

HOWTO: Compile and package standalone JavaFX applications

Article ID: 000031

Last Revised On: 14-Dec-2009

The information in this article applies to:

- Excelsior JET 7.0 and above (all editions)

You may download Excelsior JET 7.0 [here](#).

SUMMARY

This document contains step-by-step instructions for using Excelsior JET to compile a standalone JavaFX application to a native executable and package it for deployment.

INTRODUCTION

JavaFX is a new Java-based platform created by Sun Microsystems to facilitate the development of Rich Internet Applications through easy intergration of media (audio, video, graphics, rich text) and web services.

Technically, the JavaFX Runtime is a set of jars and native method libraries. Knowing that set is all that is needed to configure Excelsior JET to compile and package a standalone JavaFX application.

DETAILS

Compilation

First, determine the command line that would run your JavaFX application using the conventional `java` launcher, as opposed to the `javafx` launcher found in the JavaFX SDK. Here is how:

Suppose the `javafx` command line for your applicaion is:

```
javafx MyApp.Main
```

Then the equivalent `java` command line, broken down for readability, is:

```
java -cp JavaFX-Classpath  
-Djava.library.path=JavaFX-Libpath  
com.sun.javafx.runtime.Main MyApp.Main
```

where *JavaFX-Classpath* is the list of jars comprising the JavaFX Runtime and *JavaFX-Libpath* is the pathname of the directory containing their native method libraries. You will find these in the file `profiles/desktop.properties` in your JavaFX SDK directory under the names `execute_classpath` and `execute_nativelibpath`, respectively:

```
execute_classpath="${javafx_home}/lib/shared/javafxrt.jar; ...  
.  
.  
execute_nativelibpath="${javafx_home}/lib/desktop"
```

`${javafx_home}` expands to the actual pathname of your JavaFX SDK installation, e.g. `C:\Program Files\JavaFX\javafx-sdk1.2`.

With the above information in hand, write a simple shell script that would run your JavaFX application using the `java` launcher. **IMPORTANT:** Make sure to use double quotes if you have installed the JavaFX SDK into a directory with spaces in its pathname:

```
SET JAVAFOX_HOME=C:\Program Files\JavaFX\javafx-sdk1.2
SET JAVAFOX_CLASSPATH=%JAVAFOX_HOME%\lib\shared\javafxrt.jar; ...
SET JAVAFOX_LIBPATH=%JAVAFOX_HOME%\lib\desktop
java -cp "%JAVAFOX_CLASSPATH%" -Djava.library.path="%JAVAFOX_LIBPATH%" co
```

Run this script and verify that your JavaFX application behaves correctly.

By now, you should have noticed that the `execute_classpath` property contains lots of jar files; it is over 1,000 characters long as of JavaFX 1.2.1. The easiest way to add all those jars to an Excelsior JET project is using the Excelsior JET Launcher [1], located in the `jre/bin/` subdirectory of the active Excelsior JET profile, for instance:

```
C:\jet7.0-ent\profile1.6.0_16\jre\bin\java.exe
```

In the shell script that you have created on the previous step, replace the conventional `java` launcher with the Excelsior JET one:

```
C:\jet7.0-ent\profile1.6.0_16\jre\bin\java.exe -cp "%JAVAFOX_CLASSPATH%"
```

and run the updated script. The Excelsior JET Control Panel shall open with all JavaFX jars already added to the project, and the `java.library.path` system property set to *JavaFX-Libpath*. Go to the **Test Run** page and check that your application works on Excelsior JET Runtime, **Note:** Make sure that the name of your application's main class is specified as an argument.

After successful Test Run, proceed with project setup and compilation as usual.

Packaging

Native libraries

Excelsior JET has transformed your application jars and JavaFX Runtime jars into a native executable. It however left the native method libraries intact. You must add them to the package manually and ensure that the executable will be able to locate them at run time.

After adding your compiled JavaFX application to the package in JetPackII, do the following:

1. Go to the **Files** page.
2. (*optional*) If you want to place the JavaFX native method libraries in a separate directory, create it under **Package Files**.
3. Point the filechooser to *JavaFX-Libpath*.
4. Select all dynamic libraries (`.dll` files on Windows, `.so` files on Linux) and add them to the desired location in the package. Do not add jars - they are already compiled!

Note: As of version 1.2.1, the Windows version of the JavaFX SDK contains the file `msvcr71.dll`. That DLL is also part of the Excelsior JET Runtime, so there will be a "Duplicate File" warning displayed. As a process may not load two DLLs with the same name anyway, you may choose to omit that DLL.

5. Go to the **Resources** page.

6. Select the JavaFX executable if it is not selected yet.
7. In the **System Properties** pane, make sure that the property `java.library.path` is defined and points to the directory within the package where you have placed the JavaFX native method libraries. Use the `$(root-name)` syntax to refer to a package root, for instance:

```
java.library.path=$(Root)/lib
```

Shortcuts

When creating a shortcut for your natively compiled JavaFX application, make sure to specify the name of its main class as an argument.

ADVANCED

You may compile two or more JavaFX applications into one executable and select them by main class name at launch time.

If you have just one application and want to avoid specifying the name of the JavaFX main class on the command line, write a simple wrapper for `com.sun.javafx.runtime.Main` and specify it as the main class on the **Start** page of Excelsior JET Control Panel.

REFERENCES

1. Excelsior JET User's Guide, Chapter "Excelsior JET Launcher".