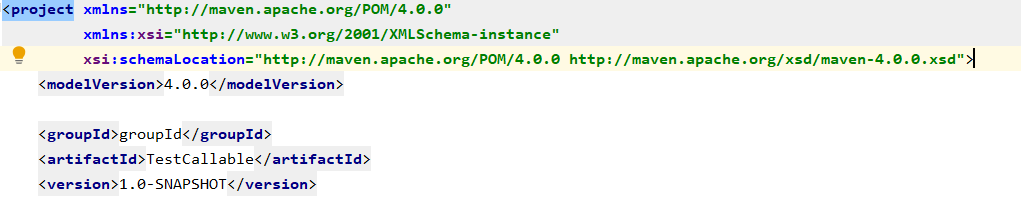
# Maven project

1. Maven là gì?
   1. Maven là công cụ để quản lý dự án Java, nó có rất nhiều các tính năng như :  
      - Builds  
      - Doucmentation  
      - Reporting  
      - Dependencies  
      - SCMs  
      - Releases  
      - Distribution
2. Maven Build Life cycle
   1. Build Lifecycle basic
      1. Maven được xây dựng xung quanh ý tưởng của build lifecycle. Có nghĩa là quá trình build và phân phối project được định nghĩa 1 cách rõ ràng một cách có quy trình và tuần tự.
      2. Đối với một người xây dựng dự án, điều này có nghĩa là chỉ cần học một vài bộ lệnh nhỏ là có thể build bất kỳ Maven project nào, với POM có thể tạo được project như mong muốn.
      3. Maven có 3 lifecycle được tích hợp sẵn trong maven đó là : default, clean và site.
         1. DEFAULT LIFECYCLE: đảm nhiệm quả trình deployment của project
         2. CLEAN LIFECYCLE: đảm nhiệm clean project
         3. SITE LIFECYCLE: đảm nhiệm tạo project’s site documentation
   2. Build Lifecycle được tạo bởi các pha (phases)
      1. Mỗi Lifecycle được xây dựng bởi danh sách các build phase khác nhau. Build phase đại diện cho một giai đoạn trong lifecycle. Ví dụ, DEFAULT LIFECYCLE có chứa các phase sau  
         - validate: kiểm tra project có chính xác hay không và mọi thông tin cần thiết đều có sẵn.  
         - compile: biên dịch mã nguồn của dự án  
         - test: test mã nguồn biên dịch với test framework  
         - package: lấy mã nguồn được biên dịch và đóng gói nó thành định dạng có thể phân phối được như : JAR hay WAR  
         - verify: chạy bất kỳ đoạn test nào để kiểm tra package hợp lệ và đáp ứng các tiêu chí về chất lượng  
         - install: cài đặt package vào repository để sử dụng trong project khác như 1 cái dependency (local)  
         - deploy: được thực hiện trong môi trường build, copy final packages để chia sẻ cho các developers và projects từ xa  
         Còn nhiều các phases của DEFAULT LIFECYCLE nữa được khai báo ở đây: [https://maven.apache.org/guides/introduction/introduction-to-the-lifecycle.html#Lifecycle\_Reference](https://maven.apache.org/guides/introduction/introduction-to-the-lifecycle.html%23Lifecycle_Reference)  
         => Các phases này được thực thi một cách tuần tự để hoàn thành cái DEFAULT LIFECYCLE.  
         => Tóm gọn lại với DEFAULT LIFECYCLE: maven sẽ kiểm tra, xác nhận dự án (validate), sau đó là biên dịch mã nguồn dự án rồi test mã nguồn biên dịch và đóng gói dự án, chạy các test được tích hợp với cái packages đó và cài đặt package đó vào local repository, cuối cùng là deploy tới remote repository.
      2. Các dòng lệnh thông thường được sử dụng
         1. mvn verify: bởi vì các build phases phải được thực thi theo tuần tự. nên trước khi chạy tới verify, ta phải chạy các phases : validate, compile, package
      3. Một build phase được tạo thành từ plugin Goals
         1. Một plugin goal đại diện cho một công việc được chỉ định góp phần xây dựng và quản lý một project. Một lifecycle thì có nhiều phases, 1 phase thì có nhiều goals.  
            - Ví dụ lệnh : mvn clean dependency:copy-dependencies package thì clean và package là các build phase trong khi đó : dependency:copy-dependencies là goals
      4. Một số Phases thường không được gọi từ command line : pre-\*, post-\*, process-\*
3. Cài đặt dự án để sử dụng Build lifecycle  
   - Build lifecycle đơn giản và đủ để sử dụng, nhưng khi xây dựng cấu trúc maven build cho 1 dự án, ta có thể phân công nhiệm vụ cho từng build phase
   1. Packaging:   
      - Hầu hết ta sử dụng cấu hình chung là (packaging). Set <packaging> cho project, mặc định nếu không set packaging thì nó là <packaging>jar</packaging>. Mỗi packaging sẽ chứa 1 danh sách goals để thực thi các giai đoạn cụ thể. Các packaging khác nhau sẽ có các build phases khác nhau.
   2. Plugin:  
      - Phương pháp thứ 2 để thêm goals cho các phases là dùng plugin. Plugin là một artifacts cung cấp goals cho Maven. Mỗi plugin có thể có một hoặc nhiều goals, mỗi goals đại diện cho một khả năng của plugin đó   
      - Plugin có thể chứa các thông tin chỉ ra sử dụng lifecycle phase nào ràng buộc cái goals. Việc tự thêm plugin là không đủ thông tin, ta phải chỉ ra cái goals muốn chạy như một phần của quá trình build.  
        
      Với thẻ <executions>, ta có thể dùng cùng một goal ở nhiều chỗ
4. A Project Object Model (Đối tượng mô hình hoá dự án) (POM)
   1. POM là cái gì ? là một đơn vị công việc cơ bản ở Maven.   
      - Nó là XML file có chứa các thông tin về dự án và chi tiết cấu hình được sử dụng bởi Maven để xây dựng dự án.   
      - Nó có chứa các config values mặc định cho hầu hết các dự án. Ví dụ như build directory là target, sources directory là src/main/java, source test là src/test/java và nhiều thứ khác.   
      - Khi mà thực thi task hoặc là goal, Maven tìm kiếm file pom ở trong thư mục, nó đọc file POM và lấy ra các thông tin cần phải cấu hình và thực thi goals.  
      - Một số các cấu hình được chỉ định ở file POM là project dependencies. Plugin hoặc goals có thể thực thi build profile.
   2. Super POM:  
      - Là Maven default POM. Tất cả các POM đều kế thừa từ SUPER POM.
   3. Minimal POM:  
      - minimum requirement for POM :  
       + project : root  
       + modelVersion: nên là 4.0.0  
       + groupId: id của nhóm dự án  
       + artifactId: id của project  
       + version: phiên bản của project  
      
   4. Project Inheritance: Kế thừa dự án
      1. Các phần tử của POM được hợp nhất như sau :   
         - Dependencies  
         - Developers and Contributors  
         - Plugin list (including reports)  
         - Plugin executeions with matching ids  
         - plugin configuration  
         - resources  
         Super POM là một ví dụ về kế thừa dự án, tuy nhiên ta có thể tạo ra parent POM của chính mình bằng cách chỉ định parent element ở trong pom  
         
   5. Tổng hợp dự án
      1. Project Aggregation (Tổng hợp dự án) cũng tương tự như Kế thừa dự án. Nhưng mà thay vì chỉ định parent POM từ module, nó chỉ định các modules từ parent POM. Bằng cách làm như vậy, parent project biết nó có bao nhiêu modules và nếu như 1 lệnh Maven được gọi để chống lại parent project, lệnh Maven đó sau đó cũng sẽ được thực thi cho modules khác của parent
   6. Nôi suy dự án và các biến (Project Interpolation và variables)
      1. Ta có thể lấy giá trị của basedir, baseUri, với lệnh ${project.basedir}, ${project.baseUri} ,…
      2. Ta cũng có thể cấu hình thêm biến với <properties>  
         
5. Maven Profiles
   1. Apache Maven đã có những bước tiến dài để đảm bảo rằng các bản build project trở trên portable. Maven cho phép build configuration ở bên trong file POM
6. DependencyManagement
   1. Quản lý các dependency khác version với nhau khi kế thừa dự án, <dependencyManagement> quản lý các version của <dependency> , nó được sử dụng ở file pom của Parent project, ở đây dependency phải khai báo version mà nó sử dụng.
7. Scope
   1. Phạm vi cung cấp thư viện (dependency)  
      - Provided : web container hoặc jdk sẽ cung cấp thư viện này  
      - Compile (mặc định) : thư viện sẽ đẩy vào classpath của project  
      - runtime: Maven nhúng thư viện này lúc chạy runtime hoặc trong test classpath, nhưng trong compile classpath thì không.  
      - test: Khai báo rằng thư viện này không cần thiết trong lúc chạy thực tế, nó chỉ chạy test compile và execute test  
      - system: chính ta phải cung cấp depedency đó một cách tường minh bằng file jar và đường dẫn tới file đó  
      - import: scope này chỉ support dependency của project type pom ở dependencyManagement