**Bài 14**

**Packaging & Dependences Management**

Module: JWBD

**Kiểm tra bàitrước**

Hỏivàtraođổivềcáckhókhăngặpphảitrongbài“JDBC Querying & Transaction"

Tómtắtlạicácphầnđãhọctừbài“JDBC Querying & Transaction”

# Mụctiêu

* Trìnhbàyđượckháiniệmđónggói
* Trìnhbàyđượckháiniệmthưviệnphụthuộc
* TrìnhbàyđượccácđịnhdạngđónggóiứngdụngJava
* TrìnhbàyđượccáccôngcụđónggóiứngdụngJavaphổbiến
* BiếtcáchsửdụngMavenđểquảnlýthưviệnphụthuộc
* BiếtcáchsửdụngMavenđểđónggóiứngdụngJava
* BiếtcáchsửdụngGradleđểquảnlýthưviệnphụthuộc
* BiếtcáchsửdụngGradleđểđónggóiứngdụngJava

# 1. Đóng gói ứng dụng Java

## Gói phần mềm chuyển giaođược

* Trongđasốtrườnghợp,khôngthểchuyểngiaomãnguồnchokhách hàng
* Hệthốngcủakháchhàng(môitrườngproduction)làmôitrườngthực thi,khôngphảimôitrườngpháttriển
* Môitrườngproductionkhôngcónhữngcôngcụđặcthùcủamôi trườngpháttriểnnhưcompiler/debugger/testingtool...
* Góiphầnmềm chuyểngiaotớikháchhàngphảiởtrạngtháisẵnsàng để hoạtđộng

### Những tài nguyên cần thiết để ứng dụng có thể hoạt động

JREcầncónhữngtàinguyênsauđểcóthểkhởiđộngvàduytrìthựcthi ứngdụng:

* Tậptấtcảcácclasscầnthiết,nằmdướidạngfile.class(byte-code)
* Tậptấtcảcáctàinguyênkhác:cấuhình,messagebundle,HTML,CSS, static files,...

## Những file .class cần thiết

* JavaAPIvàcácthưviệnbuiltin,sẵncótrongJRE.Khôngcầnthiếtphải bổ sung vào gói chuyểngiao.
* Thưviệnphụthuộcbuildsẵn.Cóthểcầnbổsungvàogóichuyểngiao trongtrườnghợpmôitrườngproductionkhôngcósẵn.
* Mã byte-code của bản thân chương trình, có được sau quá trình compile.Baogồmbyte-codecủachươngtrìnhchính–luôncầnbổ sung vào gói chuyển giao, và test cases –không bổ sung vào gói chuyểngiao.

## Gói phần mềm Java

Gói phần mềm Java có thể được đóng gói và chuyển giao tới môi trường thực thi (hệ thống) dưới hai hình thức:

* Một gói độc lập (**standalone**), được thực thi bởi JRE của hệ thống.
* Một gói tự túc (**self-contained**): bao gồm standalone application, cộng thêm một phiên bản JRE riêng, có khả năng tự khởi động được.

Lưu ý: còn có định dạng *Java Web Start application*, và *Java embered* không được kể tới ở đây.

# 2. Công cụ để đónggói

## Thư viện phụ thuộc (Dependence)

• Là các cấu phần chương trình tới từ bên ngoài mà đảm nhiệm một nhiệm vụ cụ thể. Ví dụ: :JDBC, Thymeleaf, Spring...

### Công cụ quản lý thư viện phụ thuộc (Dependence Management tool)

* Là những phần mềm giúp tích hợp thư viện phụ thuộc vào bản thân chương trình đang được phát triển. Ví dụ Composer, NPM, Ant, Maven, Gradle...
* Sử dụng những file cấu hình (chẳng hạn composer.json, package.json, pom.xml, build.gradle...) để làm rõ:
* Những phụ thuộc nào cần tìm về
* Phiên bản nào cần tìm về
* Tìm các phụ thuộc về từ repository nào

## Dependence Repository

* Kho chứa các thư viện phụ thuộc của các Công cụ Quản lý Thư viện phụ thuộc.
* NPM Repository, Maven Central, Packagist... là các ví dụ.

**Tại sao cần dùng Công cụ Quản lý Thư viện phụ thuộc**

* Đảm bảo môi trường phát triển (Dev Environment) và môi trường thực thi (Production Environment) sử dụng cùng một phiên bản của Thư viện phụ thuộc.
* Giúp việc quản lý (bổ sung, thay thế, cập nhật) các thư viện phụ thuộc trở nên dễ dàng hơn

## Building và Build tool

* **Building** là quá trình đóng gói tất cả các tài nguyên cần thiết thành một chương trình hoạt động được.
* **Build tool** là những chương trìnhg có thể tự động hóa tiến trình build. Ví dụ: Intellij builder, Ant, Maven, Gradle, Gulp, Webpack...
* Các nhiệm vụ thường gặp của build tool:
* Compile chương trình
* Thực thi các kiểm thử
* Đóng gói chương trình cùng các thư viện phụ thuộc
* Deploy
* Một số build tool đảm nhiệm thêm cả chức năng của một Trình quản lý thư viện phụ thuộc

## Ant, Maven, Gradle

* **ApacheAnt**:mộtcôngcụviếtbằngJava,đượcxâydựngvớimục đích làmmộtbuildtooluyểnchuyểnchocácứngdụngJava.Sử dụngđịnh dạngXMLchotậptincấuhình.
* **Apache Maven**: một công cụ quản lý thư viện phụ thuộc kiêm build tool cho các ứng dụng Java. Khả năng quản lý tốt hơn so với Ant, nhưng kém uyển chuyển hơn. Sử dụng định dạng XML cho tập tin cấu hình.
* **Gradle**: một công cụ quản lý thư viện phụ thuộc kiêm build tool cho các ứng dụng Java. Được xây dựng trên concept của Ant và Maven. Sử dụng ngôn ngữ Groovy cho tập tin cấu hình.

### Các thông tin trong tập tin cấu hình Gradle

* Các plugin được áp dụng
* Các Dependency Repository
* Các Dependency
* Các Task

### Cấu hình Gradle: Plugins

* Gradlesửdụngthiếtkế“mọichứcnăngđềulàplugin”.
* Mụcpluginkhaibáocácchứcnăng(cósẵnhoặctớitừdependence) muốn sửdụng

|  |  |
| --- | --- |
| plugins { i d **'java'** i d **'war'** id  **'jacoco'** i d **"org.flywaydb.flyway"**  **'5.1.4'**  } | version |

## Cấu hình Gradle: Repository

CóthểsửdụngMavenCentralhoặcrepositorykhác

|  |
| --- |
| repositories **{** mavenCentral()  maven **{** u r l **'https://jitpack.io' }**  **}** |

## Cấu hình Gradle: Dependencies

Mỗidependencyđikèmvớivịtrí,phiênbản,vàscopekhảdụng.

|  |
| --- |
| **dependencies** {  / / Lombok's Buider  **compile** 'org.projectlombok**:lombok:**1.18.10'  / / Logging  **compile** ' o r g.sl f4j : **slf4j-api**:1.7.25'  **compile** 'ch.qos.logback**:logback-classic**:1.2.3'  / / DB Migration  **providedCompile** 'org.flywaydb**:flyway-core:**5.1.4'  } |

## Các Dependency Scope quan trọng

**Compile**: dependency sẽ được đóng gói cùng với chương trình. Dependency cũng sẽ khả dụng tại mọi ngữ cảnh thực thi dự án: compile,test,execution...

* **Provided**: ngầm hiểu rằng dependency sẽ tồn tại tại ngữ cảnh thực thi chương trình trong tương lai, do đó không cần đóng gói cùng chương trình.Nhưng dependency vẫn khả dụng tại ngữ cảnh compile, test.
* **Test**:dependencykhôngđượcsửdụngtrongngữcảnhthựcthi

chương trình trong tương lai, do đó không cần đóng gói cùng chương trình. Dependency chỉ khả dụng trong ngữ cảnh test và compile test classes.

## Cấu hình Gradle: Task

Làcácđơnvịcông việccóthểphảithựcthitrongtiếntrìnhbuild.Chẳnghạn như compile, tạo document, chạy kiểm thử,...

jacocoTestReport **{** reports **{** xml.enabled **false** csv.enabled **false**

html.destination file(**"**$**{**buildDir**}/jacocoHtml"**)

**} }**

# 3. Sản phẩm của tác vụ đóng gói

## .JAR file

* Một định dạng tập tin phổ biến của gói phần mềm/thư viện Java
* Là một tập tin nén chứa các file .class, resource và metadata
* Được tạo ra bởi công cụ **”jar”** trong bộ công cụ JDK, hoặc bằng công cụ đóng gói

|  |
| --- |
| META-INF/  |--MANIFEST.MF com/  |--apwj/  |--MyApplication.class |

## .WAR file

* ĐịnhdạngphổbiếnđểđónggóiJavaWebApplication
* Chứacáctàinguyênweb,cùngvớicácclasscủaapplicationđược dùngnhưcácthưviệnchoservletcontainer
* Đượcgọithựcthibởiservletcontainer METAMANIFEST.MF -INF/

WEB-INF/ web.xml j s p / helloWorld.jsp

classes/ s t a t i c / templates/

application.properties

l i b /

/ / \* . j a r f i l e s as l i b s

## Tổng kết

* Đóng gói và quản lý phụ thuộc là tiến trình quan trọng xuyên suốt hoạt động phát triển phần mềm.
* Có nhiều công cụ đóng gói, phổ biến trong hệ sinh thái Java là Ant, Maven, Gradle
* Gradle sử dụng tập tin cấu hình là build.gradle, được viết bằng ngôn ngữ Groovy
* Tập tin JAR và WAR là hai định dạng đóng gói Java Application phổ biến. JAR dùng để đóng gói nhiều loại thư viện/ứng dụng. WAR dùng để đóng gói ứng dụng Web.



R a i s i n g t h e b a r