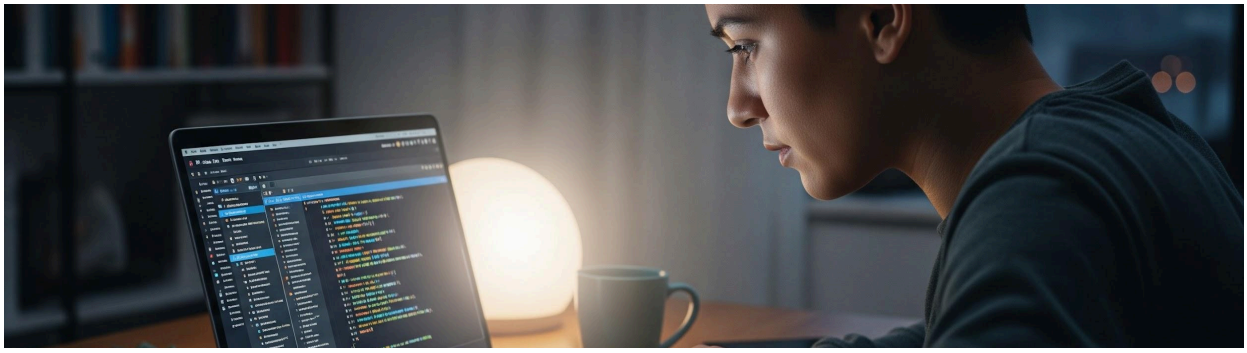

Guía de Descarga e Integración de JavaFX mediante Maven

Hola a tod@s, después de ver las necesidades actuales en el desarrollo de interfaces de escritorio, en esta guía vamos a aprender a realizar la integración y descarga de las librerías de JavaFX utilizando Maven. Aunque el procedimiento puede parecer complejo al principio, veremos que gestionar las dependencias de forma automatizada nos ahorrará muchos dolores de cabeza.

Pero antes de ponernos manos a la obra, es importante saber por qué ya no descargamos JavaFX como un simple instalador. Desde la versión 11 de Java, JavaFX fue desacoplado del JDK (el kit de desarrollo estándar) para convertirse en un módulo independiente. Su cometido ahora es funcionar como una librería externa, lo que nos obliga a "descargarla" declarándola en nuestro gestor de proyectos.

Un caso práctico en el cual se utiliza este método es al crear aplicaciones modernas que necesiten ser multiplataforma (Windows, Linux, Mac) sin obligar al usuario final a tener preinstaladas librerías específicas en su sistema, ya que Maven se encargará de empaquetar o descargar lo necesario.

Una vez teniendo claro de forma muy resumida el concepto y el cambio de gestión de librerías, tendremos que llevar a cabo diferentes pautas para tener disponible dicho entorno con sus respectivos componentes configurados correctamente:

1. Preparación del Proyecto Maven

Para la descarga de JavaFX tendremos que hacer uso de un proyecto gestionado. En lugar de buscar los ficheros `.jar` manualmente por internet, dejaremos que Maven

conecte con los repositorios centrales. Vamos a comenzar generando la estructura básica del directorio:

- Si no tienes un proyecto creado, generamos el esqueleto básico ejecutando el siguiente comando en nuestra terminal (o consola de comandos):

mvn archetype:generate -DgroupId=com.midominio -DartifactId=mi-app-javafx -DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false

- Una vez generada la carpeta con la arquitectura del proyecto, entramos en el directorio para comenzar la configuración:

cd mi-app-javafx

2. Configuración del pom.xml (La Descarga)

Una vez tenemos la estructura, procederemos a realizar la descarga real de JavaFX. En Maven, esto se realiza editando el fichero `pom.xml`. Al añadir las "dependencias", Maven descargará automáticamente los binarios compatibles con tu sistema operativo cuando compilemos el proyecto.

- Abrimos el fichero `pom.xml` con nuestro editor de texto o IDE favorito.
- Acto seguido, buscamos la sección `<dependencies>` y añadimos los módulos de `javafx-controls` y `javafx-fxml`. Estos son los componentes esenciales para pintar ventanas y manejar la interfaz gráfica.

```
<dependencies>

  <dependency>

    <groupId>org.openjfx</groupId>

    <artifactId>javafx-controls</artifactId>

    <version>17.0.0.1</version>

  </dependency>

  <dependency>

    <groupId>org.openjfx</groupId>
```

```
    <artifactId>javafx-fxml</artifactId>

    <version>17.0.0.1</version>

  </dependency>

</dependencies>
```

- A continuación, para que la compilación y la ejecución sean sencillas, añadiremos el plugin específico de JavaFX en la sección `<build><plugins>`. Esto nos permitirá ejecutar la aplicación con un comando corto más adelante sin tener que configurar rutas de librerías manualmente.

En dicha configuración estamos indicando que se utilice la versión del plugin `0.0.8` y definimos cuál será la clase principal (`mainClass`) que arrancará nuestra aplicación.<build>

```
  <plugins>

    <plugin>

      <groupId>org.openjfx</groupId>

      <artifactId>javafx-maven-plugin</artifactId>

      <version>0.0.8</version>

      <configuration>

        <mainClass>com.midominio.App</mainClass>

      </configuration>

    </plugin>

  </plugins>

</build>
```

- Cuando tengamos todos los parámetros cambiados, guardamos el fichero. La próxima vez que ejecutemos cualquier comando de Maven, este descargará internet todo lo necesario.

3. Definición de la Modularidad (module-info.java)

Desde las versiones recientes de Java, no basta con descargar la librería; debemos dar permiso explícito para usarla. Para ello, vamos a crear un fichero de definición de módulo.

- Creamos un fichero nuevo denominado `module-info.java` dentro de la ruta `/src/main/java/` e indicaremos los siguientes parámetros:

```
module com.midominio {

    requires javafx.controls;

    requires javafx.fxml;



    opens com.midominio to javafx.fxml;

    exports com.midominio;

}
```

Lo que estamos haciendo con las opciones indicadas son las descritas a continuación:

- **`requires javafx.controls`**: Indica al sistema que nuestra App no puede funcionar si no carga el módulo de controles de JavaFX.
- **`opens ... to javafx.fxml`**: Permite que el motor de JavaFX "lea" nuestras clases para inyectar el diseño gráfico.
- **`exports`**: Hace visible nuestro paquete principal para que la máquina virtual de Java pueda lanzarlo.

4. Ejecución y Comprobación

En el momento que tengamos todos los pasos anteriores realizados satisfactoriamente, ya tendremos JavaFX descargado e integrado. Para comprobar que está funcionando, podemos modificar la clase `App.java` que se creó al inicio para que muestre una ventana simple.

- Editamos `/src/main/java/com/midominio/App.java` y pegamos el siguiente código de prueba:

```
package com.midominio;

import javafx.application.Application;

import javafx.scene.Scene;

import javafx.scene.control.Label;

import javafx.scene.layout.StackPane;

import javafx.stage.Stage;

public class App extends Application {

    @Override

    public void start(Stage stage) {

        String javaVersion = System.getProperty("java.version");

        String javafxVersion = System.getProperty("javafx.version");

        Label label = new Label("Hola, JavaFX " + javafxVersion + " corriendo en Java " + javaVersion + " con Maven!");

        Scene scene = new Scene(new StackPane(label), 640, 480);

        stage.setScene(scene);

        stage.show();

    }

    public static void main(String[] args) {

        launch();

    }
```

}

- Una vez guardado el código, ejecutamos el proyecto mediante el plugin que configuramos anteriormente con el siguiente comando:

mvn clean javafx:run

Si todo ha ido bien, Maven terminará de descargar cualquier fichero restante y abrirá una ventana de 640×480 con nuestro mensaje. Con esto, habremos finalizado la instalación de JavaFX en nuestro entorno de desarrollo y estaremos listos para crear aplicaciones de escritorio complejas.