AXIOM JDK

Axiom NIK Pro Использование нативного образа в настольных приложениях

Axiom NIK Pro | Март 2025

Программное обеспечение АКСИОМ содержит программное обеспечение с открытым исходным кодом. Дополнительная информация о коде сторонних разработчиков доступна на сайте https://axiomjdk.ru/third_party_licenses. Для дополнительной информации о том, как получить копию исходного кода, можно обратиться по адресу info@axiomjdk.ru.

ДАННАЯ ИНФОРМАЦИЯ МОЖЕТ ИЗМЕНЯТЬСЯ БЕЗ ПРЕДВАРИТЕЛЬНОГО УВЕДОМЛЕНИЯ. АКСИОМ ПРЕДОСТАВЛЯЕТ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ "КАК ЕСТЬ" БЕЗ КАКИХ-ЛИБО ГАРАНТИЙ, АКСИОМ ПРЯМО ОТКАЗЫВАЕТСЯ ОТ ВСЕХ ПОДРАЗУМЕВАЕМЫХ ГАРАНТИЙ, ВКЛЮЧАЯ, НО НЕ ОГРАНИЧИВАЯСЬ ПОДРАЗУМЕВАЕМЫМИ ГАРАНТИЯМИ ТОВАРНОЙ ПРИГОДНОСТИ И ПРИГОДНОСТИ ДЛЯ ОПРЕДЕЛЕННОЙ ЦЕЛИ.

АКСИОМ НИ ПРИ КАКИХ ОБСТОЯТЕЛЬСТВАХ НЕ НЕСЕТ ОТВЕТСТВЕННОСТИ ЗА ЛЮБЫЕ КОСВЕННЫЕ, СЛУЧАЙНЫЕ, СПЕЦИАЛЬНЫЕ, ШТРАФНЫЕ ИЛИ КОСВЕННЫЕ УБЫТКИ, ИЛИ УБЫТКИ ОТ ПОТЕРИ ПРИБЫЛИ, ДОХОДА, ДАННЫХ ИЛИ ИСПОЛЬЗОВАНИЯ ДАННЫХ, ПОНЕСЕННЫЕ ВАМИ ИЛИ ЛЮБОЙ ТРЕТЬЕЙ СТОРОНОЙ, БУДЬ ТО В РЕЗУЛЬТАТЕ ДЕЙСТВИЯ ДОГОВОРА ИЛИ ДЕЛИКТА, ДАЖЕ ЕСЛИ АКСИОМ БЫЛО ПРЕДУПРЕЖДЕНО О ВОЗМОЖНОСТИ ТАКИХ УБЫТКОВ.

Использование любого программного продукта АКСИОМ регулируется соответствующим лицензионным соглашением, которое никоим образом не изменяется условиями данного уведомления. Программные продукты и фирменные наименования: Axiom JDK, Axiom JDK Pro, Axiom Runtime Container Pro, Axiom Linux, Libercat, Libercat Certified и АКСИОМ принадлежат АКСИОМ и их использование допускается только с разрешения правообладателя.

Товарный знак Linux® используется в соответствии с сублицензией от Linux Foundation, эксклюзивного лицензиата Линуса Торвальдса, владельца знака на всемирной основе. Java и OpenJDK являются товарными знаками или зарегистрированными товарными знаками компании Oracle и/или ее аффилированных лиц. Другие торговые марки являются собственностью их соответствующих владельцев и используются только в целях идентификации.

Содержание

1. Введение

Этот документ поможет вам создать нативные образы из настольных приложений, созданных с использованием технологий JavaFX и AWT.

2. Как создать нативный образ JavaFX

Включение нативной компиляции

Прежде всего вы должны скачать и установить Axiom NIK Pro.

Вы можете загрузить Axiom NIK Pro по ссылке на портале поддержки. Доступ к этому порталу предоставляется клиентам с активным договором поддержки. После входа на портал поддержки, если у вас есть несколько активных проектов, выберите нужный проект из выпадающего списка "**Выберите проект**" вверху страницы под именем пользователя. В зависимости от выбранного проекта вам будет доступна прямая ссылка на запрошенную вами версию продукта в разделе "Технологии". После завершения загрузки вы можете проверить загруженный файл, сравнив его размер на вашем диске с размером на странице загрузок.

> **Примечание:**
>
> Пользователи macOS должны устанавливать пакет `.dmg`, чтобы избежать каких-либо проблем с запуском Axiom NIK Pro.

Для дополнительной информации см. документ: *Инструментарий Нативных Образов Axiom NIK Pro: Руководство по установке*.

Теперь вы готовы к самостоятельной компиляции Java-приложений. У вас есть несколько вариантов компиляции приложений:

- Вызов вручную. Из установочного каталога Axiom NIK Pro запустите следующую команду: `native-image <параметры> -jar <имя_вашего_приложения.jar>`.
- Настройте сборку Maven, как описано в Use Maven to Build a Native Executable from a Java Application.
- Настройте сборку Gradle, как описано в Use Gradle to Build a Native Executable from a Java Application.

Во всех трех случаях ваше приложение, скорее всего, будет использовать такие ресурсы, как значки или изображения. Эти ресурсы должны быть известны программе `native-image`, чтобы она включала их в результирующий исполняемый файл. Существует два варианта включения и исключения идентификаторов ресурсов, и вы можете использовать символы подстановки:

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

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

Кроме того, ресурсы могут быть сконфигурированы с помощью файлов JSON. Многие приложения используют динамические функции языка, такие как reflective или JNI access. В этом случае используемая функция, такая как название вызываемого метода, неизвестна во время создания образа; следовательно, Axiom NIK Pro не может точно определить, какой метод вызывается.

Использование этих динамических функций должно быть настроено заранее с использованием файлов JSON, как описано здесь.

Вы можете создавать файлы JSON, используя агента нативного образа. Запустите свое приложение с включенным агентом, возможно, несколько раз, чтобы использовать все ветки алгоритма. Агент фиксирует динамические обращения к функциям и автоматически создает файлы конфигурации. Эти файлы могут быть переданы в `native-image` без дальнейшего редактирования.

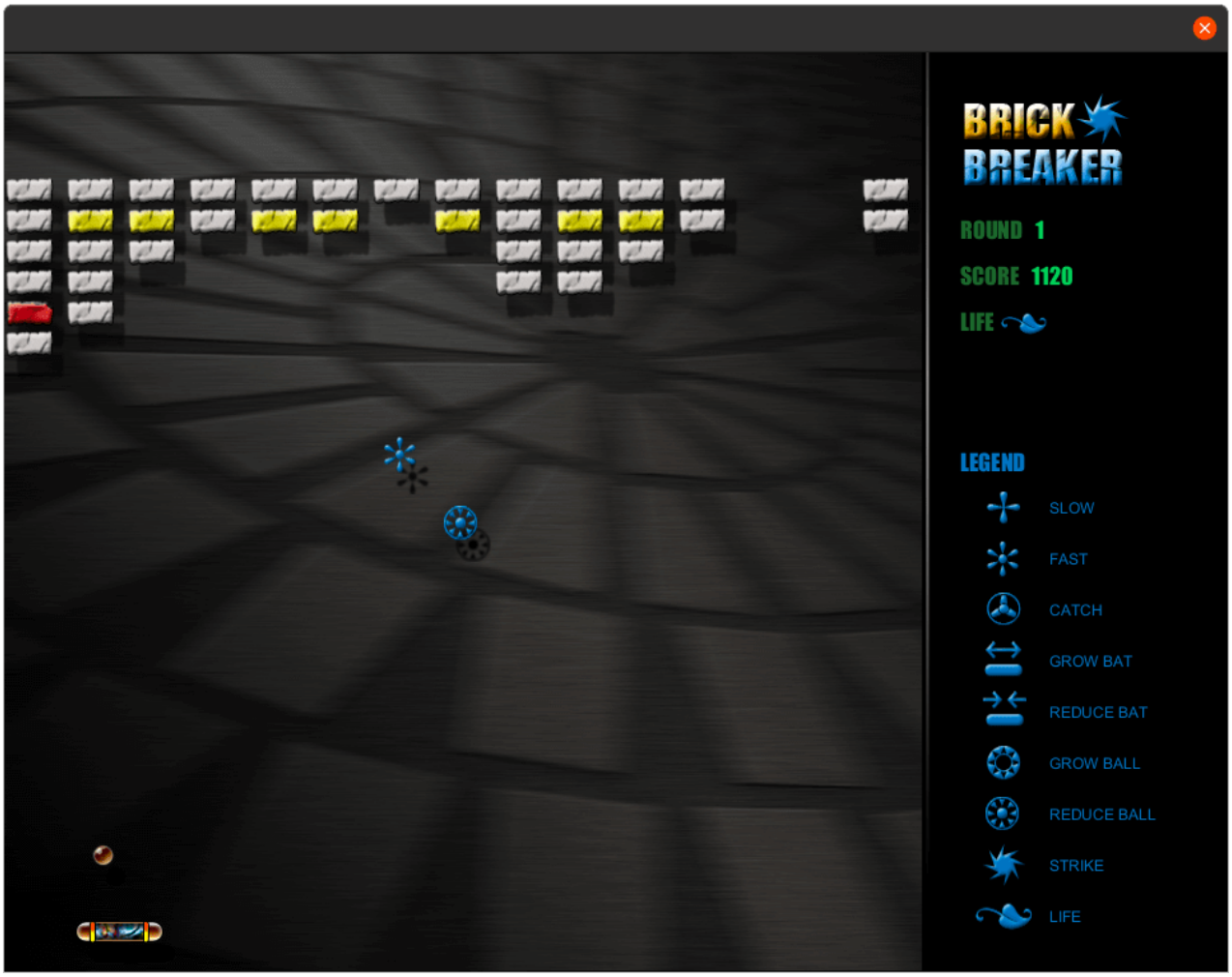
Компиляция приложения

Давайте теперь скомпилируем приложение JavaFX. В качестве примера мы будем использовать игру Brick Breaker. Вы можете использовать свое собственное приложение или выбрать демоверсию JavaFX. BrickBreaker не требует динамической настройки функций. В то же время он использует множество изображений, которые необходимо включить в результирующий исполняемый файл. В этом примере имена изображений передаются с помощью `-H:IncludeResources` с регулярным выражением:

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

После завершения компиляции запустите приложение:

```
./BrickBreaker
```



Сравнение скорости и объема используемой памяти

Приведенные ниже данные были получены на ноутбуке Intel Core i7 под управлением Ubuntu Linux.

	Axiom JRE 17.0.4.1	**Нативный образ созданный с помощью Axiom NIK Pro 22.2.0-JDK17**
Время запуска	710ms	366ms
Максимальный размер на диске	176M	133M

	Axiom JRE 17.0.4.1	**Нативный образ созданный с помощью Axiom NIK Pro 22.2.0-JDK17**
Размер при выполнении	276M (JRE+jar)	36M

Ограничения

Модули `javafx.media` и `javafx.web` в настоящее время не поддерживаются для компиляции нативного образа. Модуль `javafx.media` отвечает за добавление воспроизведения мультимедийного и аудиоконтента. Модуль `javafx.web` определяет функциональность WebView.

3. Как превратить AWT-приложение в нативный образ

AWT (the Abstract Window Toolkit) - это набор инструментов для создания виджетов с графическим интерфейсом Java. Хотя Swing в значительной степени вытеснил AWT, последний по-прежнему используется отдельно или в сочетании с компонентами Swing.

Установка Axiom NIK Pro

Axiom NIK Pro можно использовать для преобразования приложений AWT/Swing в нативные образы в Linux, Windows и Mac OS.

Для нашей демонстрации мы будем использовать Axiom NIK Pro 23.0.4 для Java 17. Скачайте и установите Axiom NIK Pro как описано в документе *Инструментарий Нативных Образов Axiom NIK Pro: Руководство по установке*. Добавьте путь к установленному Axiom NIK Pro в переменную окружения $PATH:

```
PATH=<Путь к Axiom NIK Pro>/bin:$PATH
```

Создание приложения AWT

Создайте простое AWT-приложение (код был взят отсюда):

```
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();
  }
}
```

Добавьте плагин maven для создания исполняемого файла 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>
```

Перейдите в каталог проекта и создайте jar-файл с `пакетом mvn`.

Создайте нативный образ AWT приложения

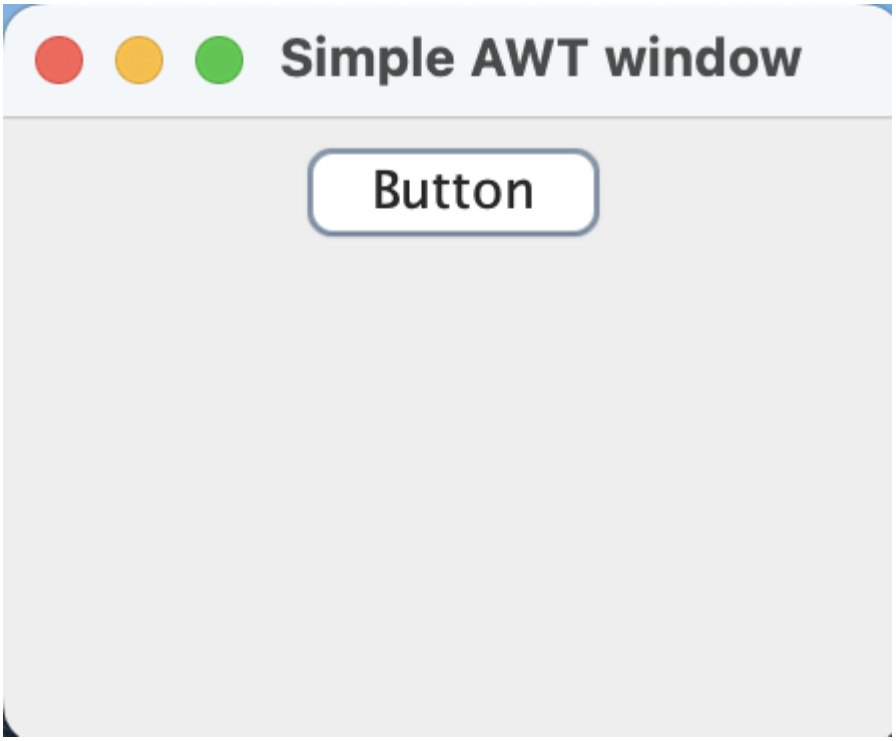
Установите для свойства `java.awt.headless` значение `false`.

Создайте нативный образ следующим образом:

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

Запустите образ следующей командой:

```
./awtdemo
```



Если ваше приложение не использует никаких ресурсов, Reflection, JNI или других функций, не поддерживаемых GraalVM, вам не нужно ничего настраивать. В противном случае явно укажите ресурсы или любые классы, используемые Reflection или сериализацией, в инструменте создания нативных образов. Для этого запустите приложение с помощью Graal tracing agent, чтобы получить доступ к ресурсам и динамическим классам, используемым приложением. После этого запустите инструмент для создания нативного образа с файлами конфигурации, которые вы сгенерировали.

AXIOM JDK

Axiom NIK Pro

Использование нативного образа в настольных приложениях