[GCS\_TRA]Topic-Maven

*Người thực hiện:*

Nguyễn Thị Lan Phương

Phạm Thị Kim Hiền

Mục Lục

[A. Tổng quát: 1](#_Toc463200682)

[I. Khái niệm: 1](#_Toc463200683)

[II. Cách hoạt động: 2](#_Toc463200684)

[III. Lợi ích 3](#_Toc463200685)

[IV. Tại sao phải sử dụng maven 4](#_Toc463200686)

[B. Nội dung chi tiết: 5](#_Toc463200687)

[I. file POM cho Maven 5](#_Toc463200688)

[II. Build Life Cycle 9](#_Toc463200689)

[1. Maven compile (mvn compile) 9](#_Toc463200690)

[2. Maven package (mvn package) 9](#_Toc463200691)

[3. Maven install (mvn install) 10](#_Toc463200692)

[4. Maven deploy (mvn deploy) 10](#_Toc463200693)

[III. Build Profile 13](#_Toc463200694)

[IV. Repositories 16](#_Toc463200695)

[1. Maven Local Repository 17](#_Toc463200696)

[2. Maven Central Repository 18](#_Toc463200697)

[3. Maven Remote Repository 19](#_Toc463200698)

[C. Demo: 20](#_Toc463200699)

[I. Creating Project 20](#_Toc463200700)

[1. Tạo Maven Project 21](#_Toc463200701)

[2. Cấu hình Maven 24](#_Toc463200702)

[3. Cài đặt Project bằng Maven 25](#_Toc463200703)

[4. Giải thích nguyên tắc hoạt động của Maven 26](#_Toc463200704)

[5 Xem Local repository 27](#_Toc463200705)

[6. Xem Maven Repository trên Internet 28](#_Toc463200706)

[7. Cấu hình Maven download source & javadoc 28](#_Toc463200707)

[8. Quản lý phụ thuộc trong Maven 30](#_Toc463200708)

[II. Build and Test Project 34](#_Toc463200710)

[III. External Dependencies 50](#_Toc463200711)

[IV. Build Automation 51](#_Toc463200712)

[D. Tài liệu tham khảo 52](#_Toc463200713)

**TÌM HIỂU APACHE MAVEN**

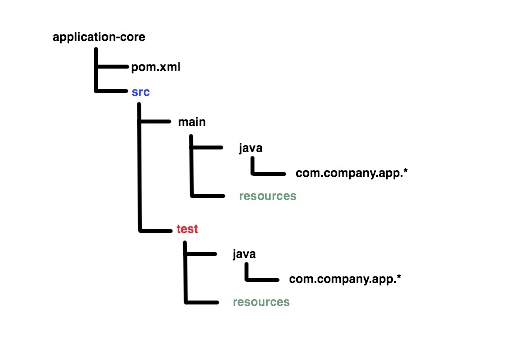
# Tổng quát:

## Khái niệm:

* Apache maven là một chương trình quản lý dự án cho phép các developers có thể quản lý về version, các dependencies ( các thư viện sử dụng trong dự án ) , quản lý build, tự động download javadoc & source, ....
* Maven được sử dụng chính trong Java, nhưng bên cạnh đó còn hỗ trợ cho các ngôn ngữ khác như PHP, C#, Ruby, Scala… Maven được quản lý bởi Apache Software Foundation, nó là một phần của dự án Jakarta Project
* Apache Maven 1.x không còn được hỗ trợ. Để biết thêm thông tin, xem trên website: <http://maven.apache.org/plugins/maven-deploy-plugin/> . Người dùng được khuyến khích chuyển sang phiên bản hiện tại của Apache Maven là 2 và 3: <http://maven.apache.org/index.html> .
* Maven sẽ tự động download các thư viện được khai báo trong file POM hay các thư viện liên quan cần thiết sử dụng cho project (ví dụ: thư viện hibernate, thư viện này có rất nhiều thư viện khác liên quan) ⇒ Hãy thử tưởng tượng nếu không có maven, bạn sẽ phải add bằng tay một đống thư viện như trên cùng với version phù hợp!
* Thư viện hibernate trong ví dụ trên sẽ nằm ở thư mục C:\Users{username}.m2\repository\org\hibernate\hibernate-core\5.0.0.CR2.
* Các bạn có thể mở file pom (hibernate-core-5.0.0.CR2.pom) của hibernate lên, để xem nó phụ thuộc những dependencies nào, và cũng để hiểu rõ hơn về maven.

## Cách hoạt động:

Maven dùng khái niệm Project Object Model (POM) để mô tả việc build project, các thành phần phụ thuộc và các module. Mỗi dự án sẽ được mô tả trong một file có định dạng XML có tên "pom.xml". File này sẽ chứa một số thông tin như tên dự án, các thư mục tài nguyên liên quan, các libraries sử dụng…Khi bắt đầu biên dịch, trình **Maven** sẽ đọc file này và tự động tải các libraries cần thiết từ repositories của nó thông qua mạng hoặc nơi người dùng đã định nghĩa.

Cấu trúc 1 project Maven

* Các thư viện này sau khi được lấy về từ trên mạng sẽ được lưu trữ tại máy tính để sử dụng cho những lần sau
* Các thư viện download nằm ở thư mục C:\Users{username}.m2\repository.
* Về nguyên tắc, khi chạy, Maven sẽ tìm các thư viện ở kho chứa local trong máy tính trước, nếu không có thì sẽ lấy từ trên mạng.
* Chỉ có lần đầu tiên build chương trình hoặc khi dự án cần sử dụng thư viện mới thì máy tính phải nối kết nối với mạng Internet, còn những lần tiếp sau đó thì có thể chạy offline.
* Tự động download các thư viện và các plug-ins từ một hay nhiều nơi từ trên mạng về ⇒ gọi là Maven remote repositories ⇒ Mặc định thì Maven sẽ lấy thư viện từ remote repository của Maven tại địa chỉ <http://repo1.maven.org/maven2>
* Maven có rất nhiều plugin và mỗi plugin sẽ được sử dụng vào trong các mục đích khác nhau. Plugin sẽ là một tập hợp của các Goal, mỗi goal sẽ quy định một công việc cụ thể sẽ được thực hiện bởi plugin.
* ví dụ: plugin archetype với goal là generate để sinh ra project với cấu trúc định nghĩa sẵn bởi Maven (cmd => gõ :mvn archetype:generate, đây là cách sử dụng một Maven plugin có tên là archetype)
* Một số plugin phổ biến có thể kể tên ra ở đây như là clean, install, surefire, jar, war, pmd, ant, assembly, dependency…

## Lợi ích

Maven giúp cho developer hiểu được toàn bộ hệ thống trong thời gian ngắn nhất, để được như vậy, Maven phải thành công trong các lĩnh vực sau:

* **Giúp việc xây dựng hệ thống một cách dễ dàng hơn:** Khi sử dụng Maven, người dùng không cần phải biết bộ máy phía dưới, Maven che chắn rất tốt các chi tiết cụ thể của hệ thống rồi.
* **Cung cấp cho người dùng một hệ thống xây dựng thống nhất:** Maven cung cấp rất nhiều thông tin hữu ích mà một phần là được trích từ file POM của java project mà bạn đã build.
* **Cung cấp thông tin dự án một cách chuyên nghiệp**
* **Đưa ra hướng dẫn cụ thể nhất cho việc phát triển thực nghiệm:** Maven giữ source code thử nghiệm của người dùng trong một nơi riêng. Maven cũng nhằm mục đích hỗ trợ trong công viện quản lý dự án, phát hiện và theo dõi vấn đề. Hướng dẫn về cách bố trí cây thư mục của dự án để người dùng tìm hiểu có thể dễ dàng hơn.
* **Cho phép tự động update những tính năng mới:** Maven cung cấp cho người dùng một cách đơn giản để cập nhật những bản update mới mà Maven đã làm.

## Tại sao phải sử dụng maven

Bạn hãy thử tượng nếu dự án của chúng ta sử dụng rất nhiều thư viện thứ 3 : **struts**, **hibernate**, spring, .... Việc import thư viện (hay gọi là các dependency trong định nghĩa của Maven) là rất vất vả, chưa kể đến việc version của các thư viện có thể config với nhau, việc import đầy đủ cácthư viện là cả một vấn đề.

VD như bạn muốn sử dụng struts, thì điều bắt buộc là chúng ta phải import cả thư viện servlet.

*Struts là gì?*

Struts là một framework phục vụ việc phát triển các ứng dụng Web trên Java. Sử dụng mẫu thiết kế Model-View-Controller (MVC)

*hibernate là gì?*

Hibernate là 1 framework giúp người lập trình không cần phụ thuộc quá nhiều vào cơ sở dữ liệu như giúp đóng băng dữ liệu, tự động chuyển các bảng từ csdl về thành các class cho người lập trình thao tác. Có 2 dạng hibernate sql và hibernate truyền thống

Hibernate kế thừa từ JDBC, nó cũng có thể chạy các lệnh SQL…

Nó có hỗ trợ một mô hình kết nối cơ sở dữ liệu mà hiện nay đa số mọi người thường áp dụng đó là mô hình ORM(Object Relational Mapping). Mô hình ORM, là một mô hình theo kiểu các đối tượng được mapping tương ứng với các bảng

⇒ code của bạn đỡ rườm rà dài dòng vs SQL

1. **Áp dụng của Apache Maven:**

* Thường cho ứng dụng web

Ứng dụng doanh nghiệp.

# Nội dung chi tiết:

## file POM cho Maven

* File pom.xml là nơi khai báo tất cả những gì liên quan đến dự án được cấu hình qua maven, như khai báo các dependency (đây là một khai báo khá quan trọng), version của dự án, tên dự án…
* Các khai báo được thể hiện dưới dạng các thẻ XML bên trong thẻ <project>
* Một số các thẻ cơ bản mà ta có thể thấy trong bất cứ file pom.xml nào
* **<groupId>, <artifactId>, <version>:** bộ ba thông tin để nhận diện ra một thành phần như project, dependency, plugin…
* <project>: Đây là phần tử trên cùng trong tất cả các file pom.xml
* <packaging>: Định dạng cho việc đóng gói project có thể là jar (Java Archive File ), war (Java Web Archive ), pom (Maven Build File), ear. Thường mặc định là file jar
* jar: thường dùng đóng gói các project app thông thường
* war: thường dùng đóng goi cho các project web
* pom: thường sử dụng để kết hợp các module con lại với nhau, file đóng gói này áp dụng cho việc đóng gói các project lớn
* <properties>: khai báo các thuộc tính có thể sử dụng ở các nơi trong file pom.xml
* <Version >: Phần tử này chỉ ra phiên bản của mô hình đối tượng mà file POM sử dụng. Phiên bản của bản thân mô hình đối tượng rất ít thay đổi nhưng nó là bắt buộc để đảm bảo tính ổn định khi các nhà phát triển Maven nghĩ rằng nó cần thiết để thay đổi mô hình.Phần tử này chỉ ra phiên bản của artifact được tạo ra bởi dự án. Bạn sẽ thường xuyên nhìn thấy chữ SNAPSHOT trong phiên bản, chỉ ra rằng dự án đang ở trong giai đoạn phát triển.

VD: version trong một dependency được khai báo sử dụng

<dependency>

<groupid>org.hibernate</groupid>

<artifactid>hibernate-core</artifactid>

<version>5.0.0.CR2</version>

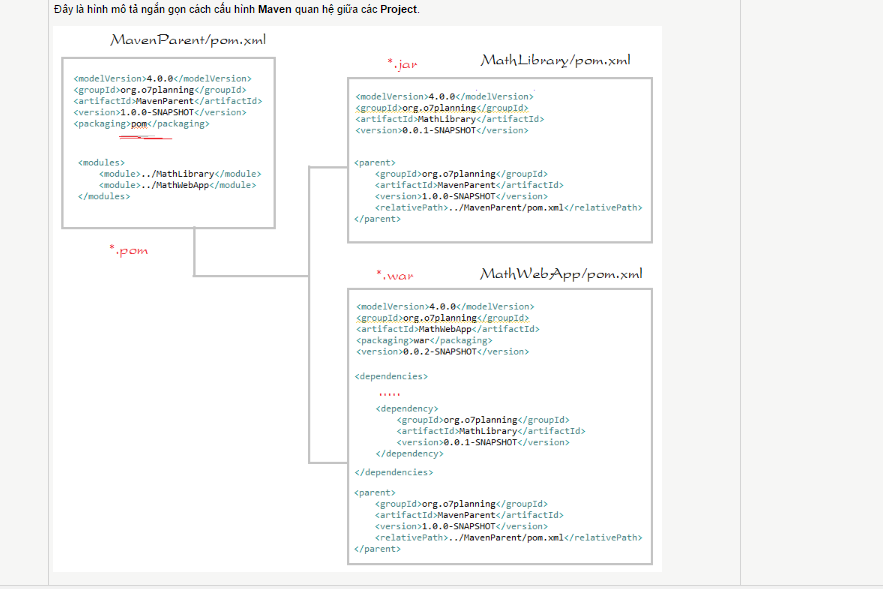
</dependency>

* <groupId >: Phần tử này chỉ ra định danh duy nhất của nhóm hoặc tổ chức tạo ra dự án. groupId là một trong định danh chính của một dự án được đặt tên dựa theo domain đầy đủ của tổ chức của bạn. Ví dụ ở đây: org.sonatype.mavenbook.simple
* <artifactId>: tên của project maven.
* <name>: Phần tử này chỉ ra tên của dự án. Điều này được sử dụng trong tài liệu được tạo ra bởi maven.
* <url> :Phần tử này chỉ ra trang của dự án có thể được tìm thấy. Điều này được sử dụng trong tài liệu được tạo ra bởi maven.
* <description> : Phần tử này cung cấp phần mô tả ngắn gọn về dự án. Điều này được sử dụng trong tài liệu được tạo ra bởi maven.
* <parent>: sử dụng khái niệm inheritance cho việc xây dựng cấu trúc file POM, các project con (module con) sử dụng tag <parent> để khai báo việc sử dụng lại các thư viện hay plugin từ project cha mà không cần phải import các thư viện hay plugin đã có từ project cha

Vậy module con tìm kiến các thư viện của project cha như thế nào?, tìm ở đâu?

Các project con sẽ tìm kiếm project cha của nó trong cùng groupID với nó, nếu không thấy nó sẽ tiếp tục tìm kiếm trong local repository (vì nó xem project cha là một thư viện)





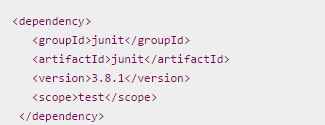
Ví dụ 1: <artifactId>-<version>.<extension> (ví dụ simple-1.0-SNAPSHOT.jar trong tình huống dự án này). Trong thuật ngữ của Maven, artifact là kết quả của quá trình build trong maven, nói chung là một fie jar, hay war trong dự án web hoặc một file thực thi khác. Một artifact trong maven được xác định bởi một hệ tọa độ gồm groupId, artifactId và version. Maven sử dụng groupId, artifactId và version để xác định sự phụ thuộc (thường là các file jar khác ) cần để build và chạy code của bạn.)

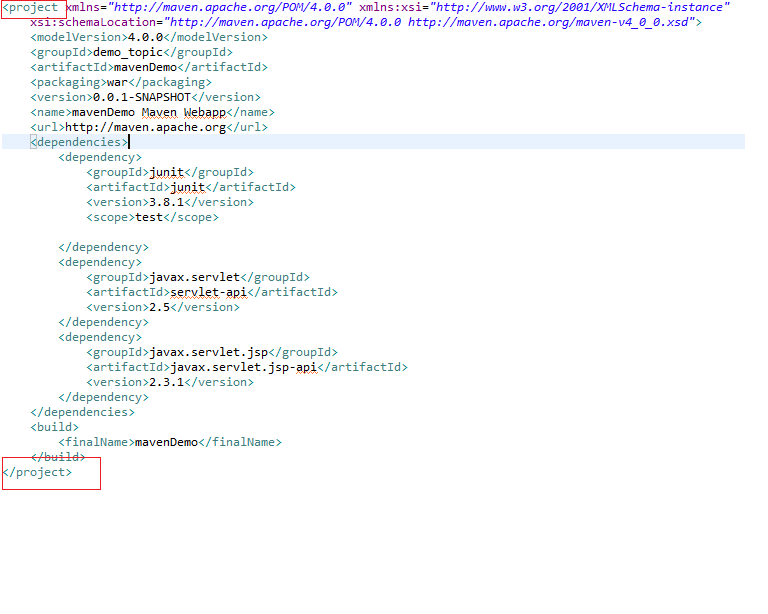
Ví dụ 2: <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

⇒ thì ở một nào đó trong pom.xml, ta có thể sử dụng giá trị UTF-8 bằng cách gọi ${project.build.sourceEncoding}

* <dependency>: Bên trong sẽ khai báo thành phần mà project sẽ phụ thuộc vào

Ví dụ 3: project có sử dụng JUnit để thực hiện unit test, thì việc project phụ thuộc vào junit sẽ được khai báo như sau





<dependencies>: quản lý các <dependency>, việc thêm bớt hay upgrade thư viện chỉ đơn giản là config vài dòng lệnh => nơi khai báo các thư viện sử dụng trong dự án => Maven sẽ tự động dowload thư viện đó về thư mục ~/.m2 trên máy local và buildpath vào project

Maven có thể quản lý cả loại internal và external dependencies

VD: 1 external dependency có thể là một thư viện như Plexus, Spring framework

<plugins> định nghĩa những Plugin sử dụng trong project trong dự án

## Build Life Cycle

[**http://maven.apache.org/guides/introduction/introduction-to-the-lifecycle.html**](http://maven.apache.org/guides/introduction/introduction-to-the-lifecycle.html)

* Goals trong maven chỉ ra mục đích dể chạy các tasks ở các phase các nhau. ví dụ mục đích chạy maven là compile, package, hay install,

⇒ Bạn có thể đặt nhiều goals theo thứ tự cách nhau bằng dấu khoảng cách.

### Maven compile (mvn compile)

Lần đầu tiên thực thi lệnh này, maven sẽ download toàn bộ các plugin và các dependencies liên quan. Nếu chúng ta thực thi lại lệnh này, maven sẽ không download lại những gì nó đã có từ lần trước, vậy nên command sẽ được thực thi nhanh hơn. Các class được compile sẽ được đưa vào ${basedir}/target/classes. Với maven vẫn còn vài lỗi phát sinh, nếu trong quá trình download các gói thư viện được định nghĩa trong pom.xml nhưng phát sinh lỗi do sự cố mạng hay một lỗi nào đó mà gói không được tải trọn vẹn, thì ở lần build sau khi maven tìm thấy gói trong repo trên máy lúc ấy maven sẽ dùng gói đó hỗ trợ build và việc build sẽ faild, trong trường hợp này cần phải tìm đến gói đó và xóa nó đi, thực hiện build lại nếu trong quá trình download lại gói đó không phát sinh lỗi thì build sẽ thành công.

### Maven package (mvn package)

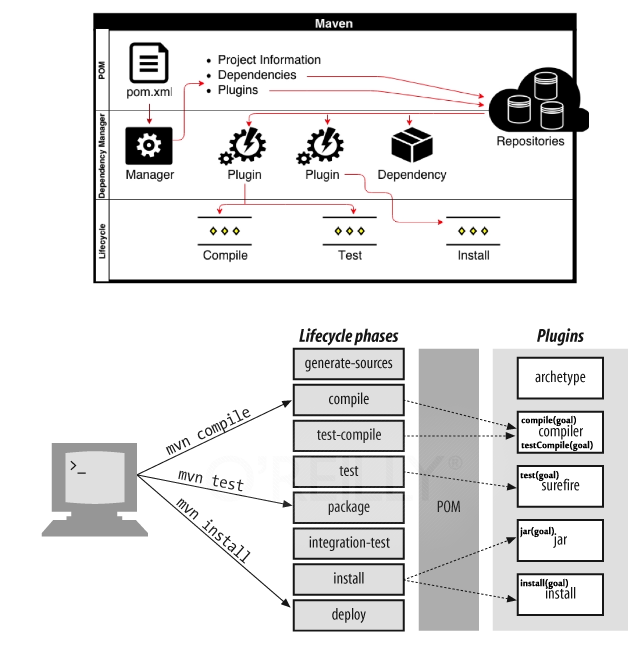
Được dùng khi muốn export project ra 1 gói jar để cho người khác sử dụng. Được định nghĩa trong pom.xml, nếu được định nghĩa jar thì khi export ra gói .jar, nếu không mặc định là .war (web app)

### Maven install (mvn install)

Với command “mvn package” gói jar chỉ được export ra ${basedir}/target, nếu muốn gói jar ấy được đưa lên repo và project có thể dependency tới thì cần sử dụng lệnh “mvn install”.

### Maven deploy (mvn deploy)

Với command “mvn deploy”: dùng để xây dựng môi trường, copy các “final package” để điều khiển cho các repository trong project cho việc chia sẻ sau này với các nhà phát triển hay các project khác.



Maven định nghĩa 3 loại lifecycles: [**https://maven.apache.org/ref/3.3.9/maven-core/lifecycles.html#clean\_Lifecycle**](https://maven.apache.org/ref/3.3.9/maven-core/lifecycles.html#clean_Lifecycle)

* [default Lifecycle](https://maven.apache.org/ref/3.3.9/maven-core/lifecycles.html#default_Lifecycle)
* [clean Lifecycle](https://maven.apache.org/ref/3.3.9/maven-core/lifecycles.html#clean_Lifecycle)
* [site Lifecycle](https://maven.apache.org/ref/3.3.9/maven-core/lifecycles.html#site_Lifecycle)

1. [**default Lifecycle**](https://maven.apache.org/ref/3.3.9/maven-core/lifecycles.html#default_Lifecycle)

<phases>

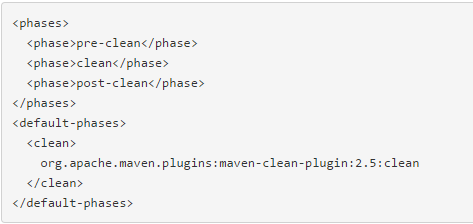
<phase>validate</phase>  
<phase>initialize</phase>

<phase>compile</phase>

<phase>test</phase>

<phase>verify</phase>  
<phase>install</phase>  
<phase>deploy</phase>  
</phases>

1. [**clean Lifecycle**](https://maven.apache.org/ref/3.3.9/maven-core/lifecycles.html#clean_Lifecycle)



|  |  |
| --- | --- |
| pre-clean | thực hiện các quy trình cần thiết trước khi thực hiện giai đoạn clean project |
| Clean | xóa bỏ hết tất cả các tập tin được tạo ra trước đó bằng lệnh build |
| post-clean | thực hiện các tiến trình cần thiết để kết thúc giai đoạn clean |

1. [**site Lifecycle**](https://maven.apache.org/ref/3.3.9/maven-core/lifecycles.html#site_Lifecycle)



|  |  |
| --- | --- |
| pre-site | thực hiện tất cả các giai đoạn cần thiết trước khi thực hiện quá trình site project |
| Site | tạo ra các document cho project |
| post-site | thực hiện tất cả các giai đoạn cần thiết để kết thúc quá trình site và chuẩn bị cho quy trình deploy |
| site-deploy | deploy các tài liệu của quy trình site đến web server theo chỉ định |

## Build Profile

* Xây dựng lại project theo ý của người dùng mà không phụ thuộc vào cấu trúc xây dựng project mà Maven quy định sẵn
* Vậy xây dựng một profile thì định nghĩa lại những thành phần nào?
* Với project: thì định nghĩa lại file POM
* Với thư viện local repository: định nghĩa lại đường dẫn
* Mô tả một profile mới



<profiles>: chứa nhiều <profile> → ghi đè sự thiết lập của file POM, các <profile> thường được đặt ở cuối cùng của file POM

<id>: đây là tên của profile được dùng để gọi khi chạy dòng lệnh cho việc build 1 profile

Một phần tử profile có thể chứa nhiều phần tử khác như <modules>, <dependencies>... cấu trúc như là 1 file POM

Để install 1 profile ta dùng lệnh: **mvn install -P<profile\_id> -X**

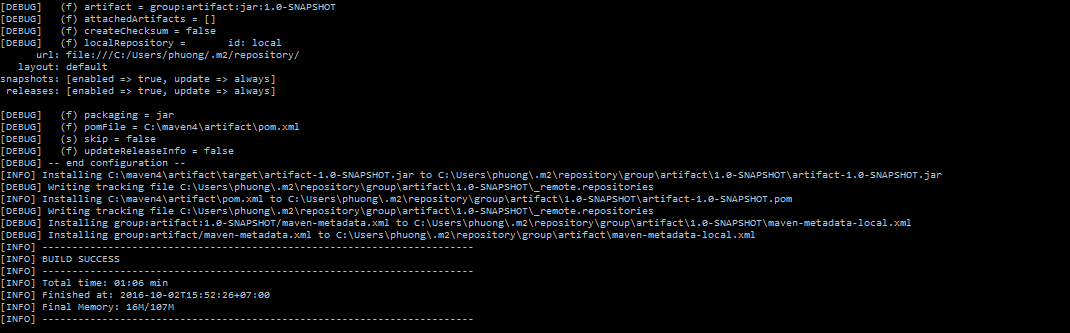
Trong đó -X: là 1 cách để gỡ lỗi trong quá trình install

**VD:**

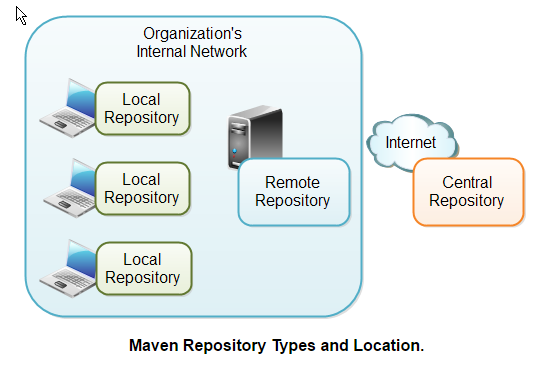
Gọi lệnh **:** 

để build profile có id: production

Sau khi install thành công 1 profile



## Repositories

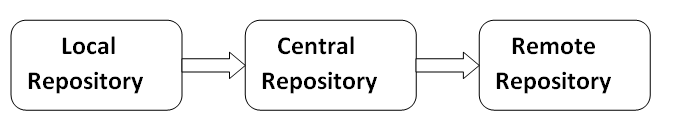


* Maven quản lý các thư viện, plugin và lưu chúng ở trong repository.
* Khi build một project, các thành phần mà project phụ thuộc vào sẽ được lấy ra từ local repository trên máy đang thực hiện quá trình build.
* Nếu như thành phần (thư viện) không được tìm thấy trên local repository thì Maven sẽ tự động download thành phần này từ remote repository về local repository. Vị trí của local repository mặc định sẽ là ${user.home}/.m2/repository, ta hoàn toàn có thể thay đổi vị trí này bằng cách cấu hình trong file /conf/settings.xml như sau:

<localRepository>E:/DEVELOPMENT/Env/apache-maven-repo</localRepository>

* Maven sử dụng một nơi lưu trữ từ xa(ví dụ <http://repo1.maven.org/> ) để lưu trữ plugin và các thư viện công cộng dùng chung. Và nó sẽ download khi dự án bạn cần về một nơi lưu trữ cục bộ (ví dụ C:\Users\Admin\.m2\repository) và nếu local repository đã có rồi thì maven không cần download nữa.
* Tự động cập nhật, mở rộng dễ dàng, đóng gói các thư viện mà project sử dụng. Maven có một repository lớn chứa toàn bộ các thư viện cần thiết (http://mvnrepository.com), nó không chỉ lưu trữ gói mà còn lưu cả version, các version này được cập nhật liên tực bởi người phát triển ra nó. Chính vì thế khi cần thay đổi version thì rất nhanh chóng.

Có 3 loại Repository



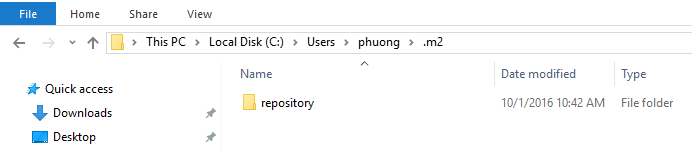
***Nếu như các dependency được khai báo trong file POM khi được thực hiện compile mà không tìm thấy trong các repository này thì sẽ có lỗi trong quá trình compile***

### Maven Local Repository

Loại Maven Local Repository này nằm trên máy tính của bạn, nó được tạo ra ngay khi bạn chạy bất cứ một lệnh maven nào như là build, package, install…

Vị trí của nó trên máy tính của bạn có giá trị đường dẫn mặc định: %USER\_HOME%/.m2

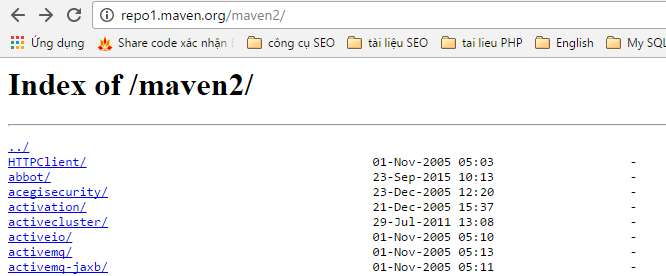
VD:



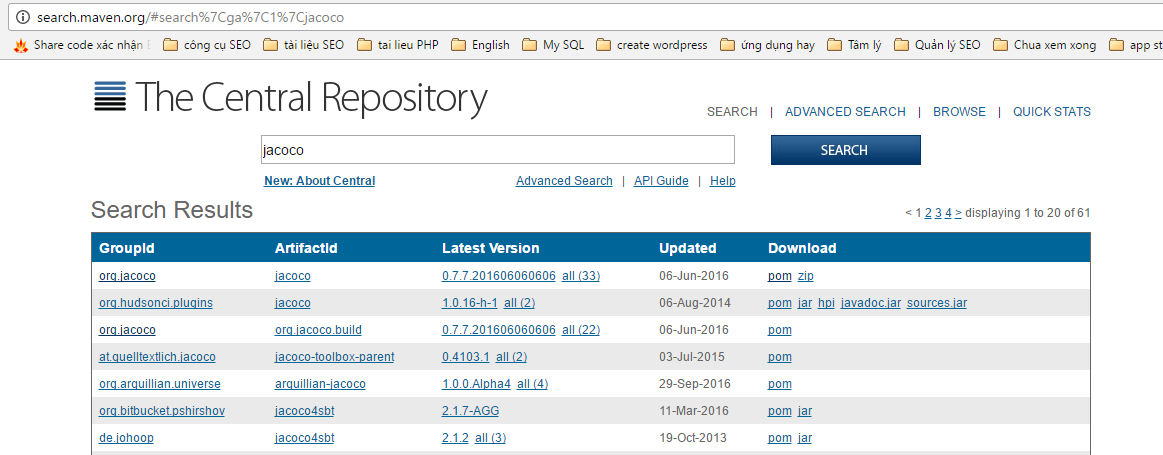
### Maven Central Repository

Maven central repository là một thư viện chung nằm trên web, nó được tạo ra bởi cộng đồng Maven Apache dùng cho chứa các thư viện

Đường dẫn của central repository là: <http://repo1.maven.org/maven2/>



Central repository chứa các thư viện thông dụng, các thư viện này bạn cũng có thể tìm trong đường dẫn <http://search.maven.org/> (đường dẫn này hỗ trợ tìm kiếm các thư viện dễ dàng hơn)



### Maven Remote Repository

Hầu hết các thư viện sẽ mất đi khi import trực tiếp bằng tay trên một project vì vậy khi có Maven bạn có thể khai báo thư viện mà bạn cần dùng trong file POM, file POM này sẽ remote repository đến thư viện chung <http://repo1.maven.org/maven2/> để download gói thư viện cần thiết cho project của bạn về lưu trong Local Repository.

VD:



## Demo:

## Creating Project

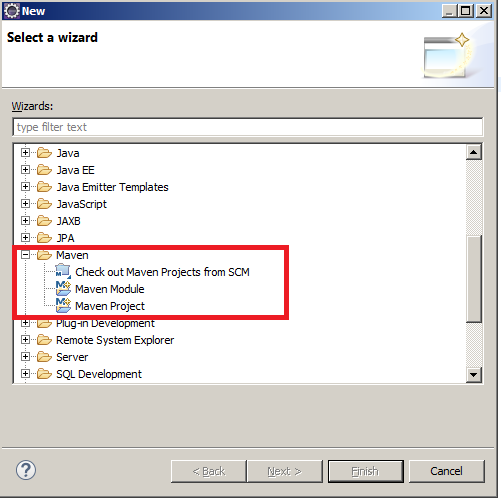
Có nhiều cách để có thể sử dụng Maven và tạo project

* **Cách 1: Cài đặt Maven như là một Plugin nhúng vào Eclipse.**
* **Cách 2: Cài đặt Maven độc lập trên Window để tạo ra project và khai báo để Eclipse có thể sử dụng project đó bằng cách import project vào eclipse, eclipse sẽ nhận dạng project thông qua 2 file là .classpath và .project**

**Dưới đây là cách tạo project từ Eclipse**

Trước hết kiểm tra xem Eclipse của bạn đã cài đặt Maven chưa.

* ***File/ Other/ Maven ...***

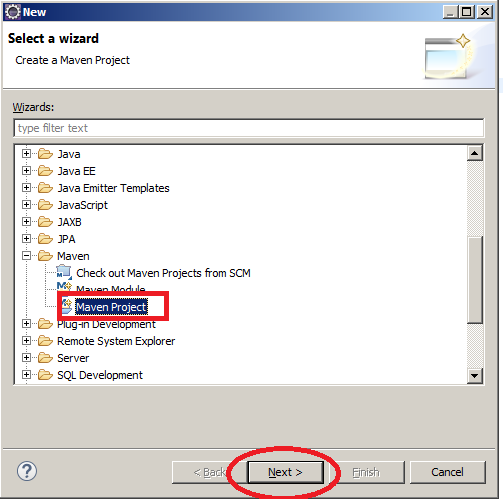


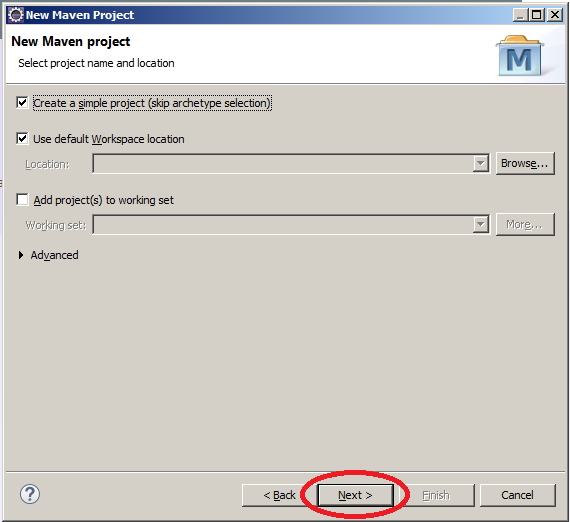
Nếu chưa cài đặt bạn có thể xem hướng dẫn tại đây:

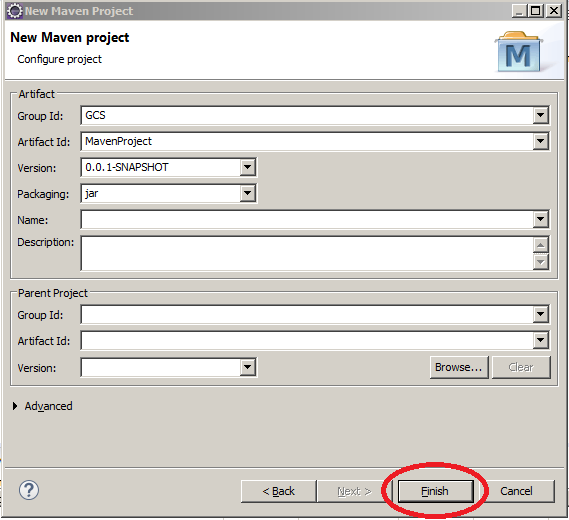
* <http://o7planning.org/en/10101/install-maven-into-eclipse>

### 1. Tạo Maven Project

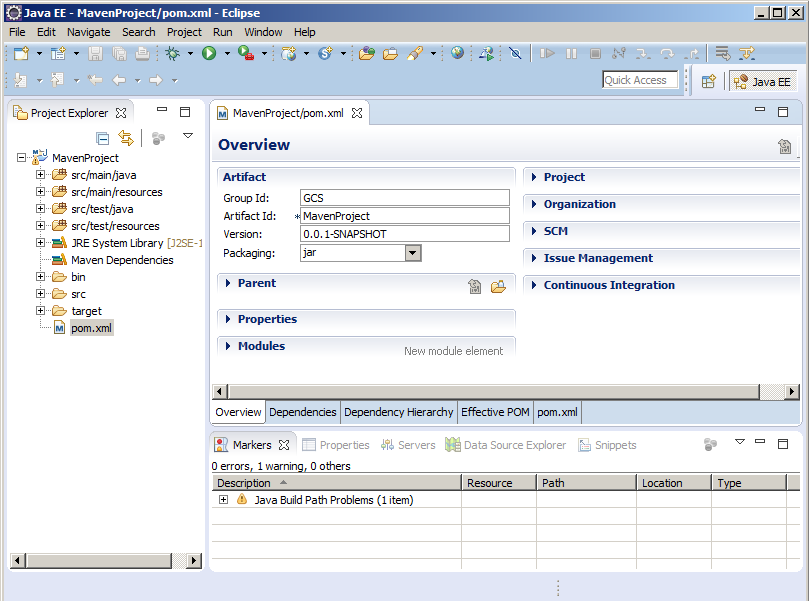
Trên Eclipse chọn ***"New/Other"***



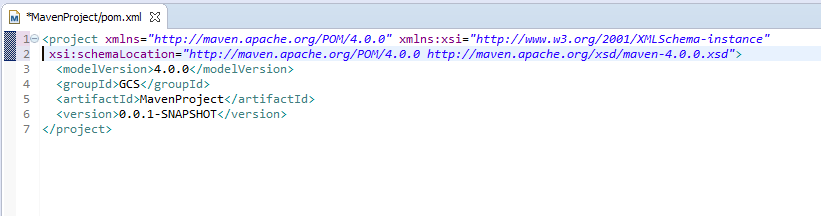




Đây là hình ảnh Project được tạo ra.

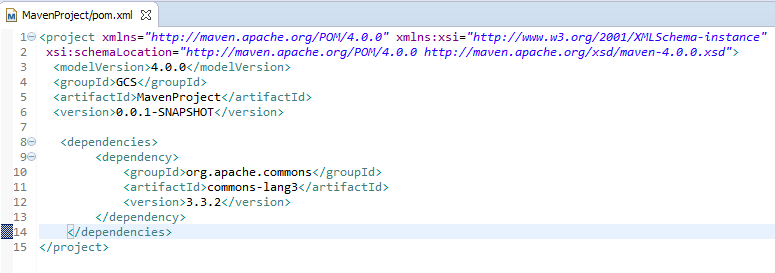


* ***pom.xml***

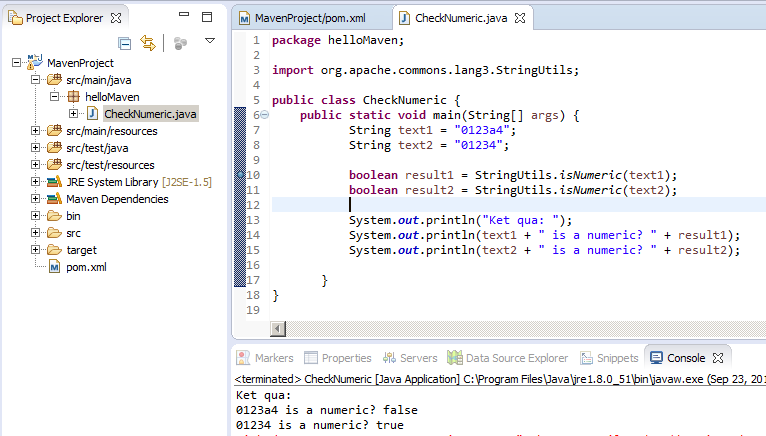


### 2. Cấu hình Maven

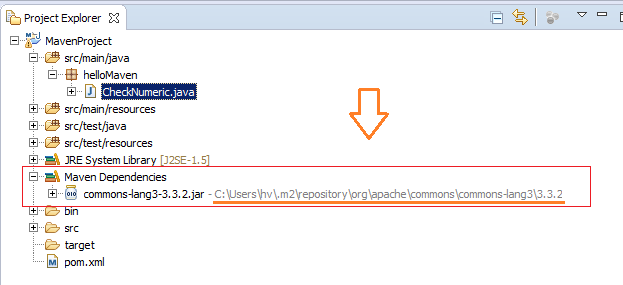
Mở file ***pom.xml*** để cấu hình các thư viện sẽ sử dụng:



Tạo class **CheckNumeric.java**



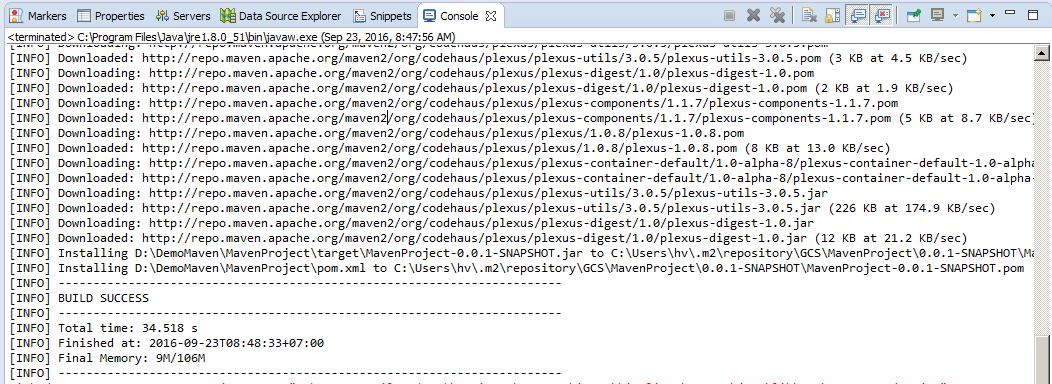
Bạn có thể thấy rõ ràng project của bạn sử dụng các thư viện nào. Vị trí của nó ở đâu trên ổ cứng.

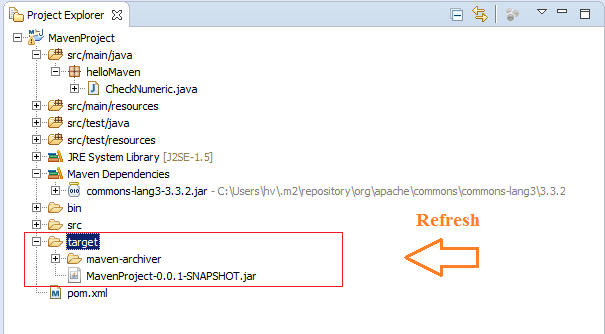


### 3. Cài đặt Project bằng Maven

Bản chất ở đây chính là dịch và đóng gói Project của bạn bằng Maven:

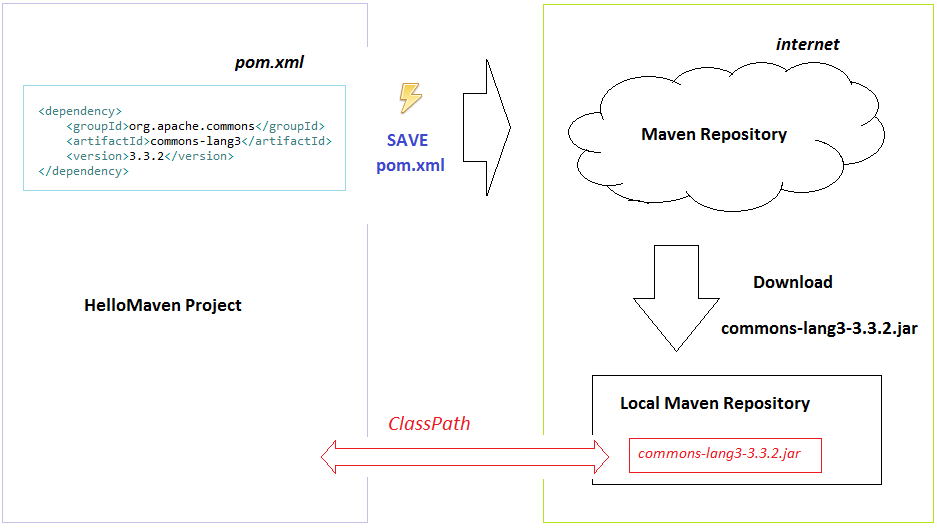
* Click chuột phải MavenProject/ Run As / Maven install





### 4. Giải thích nguyên tắc hoạt động của Maven

Ở trên bạn đã tạo project và chạy hoàn hảo. Project sử dụng tới class StringUtils là một class của Apache, vốn không nằm trong thư viện chuẩn JDK của Java. Theo truyền thống bạn phải copy thư viện vào Project và khai báo ClassPath. Tuy nhiên, trong hướng dẫn hoàn toàn không phải copy thư viện và khai báo classpath như cách truyền thống. Việc quản lý thư viện đã do Maven quản lý. Bây giờ chúng ta sẽ giải thích cách mà Maven hoạt động.



Hình minh họa ở trên chỉ ra cách hoạt động của Maven.

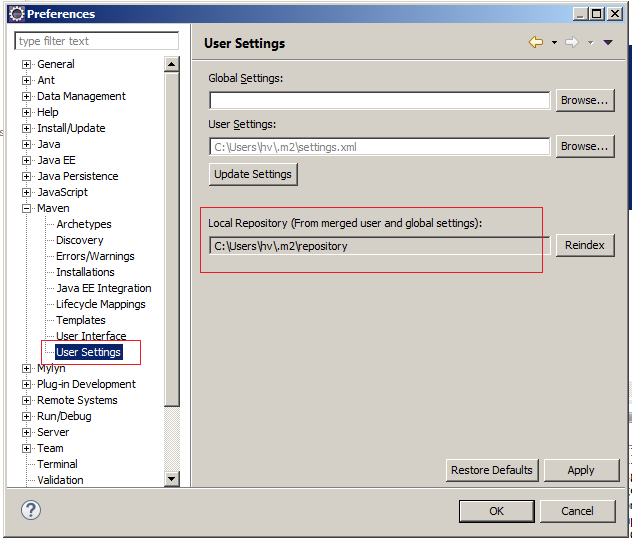
1. Bạn khai báo trên ***pom.xml*** rằng project của bạn phụ thuộc vào thư viện ***common-lang3*** phiên bản 3.3.2.
2. Ngay sau khi bạn SAVE file ***pom.xml*** Maven sẽ kiểm tra xem thư viện này đã có tại kho chứa địa phương (Local Repository) trên máy tính của bạn chưa. Nếu chưa có nó sẽ download về từ kho chứa (Repository) thư viện từ trên Internet xuống.
3. Cuối cùng là Maven sẽ tự động khai báo ClassPath cho Project tới vị trí file jar được download về.

Như vậy mọi bạn chỉ cần khai báo thư viện muốn sử dụng tại pom.xml. Việc quản lý thư viện đã do Maven lo liệu.

### 5 Xem Local repository

Câu hỏi của bạn là kho chứa thư viện địa phương (Local Repository) nằm ở đâu?

* **Windows/Preferences**



### 6. Xem Maven Repository trên Internet

Câu hỏi đặt ra là các thông tin groupId, artifactId, version tra cứu ở đâu.

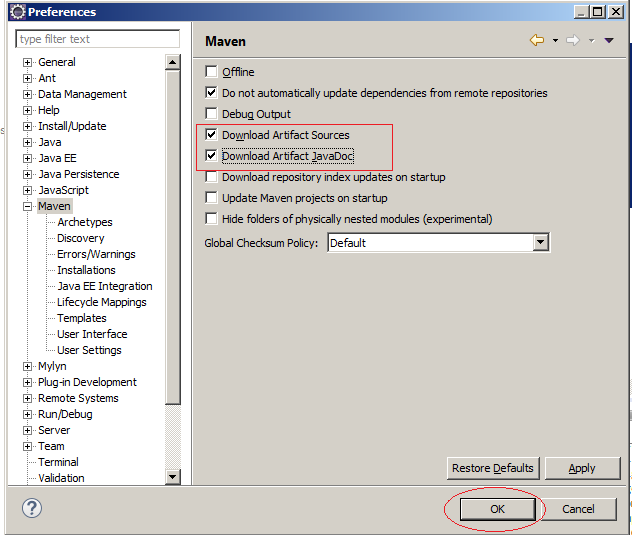
Bạn có thể vào một trong các trang:

* [http://search.maven.org](http://search.maven.org/)
* [http://mvnrepository.com](http://mvnrepository.com/)

### 7. Cấu hình Maven download source & javadoc

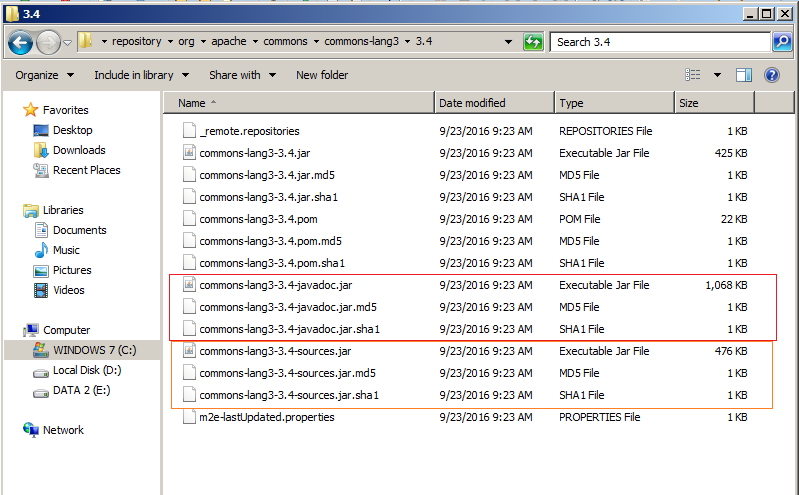
Thông thường Maven chỉ download file binary về Local Repository. Để Maven download cả source và javadoc bạn cần cấu hình trên **Eclipse**.

* **Windows/Preferences**



Thay đổi gì đó trên file ***pom.xml*** và save lại (hoặc build project), Maven sẽ download source & javadoc.

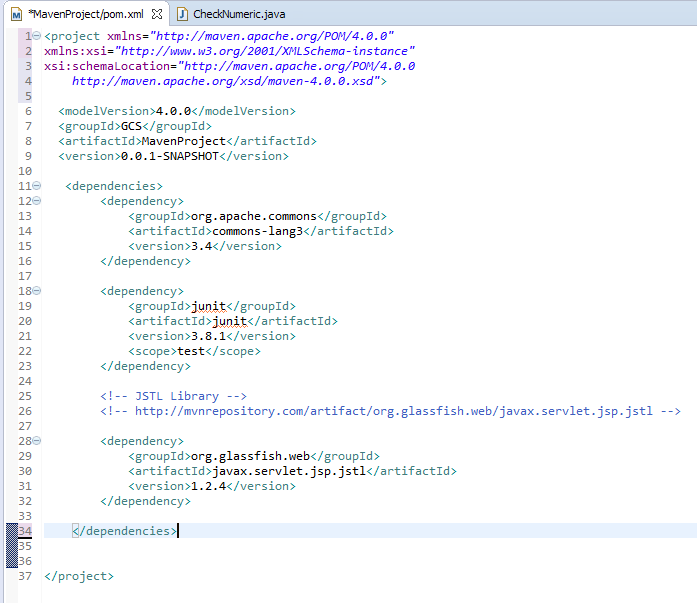
Kết quả bạn nhìn thấy trên **Local Repository**:



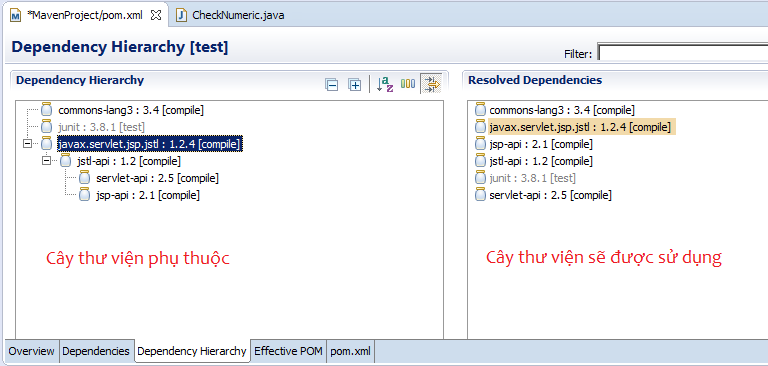
### 8. Quản lý phụ thuộc trong Maven

Trong tài liệu này tôi hướng dẫn bạn làm thế nào tránh trùng lặp thư viện trong Maven.

Để đơn giản bạn hãy xem nội dung của một file **pom.xml** sau:



File **pom.xml** trên bạn đã khai báo sử dụng thư viện **javax.servlet.jsp.jstl**, thư viện này phụ thuộc vào **jstl-api**. Và **jstl-api** lại phụ thuộc vào 2 thư viện **servlet-api** và **jsp-api**. Sự phụ thuộc đó giống như hình minh họa dưới đây:



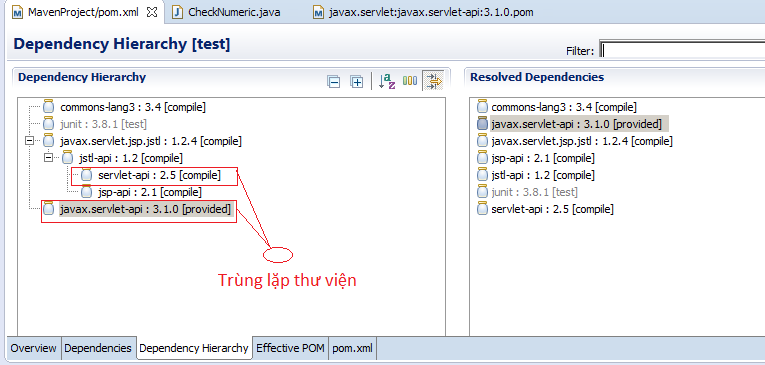
# Loại bỏ sự trùng lặp thư viện

Các thư viện bị trùng lặp có thể gây ra lỗi trong quá trình chạy của ứng dụng. Vì vậy bạn cần phải loại bỏ thư viện nào bị trùng lặp. Thông thường, ta sẽ loại bỏ phiên bản cũ hơn.

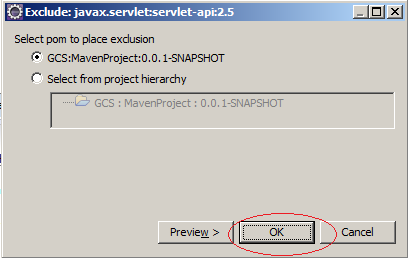
Ví dụ:

Thêm đoạn code này vào file pom.xml

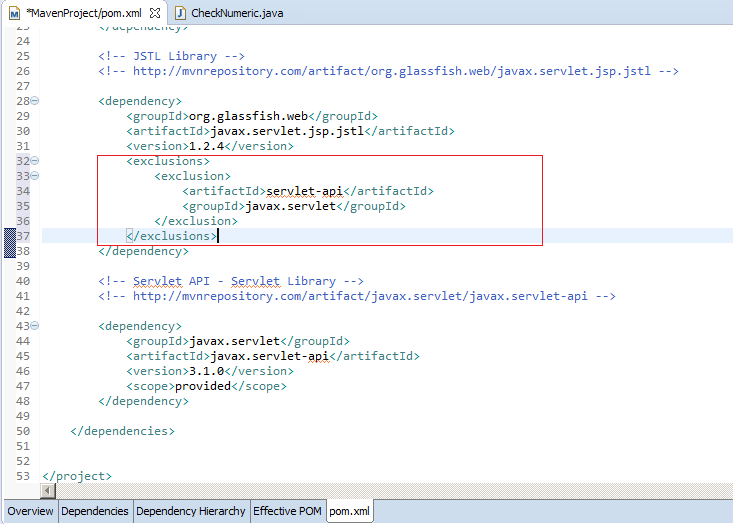




Nhấn phải chuột vào thư viện phụ thuộc cần loại trừ, chọn Exclude Maven Artifact…



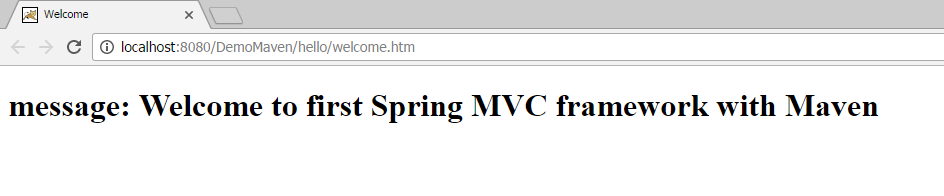
Eclipse sẽ tự động thêm đoạn code **<exclusions>** vào **pom.xml**:



## Build and Test Project

Ví dụ: Hướng dẫn tạo một spring mvc maven project.

Xuất ra nội dung:



* Eclipse
* Bước 1:

Trong Eclipse có hỗ trợ mẫu chuẩn của Spring MVC framework with Maven, ta chỉ cần cài đặt và sử dụng nó.

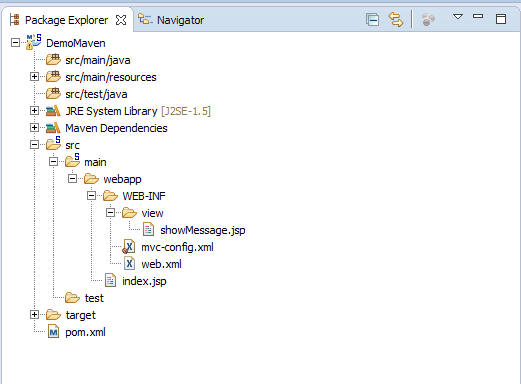
Cài đặt: Spring MVC framework with Maven

* Help -> Eclipse Marketplace… -> Tìm và cài đặt Spring IDE 3.8.1.RELEASE….

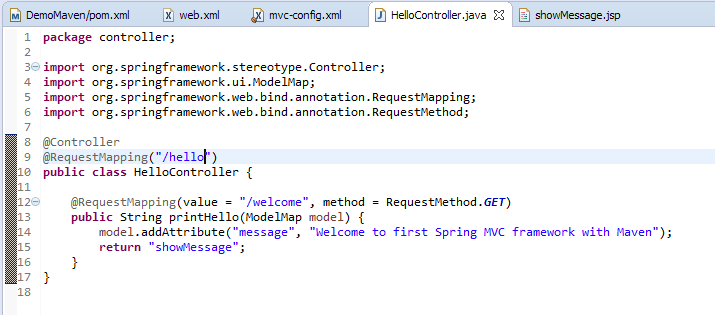
Sau khi cài đặt xong, ta tạo 1 project

* File -> New -> Other… -> Spring -> Spring Legacy Project -> Simple Projects -> chọn Simple Spring Web Maven và nhập tên project (trong khung Project name)

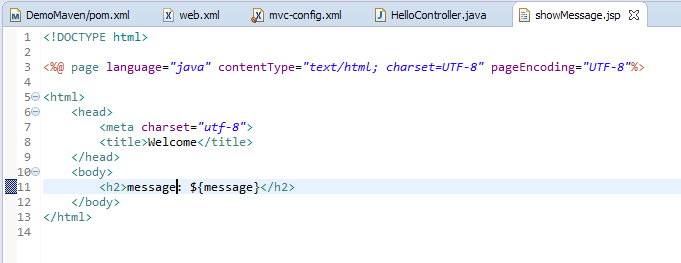
Project DemoMaven sẽ có cấu trúc như thế này:



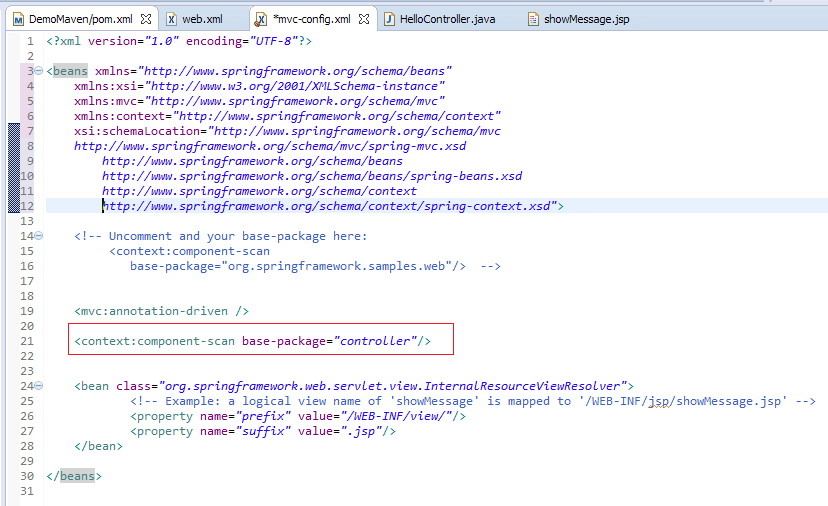
* Bước 2: Trong thư mục **src/main/java**, tạo 1 package controller. Trong package controller tạo thêm 1 class HelloController
* Bước 3: Trong class HelloController tạo phương thức để truyền đi một thông điệp (ví dụ: thông điệp xin chào), thông điệp này sẽ được gửi đến và hiển thị lên trang showMessage.jsp
* HelloController.class



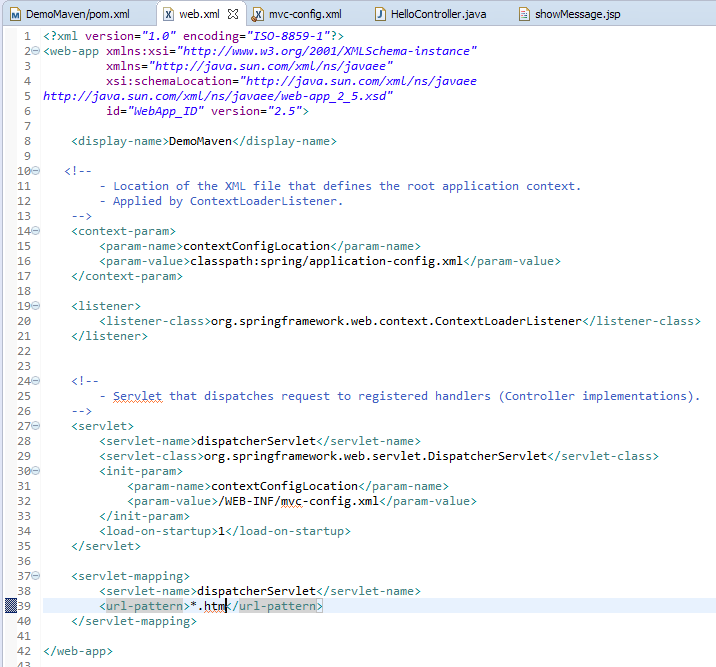
* showMessage.jsp



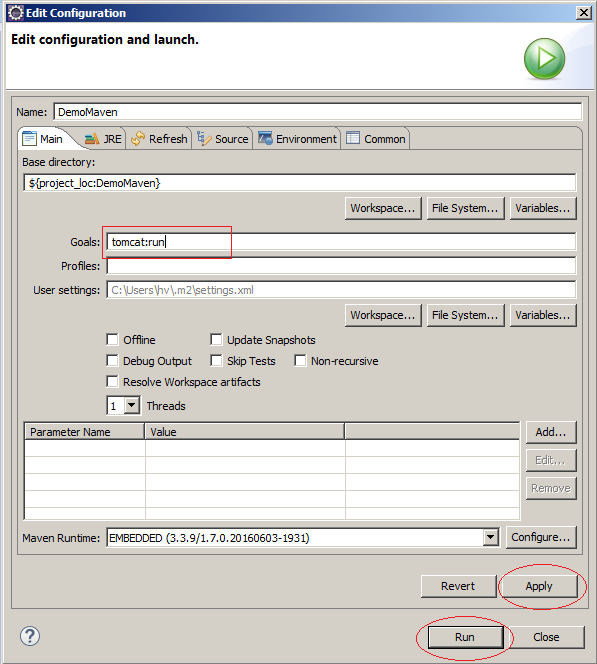
* Bước 4: Chỉnh các file cấu hình cho ứng dụng tại WEB-INF
* File mvc-config.xml



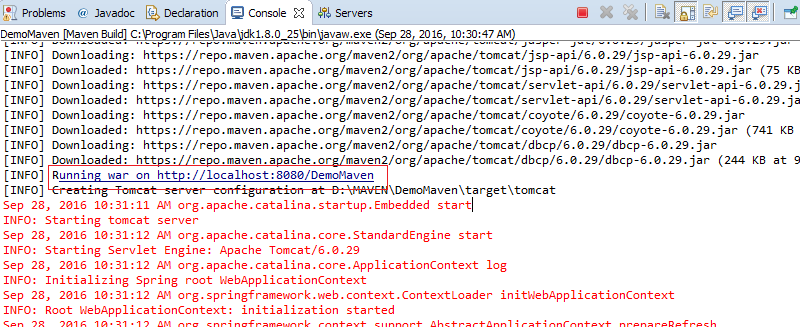
* File web.xml



* Bước 5: Chạy ứng dụng
* Click phải DemoMaven -> Run As -> Maven build...

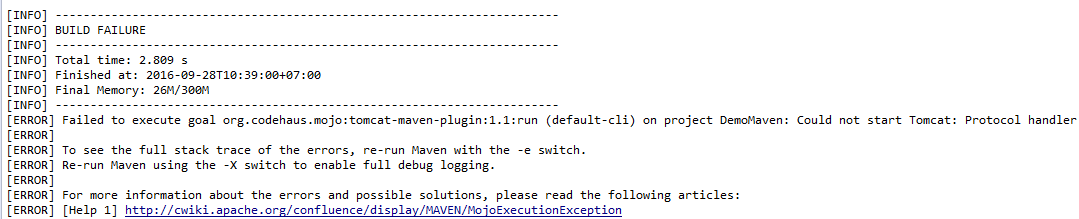


* Goals: tomcat:run : khi ta click Apply -> Run thì Maven sẽ tự động tải server về để chạy ứng dụng



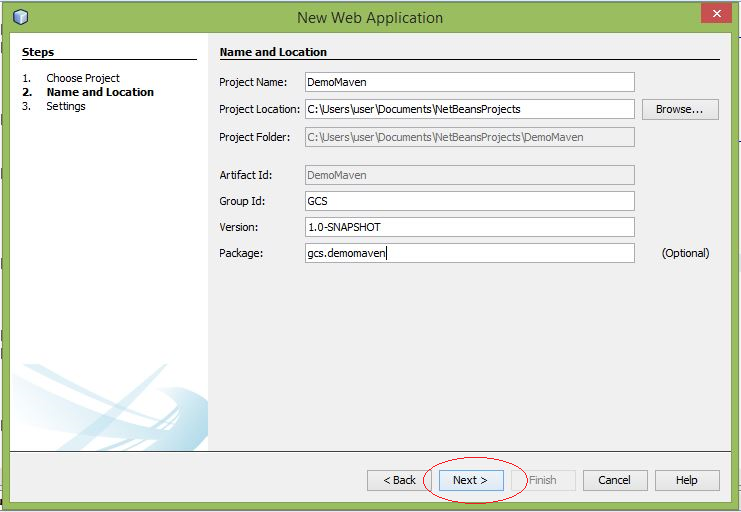
Chú ý: Khi chạy lần 2, chúng ta sẽ bị gặp lỗi

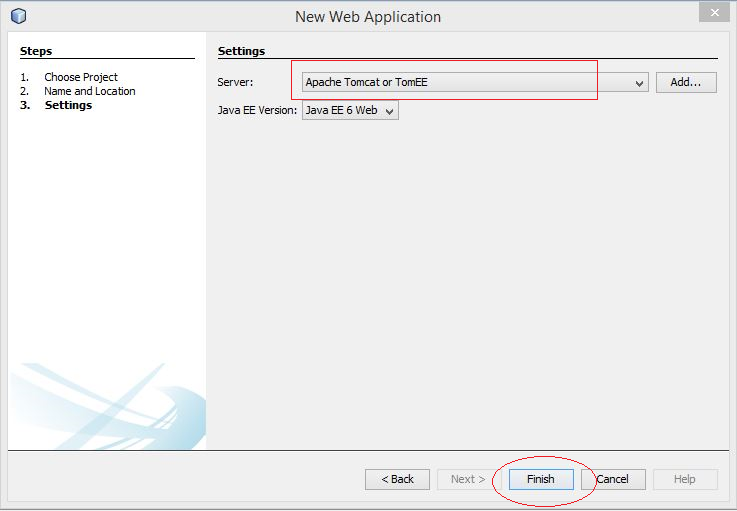




Để khắc phục lỗi này, mỗi lần ta chạy ứng dụng thì ta phải xóa tiến trình javaw.exe trước

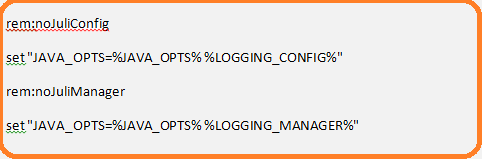
* Mở Task Manager -> Processes -> chọn javaw.exe -> End Process
* NetBean
* Bước 1:
* File/ New project/ Maven/ Web Application



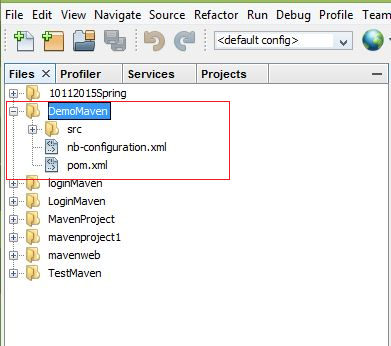


Ở đây, mình xài server: Tomcat 7. Bạn cũng có thể xài server glassfish 4 có sẵn trong netbean.

Hướng dẫn cài Tomcat 7, bạn có thể download Tomcat tại đây:

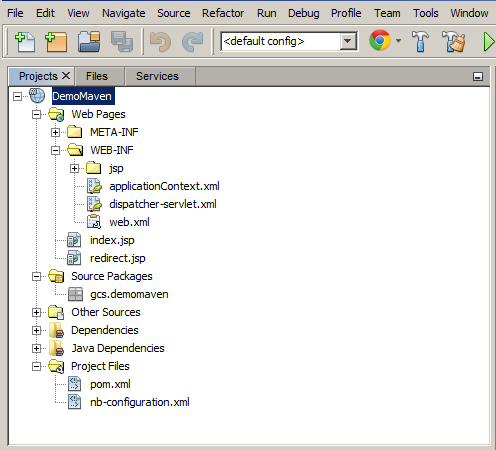
* <http://tomcat.apache.org/download-70.cgi>
* Chọn tab Services -> Servers -> Click phải chọn Add Server… -> Apache Tomcat or TomEE -> Điền các thông số cần thiết như chọn Server Location (chính là nơi bạn lưu trữ thư mục Apache Tomcat đã giải nến phía trên); nhập username và password -> Nhấn Finish -> Nhấn start để chạy thử Apache Tomcat và kiểm tra kết quả.
* Trong trường hợp Apache Tomcat không hoạt động thì:
* Mở file Apache-tomcat-7.0.64\bin\catalina.bat trong thư mục tomcat đã giải nén phía trên.
* Tìm và xóa các dòng.
* Chạy lại Tomcat

Bạn sẽ có 1 project như thế này



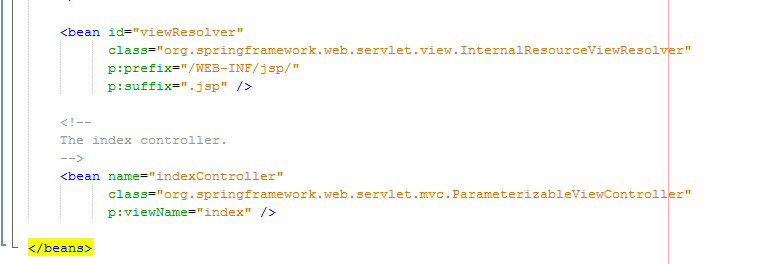
Trong netbean có hỗ trợ mẫu chuẩn của Spring MVC, chúng ta chỉ cần lấy và sử dụng nó.

* Click phải DemoMaven -> Properties -> Frameworks -> Add -> chọn Spring Web MVC -> Ok

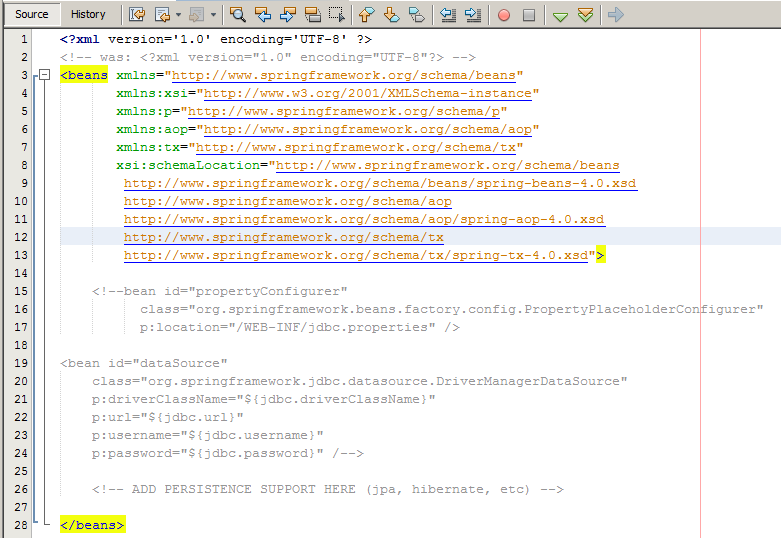


* Bước 2: Chỉnh các file cấu hình cho ứng dụng tại WEB-INF
* File dispatcher-servlet.xml

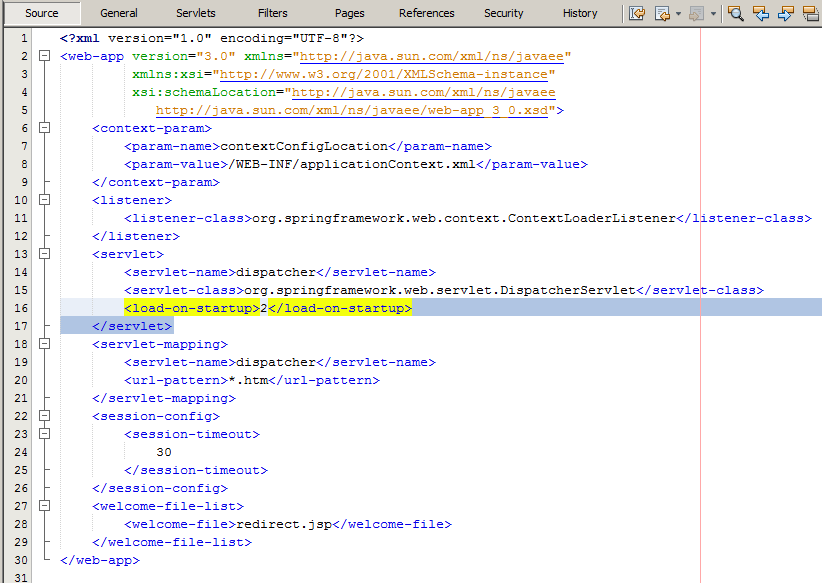




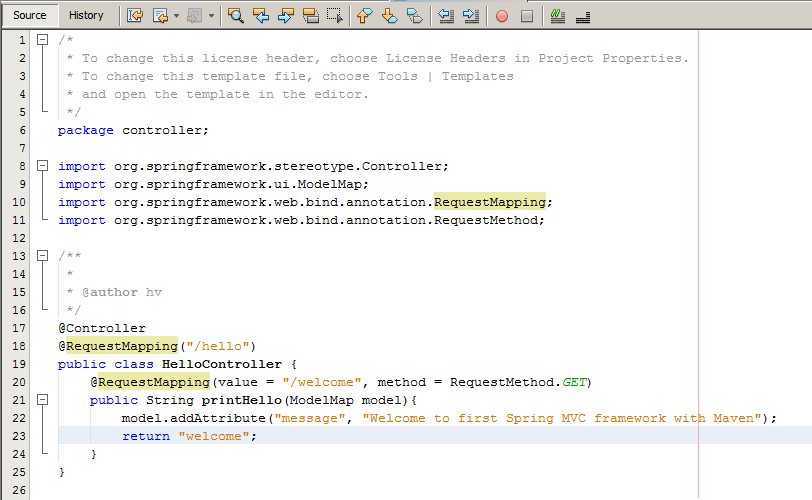
* File applicationContext.xml



* File web.xml



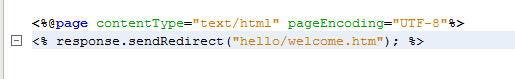
* Bước 3: Trong thư mục Source Packages, tạo 1 package controller. Trong package controller tạo thêm 1 class HelloController
* Bước 4: Trong class HelloController tạo phương thức để truyền đi một thông điệp (ví dụ: thông điệp xin chào), thông điệp này sẽ được gửi đến và hiển thị lên trang welcome.jsp
* HelloController.class



* welcome.jsp

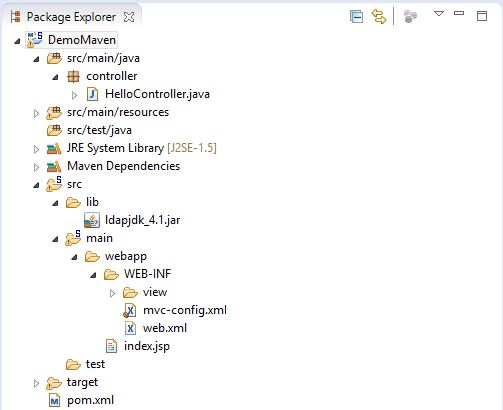


* Bước 5: Trong tập tin redirect.jsp gọi và chuyển hướng đến trang welcome.htm



## External Dependencies

* Maven dùng để quản lý dependency dùng khái niệm của Repositories. Bạn hãy nghĩ xem, chuyện gì sẽ xảy ra nếu dependency không nẳm trong Remote repositories và central repositories. Lúc đó, Maven sẽ dùng khái niệm của External Dependency.
* Ví dụ: Tôi dùng ldapjdk\_4.1.jar, thư viện này hỗ trợ cho các hoạt động của LDAP.
* Bây giờ cấu trúc project của tôi sẽ là:



* File pom.xml



* groupId và artifactId: thông thường sẽ lấy tên của thư viện
* scope: phạm vi hoạt động của thư viện
* systemPath: vị trí của thư viện

## Build Automation

* Buid Automation là quá trình giúp cho các dự án phụ thuộc lẫn nhau có tính ổn định khi có sự thay đổi nào được thực hiện.
* Ví dụ file pom.xml có 2 dự án phụ thuộc lẫn nhau

**CoreJavaTutorials Project**



**JavaSamples Project**



# Tài liệu tham khảo

<http://zeropic.onepagefree.com/files/File%20Extensions.pdf>

<https://docs.oracle.com/middleware/1213/jdev/user-guide/jdev-build-java-projects.htm#OJDUG6523>

<http://www.javacreed.com/why-should-we-use-maven/>

<http://www.mojohaus.org/versions-maven-plugin/>

<https://maven.apache.org/pom.html#Inheritance>

<http://stackoverflow.com/questions/8026447/what-does-mean-parent-in-maven-pom>

<http://maven.apache.org/guides/introduction/introduction-to-profiles.html>

<http://search.maven.org/#search%7Cga%7C1%7C>

<https://www.youtube.com/watch?v=otx2D4tKbE8>

<http://books.sonatype.com/mvnref-book/reference/>

<http://www.iet.unipi.it/a.bechini/concur/docs/Maven_slides.pdf>

<http://o7planning.org/en/10101/install-maven-into-eclipse>