Liberica NIK
Using Native Image with desktop applications

Liberica NIK
Revision 1.0
October 17, 2023

bellsoft

BellSoft software contains open source software. Additional information about third party code is available at https://bell-sw.com/third_party_licenses. You can also get more information on how to get a copy of source code by contacting info@bell-sw.com.

The use of any software product referenced in this document is governed by the applicable license agreement, which is not modified in any way by the terms of this notice.

Contents

1. Introduction

This document will guide you through generating native images out of desktop applications built with JavaFX and AWT.

2. How to create JavaFX native images

Enable native compilation

First, you need to download and install Liberica NIK. Go to the Liberica NIK Download Center and choose NIK version 22 for Java 11 or 17. You need the Full version as it includes LibericaFX, an instance of OpenJFX.

Download the package and follow the instructions to install the utility on your platform. macOS users should get the .dmg package to avoid any issues with running Liberica NIK.

You are now ready to compile Java applications natively. You have several options:

- Manual invocation. From your NIK installation, run `native-image <args> -jar <your application jar>`.
- Configure Maven build as described in Use Maven to Build a Native Executable from a Java Application.
- Configure Gradle build as described in Use Gradle to Build a Native Executable from a Java Application.

In all three cases, your application is likely to use resources such as icons or images. These resources must be made known to native-image so that it includes them in the resulting executable. There are two options to include and exclude resource identifiers, and you can use wildcards there:

```
-H:IncludeResources='com.fxapp.resources.images.*'
-H:ExcludeResources='com.fxapp.resources.unused.*'
```

Alternatively, resources can be configured using JSON files. Many applications make use of dynamic language features such as reflective or JNI access. In this case, the feature being used, such as the name of the method called, is not known at image build time, so NIK cannot figure out exactly which method is called. These dynamic feature usages must be configured beforehand using JSON files as described here.

You can create JSON files using the native image agent. Run your application with the agent enabled, probably several times to exercise different code paths. The agent captures dynamic feature accesses and creates configuration files automatically. These files can be passed to native-image without

further editing.

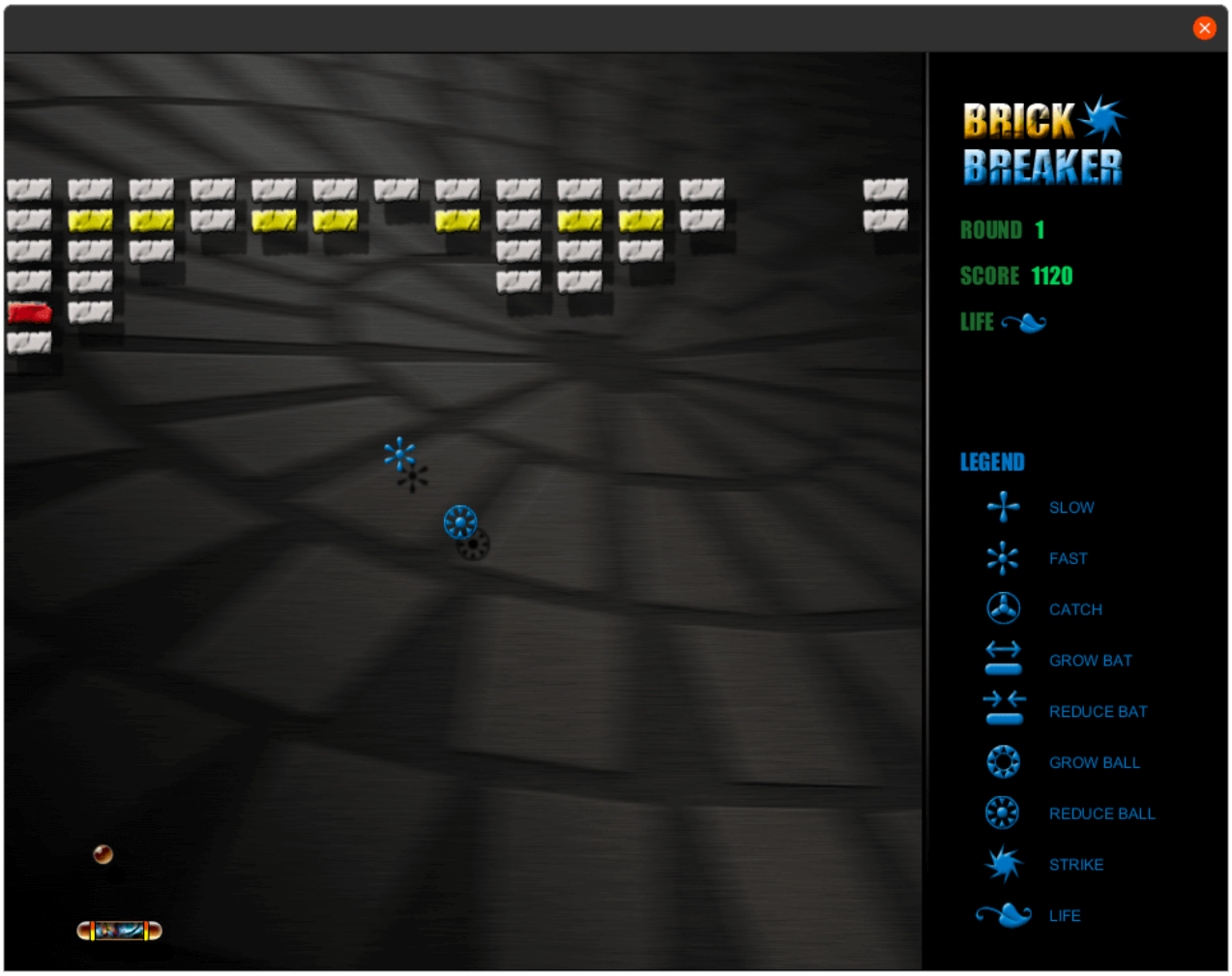
Compile the application

Let us now compile a JavaFX app natively. We will use the BrickBreaker game as an example. You can utilize your own app or select a JavaFX demo on GitHub. BrickBreaker requires no dynamic feature configuration. At the same time, it uses lots of images that need to be included in the resulting executable. In this example, image names are passed using -H:IncludeResources with a regular expression:

```
native-image -H:IncludeResources='ensemble.samples.shared
-resources.brickImages.*' -jar BrickBreaker.jar
```

After the compilation has finished, start the application:

```
./BrickBreaker
```



Speed and memory comparison

The figures below were obtained on an Intel Core i7 laptop running Ubuntu Linux.

	Liberica JRE 17.0.4.1	Native Image made with Liberica NIK 22.2.0-JDK17
Startup time	710ms	366ms
Maximum RSS	176M	133M
Executable size	276M (JRE+jar)	36M

Limitations

Modules `javafx.media` and `javafx.web` are currently not supported for native compilation. The `javafx.media` module is responsible for adding the playback of media and audio content. The `javafx.web` module defines the WebView functionality.

3. How to turn AWT applications into native images

AWT (the Abstract Window Toolkit) is a Java GUI widget toolkit. Although Swing largely superseded AWT, the latter is still used alone or in combination with Swing components.

Install Liberica NIK

The Liberica NIK Full version can be used to turn AWT/Swing applications into native images on Linux, Windows, and macOS.

For our demo, we will use Liberica NIK 22.3.1 for Java 17. Download the utility for your platform and follow the instructions to complete the installation. Set $PATH to Liberica NIK:

```
PATH=<path-to-nik>/bin:$PATH
```

Create an AWT app

Build a simple AWT application (the code was taken here):

```
import java.awt.*;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
public class Sample extends Frame {
  public Sample() {
    Button btn = new Button("Button");
    btn.setBounds(50, 50, 50, 50);
    add(btn);
    setSize(150, 150);
    setTitle("Simple AWT window");
    setLayout(new FlowLayout());
    setVisible(true);
    addWindowListener(new WindowAdapter() {
      public void windowClosing(WindowEvent we) {
        dispose();
```

```
      }
    });
  }
  public static void main(String args[]){
    new Sample();
  }
}
```

Add a maven plugin to build an executable jar:

```
<build>
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-assembly-plugin</artifactId>
      <executions>
        <execution>
          <phase>package</phase>
          <goals>
            <goal>single</goal>
          </goals>
          <configuration>
            <archive>
              <manifest>
                <mainClass>
                  Sample
                </mainClass>
              </manifest>
            </archive>
            <descriptorRefs>
              <descriptorRef>jar-with-dependencies</descriptorRef>
            </descriptorRefs>
          </configuration>
        </execution>
      </executions>
    </plugin>
  </plugins>
</build>
```

Go to the project directory and build a jar file with `mvn package`.

Build an AWT native image

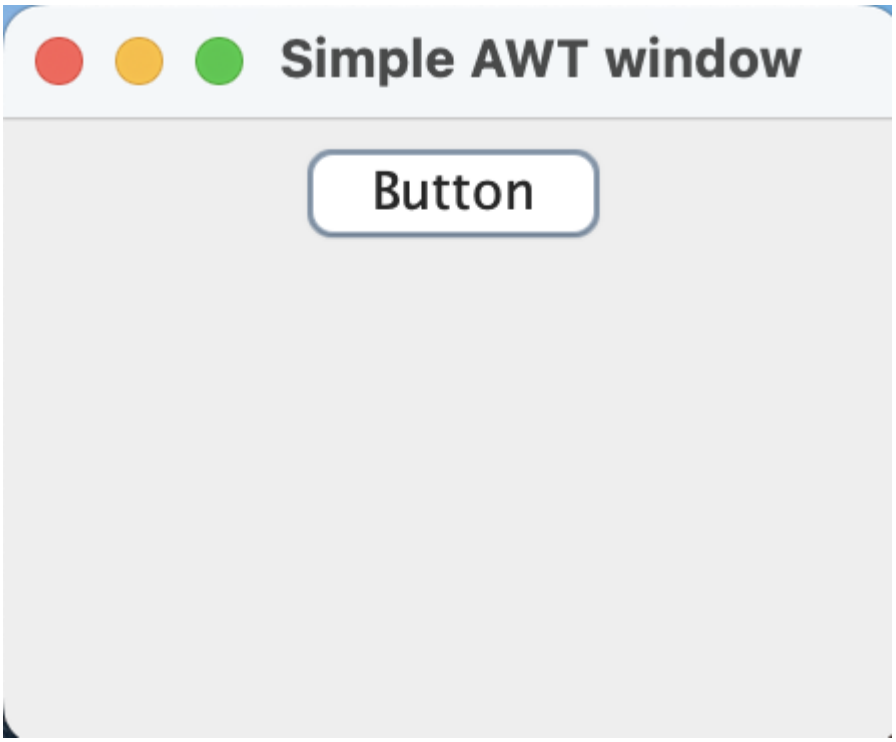
You need to set `java.awt.headless property` to false.

Build a native image as follows:

```
native-image -Djava.awt.headless=false -jar target/AwtDemo-1.0-SNAPSHOT-jar-with-dependencies.jar awtdemo
```

Run the image with the following command:

```
./awtdemo
```



If your app does not use any resources, Reflection, JNI, or other features not supported by GraalVM, you don't need to configure anything. Otherwise, explicitly provide resources or any classes used by reflection or serialization to the native-image tool. For that purpose, run the application with Graal tracing agent to dump the resources and dynamic classes used by the application. After that, run the native-image tool with configuration files that you generated.

Liberica NIK

Using Native Image with desktop applications

bell//soft