Maven ile Build ve Bağımlılık Yönetimi Eğitimi

Harezmi Bilişim Çözümleri

Maven ile Build ve Bağımlılık Yönetimi

JAVA Eğitimleri

- **Build** ve **dependency management** aracıdır
- Çıkış noktasında projelerin build sürecini **standart** bir hale getirmek
- Projelerin **standart bir dizin yapısı**na sahip olmasını sağlamak
- Proje ile ilgili bilgilerin ve **artifact**'ların kolay bir biçimde oluşturulmasını ve yayımlanmasını sağlamak
- Projelerin **ortak bağımlılıkları**nı merkezi bir noktadan yönetmek vardır

Maven'ın Çıkışı

- Ant'ın, proje ile ilgili üst bilgi (**metadata**) içermemesi
- Uzun **build script**leri yazmayı gerektirmesi
- Projeler için **standart bir dizin yapısı** sunmaması
- **Bağımlılık yönetimi**ni ele almaması gibi nedenler Maven'in ortaya çıkmasına neden olmuştur

Maven'ın Kabiliyetleri

- Maven proje geliştirme sürecinde aşağıdaki noktalarda **kabiliyetler** sunar
  - Projenin build işlemi
  - Bağımlı kütüphanelerin yönetilmesi
  - Proje sürümlerinin yönetilmesi
  - Proje artifact'larının oluşturulması ve dağıtımı
  - Dökümantasyon ve Raporlama işlemleri

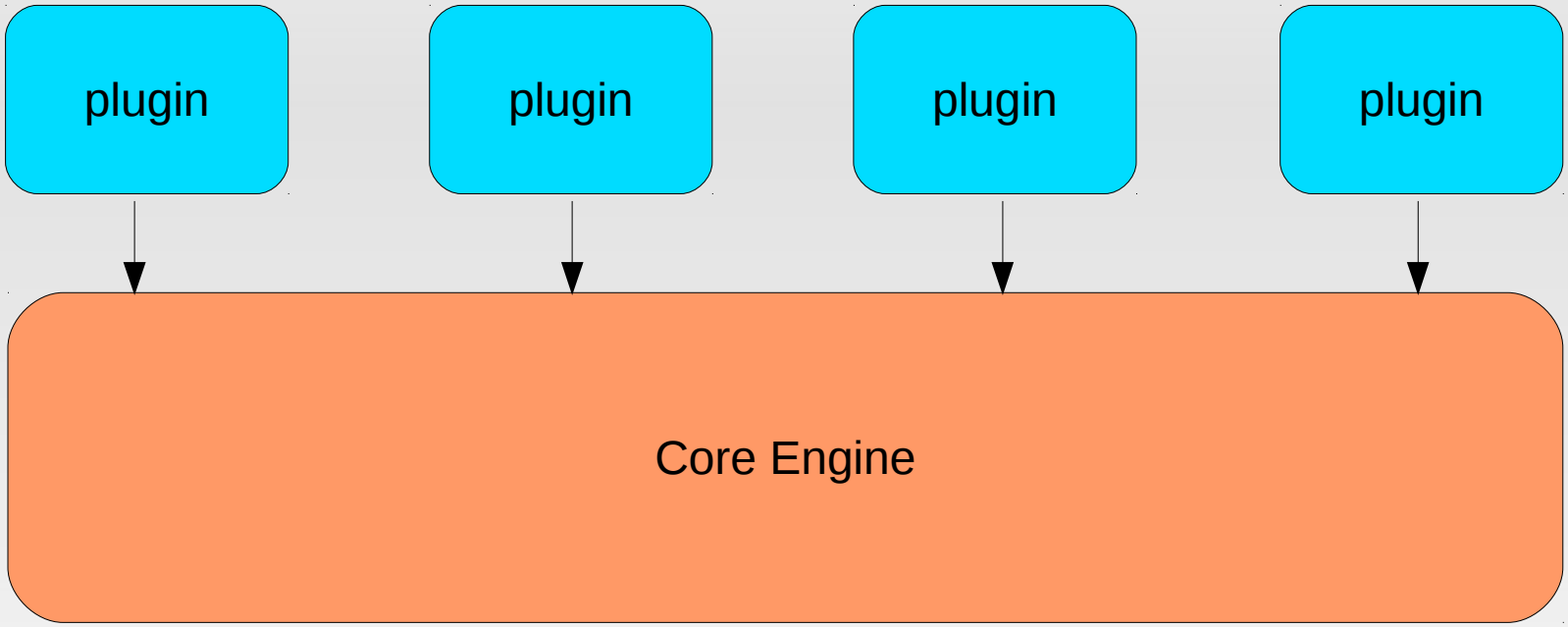
Maven Build Lifecycle

- **Build yaşam döngüleri** üzerinden build ve diğer proje yönetim işlemleri yapılır
- 3 temel **built-in** yaşam döngüsü vardır
  - **default**: projenin build ve deployment'ını yapar
  - **clean**: proje dizin yapısının temizlenmesini sağlar
  - **site**: projenin site yönetimini gerçekleştirir
- Her bir yaşam döngüsü de **fazlar**dan oluşur
  - validate, compile, test, package, install, deploy gibi

Maven Build Lifecycle

- Herhangi bir faz **doğrudan** çalıştırılabilir: mvn test
- Çalıştırılan fazdan **önceki fazlar**da sıra ile otomatik biçimde çalıştırılacaktır

Maven Mimarisi



USER_HOME/.m2/repository
Default local repo'dur

USER_HOME/.m2/settings.xml
Dosyası içinden değiştirilebilir

Birden fazla remote repo
Tanımlanabilir

Default olarak maven
Central repo vardır

Plugin'ler için de repo
tanımlanabilir

Maven Artifact

- Projelerin **paketlenmesi** sonucu oluşturulan dosyalardır
  - Jar, war, ear gibi
- Repository'lere **deploy** edilirler
- Diğer projeler tarafından **dependency** olarak kullanılabilirler
- GroupId, ArtifactId ve Version alanları ile ayrıştırılırlar

Maven POM

- Project Object Model olarak bilinir
- Projenin **kök dizini** altında yer alan bir xml dosyasıdır
  - pom.xml
- Projenin **bütün build konfigürasyonu**nu içerisinde barındırır
- Bir pom.xml başka bir pom.xml'den konfigürasyon bilgilerini inherit edebilir

Maven POM

- En tepede **super pom** vardır
- Üst pom'lardaki ayarlar alt pom'lar tarafından **override** edilebilir
- Bir pom **birden fazla modül**ü içerebilir
- Aynı zamanda bu modüllerin parent pom'u da olabilir

Maven Archetype

- **Şablon proje** vazifesi görür
- Aynı türdeki projeler için ortak konfigürasoyn tanımları, dosyalar vb içerir
- Archetype'larda repo'lara deploy edilebilir

Maven Projelerinin Dizin Yapısı

JAVA Eğitimleri

src/main/java	Uygulama kaynak kodları
src/main/resources	Uygulama resource dosyaları
src/main/filters	Resource filter dosyaları
src/main/assembly	Assembly tanım dosyaları
src/main/config	Konfigürasyon dosyaları
src/main/scripts	Script dosyaları
src/main/webapp	Web uygulamasına ait dosyalar
src/test/java	Test kaynak kodları
src/test/resources	Test resource dosyaları
src/test/filters	Test resource filter dosyaları
src/site	Projenin web sitesi içeriği
pom.xml	Projenin POM konfigürasyon dosyası
LICENSE.txt	Projenin lisans dosyası
NOTICE.txt	Projenin bağımlılıkları ile ilgili bilgilendirme ve uyarı notaları
README.txt	Projenin readme dosyası

Standalone Uygulamaların Minimum Dizin Yapısı

src/main/java	Uygulama kaynak kodları
src/main/resources	Uygulama resource dosyaları
src/test/java	Test kaynak kodları
src/test/resources	Test resource dosyaları
pom.xml	Projenin POM konfigürasyon dosyası

Web Uygulamalarının Minimum Dizin Yapısı

src/main/java	Uygulama kaynak kodları
src/main/resources	Uygulama resource dosyaları
src/test/java	Test kaynak kodları
src/test/resorces	Test resource dosyaları
src/main/webapp	Web uygulama root dizini
src/main/webapp/index.jsp	index.jsp örnek dosyası
src/main/webapp/WEB-INF	WEB-INF konfigürasyon dizini
src/main/webapp/WEB-INF/web.xml	web.xml konfigürasyon dosyası
src/main/webapp/WEB-INF/classes	Web uygulaması classes dizini
src/main/webapp/WEB-INF/lib	Web uygulaması kütüphaneler dizini
pom.xml	Projenin POM konfigürasyon dosyası

Maven Bağımlılık Yönetimi

- Projenin bağımlılıklarını otomatik olarak yönetir
- Bu bağımlılıkların da bağımlılıkları vardır
- Buna **transitive dependencies** adı verilir
- Maven bütün bağımlılıkları yönetir
- Jar'lar remote repo'lardan indirilir ve local repo'da **cache**'lenir
- Bağımlılıkların “**scope**”ları vardır
  - Compile, provided, test, runtime, system, import

Maven Bağımlılık Yönetimi

JAVA Eğitimleri

- Projenin hangi kütüphanelerin hangi versiyonlarına ihtiyaç duyduğu **pom.xml**'de tanımlanır
- Bir bağımlılık **doğrudan** veya **dolaylı** olarak gelebilir
- Spesifik versiyon pom.xml'de tanımlı değil ise ve iki farklı versiyon varsa **bağımlılık ağacında kök'e en yakın olan** tercih edilir
- Herhangi bir dolaylı bağımlılığı "**exclude**" etmek de mümkündür

Bağımlılıkların Scope Tanımları

- **Compile**
  - default scope'tur
  - Build yaşam döngüsündeki bütün classpath'lerde yer alır
  - Oluşturulan artifact'ın içerisine de eklenir
- **Provided**
  - compile ve test classpath'lerinde yer alır
  - Oluşturulan artifact'da yer almaz
  - Artifact'ın deploy edileceği container tarafından sağlanacağı varsayılır

Bağımlılıkların Scope Tanımları

- **Test**
  - Sadece test kodlarının derlenmesi ve testlerin çalıştırılması aşamasında yer alır
  - Projenin derlenmesi veya çalıştırılmasında gerekmez
  - Oluşturulan artifact'da da yer almaz
- **Runtime**
  - Compile aşamasında yer almaz
  - Sadece runtime'da ve test aşamasında yer alır

Maven Repository'leri

- Maven, **pom.xml** dosyasında belirtilen kütüphane bağımlılıklarını “**maven repository**” denilen depolardan otomatik olarak çeker
- Üç çeşit maven repository vardır
  - Local
  - Central
  - Remote

Local Repository

- Maven build komutundan sonra tüm dependency jar dosyaları local repository altına indirilir
- Böylece uzaktaki kütüphaneleri tekrar tekrar indirmeye gerek kalmaz
- Default olarak local repository path'i **$USER_HOME/.m2** şeklindedir
- **.m2/settings.xml** dosyası aracılığı ile local repository'nin path'i değiştirilebilir
  - <localRepository>C:/LocalRepository</localReposiyory>

Central Repository

- Maven topluluğu tarafından sunulan repository'dir
- Eğer bağımlılıklar local repository altında bulunamazsa default olarak buraya bakılır
- http://repo.maven.apache.org/maven2

Remote Repository

- Maven central repository altında bulunmayan jarlar için ise tanımlı remote repository'lere bakılır
- Bu tanımlar **settings.xml** veya **pom.xml** içerisinde yapılabilir

```
<repositories>
   <repository>
      <id>harezmi-repo</id>
      <name>libs-release</name>
      <url>http://artifactory.harezmi.com.tr/libs-release</url>
     </repository>
</repositories>
```

SNAPSHOT Bağımlılıkları

- Maven bağımlılıkları tanımlanırken, bazı artifactler SNAPSHOT **son eki** alabilirler
- Örneğin: foo-1.0.0-SNAPSHOT.jar
- Bu artifact'in **geliştirme süreci**nde olduğunu, kararlı bir durumda olmadığını anlatır
- Maven'ın bu eki alan artifact'leri bulma yöntemi farklıdır
- Maven bu artifact'ların build sırasında **en güncel versiyonu**nu kullanmaya çalışacaktır

SNAPSHOT Bağımlılıkları

- Local repository'de bir kopyası olsa bile, remote repository'ye giderek **daha güncel bir kopya**sı var mı diye kontrol edecektir
- Default olarak bu kontrol **günde bir kere** yapılır
- Kontrol sıklığı repository tanımı yapılan yerde **updatePolicy** bölümünde değiştirilebilir

```
<repository>
   ...
   <snapshots>
      <enabled>true</enabled>
      <updatePolicy>always</updatePolicy>
   </snapshots>
</repository>
```

SNAPSHOT Bağımlılıkları

- **updatePolicy** aşağıdakilerden değerlerden biri olabilir
  - **always**: Maven her buildde yeni bir versiyon var mı diye kontrol edecektir.
  - **daily**: varsayılan sıklık budur, günlük olarak kontrol yapılır.
  - **interval:XXX**: (XXX) dakika aralıklarla kontrol yapacaktır.
  - **never**: Kararlı sürümlere (SNAPSHOT eki almayan bağımlılıklara) nasıl davranıyorsa aynı şekilde davranacaktır.

Çoklu Modül Desteği

- Modülleri olan projeler **multi-module** veya aggregator proje olarak anılır
- Paket tipi **pom** olan bir proje, birden fazla projeyi module olarak listeleyebilir
- Parent proje build edildiği vakit listedeki bütün modüller build edilecektir

Parent Projenin POM Yapısı


```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">

   <modelVersion>4.0.0</modelVersion>

   <groupId>tr.com.harezmi</groupId>
   <artifactId>parent-project</artifactId>
   <version>1.0.0</version>

   <packaging>pom</packaging>

   <modules>
      <module>project-1</module>
      <module>project-2</module>
   </modules>
</project>
```

Modüllerin POM Yapıları

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">

   <modelVersion>4.0.0</modelVersion>

   <parent>
      <groupId>tr.com.harezmi</groupId>
      <artifactId>parent-project</artifactId>
      <version>1.0.0</version>
      <relativePath>../parent-project</relativePath>
   </parent>

   <artifactId>project-1</artifactId>
</project>
```

İletişim

- Harezmi Bilişim Çözümleri Ltd.
- Kurumsal Java Eğitimleri
- http://www.java-egitimleri.com
- info@java-egitimleri.com