Proyecto multimódulo con Maven

Un **proyecto multimódulo en Maven** permite gestionar una colección de proyectos relacionados en una sola compilación. Este enfoque es especialmente útil para aplicaciones grandes donde los diferentes módulos tienen funcionalidades distintas, pero deben gestionarse, compilarse e implementarse conjuntamente. Al usar el POM (Modelo de Objetos del Proyecto) principal, se pueden optimizar la gestión de dependencias, el control de versiones y los procesos de compilación.

En un proyecto Maven multimódulo, se cuenta con un POM principal que gestiona las configuraciones y dependencias de todo el proyecto. Cada módulo del proyecto tiene su propio archivo POM y puede compilarse de forma independiente o como parte de un proyecto más amplio. El POM principal suele residir en el directorio raíz, mientras que los módulos se encuentran en subdirectorios.

Beneficios clave:

- **Gestión de dependencias centralizada:** administre las dependencias en un solo lugar.
- **Modularización:** dividir el proyecto en partes más pequeñas y manejables.
- **Reutilización:** compartir código común entre diferentes módulos.
- **Proceso de compilación consistente:** utilice un único comando para compilar todos los módulos.

Terminologías clave

- **POM principal (modelo de objeto de proyecto principal):** el archivo POM que actúa como principal para uno o más módulos secundarios, centralizando las configuraciones y dependencias heredadas por los módulos secundarios.
- **Módulo secundario:** un subproyecto del POM principal, cada uno con su propia estructura de archivos y directorios POM.
- **Módulo:** Un subproyecto dentro de un proyecto Maven multimódulo. Cada módulo tiene su propio archivo POM y se considera un proyecto Maven independiente.
- **Empaquetado:** el tipo de artefacto que Maven producirá, como JAR, WAR y POM.
- **Gestión de dependencias:** la sección del POM principal que especifica las versiones y configuraciones utilizadas en los módulos secundarios.
- **Administración de complementos:** la sección en el POM principal que define las configuraciones, versiones y configuraciones del complemento utilizadas por el proceso de compilación.
- **Complemento:** una herramienta o extensión que realiza tareas específicas durante el proceso de compilación.
- **Artefacto:** La salida del paquete de la compilación de Maven, como un archivo JAR, WAR o ZIP

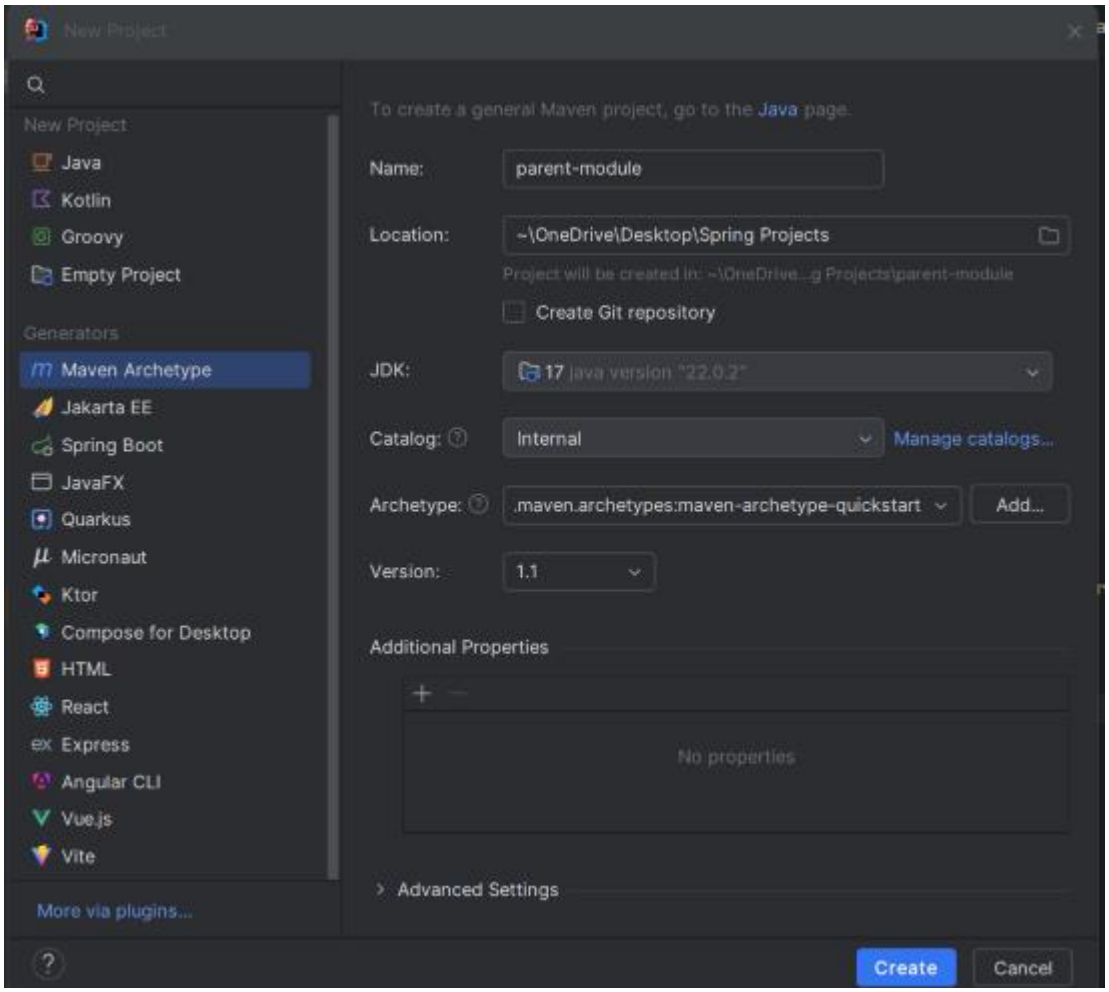
Implementación de un proyecto multimódulo con Maven

Paso 1: Crear el módulo principal

Cree un proyecto de arquetipo Maven utilizando IntelliJ IDEA con las siguientes opciones:

- Nombre: módulo principal
- JDK: 17
- Arquetipo: maven-archetype-quickstart

Haga clic en el botón **Crear** .

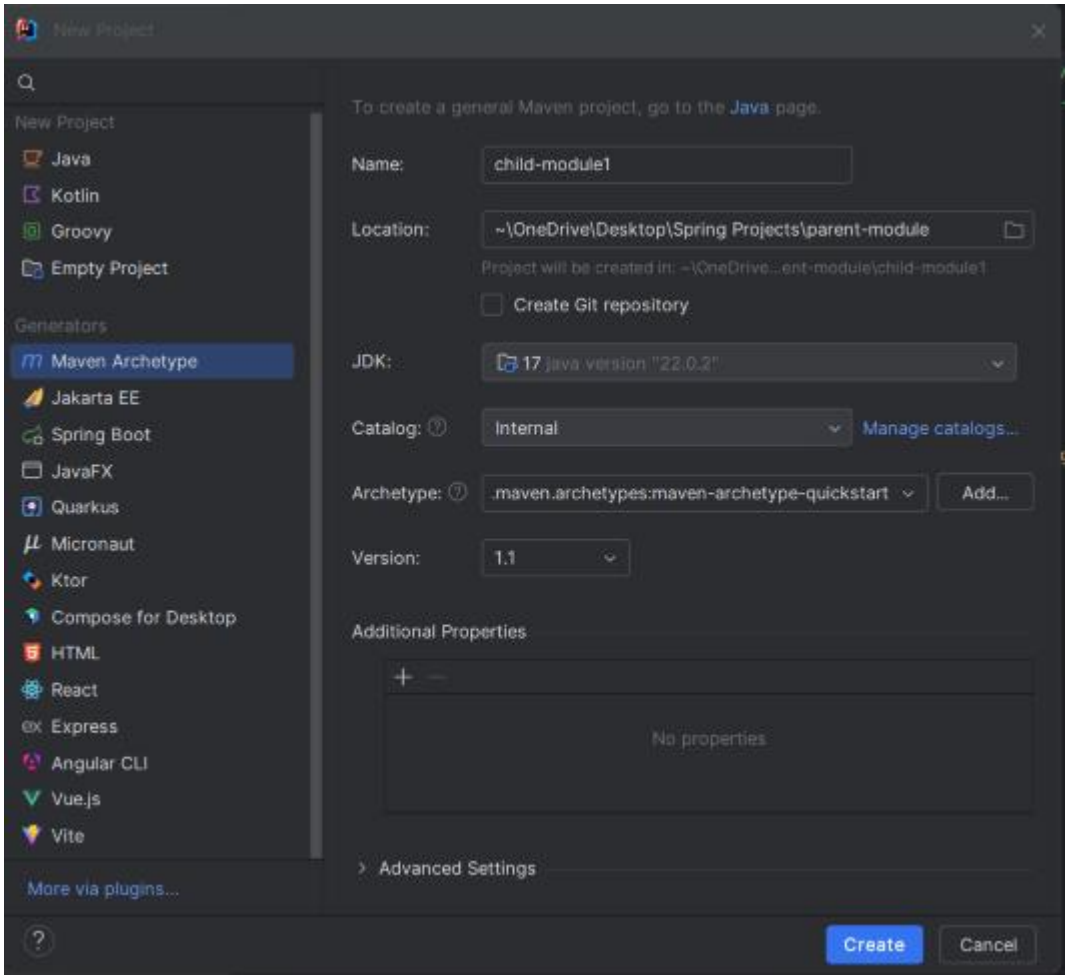


Paso 2: Crear el módulo secundario1

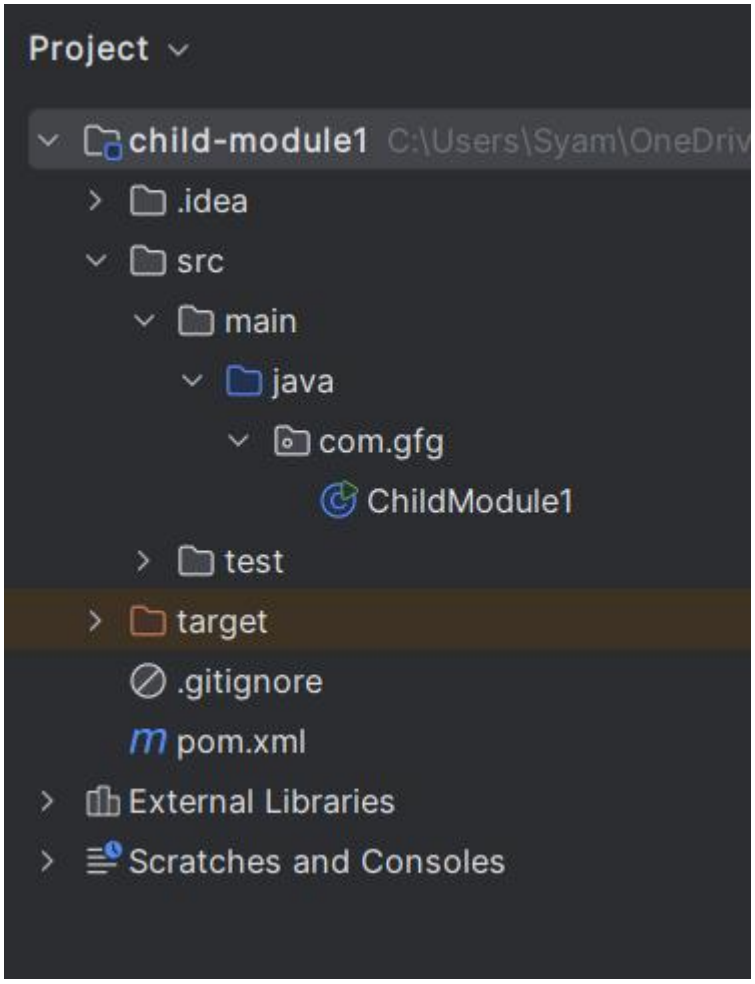
Cree otro proyecto de arquetipo Maven utilizando IntelliJ IDEA con las siguientes opciones:

- Nombre: módulo-infantil1
- JDK: 17
- Arquetipo: maven-archetype-quickstart

Haga clic en el botón **Crear** .



Después de crear el proyecto **child-module1** , la estructura del archivo se verá como la siguiente imagen.



Paso 3: Crear la clase ChildModule1

package com.gfg;

```
/**
 * Hello world!
 *
 */
public class ChildModule1
{
   public static void main( String[] args )
   {
      System.out.println( "Hello Child Module 1!" );
   }
}
```

pom.xml para child-module1:

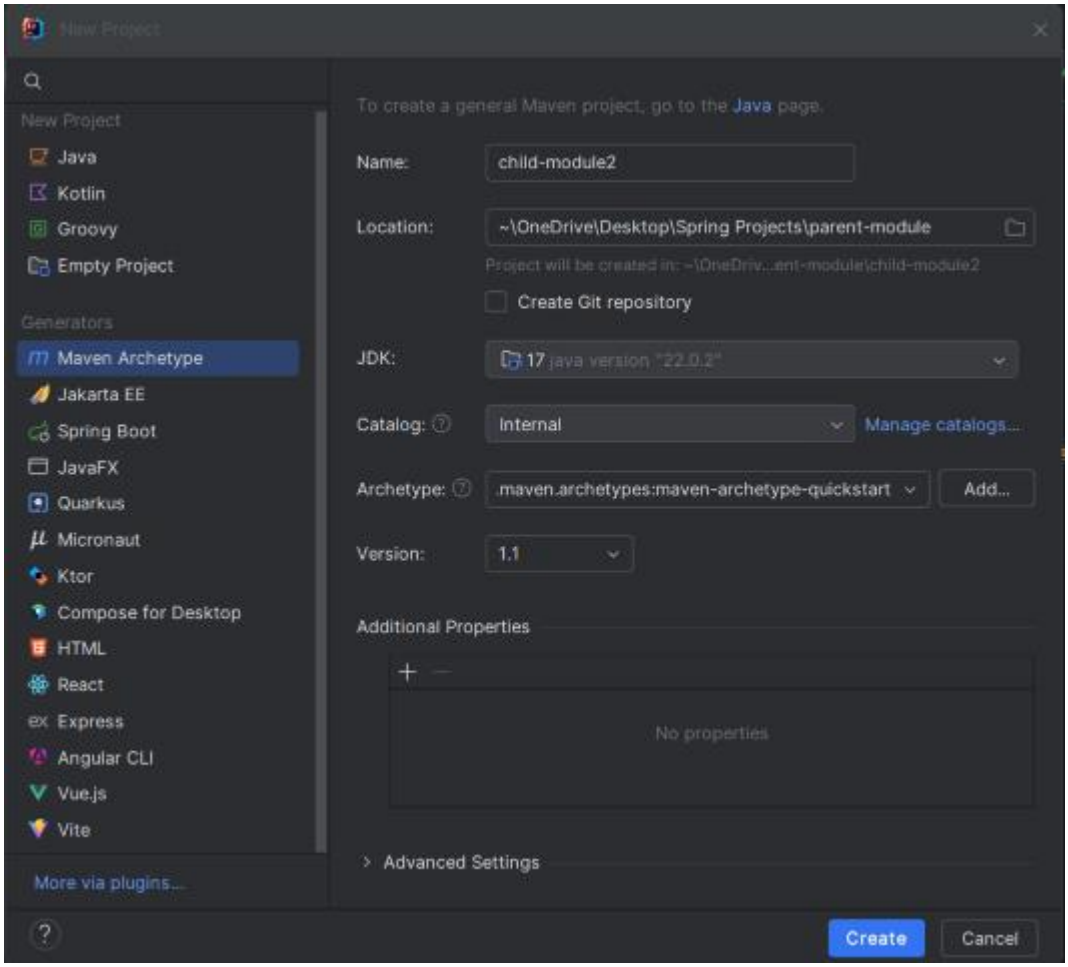
```
<project xmlns="https://maven.apache.org/POM/4.0.0"
xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="https://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
 <modelVersion>4.0.0</modelVersion>

 <groupId>com.gfg</groupId>
 <artifactId>child-module1</artifactId>
 <version>1.0-SNAPSHOT</version>
 <packaging>jar</packaging>

 <name>child-module1</name>
 <url>http://maven.apache.org</url>

 <properties>
  <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
 </properties>

 <dependencies>
  <dependency>
   <groupId>junit</groupId>
   <artifactId>junit</artifactId>
   <version>3.8.1</version>
   <scope>test</scope>
  </dependency>
 </dependencies>
</project>
```

Paso 4: Crear el módulo secundario2

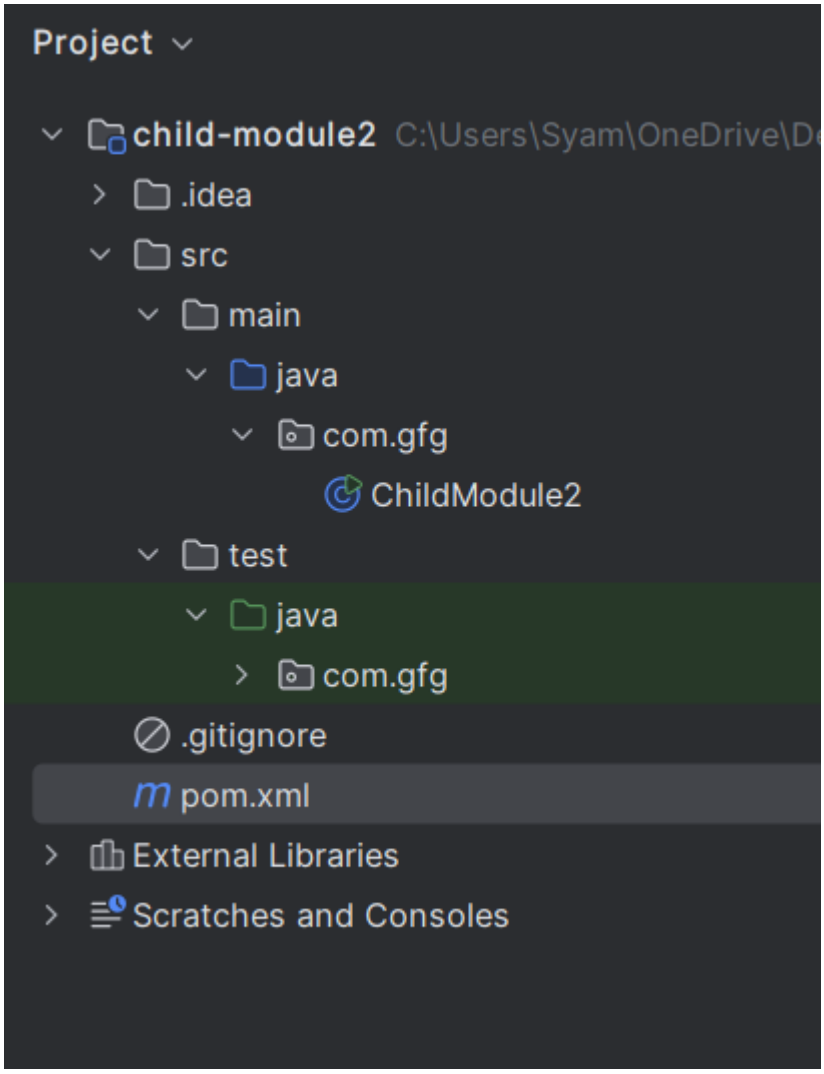
Cree otro proyecto de arquetipo Maven utilizando IntelliJ IDEA con las siguientes opciones:

- Nombre: módulo-infantil2
- JDK: 17
- Arquetipo: maven-archetype-quickstart

Haga clic en el botón **Crear** .



Después de crear el proyecto child-module2, la estructura del archivo se verá como la siguiente imagen.



Paso 5: Crear la clase ChildModule2

package com.gfg;

```
/**
 * Hello world!
 *
 */
public class ChildModule2
{
    public static void main( String[] args )
    {
        System.out.println( "Hello Child Module2!" );
    }
```

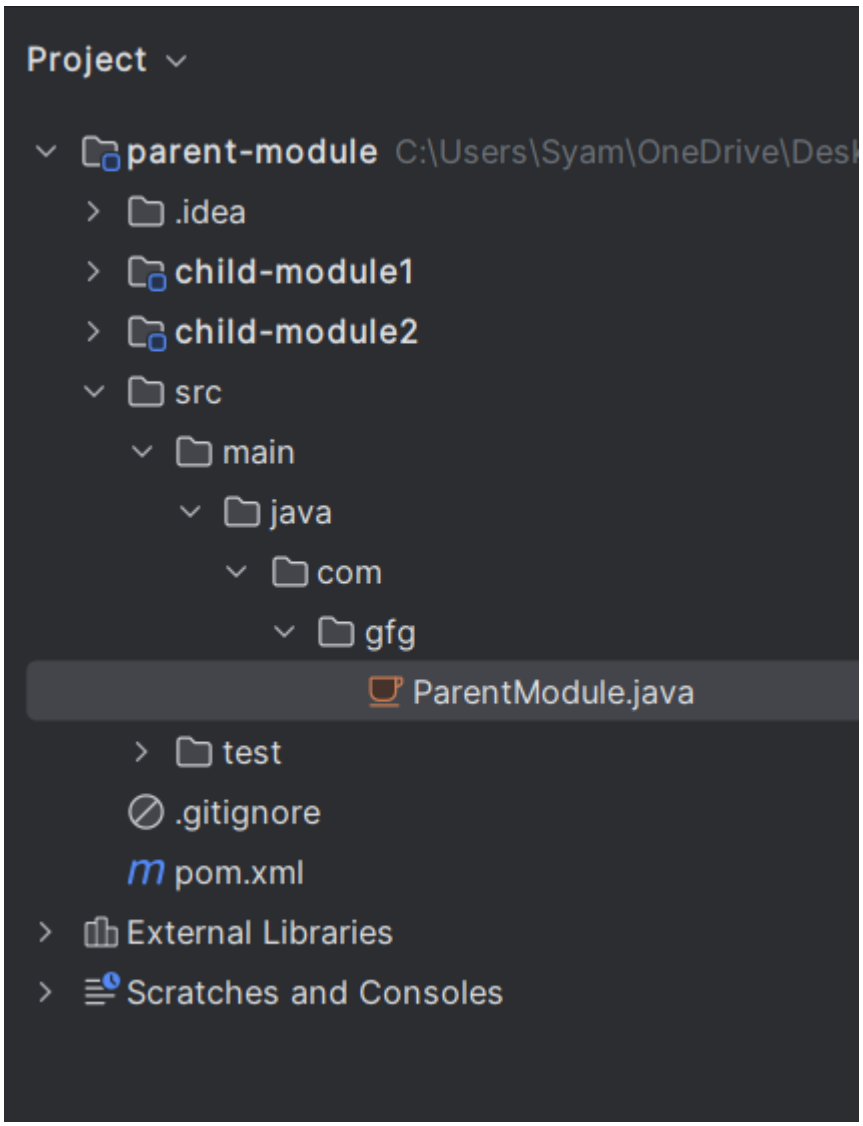
}

pom.xml para child-module2:

```
<project xmlns="https://maven.apache.org/POM/4.0.0"
xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="https://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
 <modelVersion>4.0.0</modelVersion>

 <groupId>com.gfg</groupId>
 <artifactId>child-module2</artifactId>
 <version>1.0-SNAPSHOT</version>
 <packaging>jar</packaging>

 <name>child-module2</name>
 <url>http://maven.apache.org</url>

 <properties>
  <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
 </properties>

 <dependencies>
  <dependency>
   <groupId>junit</groupId>
   <artifactId>junit</artifactId>
   <version>3.8.1</version>
   <scope>test</scope>
  </dependency>
 </dependencies>
</project>
```

Paso 6: Estructura del archivo principal

Después de configurar los módulos secundarios, la estructura del archivo principal debería verse así:



Paso 7: Agregue Spring Starter al pom.xml principal

Agregue la dependencia de inicio de Spring al padre pom.xml:

```
 <dependencyManagement>
  <dependencies>
   <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter</artifactId>
    <version>3.3.2</version>
   </dependency>
  </dependencies>
```

```
 </dependencyManagement>
```

Paso 8: Agregar el complemento Maven

Agregue el complemento del compilador Maven al padre pom.xml:

```
 <build>
  <pluginManagement>
   <plugins>
    <plugin>
     <groupId>org.apache.maven.plugins</groupId>
     <artifactId>maven-compiler-plugin</artifactId>
     <version>3.8.1</version>
     <configuration>
      <source>1.8</source>
      <target>1.8</target>
     </configuration>
    </plugin>
   </plugins>
  </pluginManagement>
 </build>
```

Paso 9: Agregar los módulos secundarios

Añade los módulos secundarios al módulo principal pom.xml:

```
 <modules>
  <module>child-module1</module>
  <module>child-module2</module>
 </modules>
```

Padre pom.xml:

```
<project xmlns="https://maven.apache.org/POM/4.0.0"
     xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
     xsi:schemaLocation="https://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
 <modelVersion>4.0.0</modelVersion>
```

```
<groupId>com.gfg</groupId>
<artifactId>parent-module</artifactId>
<version>1.0-SNAPSHOT</version>
<packaging>pom</packaging>

<name>parent-module</name>

<properties>
 <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
</properties>

<modules>
 <module>child-module1</module>
 <module>child-module2</module>
</modules>

<dependencies>
 <!-- Dependencies defined here are inherited by child modules -->
 <dependency>
  <groupId>junit</groupId>
  <artifactId>junit</artifactId>
  <version>3.8.1</version>
  <scope>test</scope>
 </dependency>
</dependencies>

<dependencyManagement>
 <dependencies>
  <dependency>
   <groupId>org.springframework.boot</groupId>
   <artifactId>spring-boot-starter</artifactId>
```

```
      <version>3.3.2</version>
    </dependency>
  </dependencies>
 </dependencyManagement>

 <build>
  <pluginManagement>
   <plugins>
    <plugin>
     <groupId>org.apache.maven.plugins</groupId>
     <artifactId>maven-compiler-plugin</artifactId>
     <version>3.8.1</version>
     <configuration>
      <source>1.8</source>
      <target>1.8</target>
     </configuration>
    </plugin>
   </plugins>
  </pluginManagement>
 </build>

</project>
```

Paso 10: Crear la clase ParentModule

```
package com.gfg;

/**
 * Hello world!
 *
 */
public class ParentModule
{
```

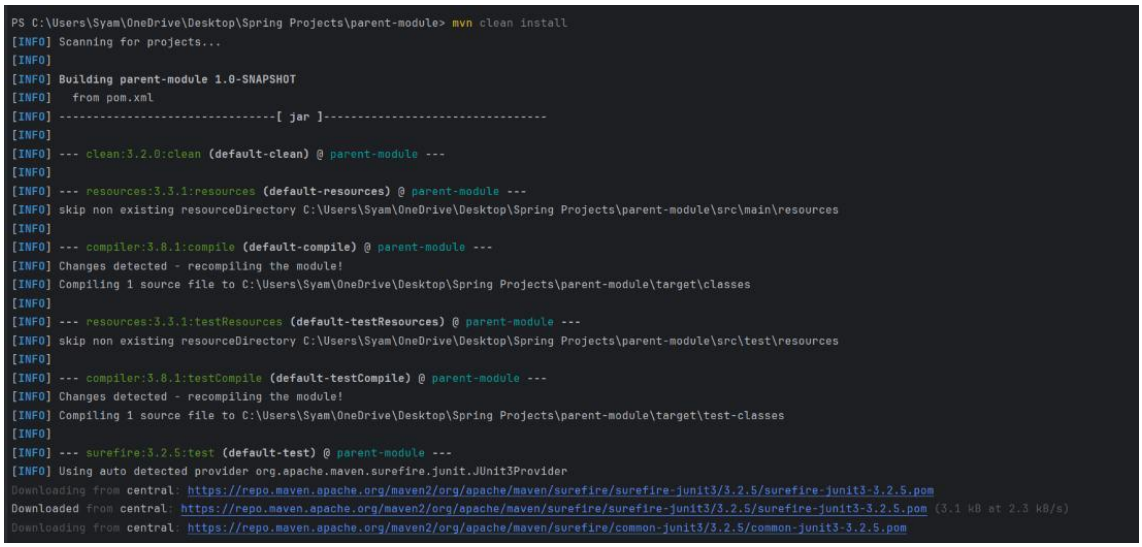
```
    public static void main( String[] args )
    {
        System.out.println( "Hello Parent Module!" );
    }
}
```

Paso 11: Instalar Maven

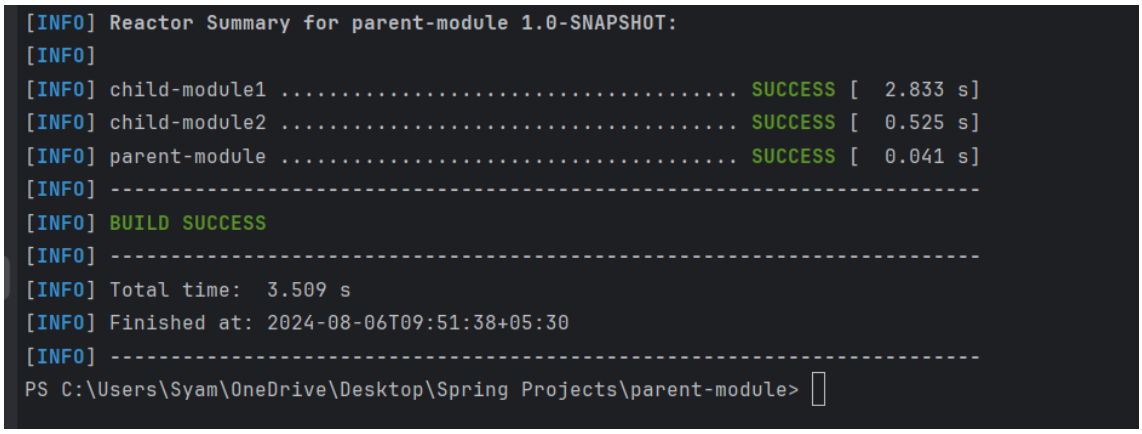
Utilice el siguiente comando para instalar el Maven del módulo principal:

mvn clean install

Producción:



Resultado de la prueba:



Paso 12: Ejecutar el módulo secundario 1

Utilice el siguiente comando para ejecutar el módulo secundario 1:

java -cp objetivo/clases com.gfg.ChildModel1

Producción:

```
PS C:\Users\Syam\OneDrive\Desktop\Spring Projects\parent-module\child-module1> java -cp target/classes com.gfg.ChildModule1
Hello Child Module 1!
PS C:\Users\Syam\OneDrive\Desktop\Spring Projects\parent-module\child-module1>
```

Paso 13: Ejecutar el módulo secundario 2

Utilice el siguiente comando para ejecutar el módulo secundario 2:

java -cp objetivo/clases com.gfg.ChildModel2

Producción:

```
PS C:\Users\Syam\OneDrive\Desktop\Spring Projects\parent-module\child-module2> java -cp target/classes com.gfg.ChildModule2
Hello Child Module2!
PS C:\Users\Syam\OneDrive\Desktop\Spring Projects\parent-module\child-module2>
```

Conclusión

Configurar un proyecto multimódulo con Maven permite una mejor gestión de proyectos complejos al dividirlos en módulos más pequeños y manejables. Este enfoque mejora la modularidad y la reutilización, y simplifica el proceso de compilación, facilitando el mantenimiento y el desarrollo de aplicaciones de gran tamaño.