# **I Buổi I**

## **1.1 Tìm hiểu maven**

### **1.1.1 Dựng một dự án (project) bằng Eclipse**

1. \*\*Cài đặt Eclipse\*\*: Nếu bạn chưa cài đặt Eclipse, bạn cần tải và cài đặt nó trước. Truy cập trang web chính thức của Eclipse (https://www.eclipse.org/) và tải phiên bản phù hợp với môi trường lập trình của bạn (Java, C++, PHP, v.v.).

2. \*\*Khởi động Eclipse\*\*: Sau khi cài đặt xong, mở Eclipse.

3. \*\*Tạo một Workspace\*\*: Eclipse sẽ yêu cầu bạn chọn một thư mục làm Workspace, nơi các dự án của bạn sẽ được lưu trữ. Chọn hoặc tạo một thư mục để làm Workspace.

4. \*\*Tạo một Dự án (Project)\*\*:

- Nhấp chuột phải vào phần trống trong khung làm việc (Workbench) và chọn "New" > "Project...".

- Trong hộp thoại "Select a wizard", bạn có thể chọn loại dự án bạn muốn tạo. Ví dụ, nếu bạn muốn tạo một Java Project, hãy chọn "Java" > "Java Project". Nếu bạn muốn tạo một Web Project, hãy chọn "Web" > "Dynamic Web Project" và còn nhiều loại khác tùy thuộc vào nhu cầu của bạn.

- Nhấp "Next" để tiếp tục.

5. \*\*Đặt tên và cấu hình Dự án (Project)\*\*:

- Đặt tên dự án trong trường "Project name".

- Thực hiện các cấu hình khác cần thiết cho dự án của bạn. Các tùy chọn sẽ phụ thuộc vào loại dự án mà bạn đang tạo.

- Tạo working set nếu cần

6. \*\*Chọn các thư viện và thư mục (folders)\*\*: Đối với các dự án Java, bạn có thể chọn các thư viện và thư mục gốc (source folders) cho dự án của bạn. Sau đó, nhấp "Finish" để tạo dự án.

7. \*\*Mở Dự án (Project)\*\*: Sau khi tạo dự án, nó sẽ xuất hiện trong thư mục Workspace của bạn. Bạn có thể mở dự án bằng cách nhấp đúp vào tên dự án trong khung làm việc (Workbench).

8. \*\*Thêm và viết mã (code)\*\*: Bây giờ bạn có thể bắt đầu thêm tệp mã và bắt đầu viết code cho dự án của bạn.

Lưu ý rằng các bước cụ thể có thể khác nhau tùy thuộc vào loại dự án và phiên bản Eclipse bạn đang sử dụng. Ngoài ra, Eclipse có rất nhiều tính năng và plugin hỗ trợ cho nhiều ngôn ngữ lập trình, vì vậy bạn cần xem xét tài liệu cụ thể cho loại dự án mà bạn muốn tạo để biết thêm chi tiết về cách sử dụng Eclipse.

### **1.1.2 Tính kế thừa của maven**

Trong Maven, tính kế thừa (inheritance) là một khái niệm quan trọng và cho phép bạn chia sẻ cấu hình, cài đặt, và plugin giữa các dự án Maven. Điều này giúp giảm sự lặp lại, quản lý dự án dễ dàng hơn và đảm bảo tính nhất quán giữa các dự án. Cụ thể, tính kế thừa trong Maven thể hiện qua việc sử dụng thẻ `<parent>` trong tệp pom.xml của một dự án con để kế thừa các thông tin từ dự án cha.

Dưới đây là một ví dụ về cách tính kế thừa hoạt động trong Maven:

1. \*\*Dự án cha (Parent Project)\*\*:

- Bạn tạo một dự án Maven gốc hoặc cha (parent project).

- Dự án cha có một tệp pom.xml chứa các thông tin cơ bản về dự án, cũng như các plugin, dependencies, và cài đặt chung.

2. \*\*Dự án con (Subproject)\*\*:

- Bạn tạo một hoặc nhiều dự án con (subprojects), mỗi dự án con có tệp pom.xml riêng.

- Trong tệp pom.xml của dự án con, bạn sử dụng thẻ `<parent>` để chỉ định dự án cha mà nó sẽ kế thừa.

Khi bạn chạy các mục tiêu Maven cho dự án con, Maven sẽ tự động kế thừa các cài đặt và plugin từ dự án cha. Điều này giúp đảm bảo rằng các dự án con duy trì tính nhất quán với dự án cha và giúp tránh lặp lại cài đặt và cấu hình chung.

Tính kế thừa trong Maven không chỉ giúp quản lý các dự án con dễ dàng mà còn giúp tạo ra một cấu trúc dự án chặt chẽ và sáng sủa, đồng thời tối ưu hóa quản lý dự án trong môi trường phát triển phần mềm.

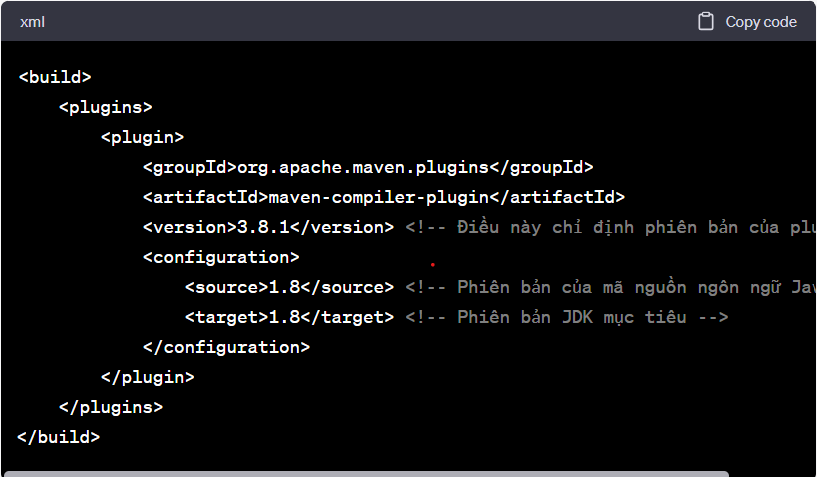
### **1.1.3 Tính mở rộng của maven**

Maven là một công cụ quản lý dự án và xây dựng phần mềm mạnh mẽ và phổ biến trong cộng đồng phát triển phần mềm Java. Maven có khả năng mở rộng thông qua các plugin và tính năng linh hoạt, cho phép bạn tùy chỉnh và mở rộng khả năng của nó theo nhu cầu cụ thể của dự án của bạn. Dưới đây là một số cách mở rộng Maven:

1. \*\*Sử dụng Plugin Maven\*\*: Maven cho phép bạn sử dụng các plugin để thực hiện các nhiệm vụ cụ thể. Có nhiều plugin có sẵn cho Maven để làm việc với nhiều nhiệm vụ khác nhau như biên dịch mã nguồn, đóng gói ứng dụng, chạy kiểm thử, tạo tài liệu, v.v. Bạn cũng có thể viết các plugin tùy chỉnh để mở rộng Maven cho nhu cầu cụ thể của dự án.

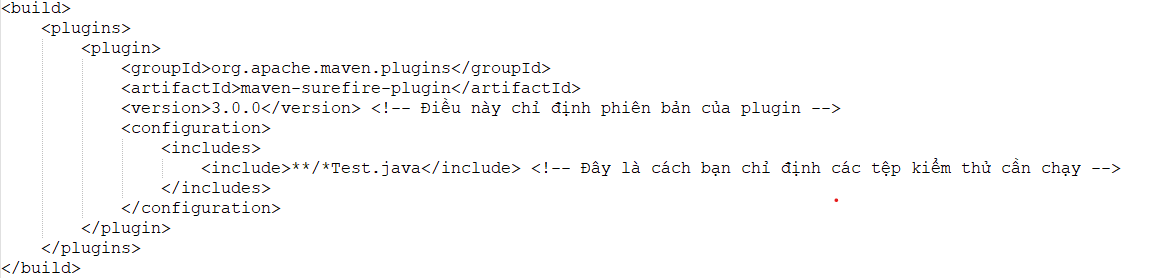
#### **Plugin Compiler (maven-compiler-plugin)**

: Plugin này cho phép bạn cấu hình và thực hiện biên dịch mã nguồn Java trong dự án. Bạn có thể chỉ định phiên bản của JDK, cấu hình thư mục mã nguồn và thư mục mã sau khi biên dịch

****

#### **Plugin Surefire (maven-surefire-plugin)**

**:** Plugin này sử dụng để chạy các bài kiểm thử (unit tests) trong dự án. Bạn có thể cấu hình nó để tìm và chạy các bài kiểm thử và xem kết quả.



1. Trong ví dụ trên, chúng ta đã chỉ định phiên bản của plugin và sử dụng cấu hình để chỉ định các tệp kiểm thử (unit test files) cần chạy. Ở đây, chúng ta đã sử dụng **<includes>** để chỉ định rằng tất cả các tệp có tên kết thúc bằng "Test.java" sẽ được chạy là các bài kiểm thử.
2. Sau khi bạn đã cấu hình plugin này trong **pom.xml**, bạn có thể sử dụng lệnh Maven để chạy các bài kiểm thử. Ví dụ, để chạy bài kiểm thử, bạn có thể sử dụng lệnh sau:



1. Plugin Surefire sẽ tìm kiếm và chạy các bài kiểm thử theo cấu hình đã được thiết lập trong **pom.xml**. Kết quả của các bài kiểm thử sẽ được hiển thị trong đầu ra của lệnh Maven.

Lưu ý rằng bạn có thể tùy chỉnh nhiều cài đặt khác trong plugin Surefire để đáp ứng các yêu cầu cụ thể của dự án của bạn, chẳng hạn như cấu hình reports, thực thi bài kiểm thử đồng thời, hoặc thiết lập hạn chế thời gian chạy bài kiểm thử.

#### **Plugin Assembly (maven-assembly-plugin)**

: Được sử dụng để tạo các gói (package) tùy chỉnh từ các tài nguyên trong dự án. Bạn có thể tạo các gói như tệp JAR, WAR, ZIP, hoặc các kiểu gói tùy chỉnh khác.

Dưới đây là một ví dụ về cách sử dụng Maven Assembly Plugin (maven-assembly-plugin) để tạo các gói tùy chỉnh từ các tài nguyên trong một dự án Maven:

1. Đầu tiên, bạn cần thêm plugin này vào tệp `pom.xml` của dự án Maven của mình. Dưới đây là một ví dụ:

```xml

<build>

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-assembly-plugin</artifactId>

<version>3.3.0</version> <!-- Điều này chỉ định phiên bản của plugin -->

<executions>

<execution>

<id>make-assembly</id>

<phase>package</phase> <!-- Đặt giai đoạn trong quá trình build maven -->

<goals>

<goal>single</goal>

</goals>

<configuration>

<descriptors>

<descriptor>src/assembly/assembly.xml</descriptor> <!-- Đường dẫn đến tệp cấu hình assembly -->

</descriptors>

</configuration>

</execution>

</executions>

</plugin>

</plugins>

</build>

```

2. Trong ví dụ trên, chúng ta đã chỉ định phiên bản của plugin và cấu hình nó để thực hiện trong giai đoạn "package". Điều này có nghĩa rằng khi bạn chạy lệnh `mvn package`, plugin Assembly sẽ được kích hoạt để tạo gói tùy chỉnh.

3. Bạn cần tạo một tệp cấu hình assembly (ví dụ: `assembly.xml`) tại đường dẫn `src/assembly/`. Trong tệp này, bạn có thể định rõ cách gói tùy chỉnh sẽ được tạo, bao gồm tài nguyên và định dạng gói. Dưới đây là một ví dụ cơ bản:

```xml

<assembly>

<id>my-assembly</id>

<formats>

<format>zip</format> <!-- Định dạng gói (ví dụ: ZIP) -->

</formats>

<fileSets>

<fileSet>

<directory>src/main/resources</directory> <!-- Thư mục chứa tài nguyên bạn muốn đóng gói -->

<outputDirectory>/</outputDirectory> <!-- Thư mục đích trong gói -->

</fileSet>

</fileSets>

</assembly>

```

4. Sau khi bạn đã cấu hình plugin Assembly trong `pom.xml` và tạo tệp cấu hình assembly, bạn có thể chạy lệnh Maven để tạo gói tùy chỉnh. Ví dụ, để tạo gói ZIP, bạn có thể sử dụng lệnh sau:

```

mvn package

```

Gói tùy chỉnh sẽ được tạo và lưu trữ trong thư mục `target`.

Lưu ý rằng bạn có thể tùy chỉnh nhiều cài đặt khác trong plugin Assembly để đáp ứng các yêu cầu cụ thể của dự án của bạn, chẳng hạn như đặt tên gói, bao gồm hoặc loại trừ các tệp, thay đổi định dạng gói, và nhiều cài đặt khác.

#### **Plugin Site (maven-site-plugin)**

: Sử dụng để tạo trang web tài liệu cho dự án. Bạn có thể tạo tài liệu bằng cách sử dụng định dạng AsciiDoc, Markdown, HTML, hoặc một số định dạng tài liệu khác.

Dưới đây là một ví dụ về cách sử dụng Maven Site Plugin (maven-site-plugin) để tạo trang web tài liệu cho dự án Maven:

1. Đầu tiên, bạn cần thêm plugin này vào tệp `pom.xml` của dự án Maven của mình. Dưới đây là một ví dụ:

```xml

<build>

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-site-plugin</artifactId>

<version>3.10.1</version> <!-- Điều này chỉ định phiên bản của plugin -->

</plugin>

</plugins>

</build>

```

2. Bạn cũng có thể chỉ định định dạng tài liệu mà bạn muốn sử dụng trong tệp `pom.xml`. Dưới đây là một ví dụ:

```xml

<reporting>

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-project-info-reports-plugin</artifactId>

<version>3.1.2</version> <!-- Phiên bản của plugin -->

</plugin>

</plugins>

</reporting>

```

3. Bây giờ bạn có thể sử dụng các định dạng tài liệu như AsciiDoc, Markdown, HTML hoặc một số định dạng tài liệu khác để tạo tài liệu cho dự án của bạn. Đối với ví dụ này, chúng ta sẽ sử dụng định dạng Markdown.

4. Tạo thư mục `src/site/markdown` trong dự án của bạn và thêm các tệp Markdown (ví dụ: `index.md`) vào thư mục này để viết tài liệu.

5. Tiếp theo, bạn cần chỉ định trang chính (index) trong tệp `pom.xml`. Dưới đây là một ví dụ:

```xml

<project>

...

<distributionManagement>

<site>

<id>project-site</id>

<url>scp://your-server/path/to/site</url>

</site>

</distributionManagement>

<scm>

...

</scm>

<reporting>

<outputDirectory>target/site</outputDirectory>

</reporting>

</project>

```

Trong phần này, bạn cần thay thế `<id>project-site</id>` và `<url>scp://your-server/path/to/site</url>` bằng các giá trị phù hợp cho máy chủ của bạn và đường dẫn đến trang web tài liệu của dự án.

6. Bây giờ, bạn có thể chạy lệnh Maven để tạo trang web tài liệu. Ví dụ, để tạo trang web tài liệu trong định dạng HTML, bạn có thể sử dụng lệnh sau:

```

mvn site

```

Sau khi lệnh này chạy xong, trang web tài liệu sẽ được tạo và lưu trữ trong thư mục `target/site`.

Lưu ý rằng bạn có thể tùy chỉnh nhiều cài đặt khác trong Maven Site Plugin để đáp ứng các yêu cầu cụ thể của dự án của bạn, chẳng hạn như tùy chỉnh giao diện trang web tài liệu hoặc thêm bất kỳ trang con nào.

#### **Plugin Docker (fabric8-maven-plugin)**

: Cho phép bạn tạo và quản lý các hình ảnh Docker và triển khai ứng dụng vào container Docker

Dưới đây là một ví dụ về cách sử dụng plugin Docker (fabric8-maven-plugin) để tạo và quản lý hình ảnh Docker và triển khai ứng dụng vào container Docker trong một dự án Maven:

1. Đầu tiên, bạn cần thêm plugin này vào tệp `pom.xml` của dự án Maven của mình. Dưới đây là một ví dụ:

```xml

<build>

<plugins>

<plugin>

<groupId>io.fabric8</groupId>

<artifactId>fabric8-maven-plugin</artifactId>

<version>5.5.2</version> <!-- Điều này chỉ định phiên bản của plugin -->

<executions>

<execution>

<id>build-docker-image</id>

<phase>install</phase> <!-- Đặt giai đoạn trong quá trình build maven -->

<goals>

<goal>build</goal>

</goals>

</execution>

</executions>

</plugin>

</plugins>

</build>

```

2. Trong ví dụ trên, chúng ta đã chỉ định phiên bản của plugin và cấu hình nó để thực hiện trong giai đoạn "install". Điều này có nghĩa rằng khi bạn chạy lệnh `mvn install`, plugin Docker sẽ được kích hoạt để xây dựng hình ảnh Docker từ dự án của bạn.

3. Bạn cần cung cấp cấu hình Docker thông qua tệp `src/main/fabric8/docker.json` hoặc `src/main/fabric8/docker.properties` để định cấu hình cụ thể cho hình ảnh Docker mà bạn muốn xây dựng. Cấu hình này có thể bao gồm thông tin như cổng, biến môi trường, tài nguyên, v.v.

4. Sau khi bạn đã cấu hình plugin Docker trong `pom.xml` và tạo tệp cấu hình Docker, bạn có thể chạy lệnh Maven để xây dựng hình ảnh Docker. Ví dụ, để xây dựng hình ảnh Docker, bạn có thể sử dụng lệnh sau:

```

mvn install

```

5. Plugin Docker sẽ sử dụng cấu hình đã được thiết lập để tạo hình ảnh Docker và lưu trữ nó trong Docker Registry hoặc máy chủ Docker cục bộ.

6. Bạn cũng có thể triển khai ứng dụng vào container Docker bằng cách sử dụng các mục tiêu khác của plugin Docker, chẳng hạn như `start`, `stop`, `deploy`, v.v.

Lưu ý rằng plugin Docker (fabric8-maven-plugin) cung cấp nhiều tính năng và tùy chọn cấu hình cho việc quản lý và triển khai ứng dụng vào Docker containers. Điều này cho phép bạn linh hoạt thích ứng với các yêu cầu cụ thể của dự án của bạn trong việc tạo và quản lý các hình ảnh Docker.

2. \*\*Cấu hình Tùy chỉnh (Custom Configuration)\*\*: Bạn có thể tùy chỉnh tệp pom.xml của dự án để cấu hình Maven theo cách bạn muốn. Bạn có thể thêm dependencies, cài đặt, và thông số tùy chỉnh riêng để đáp ứng yêu cầu của dự án.

3. \*\*Sử dụng Profiles\*\*: Maven cho phép bạn sử dụng profiles để kích hoạt cài đặt và cấu hình khác nhau dựa trên môi trường hoặc mục tiêu cụ thể. Điều này giúp bạn quản lý môi trường phát triển và triển khai.

4. \*\*Kết hợp với Các Công Cụ Bên Ngoài (External Tools)\*\*: Maven có thể tích hợp với nhiều công cụ khác nhau như Jenkins, SonarQube, Nexus, và nhiều công cụ quản lý mã nguồn (VCS) như Git hoặc Subversion. Sử dụng các tích hợp này, bạn có thể mở rộng quá trình tích hợp liên tục (CI/CD) và quản lý mã nguồn.

5. \*\*Sử dụng Repository Cá Nhân (Private Repository)\*\*: Maven cho phép bạn sử dụng kho lưu trữ cá nhân (private repository) để quản lý các thư viện và plugin tùy chỉnh. Điều này giúp bạn quản lý và chia sẻ các thành phần phần mềm nội bộ của tổ chức hoặc dự án.

1. \*\*Xây dựng Tự động (Automatic Builds)\*\*: Bạn có thể sử dụng các công cụ tích hợp Maven vào các môi trường tích hợp liên tục (CI) để xây dựng và triển khai tự động dự án của bạn sau mỗi lần commit hoặc push code.

### **Scope**

Trong Maven, có nhiều loại scope khác nhau để xác định cách dependency (thư viện) được quản lý và sử dụng trong dự án. Dưới đây là danh sách các loại scope phổ biến trong tệp `pom.xml` của Maven:

1. \*\*compile (Mặc định)\*\*:

- Dependency có scope compile sẽ được bao gồm trong quá trình biên dịch và quá trình chạy ứng dụng.

2. \*\*provided\*\*:

- Dependency có scope provided được coi là được cung cấp bởi môi trường triển khai (runtime environment) và không cần được bao gồm trong quá trình biên dịch.

- Ví dụ: Servlet API trong môi trường Java EE.

3. \*\*runtime\*\*:

- Dependency có scope runtime không được bao gồm trong quá trình biên dịch, nhưng nó sẽ được bao gồm trong quá trình chạy ứng dụng.

4. \*\*test\*\*:

- Dependency có scope test chỉ được sử dụng trong quá trình thực hiện kiểm thử (test) và không được bao gồm trong quá trình biên dịch hoặc chạy ứng dụng.

5. \*\*system\*\*:

- Dependency có scope system cho phép bạn chỉ định một tệp JAR hoặc thư mục trên hệ thống tệp local của bạn. Maven không sử dụng kho lưu trữ để tìm dependency này.

6. \*\*import\*\*:

- Scope import được sử dụng cho các dependency import từ một tệp POM khác.

7. \*\*compile-only\*\*:

- Scope compile-only tương tự scope compile, nhưng không được đánh dấu cho việc bao gồm trong quá trình chạy ứng dụng. Nó thường được sử dụng cho các annotation processors và giúp tránh xung đột trong quá trình chạy ứng dụng.

8. \*\*test-compile\*\*:

- Scope test-compile tương tự scope compile, nhưng chỉ áp dụng cho quá trình biên dịch mã kiểm thử.

9. \*\*test-runtime\*\*:

- Scope test-runtime tương tự scope runtime, nhưng chỉ áp dụng cho quá trình chạy mã kiểm thử.

Những scope này cho phép bạn quản lý dependency một cách hiệu quả và đảm bảo rằng các thư viện được sử dụng trong dự án của bạn đúng cách trong quá trình biên dịch và chạy ứng dụng.

* + 1. **Maven đóng file riêng**

Maven không cung cấp chức năng để đóng tệp (zip file) riêng trong quá trình biên dịch và xây dựng dự án. Tuy nhiên, bạn có thể sử dụng plugin Maven để thực hiện công việc này. Một plugin phổ biến cho việc đóng tệp là `maven-assembly-plugin`. Plugin này cho phép bạn tạo các gói tệp nén (archive) tùy chỉnh, chẳng hạn như ZIP hoặc JAR, từ các tài nguyên trong dự án.

Dưới đây là cách bạn có thể sử dụng `maven-assembly-plugin` để đóng tệp ZIP riêng trong dự án Maven:

1. Thêm plugin `maven-assembly-plugin` vào tệp `pom.xml` của dự án:

```xml

<build>

<plugins>

<plugin>

<artifactId>maven-assembly-plugin</artifactId>

<version>3.3.0</version> <!-- Sử dụng phiên bản mới nhất của plugin -->

<executions>

<execution>

<id>make-assembly</id>

<phase>package</phase>

<goals>

<goal>single</goal>

</goals>

</execution>

</executions>

<configuration>

<descriptors>

<descriptor>src/assembly/your-assembly.xml</descriptor>

</descriptors>

</configuration>

</plugin>

</plugins>

</build>

```

2. Tạo một tệp XML (ví dụ: `your-assembly.xml`) trong thư mục `src/assembly` để cấu hình gói tệp ZIP của bạn. Dưới đây là một ví dụ đơn giản của `your-assembly.xml`:

```xml

<assembly xmlns="http://maven.apache.org/ASSEMBLY/2.0.0">

<id>zip-assembly</id>

<formats>

<format>zip</format>

</formats>

<includeBaseDirectory>false>

<fileSets>

<fileSet>

<directory>${project.build.directory}</directory>

<outputDirectory>/</outputDirectory>

<includes>

<include>\*.jar</include>

</includes>

</fileSet>

</fileSets>

</assembly>

```

3. Chạy lệnh sau để xây dựng dự án và tạo tệp ZIP:

```

mvn clean package

```

Tệp ZIP được tạo ra sau quá trình biên dịch trong thư mục `target` của dự án.

Chú ý rằng Maven có rất nhiều plugin và cách khác nhau để đóng tệp, và bạn có thể tùy chỉnh cấu hình theo nhu cầu cụ thể của bạn bằng cách sử dụng các plugin và descriptors (tệp XML).

* + 1. **Git đóng file riêng**

Trong Git, để đóng một tệp riêng (đóng gói một tệp hoặc thư mục thành một tệp riêng biệt), bạn có thể sử dụng các lệnh `git archive` hoặc `git bundle`. Dưới đây là cách bạn có thể sử dụng mỗi lệnh:

### Sử dụng `git archive`:

1. \*\*Tạo một tệp archive (ví dụ: file.zip) cho một thư mục hoặc tệp cụ thể:\*\*

```bash

git archive --format=zip -o file.zip HEAD:path/to/your/file/or/directory

```

- `--format=zip` xác định định dạng tệp archive là ZIP.

- `-o file.zip` xác định tên của tệp archive đầu ra.

2. \*\*Tệp ZIP (hoặc tệp riêng) sẽ được tạo trong thư mục làm việc hiện tại.

### Sử dụng `git bundle`:

1. \*\*Tạo một bundle file (ví dụ: bundlefile.bundle) cho một phạm vi commit cụ thể (commit range) bằng cách chỉ định commit cuối cùng và commit gốc cho phạm vi:

```bash

git bundle create bundlefile.bundle HEAD~3..HEAD

```

- `HEAD~3` là commit cuối cùng (HEAD) cộng với 3 commit trước đó.

- `HEAD` là commit gốc.

2. \*\*Tệp bundle (hoặc tệp riêng) sẽ được tạo trong thư mục làm việc hiện tại.

Lưu ý rằng tệp bundle có thể được chia sẻ và sau đó được giải nén trở lại bằng `git clone`, đây là một cách tiện lợi để chia sẻ một phạm vi commit cụ thể với người khác mà không cần sử dụng kho lưu trữ từ xa (remote repository).

Table of Contents

[**I Buổi I** 1](#_Toc148565795)

[**1.1 Tìm hiểu maven** 1](#_Toc148565796)

[**1.1.1 Dựng một dự án (project) bằng Eclipse** 1](#_Toc148565797)

[**1.1.2 Tính kế thừa của maven** 2](#_Toc148565798)

[**1.1.3 Tính mở rộng của maven** 3](#_Toc148565799)

[**1.1.4** **Scope** 13](#_Toc148565800)

[**1.1.5** **Maven đóng file riêng** 14](#_Toc148565801)

[**1.1.6** **Git đóng file riêng** 16](#_Toc148565802)

[**1.1.7** **Dựng dự án đóng file riêng** 17](#_Toc148565803)

### **Dựng dự án đóng file riêng**

Nếu bạn muốn xây dựng một dự án và đóng gói một hoặc nhiều tệp hoặc thư mục riêng lẻ thành một tệp archive (ví dụ: ZIP) trong quá trình xây dựng dự án của mình, bạn có thể sử dụng các công cụ build automation như Maven hoặc Gradle để thực hiện công việc này. Dưới đây là cách thực hiện điều đó trong Maven:

1. \*\*Sử dụng Maven Assembly Plugin\*\*:

- Thêm `maven-assembly-plugin` vào tệp `pom.xml` của dự án. Đảm bảo bạn cấu hình plugin để đóng gói các tệp hoặc thư mục cụ thể vào tệp ZIP hoặc bất kỳ định dạng nén nào khác.

Ví dụ về cấu hình plugin trong `pom.xml`:

```xml

<build>

<plugins>

<plugin>

<artifactId>maven-assembly-plugin</artifactId>

<version>3.3.0</version> <!-- Phiên bản mới nhất của plugin -->

<executions>

<execution>

<id>make-assembly</id>

<phase>package</phase>

<goals>

<goal>single</goal>

</goals>

</execution>

</executions>

<configuration>

<descriptors>

<descriptor>src/main/assembly/your-assembly.xml</descriptor>

</descriptors>

</configuration>

</plugin>

</plugins>

</build>

```

- Tạo một tệp XML (ví dụ: `your-assembly.xml`) trong thư mục `src/main/assembly` để cấu hình đóng gói các tệp hoặc thư mục bạn muốn vào tệp archive. Chạy `mvn package` để xây dựng dự án và tạo tệp ZIP.

2. \*\*Sử dụng Maven Exec Plugin\*\* (đối với đóng gói tệp riêng lẻ):

- Sử dụng `maven-exec-plugin` để thực hiện các lệnh tùy chỉnh trong quá trình xây dựng dự án. Bạn có thể sử dụng lệnh nén tệp, chẳng hạn như `zip` hoặc `tar`, để tạo một tệp ZIP hoặc nén khác trong quá trình xây dựng.

Ví dụ về cấu hình `maven-exec-plugin` trong `pom.xml`:

```xml

<build>

<plugins>

<plugin>

<groupId>org.codehaus.mojo</groupId>

<artifactId>exec-maven-plugin</artifactId>

<version>3.0.0</version>

<executions>

<execution>

<id>create-zip</id>

<phase>package</phase>

<goals>

<goal>exec</goal>

</goals>

</execution>

</executions>

<configuration>

<executable>zip</executable>

<arguments>

<argument>-r</argument>

<argument>output.zip</argument>

<argument>file1.txt</argument>

<argument>file2.txt</argument>

<!-- Thêm các tệp hoặc thư mục khác cần đóng gói -->

</arguments>

</configuration>

</plugin>

</plugins>

</build>

```

- Chạy `mvn package` để xây dựng dự án và thực hiện lệnh đóng gói tệp riêng lẻ.

Chú ý rằng bạn cần sử dụng cơ bản hoặc kịch bản thích hợp để đóng gói các tệp hoặc thư mục cụ thể và tạo tệp archive như bạn muốn. Điều này cho phép bạn tạo một quy trình xây dựng tùy chỉnh để đáp ứng nhu cầu cụ thể của dự án của bạn.