



Appendix Apache Maven

Presented by Hosein Zare
Twitter: @zare88r





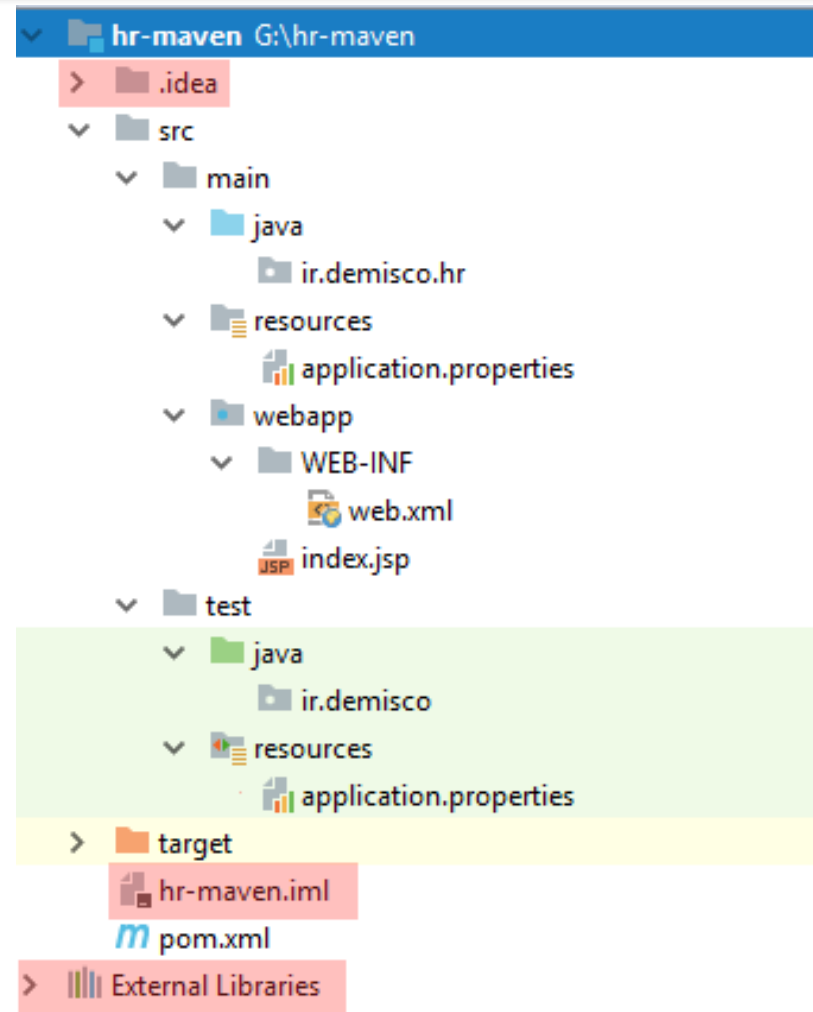
- Apache Maven یک فریم ورک مدیریت پروژه استاندارد محور است.
- Maven در ساخت ، تست ، گزارش گیری و پکیج کردن پروژه ها کمک میکند.
- ورژن اول آن در سال 2004 منتشر شد
- آخرین ورژن آن (تا اکنون) 3.5.3 میباشد



Standardized Directory Structure

- در maven از یک سیستم استاندارد جهت پوشه بندی فایل های مختلف از جمله کدها ، تست ها ، و فایل های تنظیمات استفاده میشود .
- با این روش :
 - ساختار پروژه ها یکسان شده و بالطبع باعث سردرگمی برنامه نویسان نمیشود
 - در شروع پروژه ها تصمیم گیری جهت نحوه ی چیدمان فایل ها نیاز نمیباشد
 - این قرارداد تطبیق پذیری پروژه ها بین ابزار های مختلف ایجاد میکند


Standardized Directory Structure





Archetypes

- Maven archetype ها پروژه های نمونه ی از قبل تعریف شده ای هستند که برای تعریف پروژه های جدید مورد استفاده قرار میگیرند.
- با استفاده از این سیستم میتوانید ساختار استاندارد پوشه بندی گفته شده را به صورت اتوماتیک ساخت



Plug-ins

- Maven از یک معماری plugin محور تبعیت میکند
- این پلاگین ها میتوانند وظایف مختلفی را انجام دهند از جمله کامپایل ، پاک کردن خروجی ها ، ساخت ear , war , jar ، تولید گزارش و ...
- لیست تعدادی از پلاگین ها:

<https://maven.apache.org/plugins/>



Uniform Build Abstraction

- Maven روشی یکپارچه برای ساخت پروژه ها معرفی میکند
- اینکار توسط یک سری command صورت میگیرد
- این موضوع باعث میشود با یکبار یادگیری نحوه ی ساخت ، تمرکز بیشتری روی develop بوجود بیاید.



مدیریت dependency

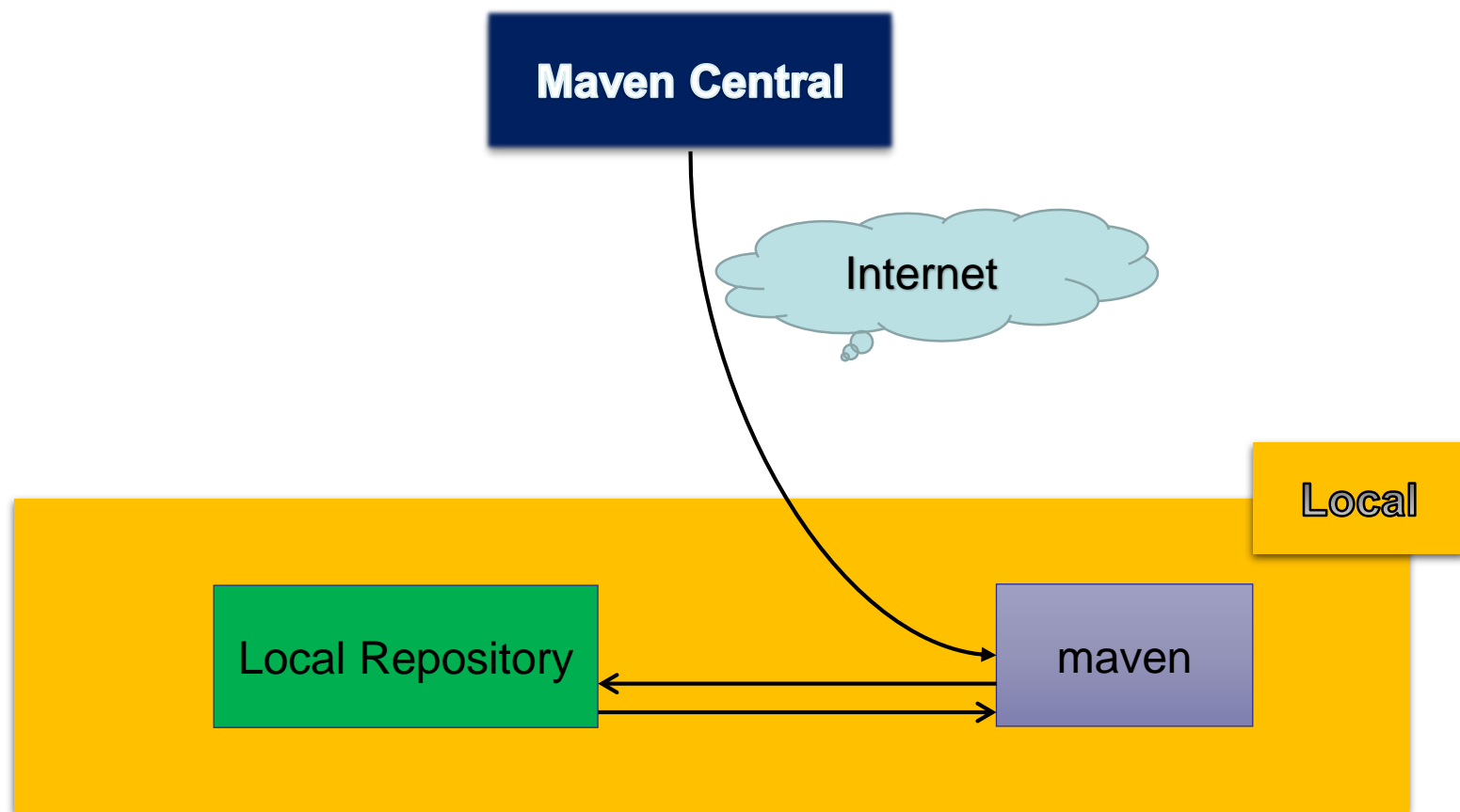
- اکثر پروژه ها برای کامپایل و اجرا به فریم ورک ها و یا کتابخانه های دیگر احتیاج دارند ، دانلود کردن jar فایل ها به صورت دستی ، نگهداری آنها و همچنین کنترل ورژن های آن کار دشواری است
- Maven روشی جهت مدیریت آن دارد:
 - ابتدا این وابستگی ها را درون فایلی به نام pom.xml
 - سپس کتابخانه های مورد نیاز به صورت خودکار دانلود شده و درون پروژه قرار میگیرند



مدیریت dependency

- Maven برای مدیریت وابستگی ها از Repository استفاده میکند
- کتابخانه هایی که با این عنوان در pom.xml معرفی میشوند از یک Remote Repository دانلود میشوند و درون یک Local Repository (روی کامپیوتر خودتان) ذخیره میشوند

مدیریت dependency



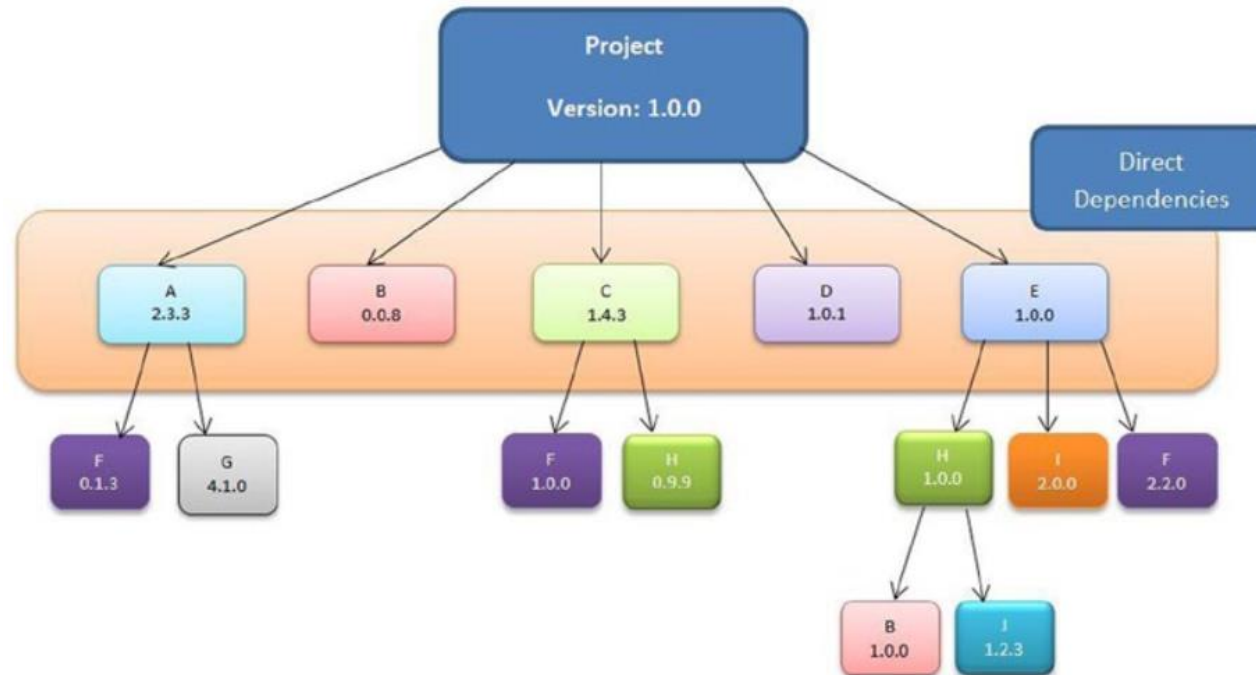


مشخصات dependency

- groupId : مشخص کننده گروه یا سازمان مسئول پروژه مثل :
org.hibernate , org.springframework , log4j –
- artifactId : مشخص کننده محصول تولید شده مثل :
hibernate-tools , spring-core –
- version : ورژن محصول مثل
1.0.0 , 2.3.1-SNAPSHOT , 4.3.6-FINAL –
- type : چه پکیجی از محصول مورد نیاز است مثل :
JAR , WAR , EAR –

Transitive Dependencies

- ممکن است یک dependency خود به artifact های دیگری dependency داشته باشد. به این حالت transitive dependency میگویند.
 - مثلا hibernate به jboss logging ، dom4j ، javaassist و ...





Dependency Scope

- میتوان به maven توضیح داد یک وابستگی در چه زمانی مورد نیاز است:
 - compile (پیش فرض) : یعنی در تمامی مراحل کامپایل ، تست و اجرا این وابستگی در class path وجود داشته باشد
 - provided : فقط در زمان کامپایل و تست باشد و در مرحله ساخت محصول و اجرا نیازی نیست
 - runtime : فقط در زمان اجرا استفاده شود
 - test : فقط برای اجرای تست ها
 - system : شبیه به provided با این تفاوت که آن را از file system آن را لود میکند
 - import : صرفاً برای استفاده از تنظیمات یک pom فایل دیگر استفاده میشود



Maven Life Cycle

- برای ساخت یک پروژه معمولاً مجبوریم چندین مرحله را طی کنیم مانند :
 - کامپایل ، تست ، پکیج و ...
- Maven از مفهوم goal برای تعریف هر مرحله استفاده میکند
- goal ها درون plugin ها پکیج میشوند

```
mvn plugin_identifier:goal_identifier
```

```
mvn clean:clean
```

```
mvn compile:compile
```



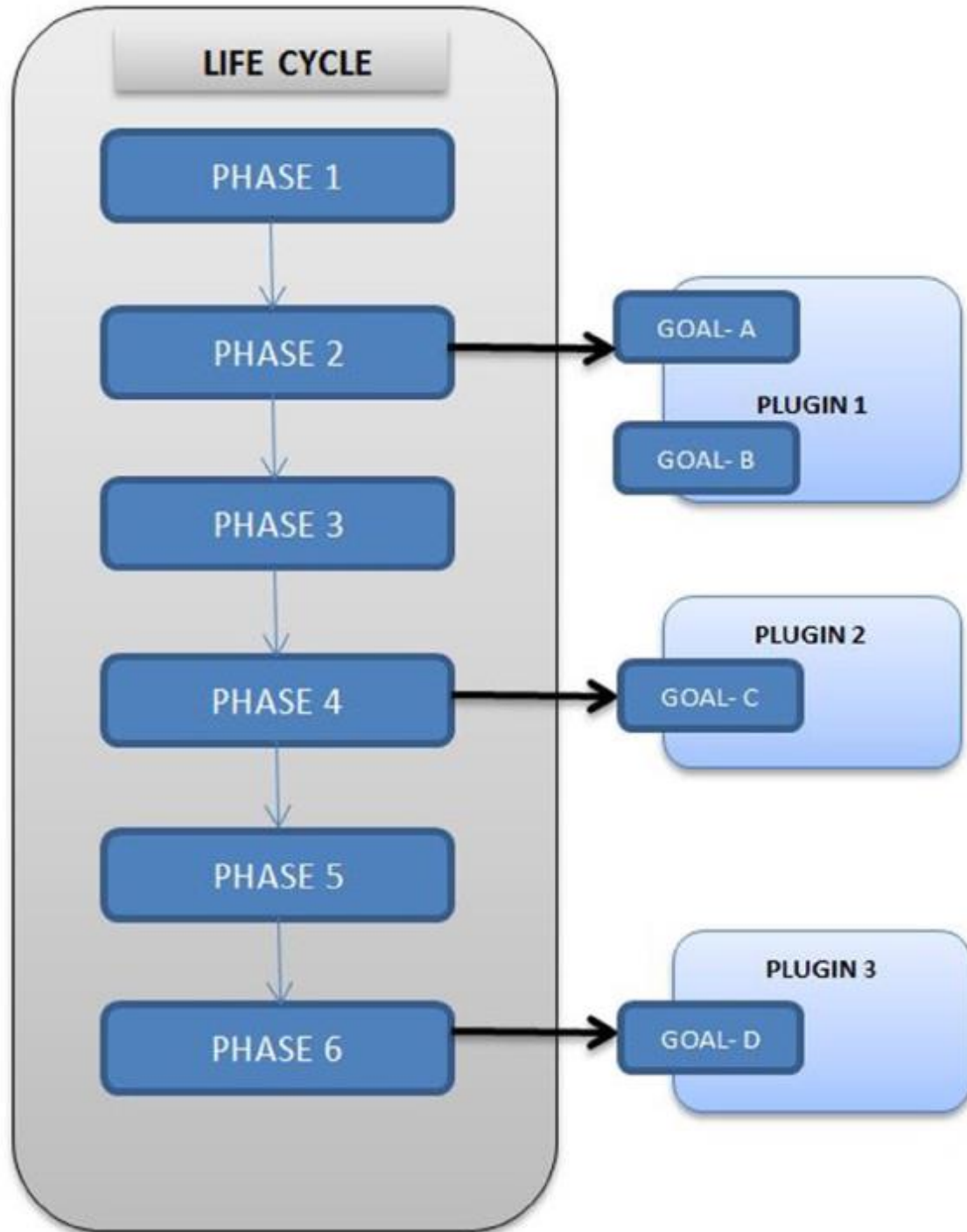
Life Cycle And Phases

- Maven از یک life cycle برای ساخت پروژه ها استفاده میکند
- این چرخه حیات شامل مراحل برای اجرای پشت سر هم میباشد . به این مراحل phase گفته میشود.
- سه phase که در maven به طور پیش فرض تعریف شده اند شامل :
 - Default : شامل کامپایل ، تست ، پکیج و دیپلوی
 - Clean : شامل حذف فایل های موقت مانند class های ساخته شده
 - Site : شامل ساخت documentation ها و مستقر کردن پروژه



Default Phase

- Validate : جهت چک کردن صحیح بودن پروژه و وابستگی هایش
- Compile
- Test
- Package : پکیج کردن پروژه در فرمت هایی مثل jar یا war
- Install : نصب پکیج ها در local repository
- Deploy : ارسال پکیج به remote repository



Goal	Plugin
process-resources	resources:resources
compile	compiler:compile
process-test-resources	resources:testResources
test-compile	compiler:testCompile
test	surefire:test
package	jar:jar
install	install:install
deploy	deploy:deploy

<https://maven.apache.org/guides/introduction/introduction-to-the-lifecycle.html>