced due to compression.

Now you should be able to adapt everything to your needs. If you want to build for MacOS or Linux, you need to add a new execution tag with the type you want to go for. (You will find them on the jPackage Documentation)

Have fun coding!

Best regards,

Lorenz Geyser