## What is a Dependency?

Whenever a class A uses another class or interface B, then A depends on B. A cannot carry out it's work without B, and A cannot be reused without also reusing B.

## Maven’s Objectives

Maven’s primary goal is to allow a developer to comprehend the complete state of a development effort in the shortest period of time. In order to attain this goal, Maven deals with several areas of concern:

* Making the build process easy
* Providing a uniform build system
* Providing quality project information
* Encouraging better development practices

### Making the build process easy

While using Maven doesn’t eliminate the need to know about the underlying mechanisms, Maven does shield developers from many details.

### Providing a uniform build system

Maven builds a project using its project object model (POM) and a set of plugins. Once you familiarize yourself with one Maven project, you know how all Maven projects build. This saves time when navigating many projects.

### Providing quality project information

Maven provides useful project information that is in part taken from your POM and in part generated from your project’s sources. For example, Maven can provide:

* Change log created directly from source control
* Cross referenced sources
* Mailing lists managed by the project
* Dependencies used by the project
* Unit test reports including coverage

Third party code analysis products also provide Maven plugins that add their reports to the standard information given by Maven.

### Providing guidelines for best practices development

Maven aims to gather current principles for best practices development and make it easy to guide a project in that direction.

For example, specification, execution, and reporting of unit tests are part of the normal build cycle using Maven. Current unit testing best practices were used as guidelines:

* Keeping test source code in a separate, but parallel source tree
* Using test case naming conventions to locate and execute tests
* Having test cases setup their environment instead of customizing the build for test preparation

Maven also assists in project workflow such as release and issue management.

Maven also suggests some guidelines on how to layout your project’s directory structure. Once you learn the layout, you can easily navigate other projects that use Maven.

While takes an opinionated approach to project layout, some projects may not fit with this structure for historical reasons. While Maven is designed to be flexible to the needs of different projects, it cannot cater to every situation without compromising its objectives.

If your project has an unusual build structure that cannot be reorganized, you may have to forgo some features or the use of Maven altogether.

**Maven’in Amaçları Nelerdir ?**

* Projenin geliştirme adımlarını kolaylaştırmak
* Standartlaştırma
* Projenin niteliklerini bilgi olarak sağlamak
* Tekrar kullanılabilirlik
* Kararlılık
* Bağımlılık yönetimi
* Ölçeklenebilirlik

**Ne işe yarar bu Maven?**

**Maven**, proje geliştirirken proje içerisinde bir standart oluşturmamızı, geliştirme sürecini basitleştirmemizi, dokümantasyonumuzu etkili bir şekilde oluşturmamızı, projemizdeki kütüphane bağımlılığını ve IDE bağımlılığını ortadan kaldırmamızı sağlayan bir araçtır.

**POM Nedir ?**

**Proje Nesnesi Modeli (Project Object Model) veya POM**, Maven’deki temel iş birimidir. Proje kurmak için Maven tarafından kullanılan proje ve yapılandırma detayları hakkında bilgi içeren bir XML dosyasıdır.

**Maven nedir?**

Maven özetle Apache tarafından geliştirilmiş, yazılım projelerinizin geliştirilmesi sırasında projenin geliştirilmesini kolaylaştıran, basitleştiren ve proje bağımlılıklarının kontrolünü yapmayı sağlayan bir araçtır. Maven projedeki bağımlılıkların(dependency) yönetimi dışında IDE(Integrated Development Environment) bağımsız yazılım geliştirmeyi de amaçlar.

Maven, projenizi geliştirirken projenin birden çok geliştirme ortamına uyumlu olmasını sağlar. Yani projenizin belli standartlara uygun geliştirilmesini ister. Bu sayede A yazılımcısı ile B yazılımcısı farklı bilgi düzeylerinde olsalar bile aynı standartlarda, aynı yapı içerisinde kodu geliştirirler. Projeyi Eclipse’de ya da Netbeans’de yazmanız bir şeyi değiştirmez(en azından teoride). Peki bunu yapmayı kolaylaştıran ne? Cevap: [Standart Directory Structure](https://maven.apache.org/guides/introduction/introduction-to-the-standard-directory-layout.html). Bu yapı sayesinde geliştirdiğiniz projenin kaynak kodlarına bakan diğer bir yazılımcı neyin nerede olduğunu kolaylıkla bulabilir.

Şimdi Maven’ın asıl önemli olduğu noktalara gelelim. Java’da kütüphanelerin JAR(Java Archive) dosyaları ile projeye dahil edildiğini biliyor olmalısınız. JAR, aynı zip gibi çalışan ancak içerisinde genellikle derlenmiş Java sınıfları ve kütüphane ek dosyalarını barındıran arşiv formatıdır. Maven gibi bağımlılık(dependency) yönetim araçları yokken geliştiriciler projelerine başka bir framework ya da kütüphaneyi eklemek istediğinde o projenin sitesinden JAR dosyalarını indirip kendi projelerine ekliyorlardı. Projeye Hibernate eklemek istiyorsanız Hibernate sitesine girip JAR dosyalarını indirip projeye ekliyordunuz. Ancak projede kullanacağınız dependency sayısı artınca bu durum içinden çıkılmaz bir hal alıyor. Her eklenecek bağımlılık için JAR dosyalarını elle eklemek geliştirici için büyük yük. İşte Maven burada devreye giriyor. Maven tüm bunları sizin için yapıyor. Üstelik bu eklediğiniz bağımlılıkların bağımlı olduğu başka dependencyler varsa onları da sizin için projenize dahil ediyor. Mükemmel dimi?