Maven

Thursday, 9 September 2021 14:39

Basics of Java
Done: View, To do: Go through the activity to the end
History of Java

From <https://lms.clarusway.com/mod/lesson/view.php?id=5232>

Java is a general-purpose programming language that is class-based, object-oriented, and designed to have as few dependencies as possible. It is intended to let application developers Write Once, Run Anywhere (WORA), meaning that compiled Java code can run on all platforms that support Java without the need for recompilation. Java applications are typically compiled to bytecode that can run on any Java Virtual Machine (JVM) regardless of the underlying computer architecture. The syntax of Java is similar to C and C++ programming languages, but it has fewer low-level facilities than either of them.

From <https://lms.clarusway.com/mod/lesson/view.php?id=5232>

TURKISH
Java, sınıf tabanlı, nesne yönelimli ve mümkün olduğunca az bağımlılığa sahip olacak şekilde tasarlanmış genel amaçlı bir programlama dilidir. Uygulama geliştiricilerinin Bir Kez Yaz, Her Yerde Çalıştır (WORA) sağlaması amaçlanmıştır; bu, derlenmiş Java kodunun yeniden derlemeye gerek kalmadan Java'yı destekleyen tüm platformlarda çalışabileceği anlamına gelir. Java uygulamaları tipik olarak, temeldeki bilgisayar mimarisinden bağımsız olarak herhangi bir Java Sanal Makinesinde (JVM) çalışabilen bayt koduna derlenir. Java'nın sözdizimi, C ve C++ programlama dillerine benzer, ancak her ikisinden de daha az düşük seviyeli olanaklara sahiptir.

Sun Microsystems released the first public implementation as Java 1.0 in 1996. It promised to Write Once, Run Anywhere (WORA), providing no-cost run-times on popular platforms. Fairly secure and featuring configurable security, it allowed network- and file-access restrictions. Major web browsers soon incorporated the ability to run Java applets within web pages, and Java quickly became popular. With the advent of Java 2 (released initially as J2SE 1.2 [Java 2 Standart Edition] in December 1998 – 1999), new versions had multiple configurations built for different types of platforms. J2EE (Java 2 Enterprise Edition) included technologies and APIs (Application Programming Interfaces) for enterprise applications typically run in server environments, while J2ME (Java 2 Micro Edition) featured APIs optimized for mobile applications. The desktop version was renamed J2SE. In 2006, for marketing purposes, Sun renamed new J2 versions as Java EE, Java ME, and Java SE, respectively

From <https://lms.clarusway.com/mod/lesson/view.php?id=5232>

TURKISH
Sun Microsystems, 1996'da Java 1.0 olarak ilk genel uygulamayı yayınladı. Bir Kez Yaz, Her Yerde Çalıştır (WORA) sözü vererek popüler platformlarda ücretsiz çalışma süreleri sağladı. Oldukça güvenli ve yapılandırılabilir güvenlik özelliğiyle ağ ve dosya erişimi kısıtlamalarına izin verdi. Büyük web tarayıcıları kısa süre içinde Java uygulamalarını web sayfalarında çalıştırma yeteneğini dahil etti ve Java hızla popüler oldu. **Java 2'nin (başlangıçta Aralık 1998 – 1999'da J2SE 1.2 [Java 2 Standart Sürüm] olarak piyasaya sürüldü) ortaya çıkmasıyla birlikte,** yeni sürümler farklı platform türleri için oluşturulmuş çoklu konfigürasyonlara sahipti. J2EE (Java 2 Enterprise Edition), tipik olarak sunucu ortamlarında çalışan kurumsal uygulamalar için teknolojileri ve API'leri (Uygulama Programlama Arayüzleri) içerirken, J2ME (Java 2 Micro Edition), mobil uygulamalar için optimize edilmiş API'leri içeriyordu. Masaüstü sürümü J2SE olarak yeniden adlandırıldı. 2006'da Sun, pazarlama amacıyla yeni J2 sürümlerini sırasıyla Java EE, Java ME ve Java SE olarak yeniden adlandırdı.

As of 2006, Sun released much of its Java Virtual Machine (JVM) as free and open-source software (FOSS), under the terms of the GNU General Public License (GPL). In 2007, Sun finished the process, making all of its JVM's core code available under free software/open-source distribution terms, aside from a small portion of code to which Sun did not hold the copyright.

Following Oracle Corporation's acquisition of Sun Microsystems in 2009–10, Oracle has described itself as the steward of Java technology with a relentless commitment to fostering a community of participation and transparency. This did not prevent Oracle from filing a lawsuit against Google shortly after that for using Java inside the Android SDK. Java software runs on everything from laptops to data centers, game consoles to scientific supercomputers

From <https://lms.clarusway.com/mod/lesson/view.php?id=5232>
TURKISH
2006 itibariyle Sun, **Java Sanal Makinesi'nin (JVM**) çoğunu GNU Genel Kamu Lisansı (GPL) koşulları altında ücretsiz ve açık kaynaklı yazılım (FOSS) olarak **yayınladı**. 2007'de Sun, Sun'ın telif hakkına sahip olmadığı küçük bir kod kısmı dışında, JVM'nin tüm çekirdek kodunu ücretsiz yazılım/açık kaynak dağıtım koşulları altında kullanıma sunarak süreci tamamladı.

Oracle Corporation'ın 2009-10'da Sun Microsystems'i satın almasının ardından Oracle, kendisini bir katılım ve şeffaflık topluluğunu geliştirmeye amansız bir bağlılıkla Java teknolojisinin koruyucusu olarak tanımladı. Bu, Oracle'ın bundan kısa bir süre sonra Java'yı Android SDK içinde kullandığı için Google'a dava açmasını engellemedi. Java yazılımı, dizüstü bilgisayarlardan veri merkezlerine, oyun konsollarından bilimsel süper bilgisayarlara kadar her şeyde çalışır

Java Specification
Java language specification

From <https://lms.clarusway.com/mod/lesson/view.php?id=5232&pageid=5967>

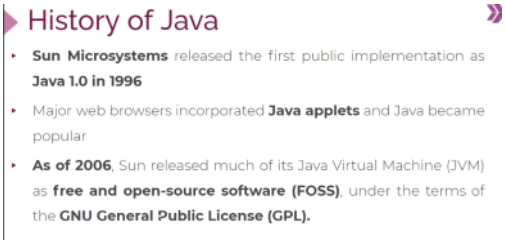
Computer languages have strict rules of usage. If you do not follow the rules when writing a program, the computer will not be able to understand it. The Java language specification and the Java API define the Java standards. The application program interface (API), also known as library, contains predefined classes and interfaces for developing Java programs. The Java language specification is a technical definition of the Java programming language's syntax and semantics. You can find the complete Java language specification at Java Language and Virtual Machine Specifications.
TURKISH
Java Spesifikasyonu Java dili belirtimi Bilgisayar dillerinin katı kullanım kuralları vardır. Bir program yazarken kurallara uymazsanız, bilgisayar onu anlayamaz. Java dili belirtimi ve Java API, Java standartlarını tanımlar. Kütüphane olarak da bilinen uygulama programı arayüzü (API), Java programları geliştirmek için önceden tanımlanmış sınıfları ve arayüzleri içerir. Java dili belirtimi, Java programlama dilinin sözdizimi ve anlambiliminin teknik bir tanımıdır. Java dili spesifikasyonunun tamamını Java Language and Virtual Machine Spesifikasyonlarında bulabilirsiniz.

What is JVM?
JVM (Java Virtual Machine) is a virtual machine. It is called a virtual machine because it doesn't physically exist. It is a specification that provides a runtime environment in which Java bytecode can be executed. It can also run those programs which are written in other languages and compiled to Java bytecode.

JVM nedir? JVM (Java Virtual Machine) sanal bir makinedir. Fiziksel olarak var olmadığı için sanal makine olarak adlandırılır. Java bayt kodunun yürütülebileceği bir çalışma zamanı ortamı sağlayan bir belirtimdir. Diğer dillerde yazılmış ve Java bayt koduna derlenmiş programları da çalıştırabilir.

JVMs are available for many hardware and software platforms. JVM, JRE, and JDK are platform dependent because the configuration of each OS is different from each other. However, Java is platform-independent. There are three notions of the JVM: specification, implementation, and instance.
The JVM performs the following main tasks:
- Loads code
- Verifies code
- Executes code

Istedigimiz her platformda avayi calistirabilmis olmamiz en buyuk avantajidir



Java daha kati bir dil yapisina sahip

JRE = Libarry

Java buyuk kucuk harf duyarkikigi var

Her java uygulamasinda bir main methodu olmal zorunda

// command tek satirli comment

/* birden fazla satirli comment

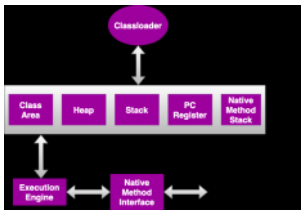
System clas out alt

Main programin giris satiri giris uygulamasi,

JVMs are available for many hardware and software platforms. JVM, JRE, and JDK are platform dependent because the configuration of each OS is different from each other. However, Java is platform-independent. There are three notions of the JVM: specification, implementation, and instance.
The JVM performs the following main tasks:

- Loads code
- Verifies code
- Executes code
- Provides runtime environment

TURKISH
JVM'ler birçok donanım ve yazılım platformu için mevcuttur. JVM, JRE ve JDK, her işletim sisteminin yapılandırması birbirinden farklı olduğu için platforma bağlıdır. Ancak, Java platformdan bağımsızdır. JVM'nin üç kavramı vardır: belirtim, uygulama ve örnek. JVM aşağıdaki ana görevleri gerçekleştirir: Kodu yükler Kodu doğrular Kodu yürütür
Çalışma zamanı ortamı sağlar



JVM Structure

What is JRE?

The Java Runtime Environment (JRE) is a software package which bundles the libraries (jars) and the Java Virtual Machine, and other components to run applications written in the Java. JVM is just a part of JRE distributions. To execute any Java application, you need JRE installed in the machine. It's the minimum requirement to execute Java applications on any machine.
JREs can be downloaded as part of JDKs or you can download them separately. JREs are platform dependent. It means that based on the type of machine (OS and architecture), you will have to select the JRE bundle to import and install.

URKISH
Java Runtime Environment (JRE), kitaplıkları (kavanozları) ve Java Sanal Makinesini ve Java'da yazılmış uygulamaları çalıştırmak için diğer bileşenleri bir araya getiren bir yazılım paketidir. JVM, JRE dağıtımlarının sadece bir parçasıdır. Herhangi bir Java uygulamasını çalıştırmak için makinede JRE'nin kurulu olması gerekir. Java uygulamalarını herhangi bir makinede yürütmek için minimum gereksinimdir.
TURKISH
JRE'ler, JDK'ların bir parçası olarak indirilebilir veya ayrı olarak indirebilirsiniz. JRE'ler platforma bağlıdır. Bu, makinenin türüne (işletim sistemi ve mimari) bağlı olarak, içe aktarmak ve yüklemek için JRE paketini seçmeniz gerekeceği anlamına gelir.

For example, you cannot install a 64-bit JRE distribution on a 32-bit machine. Similarly, JRE distribution for Windows will not work in Linux; and vice-versa.

What is JDK?

The Java Development Kit (JDK) is a superset of JRE. JDK contains everything that JRE has along with development tools for developing, debugging, and monitoring Java applications. You need JDK when you need to develop Java applications.
Same as JREs, JDKs are also platform dependent. So take care when you download the JDK package for your machine.

TURKISH
Örneğin, 32 bit bir makineye 64 bit JRE dağıtımı yükleyemezsiniz. Benzer şekilde, Windows için JRE dağıtımı Linux'ta çalışmayacaktır; ve tam tersi. JDK nedir? Java Geliştirme Kiti (JDK), JRE'nin bir üst kümesidir. JDK, Java uygulamalarını geliştirmek, hata ayıklamak ve izlemek için geliştirme araçlarıyla birlikte JRE'nin sahip olduğu her şeyi içerir. Java uygulamaları geliştirmeniz gerektiğinde JDK'ya ihtiyacınız var. JRE'ler gibi, JDK'lar da platforma bağlıdır. Bu nedenle, makineniz için JDK paketini indirirken dikkatli olun.

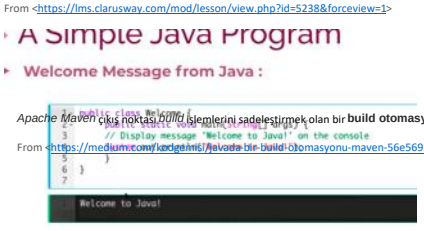
Previous
Next

From <https://lms.clarusway.com/mod/lesson/view.php?id=5232&pageid=5967>

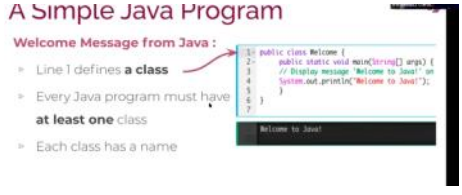
A Simple Java Program
Done: View To do: Go through the activity to the end

A Simple Java Program
Let's begin with a simple Java program that displays the message Welcome to Java! on the console

From <https://lms.clarusway.com/mod/lesson/view.php?id=5238&forceview=1>

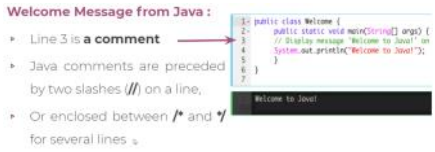


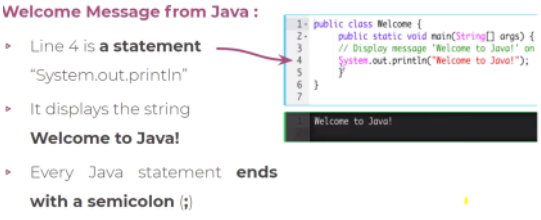
Apache Maven çıkış noktası *build* işlemlerini sadeleştirmek olan bir **build otomasyon** aracıdır.
From <https://medium.com/kodgemisi/javada-bir-build-otomasyonu-maven-56e56932b6f3>

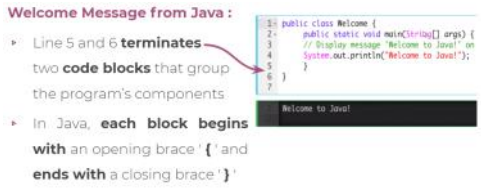


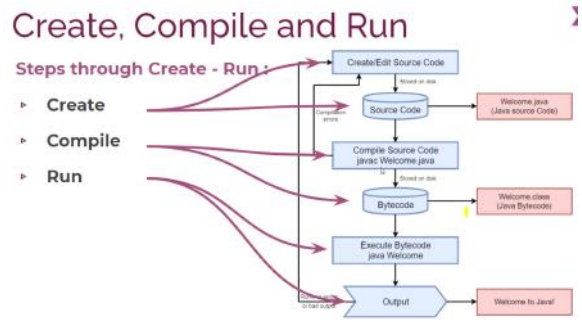
System clas out alt

Main programin giris satiri giris uygulamasi,









Tek bir source codun tek bir coda dönusturulmesine source kodda önce oindir dependanc var ise onu da indir sonra calisir

Artifac indirdigim anda calismaya hazir demek al kullan

Public ile bir sinif tanimlama var ersisim iznini veriyor

Minimum gereklilik bir public clastir. Public Erisim in olu olmayacagini ifade eder

What is Maven
Done: View **To do:** Go through the activity to the end
Introduction to Maven

From <https://lms.clarusway.com/mod/lesson/view.php?id=5236>

Maven began its life in Apache's Jakarta Alexandria Project in 2001. After 5 months of development, it was implemented in the project, Jakarta Turbine. Maven's contribution to the project was to simplify the building processes.

TURKISH

Maven, 2001 yılında Apache'nin Jakarta Alexandria Projesi'nde hayatına başladı. 5 aylık geliştirmeden sonra Jakarta Turbine projesinde uygulandı. Maven'in projeye katkısı, inşaat süreçlerini basitleştirmekti.

From <https://lms.clarusway.com/mod/lesson/view.php?id=5236>

As a project management tool, Apache Maven helps to build multiple projects easily, publish documentation for the projects, accomplish an easy deployment, share JARs across several other projects and help in collaboration with development teams.

From <https://lms.clarusway.com/mod/lesson/view.php?id=5236>

Maven can **manage a software project's builds with various versions, compile source code into binary, download dependencies, put additional JAR (Java Archive) files on a classpath, add documentation, run tests, package compiled code into deployable artifacts such as JAR, WAR, EAR and ZIP files, and deploy these artifacts to an application server or a repository**

From <https://lms.clarusway.com/mod/lesson/view.php?id=5236>

TURKISH

Maven, bir yazılım projesinin çeşitli sürümlerle derlemelerini yönetebilir, kaynak kodunu ikili olarak derleyebilir, bağımlılıkları indirebilir, bir sınıf yoluna ek JAR (Java Arşivi) dosyaları koyabilir, belgeler ekleyebilir, testler çalıştırabilir, derlenmiş kodu JAR, WAR gibi konuşlandırılabilir eserlere paketleyebilir, EAR ve ZIP dosyaları ve bu yapıtları bir uygulama sunucusuna veya bir havuza dağıtın

Also, Maven can automate these tasks and eliminate the risks of manual compiling. If we made all these tasks mentioned manually, we could have ended up with many problems. Consider a project that grows day by day. Some of your struggles might include the followings: Dependency management and adding so many JAR files into your project which is maybe the biggest problem and cumbersome task. For example, if we need some of the big and commonly used dependencies or frameworks, we should add sets of or hundreds of JAR files in each project. Building the right structure for your project is also an issue. You must put the right files into the right folders. Otherwise, you can't make your project work seamlessly. Also, the developers new to the project would need much more time to get used to the project structure. On the way to releasing the project, building and deploying would cause you a lot of trouble.

TURKISH

Ayrıca Maven bu görevleri otomatikleştirebilir ve manuel derleme risklerini ortadan kaldırabilir. Bahsedilen tüm bu görevleri manuel olarak yapsaydık, birçok sorunla karşılaşabilirdik. Her geçen gün büyüyen bir proje düşünün. Mücadelelerinizden bazıları aşağıdakileri içerebilir: Bağımlılık yönetimi ve projenize bu kadar çok JAR dosyası eklemek, belki de en büyük sorun ve hantal bir görevdir. Örneğin, bazı büyük ve yaygın olarak kullanılan bağımlılıklara veya çerçevelere ihtiyacımız varsa, her projeye bir dizi veya yüzlerce JAR dosyası eklemeliyiz. Projeniz için doğru yapıyı inşa etmek de bir sorundur. Doğru dosyaları doğru klasörlere koymalısınız. Aksi takdirde projenizin sorunsuz çalışmasını sağlayamazsınız. Ayrıca, projeye yeni başlayan geliştiricilerin proje yapısına alışmaları için çok daha fazla zamana ihtiyacı olacaktır. Projeyi yayınlama yolunda, inşa etmek ve dağıtmak size çok fazla sorun çıkarır.

Convention over Configuration

TURKISH

Konfigürasyon Üzerinden Konvansiyon

From <https://lms.cl
Maven adopts Convention over Configuration. This means that the developers should not deal with a building environment and should not do the build processes manually. They must not struggle with the smallest configuration item.

Maven handles the cumbersome and detailed processes for developers. It starts with a clean and well-defined project structure. It helps to define life-cycle goals within plugins (will be discussed later) and dependencies. Plugins, as part of the POM (Project Object Model) file, control the stages like compiling, building, testing, or packaging.

However, it's not necessary for a developer to fully understand how the plugins work because you can start a correctly structured and configured project with just a few clicks. There is only one critical thing that a developer should do. That is to use the directory structure and files in a proper way. Especially **the POM file is the most important file** for the overall health of the project.

From <https://lms.clarusway.com/mod/lesson/view.php?id=5236&pageid=5994>

arusway.com/mod/lesson/view.php?id=5236&pageid=5994>

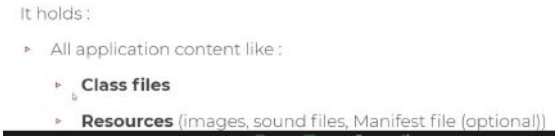
TURKISH

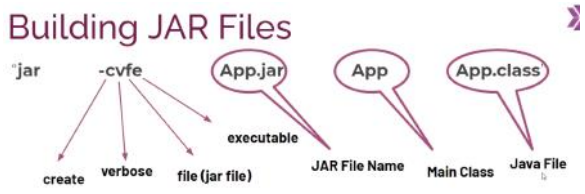
Maven, Konfigürasyon Üzerinden Sözleşmeyi benimser. Bu, geliştiricilerin bir bina ortamıyla ilgilenmemeleri ve inşa işlemlerini manuel olarak yapmamaları gerektiği anlamına gelir. En küçük yapılandırma öğesiyle mücadele etmemelidirler. Maven, geliştiriciler için hantal ve ayrıntılı süreçleri yönetir. Temiz ve iyi tanımlanmış bir proje yapısı ile başlar. Eklentiler (daha sonra tartışılacaktır) ve bağımlılıklar içindeki yaşam döngüsü hedeflerini tanımlamaya yardımcı olur. Eklentiler, POM (Proje Nesne Modeli) dosyasının bir parçası olarak derleme, oluşturma, test etme veya paketleme gibi aşamaları kontrol eder. Ancak, bir geliştiricinin eklentilerin nasıl çalıştığını tam olarak anlaması gerekli değildir, çünkü doğru yapılandırılmış ve yapılandırılmış bir projeyi yalnızca birkaç tıklamayla başlatabilirsiniz. Bir geliştiricinin yapması gereken tek bir kritik şey vardır. Yani dizin yapısını ve dosyalarını uygun şekilde kullanmaktır. Özellikle POM dosyası, projenin genel sağlığı için en önemli dosyadır.

- It's **easy to start** with Maven.
- You can start with a **variety of options** according to your needs.
- It has the **same structure** across a variety of different projects.
- It's **easy to integrate** into a developing team when they are working on Maven.
- It has a **powerful dependency management tool**.
- There is a **large repository of libraries**. You can find what you want among them.
- You have the chance to reach **extra features with plugins** in Java or scripting languages. You can also write plugins.
- Maven can give **different outputs** like a jar, ear, war, or metadata for the same project.
- Maven can **generate a website and a PDF** with the documentation in the project.
- Maven can **integrate with your source control system** such as CVS and manages the release of a project.
- Maven can **support the older versions**.

TURKISH

Maven ile başlamak kolaydır. İhtiyaçlarınıza göre çeşitli seçeneklerle başlayabilirsiniz. Çeşitli farklı projelerde aynı yapıya sahiptir. Maven üzerinde çalışırken gelişmekte olan bir takıma entegre etmek kolaydır. Güçlü bir bağımlılık yönetimi aracına sahiptir. Büyük bir kütüphane deposu var. Aralarında istediğinizi bulabilirsiniz. Java veya script dillerinde eklentiler ile ekstra özelliklere ulaşma şansınız var. Eklentiler de yazabilirsiniz. Maven aynı proje için jar, ear, war veya metadata gibi farklı çıktılar verebilir. Maven, projedeki belgelerle bir web sitesi ve bir PDF oluşturabilir. Maven, CVS gibi kaynak kontrol sisteminizle entegre olabilir ve bir projenin yayınlanmasını yönetebilir. Maven eski sürümleri destekleyebilir.





Java ile ilgili yazilmis dosyaları deployedtme bizim acimizdan önemli

jar ==> kavanoz demek ayni zamanda konservelemek gibi ihtiyac aninda cikartip yemek icin

From <https://app.slack.com/client/T0227UVRJU8/C021BG84YJJ/thread/C021BG84YJJ-1631209491.390800>

Jazva Run Time enviroment bilgisayarda java kullanmak icin gerekli paket

Maven ve Gradal ve Jaava ile en fazla kullanila Tool dur

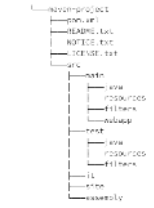
Birden fazla projeyo deploy et,meyi sagliyor ayrica dokumantasyon önemli oalzn bir tarafi

TURKISH

Maven ile başlamak kolaydır. İhtiyaçlarınıza göre çeşitli seçeneklerle başlayabilirsiniz. Çeşitli farklı projelerde aynı yapıya sahiptir. Maven üzerinde çalışırken gelişmekte olan bir takıma entegre etmek kolaydır. Güçlü bir bağımlılık yönetimi aracına sahiptir. Büyük bir kütüphane deposu var. Aralarında istediğinizi bulabilirsiniz. Java veya script dillerinde eklentiler ile ekstra özelliklere ulaşma şansınız var. Eklentiler de yazabilirsiniz. Maven aynı proje için jar, ear, war veya metadata gibi farklı çıktılar verebilir. Maven, projedeki belgelerle bir web sitesi ve bir PDF oluşturabilir. Maven, CVS gibi kaynak kontrol sisteminizle entegre olabilir ve bir projenin yayınlanmasını yönetebilir. Maven eski sürümleri destekleyebilir.

From <https://lms.clarusway.com/mod/lesson/view.php?id=5236&pageid=5995>

Directory Structure
Having a common project directory layout would bring the developers to "Oh, I'm at home." feeling in separate Maven projects.
The directory layout expected by Maven and the directory layout created by Maven is as in the picture below. So a maven project's directory structure should conform to this structure. But it's also possible to modify the structure.



Directory	Explanation
src/main/java	Application/Library source code
src/main/resources	Application/Library resources
src/main/filters	Resource filter files
src/main/webapp	Web application sources
src/test/java	Test source code
src/test/resources	Test resources
src/test/filters	Test resource filter files
src/it	Integration Tests (primarily for plugins)
src/assembly	Assembly descriptors
src/site	Site
LICENSE.txt	Project's license
NOTICE.txt	Notices and attributions required by libraries that the project depends on
README.txt	Project's readme

From <https://lms.clarusway.com/mod/lesson/view.php?id=5236&pageid=5995>

----- Lets go Handsone----

- **Firstly open the folder with the vscode**
- **And change the ket name in the maven-java-terraform.tf file**
- **After that got terminal in this file and write**
- **$ terraform init command**
- **Terraform apply**

After that connection ec2 with ssh

If you connect with ec2 in terraform use this code :

```
sudo yum update -y
sudo yum install java-11-amazon-corretto -y
cd /home/ec2-user/
wget https://ftp.itu.edu.tr/Mirror/Apache/maven/maven-3/3.6.3/binaries/apache-maven-3.6.3-bin.tar.gz
tar -zxvf $(ls | grep apache-maven-*-bin.tar.gz)
rm -rf $(ls | grep apache-maven-*-bin.tar.gz)
echo "M2_HOME=/home/ec2-user/$(ls | grep apache-maven)" >> /home/ec2-user/.bash_profile
echo 'PATH=$PATH:$M2_HOME/bin' >> /home/ec2-user/.bash_profile
echo 'export $PATH' >> /home/ec2-user/.bash_profile
```

Kisaca nasil calistigina bakarsak;

- **bir tane pipelinemiz olacak ve bu her zaman acik olacak building icin calisacak**
- **Sadece build ettikten sonra unit Testleri yapacak bunun surekli aktif olmasina gerek yok bu gunde bir kere calisacak aayaga kalkacak ve sonrasinda isi bitecek**
- **Bir tane piplene Functional Testler icin olacak**
  Bunlarin hepsi farkli veri tanbanlari ile cagrilacak

Profilleri biz olusturacagiz pom dosyalari hazirlandiktan sonra islemler basliyor

User data ile önce javakit i yukluyoruz daha sonra Maven i yukluyoruz ve bunu path in altina kopyaliyoruz

Hands-on Maven-01 : Using Maven As a Build Tool

Purpose of the this hands-on training is to teach the students how to use Maven with Java as a build tool.

- **Easy to integrate** into a developing team
- It has a **powerful dependency management tool**
- **Large repository** of libraries

Features of Maven

- **Extra features** with plugins
- **Different outputs** like a **jar**, **ear** or **war**
- Maven can **generate a website**
- Maven can **support the older versions**

Mevcut Configurasyona ekleme yapmayi ögrenmemiz gerekiyor maven icin

Directory Structure

- Project structure **should conform** to
- The most important file is the **pom file**
  - defines project's **config details**

```
maven-project
├── pom.xml
├── README.txt
├── NOTICE.txt
├── LICENSE.txt
└── src
    ├── main
    │   ├── java
    │   ├── resources
    │   ├── filters
    │   └── webapp
    ├── test
    │   ├── java
    │   ├── resources
    │   └── filters
    ├── it
    ├── site
    └── assembly
```

Introduction to POM File

Introduction to POM File

- It is an XML file
- **P**roject **O**bject **M**odel is the **starting point** for a Maven project
- It contains **configurations** about the project
- When a task or goal is executed, **Maven searches for** the **POM file**

Mave icin en önemli dosyalardan biridir

Introduction to POM File

POM defines

- Project **dependencies**
- Plugins and **goals** to be executed
- **Build profiles**
- Other information like the **project version**, **description**, **developers**, **mailing lists**, and more...

Introduction to POM File

There **must** be a POM file in every Maven project

All POMs need at least

- Project tag
- modelVersion tag
- groupId tag
- artifactId tag
- version (Last three called as **gav** in short)

```xml
<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>com.companyname.project-group</groupId>
   <artifactId>project</artifactId>
   <version>1.0</version>
</project>
```

Introduction to POM File

Group Id should be long enough to give **uniqueness** to the project

Artifact id is the id for specifying the project under the group

It shows the **name of the project** like pet-clinic-server

Version defines the version number of the project

Hands-on Maven-01 : Using Maven As a Build Tool

Purpose of the this hands-
on training is to teach the students how to use Maven with Java as a buil d tool.

#Learning Outcomes

At the end of the this hands-on training, students will be able to;
- install Maven and Java-11 on Amazon Linux 2 EC2 instance
- explain various build phases of a Java Application
- use Maven's clean, compile, package, install and site commands

#Outline

- Part 1 - Launch Amazon Linux 2 EC2 Instance with Cloudformation Temp late
- Part 2 - Generate a Java application using Maven's Archetype Plugin
- Part 3 - Run Maven Commands

#Part 1 - Launch Amazon Linux 2 EC2 Instance and Connect with SSH

- Launch an EC2 instance using ```maven-java-template.yml``` file located in this folder.
- This template will create an EC2 instance with Java-11 and Maven.
- Connect to your instance with SSH.
- Check if you can see the Maven's binary directory under ```/home/ec2-user/```.
- Run the command below to check if Java is available.

```bash
java -version
```

- Cat ```~/.bash_profile``` file to check if Maven's path is correctly transferr ed. If not paste the necessary lines manually.

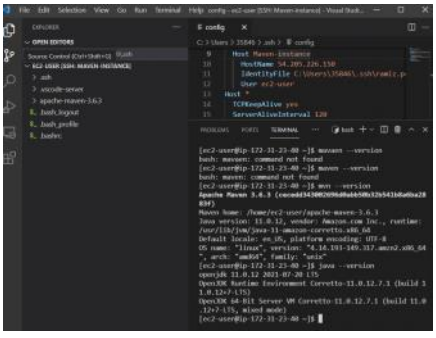
```
# Append the lines below into ~/.bash_profile file by making necessary ch anges.
M2_HOME=/home/ec2-user/<apache-maven-directory-with-its-version>
PATH=$PATH:$M2_HOME/bin
export $PATH
```

- Run the command below to check if mvn commands are available.

```bash
mvn --version
```

- If not, run the command below and wait for the EC2 machine to get read y.

```bash
sudo reboot
```

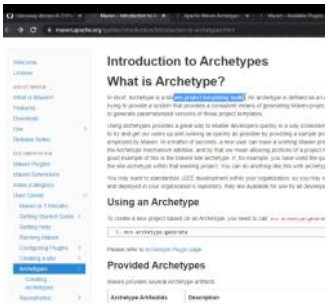


User data ile olusturdugumuz ec2 ya ssh ile baglaniyoruz

mvn --version # ---- yuklendigini görduk
Java --version # ayni sekilde versiyonunu görduk

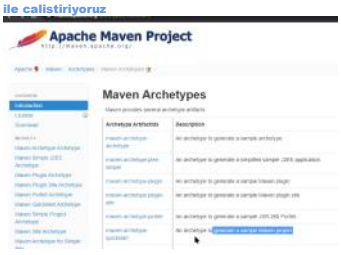
#Part 2 - Generate a Java application using Maven's Archetype Plugin

-



Architect Maven de bir template bir iskelet template kuruyor

Run the command below to produce an outline of a Java project with Ma ven.
Biz burda quick starti kullanacagiz bu artichetype altinda bulunan komut ile calistiriyoruz



it shows the **name of the project** like pet-clinic-server

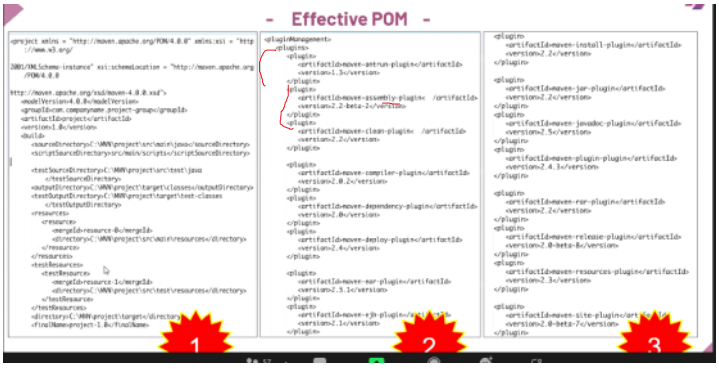
Version defines the version number of the project

Maven ile birlik te gepen dosyayay sSuper Pom diyoruz . Bizim kendi dosyamiz ile bir ararya gelince effective pom u oluturuyo r

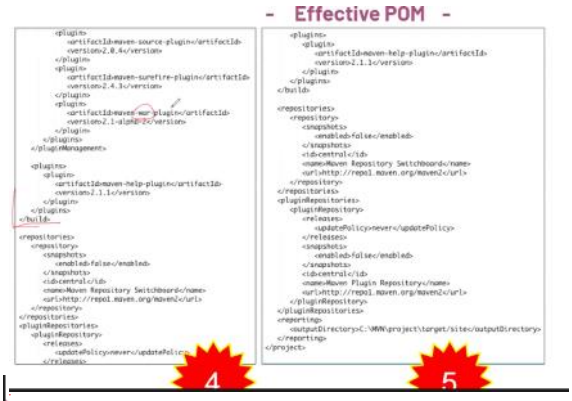
Super POM

- Super POM is Maven's **default POM**
- **All POMs extend** the Super POM **unless explicitly set**
- Super POM and project POM creates the **Effective POM**
- Which is the **overall configuration** file
- Effective POM can be examined by running

"**mvn help:effective-pom**"



Bizim asil ekleme yapacagimiz yere pluging kismi ortada ki kisim



3 Project Inheritance

Project Inheritance

- As in the object-oriented programming, POM files **can also be inherited** by other POM files
- Child POM can either **inherit** or **override**
- Parent POM is a **general template**
- **Not every item** in the parent is inherited
- Some elements should be declared specifically
- Like **artifactId**, **name**, and **prerequisites**

POM ile Bir projeyi inherite ederek getirebilirsiniz

Bi Maven Dosyasi inherit ederek baska dosyada kullanilabiliniyor

Inherit ; miras etmek

Project Inheritance

Parent POM's **packaging** tag should have the value "**pom**"

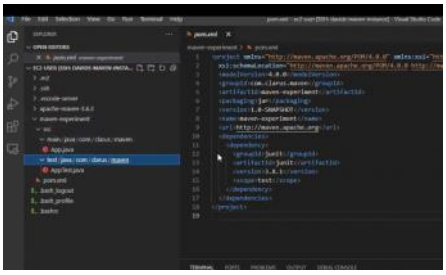
Parent

```
<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
                      https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>com.clarusway.mojo</groupId>
  <artifactId>my-parent</artifactId>
  <version>2.0</version>
  <packaging>pom</packaging>    <!-- NOTICE THE PACKAGING TYPE -->
</project>
```

Maven kullanmak icin bir template kuracagiz

```bash
mvn archetype:generate -DgroupId=com.clarus.maven
-DartifactId=maven-experiment -DarchetypeArtifactId=maven-archetype-
quickstart -DinteractiveMode=false
```

Calistirinca ekranimiza dosyaalarimiz eldi



- Cd into the project folder.
- Run the command below to be able to use tree command.

```bash
sudo yum install -y tree
```

- Run the command below to show the directory structure of the project.

```bash
tree
```

- Go into the folder where App.java resides and ```cat``` the auto generated ```hello world``` application in Java.
- Replace the content of the App.java file with the content of the App.java file in this repo.
- Go into the project's root folder.
- Replace the content of the ```pom.xml``` file with the content of the pom.xml file in this repo.
- Since we've install Java-11 on the EC2 machine, uncomment the ```properties``` tag of the new pom.xml file.
- Explain that the ```maven.compiler.source``` property specifies the version of source code accepted and the ```maven.compiler.target``` generates class files compatible with the specified version of JVM.
- Explain that ```dependencyManagement``` section in the pom file will import multiple dependencies with compatible versions.

Pom dosyamizi ve main App.java dosyamizi kendi repomuzdan kopyaladik

Java uygulamasinin ne yaptigina bakacak olursak öncelikler aws hesabimiza baglaniyor bir adet s3 bucket olusturuyor ve bunu buc kota atiyor mevcut bucketlarin listesini bize gösteriyor cikarken hem bucketi siliyor hem de dosyayi siliyor

Part 3 - Run Maven Commands

>### mvn compile

compile derlemek

- Run the command below.

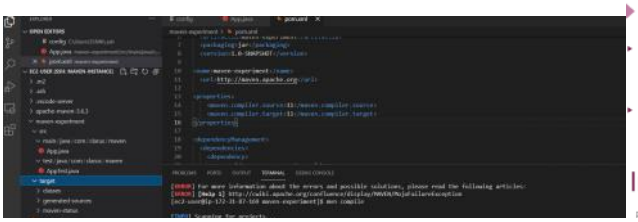
```bash
mvn compile
```

- Go into the folder ```<project-root>/target/classes/``` and show the class file.
- Run the command below to show how to test a Maven project.

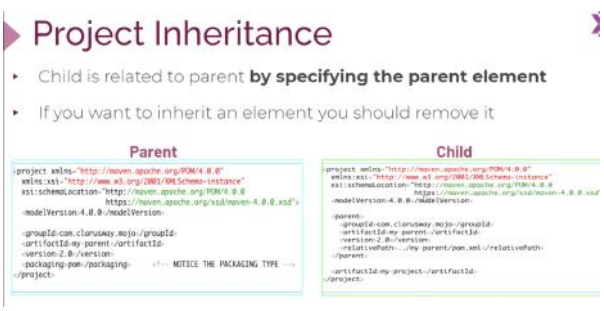
>### mvn clean test

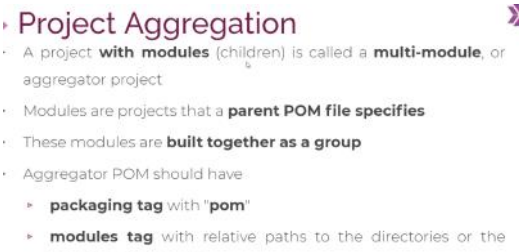
```bash
mvn clean test
```

- Show that there is a new folder named ```target``` in the project root.
- inspect the target folder with tree command.
- Show the content of the file ```<project-root>/target/surefire-report/com.clarus.maven.AppTest.txt``` as the output of the test.

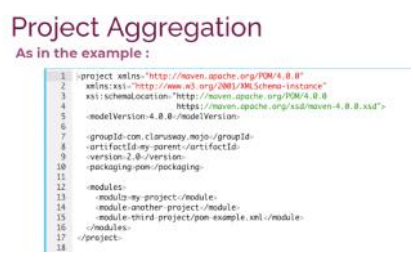


Packing kisminda Pom yaziyorsa inherit edilmis demek



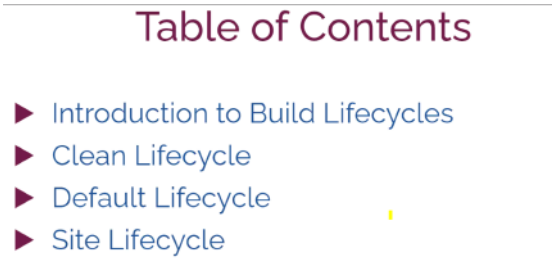


Pom dosyasina Pluginleri ekleyecehgzi

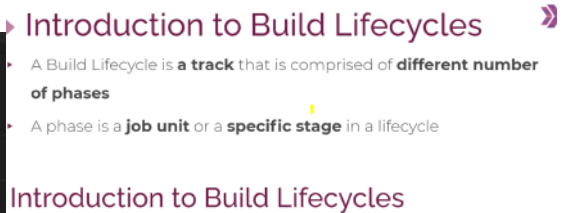


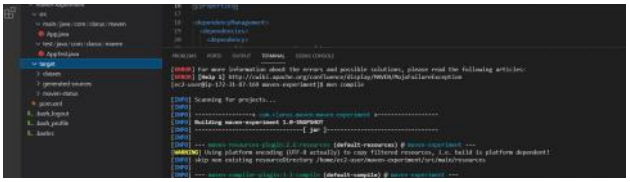
Mcroservice ozelligi yapisi gelistirmek icin daha avantaslidir

En ömemli konu Maven icin



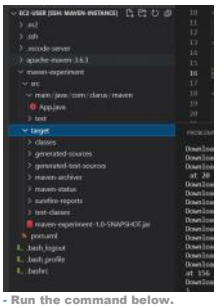
Maven calisirken belli basli asamalardan gecmesu gerekiyor toplam 3 asamasi var



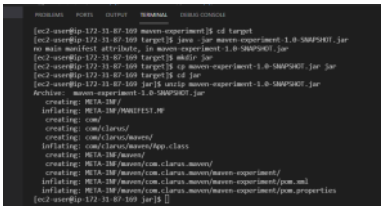


Target klasöru olustu bu klasör mvn in compile islemi sonrasi dosyalarini sakladigi klsör

>### mvn package # komutu ile target in altında . jar dosyasi olusturuyor



- Run the command below.
```bash
mvn clean package  # bir önceki klasörde olusturdumuz Target gurubu ve diger dosyalari siliyor
```
- Go into the folder ```<project-root>/target/``` and show the ```maven-experiment-1.0-SNAPSHOT.jar``` file as the output of the ```mvn package``` command.
- Run the command below to start the application.
```bash
java -jar maven-experiment-1.0-SNAPSHOT.jar
```



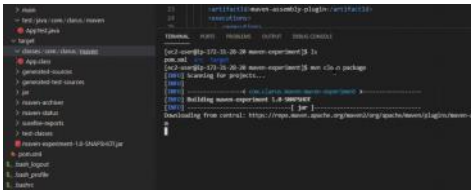
Ve jar dosyamizi olusturduk icerisne cd komutu ile girip unzip komutunu uyguladik
Jar bildigimiz bir zip dosyasidir.

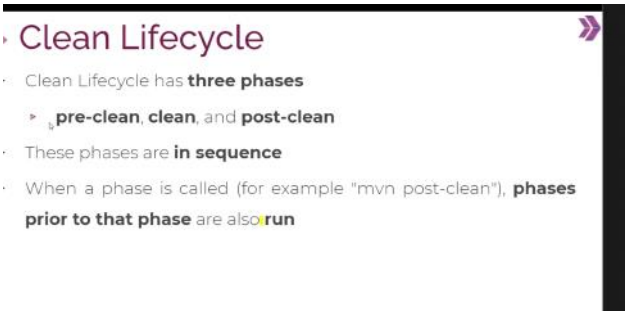
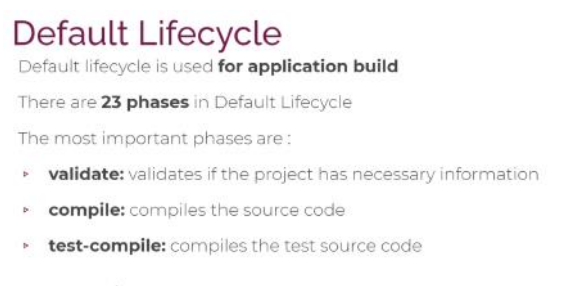
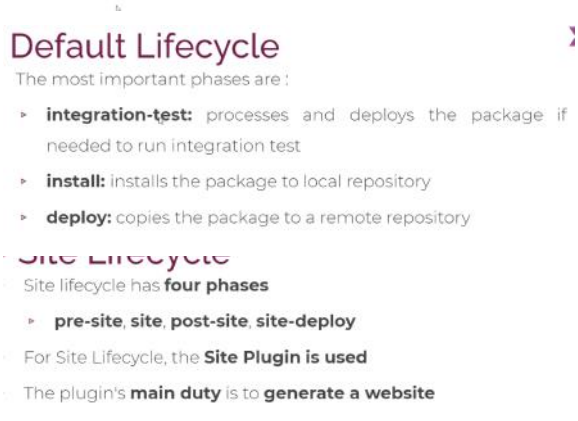
- Explain the error in the standard output.
  - Maven's jar file is not an executable jar file. The jar file does not have both the ```Main Class``` and the necessary packages to run the application.
- Add the plugin below to the pom file and run ```mvn clean package``` command again.

```xml
<build>
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-assembly-plugin</artifactId>
      <executions>
        <execution>
          <phase>package</phase>
          <goals>
            <goal>single</goal>
          </goals>
          <configuration>
            <archive>
            <manifest>
              <mainClass>
                com.clarus.maven.App
              </mainClass>
            </manifest>
            </archive>
            <descriptorRefs>
              <descriptorRef>jar-with-dependencies</descriptorRef>
            </descriptorRefs>
          </configuration>
        </execution>
      </executions>
    </plugin>
  </plugins>
</build>
```

```bash
mvn clean package
```

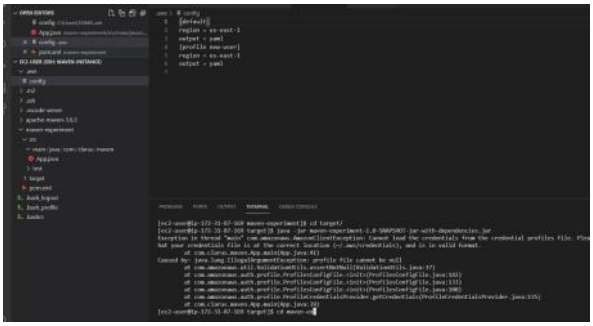
Bu komutu pom dosyamizin da icerisinde oldugu yerde calistiriyoruz



>### Run The Application
- Now, open up a fresh terminal on your local computer and run the command below.
```bash













**Lifecycle komutu calistirdigimizda art arda komutlar**


```

>### Run The Application
- Now, open up a fresh terminal on your local computer and run the comm and below.
```bash
scp -i <path-to-your-pem-file> -r <path-to-your-home-directory>/.aws ec2-user@<IP-of-your-instance>:/home/ec2-user
```
- Check if the the credentials are transferred to EC2 instance.
- Go into your target folder.
- Run the command below to start the application. This time we are runni ng the executable jar file with suffix ```jar-with-dependencies```.
```bash
java -jar maven-experiment-1.0-SNAPSHOT-jar-with-dependencies.jar
```
- Explain what the application does in the background.
    - Note that to be able see the object and the S3 bucket, we should com ment the lines 142 and 150.

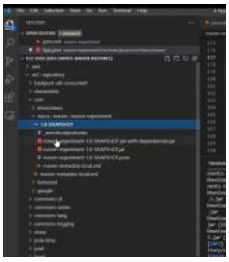
Uygulamayi calistirmaya calistik anvcak crediantal dosyamizi olmadigi icin hata verdi bizde .aws klasöru olusturduk ve icerisine crediantel imizi kopyaladik

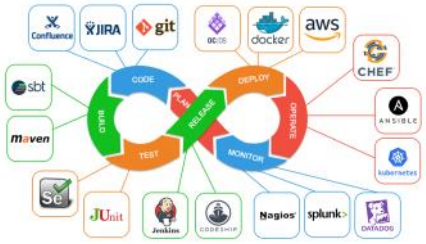


>### mvn install
- Run the command below to install our own package into .m2 folder.
```bash
mvn install
```
- Go into ```~/.m2/repository``` folder and show where our package is instal led.

Install etemek icin önce;
- **compile etmek lazim (mvn compile)**
- **Test (mvn clean test)**
- **built etmek lazim built etmeden önce**
- **Package (mvn clean package)**
- **En son install**

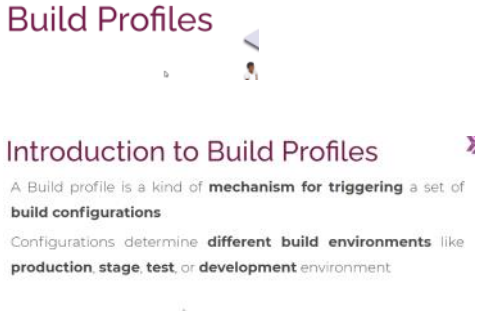
Instal komutu ile m2 dosyasi tam bir architec reposu olustu executable yani calistuirailabilir ve calsiacak butun dosyalar bir dosyanun icerisindedir.



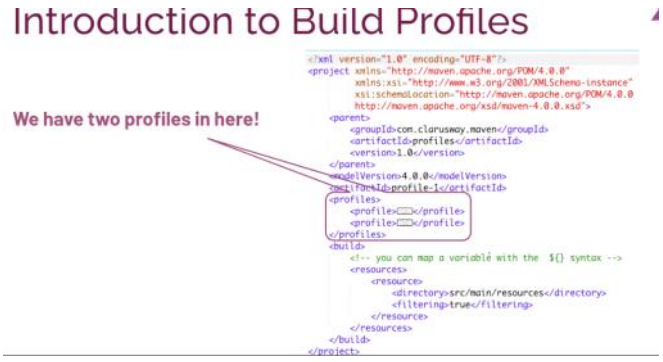


Built etmek farkli source kodlari compaile edip test edip calistirilabilir code haline getirilmesidir.

Örnegibn 6 Developer in hazirladigi projenin toplamda 12



Developerlar kodu yazdıktan sonra testerlara gşnderecekler



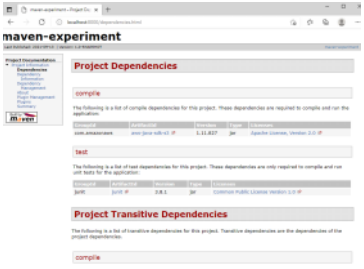
Bir developer profile olustururken hangi veritabanini kullanacaksa onu belirtir.

FARKLI
Pom dosyasi mave in ana kombinasyon kaynagidir. Efective Pom dosyasi kullaniliyordu Maven ile birlikte dedfault olarak gedlen super pom ve proje pom ile bir lesir effective pom u olusturur

>### mvn site
- Add two more plugins to run the command ```mvn site```
```xml
<plugin>
   <groupId>org.apache.maven.plugins</groupId>
   <artifactId>maven-site-plugin</artifactId>
   <version>3.7.1</version>
</plugin>
<plugin>
   <groupId>org.apache.maven.plugins</groupId>
   <artifactId>maven-project-info-reports-plugin</artifactId>
   <version>3.0.0</version>
</plugin>
```
- Run the command below.
```bash
mvn clean site
```
- Show the output ```site``` directory under target directory.
- Run the command below to install Apache Server.
```bash
sudo yum install -y httpd
sudo systemctl start httpd
sudo systemctl enable httpd
```
- Run the command below to copy the contents of the site folder under ```/var/www/html``` folder.
```bash
sudo cp -a site/. /var/www/html
```

Pom dosyamizda
Built in hemen altina 2 plugin ekledik

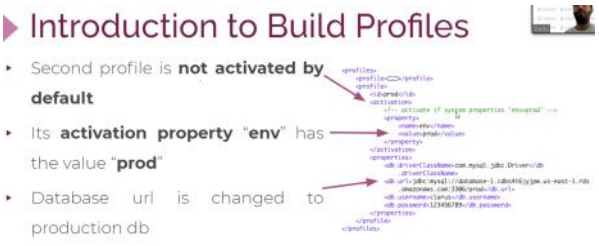
bir diger yol da calistirmanin
Python -mSimpleHTTPServer
Komutu ile sayfamizi göruntuledik



Ded Profilini genelde development icin kullanilir Testerlar lokalde RRDS gibi remote da calisirlar

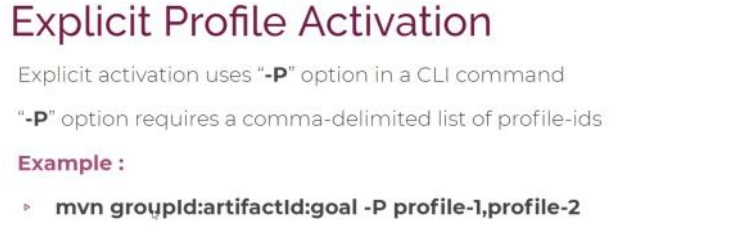
Developerlar kendi testini yapar ken debug in kulalnir

Ayni built konfegerasyonunda farkli profiller olusturulabilir

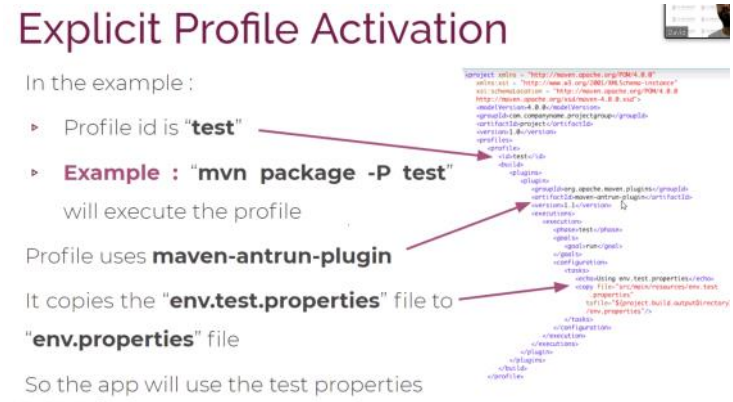


Default build development olandır digerlerını secek ısek onları action da belirtmemız gerekıyor

Profl aktıve temenın 5 ayrı yşntemı hangi fase kullaniyorsak onu yazacazgiz



Bu komutta demek istedigimiz sudur; groupId ile baszlayan faze aktiviide ederken profile 1 ve profile 2 yi kullan berdesn faz la oprofile calistir demektir bu komutun anlami



mvn package -P + sonra hangi enviroment calismasini istiyorsak yazacagiz birden fazla olabilir mvn alttaki test calistiriyor

Maven bu pom dosyasinda ki islemleri yaparke test isimlinprofili calistiracaktir. Calistiriken Test plugungin özellilerini di kkate aliyor

Test profilinde kullandigi plugini ikinkci ok isareti ile görebiliyoruz compile etmek icin building etmek icin kullanilan bi r seydir

Echo ile stringe yazdiriyoruz. Program ayaga kalkarken run edip built ederken properties icerigini degistiriyoruz.

Projeye baslarken Developerlar Pom dosyasini hazirlarlar Dedveloperlar pluginler belirlendikten sonra pom dosyasini hazirlarl ar

Dev-Ops icin hazirlana dökumantASYONU IYI OKUYUP ONA GLRE CALISZMA K COK ÖNEMLIDIR.

En az 3 adet profile hazirlanir

- Production

- **Development**
- **Test**

Bynlar 3 ayri Alanda calismalarini yaparlar farkli alanlar icin farkli profile olusturma sorumlulugu bizimdir. Test

Alfa Test sirketin kenndi bunyesinde apilir

Beta Test de disarida acik olara insanlara davet gönderirisiniz sizin serverinuza gierme yetksi verirsiniz log kayitlarini to plar developerlara gönderirisiniz.

Testenviroment
Productionenviriment
Staging enbviroment

Hepsi b irbirinden farklidir her enviroment icin farkli bir profile belirleyip buinlar icinde dökumantasyona bakip

Profile Activation via Maven Settings

When you install Maven, a directory named "**.m2**" is created **under** your **Home Directory**

"**settings.xml**" (user settings) is located in that directory

If it's not there, you can create one

To activate a profile with settings.xml, **profile id** should be declared under **<activeProfile>** tag

```
<settings 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/settings-1.0.0.xsd">
  <mirrors>
    <mirror>
      <id>maven.dev.snapongloal.com</id>
      <name>Internal Artifactory Maven repository</name>
      <url>http://repo1.maven.org/maven2/</url>
      <mirrorOf>*</mirrorOf>
    </mirror>
  </mirrors>
  .
  .
  <activeProfiles>
    <activeProfile>test</activeProfile>
  </activeProfiles>
</settings>
```

Developerlar .m2 isimli bir repo oluturu ve bu dosyanin icerisine settins.xml dosyasi olusturraarak olusturmak istedikleri p rofil leri buraya kaydedebilirler. Bu sekilde olursa ayri olarak cagrilmasina gerek kalmaz

Java ile geliştirme yaptığım her projede 3 farklı profilim olur benim. **dev , prep ve prod** diye..

- **dev : Development(geliştirme yaptığım)** ortamını ifade eder
- **prep : preProduction** dan gelir. **Test** ortamını ifade eder.
- **prod : production** ortamını ifade eder.

Peki neden 3 farklı profil size kısaca anlatayım. Öncelikle genel olarak şirketlerde kodları geliştirdiğiniz bir makine oluyor(Bu size verilen laptop). Geliştirme işlemleri bitince sizden geliştirme isteyenlerin kodlarınızı görebilmesi ve test uzmanlarının sizin geliştirdiğiniz kodu test edebilmesi için bu projeye ait bir test sunucusu oluyor. Birde testlerden onay alındıktan sonra amacına hizmet etmek üzere kodları attığınız gerçek(Erişime açılan) sunucular...
Bir projenin geliştirilip yayına çıkması bu 3 aşamadan geçerek gerçekleşir. Dolayısıyla siz geliştirme yaparken localde kullandığınız **veri tabanı** bilgileri ayrı, testte kullandıklarınız ayrı ve prodda kullandıklarınız ayrı olacaktır. Bu farklılıklara istinaden projenizi **deploy** ederken ya her sunucunun bilgilerini elle güncelleyip, **compile** edip, **deploy** edeceksiniz, yada birazdan göstereceğim şekilde **maven ile profil** oluşturarak bu üç farklı sunucu için üç farklı ayar dosyası oluşturacaksınız.
Ve **compile** ederken **profil** seçeceksiniz.
Kısacası ; localde çalışırken maven'a kodlarını **dev profiline göre compile et,** teste giderken **prep'e göre** ve son olarak production a çıkarken **prod'a göre compile et** diyeceksiniz..

From <http://www.leventyildiz.com.tr/2014/09/maven-profil-olusturma-dev-preprod-prod.html>

NOT : Bu yazıyı sadece 4 satırlık veri tabanı ayar dosyaları üzerinden örneklendiriyorum ancak gerçek ortamlarda kod yazarken bu ayar dosyalarının farklılıkları onlarca satırı bulur ve eğer profil kullanmazsanız her deploy etmede(dev+test+prod) bu onlarca satır kodu ilgili sunucuya göre değiştirmek durumunda kalırsınız.
şimdi örneklendirmeye geçiyorum.Geliştirme yaptığımız ortamda(development) bir MySQL veri tabanımız olsun erişim bilgileri aşağıdaki gibi olsun

From <http://www.leventyildiz.com.tr/2014/09/maven-profil-olusturma-dev-preprod-prod.html>

Profile Activation via System Variables

Activation occurs when **system property** is specified with **-D** option

```
<profiles>
  <profile>
    <activation>
      <property>
        <name>debug</name>
      </property>
    </activation>
    .
    .
    .
  </profile>
</profiles>
```

mvn groupId:artifactId:goal -Ddebug

mvn groupId:artifactId:goal

```
<profiles>
  <profile>
    <activation>
      <property>
        <name>!debug</name>
      </property>
    </activation>
    .
    .
    .
  </profile>
</profiles>
```

Maven de Aktivide edilmesi gereken dosya farkli profilde active edilir farkli asamalarda aktivide edilier Örnegin test asamas in da farkli production asamasinda farkli.

Her halukarda belirtmedigim surette calismasin diyorsak !debug diyebiliriz

Activation occurs when **system property** is specified with **-D** option

```
<profiles>
  <profile>
    <activation>
      <property>
        <name>debug</name>
        <value>!true</value>
      </property>
    </activation>
    .
    .
    .
  </profile>
</profiles>
```

mvn groupId:artifactId:goal -Ddebug=false

mvn groupId:artifactId:goal -Denvironment=test

```
<profiles>
  <profile>
    <activation>
      <property>
        <name>environment</name>
        <value>test</value>
      </property>
    </activation>
    .
    .
    .
  </profile>
</profiles>
```

LARUSWAY©

5. olarak Key Value degerleri calistirabiliriz. Biz -D ile valulari cagirabiliriz ve bu sekilde calistirabiliriz.

Profile Activation via Operating System

It is defined under **<os>** tag

In the example **Windows XP** will **trigger** the profile

```
<profile>
    <id>test</id>
    <activation>
        <os>
            <name>Windows XP</name>
            <family>Windows</family>
            <arch>x86</arch>
            <version>5.1.2600</version>
        </os>
    </activation>
</profile>
```

6 olarak isletim sistemine göre built edilen isletim sistemi ile calistirmaktir. Burdaki husus maven i builtettigimiz system Windows ise bu profildeki huusulari uygula demek istiyoruz.

Profile Activation via Present/Missing File

In the example profile is triggered when **the file**

Profile Activation via Present/Missing File

In the example, profile is triggered when **the file** target/generated-sources/axistools/wsdl2java/org/apache/maven is **missing**

```
<profiles>
  <profile>
    <activation>
      <file>
        <missing>target/generated-sources/axistools/wsdl2java/org/apache
                 /maven</missing>
      </file>
    </activation>
    ...
  </profile>
</profiles>
```

APLICWAY©

7 Activasyon ise ; uygulamayi calisirken ihtiyaclari olan dosya missing ise bu profile ile calistir diye biliyoruz.

Önemli olan kisim neyi ariyabilecegimizi bilip internetten arayip bulmak ve

Her asama icin farkli pom dosySI HAZIRLAMAMK ICIN ay

Her safha icin ayri profilleri konuluyor

Her uygulamamnin bir debuging var developerlar sadece görebilir

Farkli safhalar icin farkli profiller olusturulur.

Bir urun hazirlanirken ilk olarak development safhasi olur sonrasinda staging safhasi daha son Testin safhasi ve son olazrak Producting asamasi olur

Arka tarafinda veri tabanlari var ise Her asamada farkli veri tabanlari kullanilir Uygulama ayni uygulama . Ne zaman hangi ve ri tabaninin kullanilacagini Maven in Built Configurasyonunda belirliyoruuz. Pom Dosyasinda belirleniyor.
Farkli veri tabanlarinin nasil calisacagini pom dosyasina koydugumuz profiller sile belirliyoruz.

Bizim ana isimiz dökumasyonda gecen konulari built codlarina cevirmek

Örnegon chrom da farkli mailler icin farkli gmaillerin cagrilmasi gibi

Dev-Ops adi ustunde otomasyon ile hayati kolaylastirma amaclidir.

http://www.leventyildiz.com.tr/2014/09/maven-profil-olusturma-dev-preprod-prod.html

Testing Publine calisirken

Nasil aktivie edilir

Aktivasyonu tekrar edersek ilk olarak active by default True dersek aktivasyon hicbir sey yapmasak da sadece maven komutu ca lisinca calisacaktir.

First profile is **activated by default** **unless** another profile in the same POM is activated

Explicit Profile Activation

Explicit activation uses "**-P**" option in a CLI command

"**-P**" option requires a comma-delimited list of profile-ids

Example :

- **mvn groupId:artifactId:goal -P profile-1,profile-2**

mnv komutu ile ile calistiriyoruz .

Explicit Profile Activation

In the example :

- Profile id is "**test**"
- **Example : "mvn package -P test"** will execute the profile

Profile uses **maven-antrun-plugin**

It copies the "**env.test.properties**" file to "**env.properties**" file

So the app will use the test properties

APLICWAY©

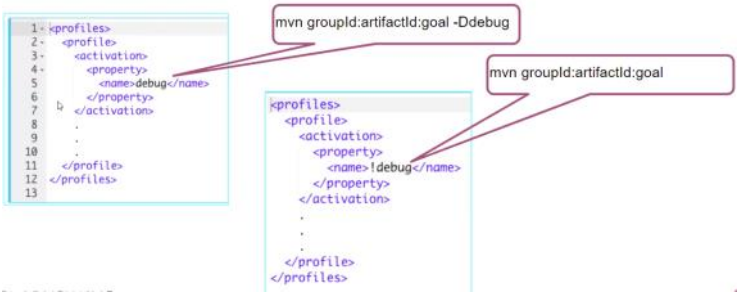
Profile Activation via Maven Settings

- When you install Maven, a directory named "**.m2**" is created **under** your **Home Directory**
- "**settings.xml**" (user settings) is located in that directory
- If it's not there, you can create one
- To activate a profile with settings.xml, **profile id** should be declared under **<activeProfile>** tag
- **No need to trigger** the profile

Calistirildiktan sonra .m2 dosyasi olusur

Profile Activation via System Variables

- Activation occurs when **system property** is specified with **-D** option



Ucuncu bir özellik

Profile Activation via System Variables

- Activation occurs when **system property** is specified with **-D** optior

Bir sonraki özellik key valu ile aktive ediliyor

Profile Activation via Operating System

It is defined under **<os>** tag

In the example **Windows XP** will **trigger** the profile

Isletim sistemine göre aktivite edilmesi

Profile Activation via Present/Missing File

In the example, profile is triggered when **the file** target/generated-sources/axistools/wsdl2java/org/apache/maven is **missing**

Introduction to Repositories

- Repository is a source where all library **jars**, **plugins**, **dependencies**, or any other project-specific **artifacts** are stored
- While your project runs, **these resources** are **used silently**
- There are **two types** of repositories
  - **Local** and **remote**
- Local repository is **your own computer**

Internet baglantisi olmadan calisabiliyor. Burda bir architeck reposu gibi dusunebiliriz yani uygulamanin calismasi icin gere kli olan her seyin oldu repostory diyebiliiriz

Resource lar var ise onlari alip reponiun icerisine atiyor

Introduction to Repositories

- **Remote** repository can be **separated into two**
  - **Central** repository and **Third-Party** repository
- **By default central** repository **is used** as the remote repository
- You can also configure to use a third-party repository

Remote repo 2 ye ayriliyor

Local Repository

As mentioned, local repository is **in your local computer**

Maven creates this directory

It **continuously develops** it whenever you use a resource from a remote repository

.M2 KLASÖRUDUR ihtiyaci olan dosyalari internetten indirebiliyor
Loakal olan repo silindigi taktirde kendiliginden internet baglantisi ile yeniden olusacaktir

Local Repository

Tip: In general, you should not need to do anything with the local repository on a regular basis, except clean it out (~/.m2 directory) if you are short on disk space (or erase it completely if you are willing to download everything again).

Maven in kendi reposu

Central Repository

- Maven central repository is the **default remote repository**
- When Maven **cannot find a dependency** in the local repository, it tries to find it in the central repository
- Central repo is **located in** this url **https://repo.maven.apache.org/maven2/**
- **No configuration** is needed to use the central repo

Third-Party Repository

Central repository is not the only choice

Any organization or **any individual** can host a remote repository

You need to configure it in the POM file

Third-Party Repository

In the example, third-party repositories are specified under **<repositories>** and **<repository>** fields

```
<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>com.companyname.projectgroup</groupId>
  <artifactId>project</artifactId>
  <version>1.0</version>
  <dependencies>
     <dependency>
        <groupId>com.companyname.common-lib</groupId>
        <artifactId>common-lib</artifactId>
        <version>1.0.0</version>
     </dependency>
  </dependencies>
  <repositories>
     <repository>
        <id>companyname.lib1</id>
        <url>http://download.companyname.org/maven2/lib1</url>
     </repository>
     <repository>
        <id>companyname.lib2</id>
        <url>http://download.companyname.org/maven2/lib2</url>
     </repository>
  </repositories>
</project>
```

Repo genel olarak; Dependacy leri ndirdigimiz ve dosyalarimizi sakladigimiz yer

Github da sadece source dosyaalari bulunurken ; bu reponun farkli ihtiyac duydugumuz dosyalarinda icerisinde bulundugu archit ectuure depodur

What is a Plugin?

Plugin is the **heart of Maven** framework

A **unit work** in Maven or a **single output** is produced by a specific Maven Plugin

Some of the plugins are **bound to** some of the **phases** of Maven Build Lifecycles

But **some** are **independent**

Plugin temel seylerin yaninda ileri seviyede islemler yapiyorlar; Temel islemler

Paketleme, Compail etme, test etme, bunun yani sira reporting islemler clean islemi Plugingler araciligiyla yapilir

Project management TOOL

Plugins do the works like **creating jar** files, **war** files, **compiling code**, **compiling unit test** code, creating **project documentation** or **JavaDoc** (Java Documentation), and so on

One of the **simplest plugins** in Maven is the **clean plugin**

Maven Clean Plugin is responsible for **removing the target directory** of a Maven project

When you run **mvn clean**, Maven executes the **clean goal** as defined **in** the **clean plug-in**

Goals in Maven can be executed via the command-line interface within the format specified below :

- **mvn [plugin-name]:[goal-name]**

Her plugin de mutlaka bir group id birde artifactid ve version olmak zorundadir

If you want to run **both the clean phase** and compiler plugin's **compile goal**, you should run the command

- **mvn clean compiler:compile**

All plugins should have the **minimum requirement** of having the **groupId**, **artifactId**, and **version** elements

Maven'ı bir **plugin** çalıştırma framework'u olarak tanımlayabiliriz. **Maven** yaptığı bütün işleri pluginler aracılığıyla yapar. Kaynak kodun derlenmesi, test kodlarının çalıştırılması, projenin paketlenip bir jar dosyası haline getirilmesi vb. pluginler aracılığıyla ger çekleşir.

From <https://www.google.com/search?q=plugin+nedir+maven+&sxsrf=AOaemvJBcmv75G8C_-m2ZI_PHEN9IRTwbQg%3A1631352325242&ei=BXY8YfuYDp2Xxc8PuNK_2As&oq=plugin+nedir+maven+&gs_lcp=Cgdnd3Mtd2l6EAMyBggAEBYQHjIGCAAQFhAeOgcIABBHELADOgcIABCwAxBDOg0ILhDHARDRAxCwAxBDOggIABDkAhC_wAzoFCAAQgARKBQg8EgExSgQIQRgBUOU6WN9JYJ9OaAFwAHgAgAGaAYgB-wWSAQM0LjOYAQCgAQHIAQ3AAQE&sclient=gws-wiz&ved=0ahUKEwi7r-X-zPbyAhWdS_EDHTjpD7sQ4dUDCA4&uact=5>

Iki tur var ; build plugingler buildingin altinda, reportingler ler reportinglerin altinda yer alir.

Types of Plugins

- There are **two types** of plugins :
  - **Build** Plugins and **Reporting** Plugins
- Build plugins are configured under **<build>** tag
- They **run** during the **build time**

Types of Plugins

Reporting Plugins are configured under **<reporting>** tag

They run while you are **generating the site** for the project

Maven plugins are configured by specifying a **<configuration>** element

Maven Plugin Nedir ?

Murat Gülcü6 Şubat 2021

- Maven'ın temelinde pluginler vardır. Maven'ı bir plugin çalıştırma framework'u olarak tanımlayabiliriz.
- Maven yaptığı bütün işleri pluginler aracılığıyla yapar. Kaynak kodun derlenmesi, test kodlarının çalıştırılması, projenin paketlenip bir jar dosyası haline getirilmesi vb. pluginler aracılığıyla gerçekleşir.
- Pluginler bir veya daha fazla işin gruplandığı yapılardır. Yapılacak işi belirten yapılara "**goal**" denir. Plugin goal'lerin bir araya gelmesinden oluşmaktadır.

From <http://muratgulcu.com/java/maven-plugin-nedir/>