MAVEN 2.X

Mustafa Sait Özen
Elektrik-Elektronik Mühendisi
Anadolu Üniversitesi
Bilgisayar Araştırma ve Uygulama Merkezi
msaitozen@gmail.com

İÇERİK

1. MAVEN NEDİR?

Maven, Java projelerini yönetmek için kullanılan bir araçtır. Proje gelişim süreci geniş bir alanda basitleştirilerek gerçekleştirilmektedir. Maven hakkında her ne kadar site ve dokümantasyon aracı ya da Ant için bir soyutlama katmanı sunuyor olması gibi tanımlamalar yapılıyor olsa da Maven daha fazlasını yapabilme özelliğine sahiptir.

Maven 1'den sonra Maven 2'nin geliştirilmesinin temel amacı daha güçlü bir java yapılandırması, kaliteli rapor bilgisi, gelişmiş araçlar ve daha kapsamlı bir yaşam döngüsü (Built-Life-Cycle)'ne sahip olmasıdır.

NOT: Maven bir Java aracı olduğu için Java'nın kurulu olması gerekir.

Maven'la ilgili bir foruma dahil olmanız konu ile ilgili sorunlarınızı çözmeniz için bir kaynak teşkil edecektir. Forum'a kayıt işlemleri hakkındaki bilgi için Geliştirici Mail Listesi konu başlığını okuyunuz.

1.1. Maven ve Ant

Ant'ı C gibi modüler bir dil olarak düşünürsek Maven da C++ gibi nesneye yönelik bir dil olarak kıyaslanabilmektedir.

Java'da iyi bir web uygulaması geliştirirken derle, paketle, kopyala, deploy et, cache'leri sil, sunucuyu tekrar başlat gibi sürekli tekrarlanan işlemler Ant ile birlikte bir *.xml* dosyasında sırası ile çalıştırılacak şekilde tanımlanmış. Bu kolaylık Maven ile de sağlanmış, işlem adımları bir *.xml* dosyası ile sırası ile tanımlanmıştır ve otomatik olarak gerçekleştirilmektedir.

1.2. Neden Maven ?

- Etkin bir sistem yönetimi sunmaktadır.
- Projenin bir düzen içerisinde geliştirilmesini sağlar.
- Yapılandırma işlemi daha kolaydır.
- Kaliteli bir proje bilgisi sağlar.
- Tam ve dikkatli test etme pratiğini kazandırır.
- Projeler için kalıtım olanağı vermektedir.
- Yeni niteliklere saydam bir geçiş sağlar.

NOT: **Lütfen sürümü açık bir şekilde aşağıda belirtilmiş olan programın x olarak belirtilen kısmını olması gereken sürümü yazarak güncelleyiniz.**

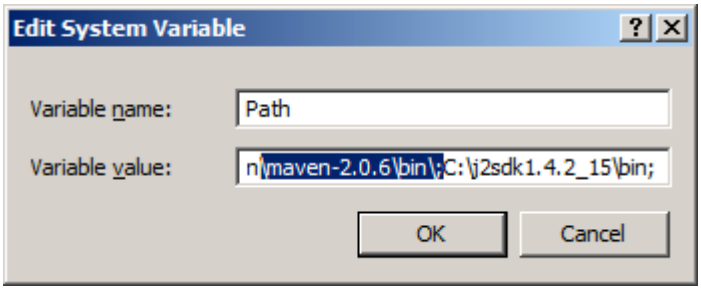
2. Maven'ın Kurulumu ?

Maven'ın en son sürümüne bu linkten ulaşabilirsiniz : http://maven.apache.org/download.html. Kurulum işlemleri sistemdeki mevcut işletim sistemine göre farklılıklar taşımaktadır. İndirilen dosyayı **C: /maven** içerisine açınız.

2.1. Windows XP için…

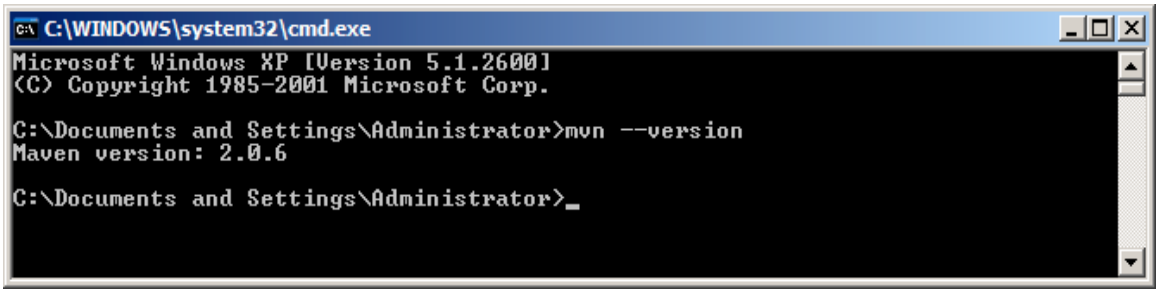
- **1. Adım** : http://maven.apache.org/download.html adresinden Maven 'in son sürümüne ulaşabilirsiniz.
- **2. Adım** : C:\Program Files\Apache Software Foundation\maven-2.0.6 olarak belirtilen yere kaydedilir.
- **3. Adım** : Maven için bir path ayarı yapmanız gerekmektedir.
  (Windows Tuşu + Pause) → Advanced → Environment Variables

Path'e eklenecek olan yol : C:\Program Files\Apache Software Foundation\maven-2.0.6\bin\;



Şekil 2-1.1 Path belirleme.

- **4. Adım** : **mvn --version** ya da **mvn –v** (İlgili kısa yollara **?** işareti ile ulaşabilirsiniz. Örneğin: mvn -?) komutu komut satırında yazılarak versiyon görülebilir. Kurulumun başarı ile gerçekleşip gerçekleşmediğini kontrol etmek için **mvn --version** komutunu komut satırında yazarak **Şekil 2.1-2** 'deki örneğe benzer bir çıktı almalısınız.

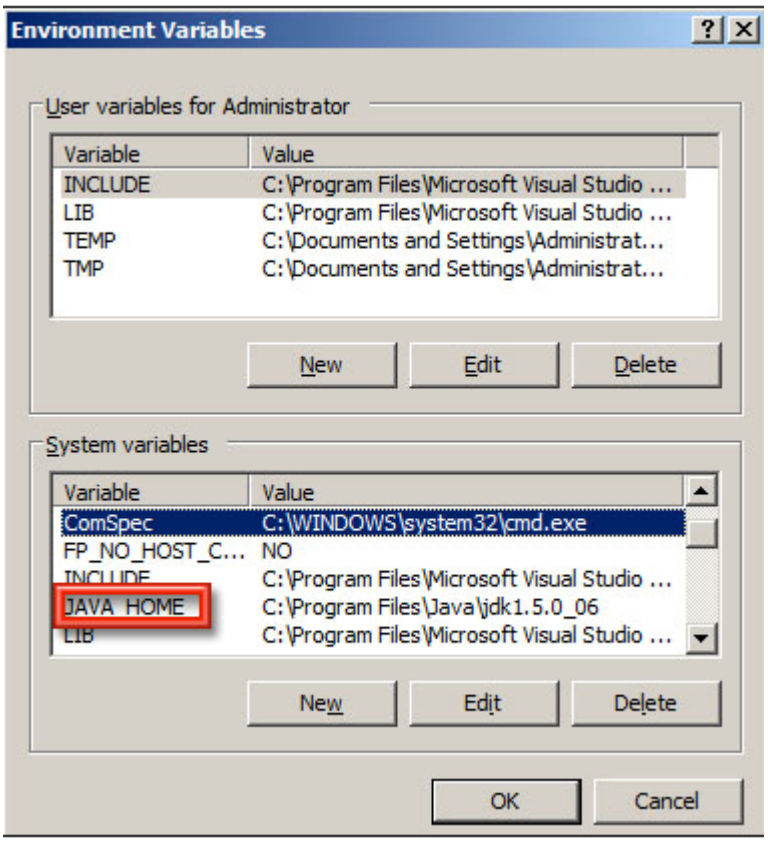


Şekil 2.1-2 Maven'ın versiyon bilgisine ulaşma.

yukarı

5. Adım : Java SDK sisteminizde kurul olduğundan emin olun. SDK – JDK – JRE hakkında bilgi için ilgili linke gidiniz. **(JSF,JDK,SDK ve JRE arasındaki fark nedir?)**

NOT: JAVA_HOME sistem değişkeninde tanımlanmış olmalı.



Şekil 2-1.3 JAVA_HOME değerini eklemek.

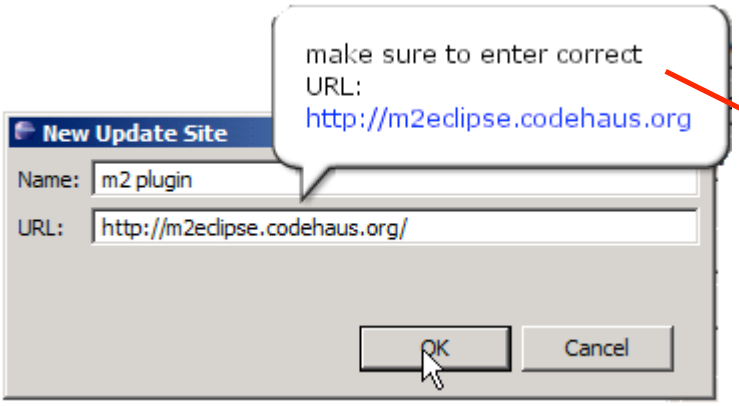
2.2. Linux, Solaris ve Mac OS X için :

- Maven ../usr/local/maven-2.0.6 altında kaydedilir.
- **bin** için yol tanımlaması yapılır. PATH=/usr/local/maven-2.0.6/bin
- Windows kurulumunda olduğu gibi JDK için bir yol tanımlamasının olduğundan emin olunmalıdır. JAVA_HOME=/usr/java/jdk1.5.0_02
- mvn --version komutu komut satırında yazılarak yükleme işleminin sorusuz olup olmadığı konusunda emin olabilirsiniz.

3. Eclipse ve Maven

http://m2eclipse.codehaus.org/Installing_Maven_2.0_plugin_for_Eclipse.html linkinden Eclipse programına maven'ın modifiye edilmesi hakkında görsel bir dokümana ulaşabilirsiniz. Dikkat etmeniz gereken nokta plugin'lerin hala aynı linkte olup olmadığıdır.

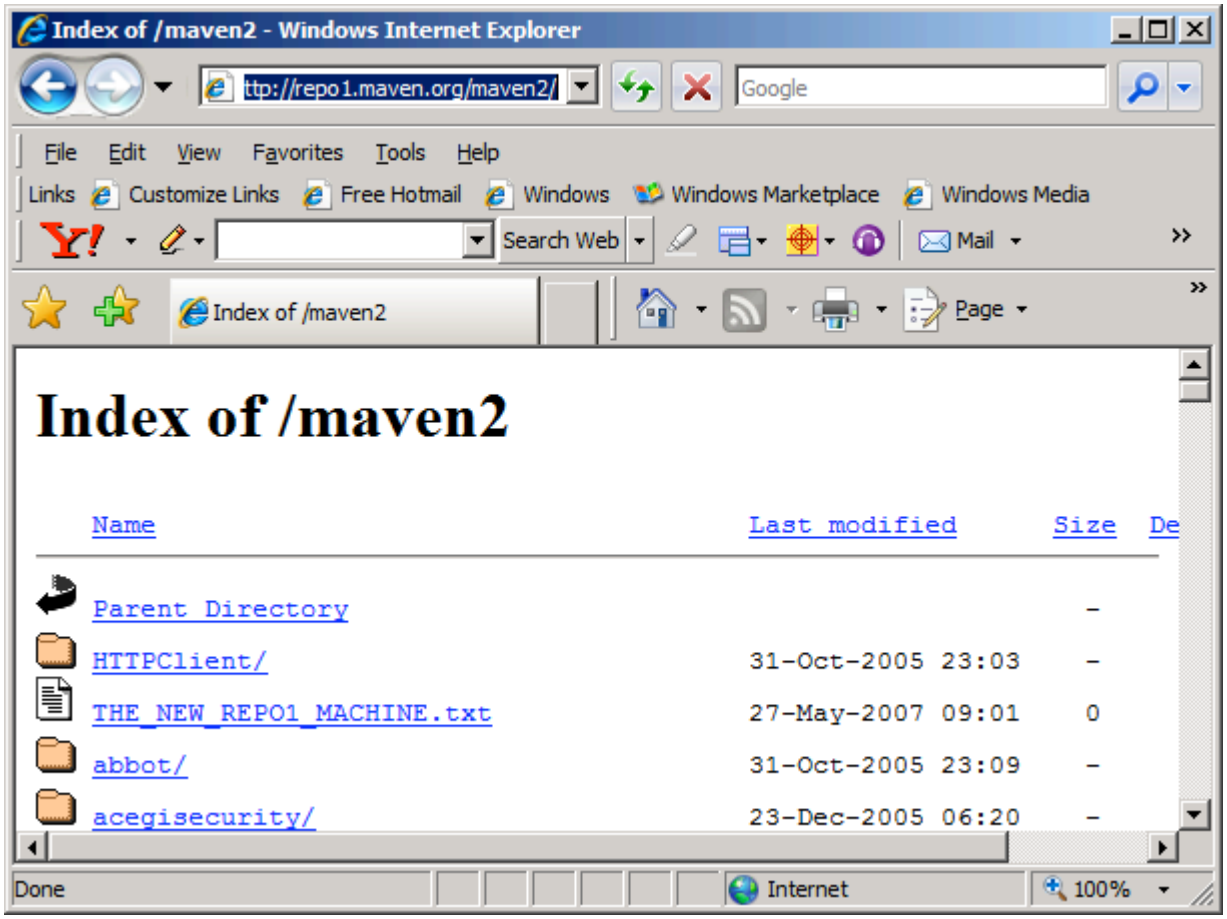
yukarı



Bu linkin değişip değişmediğinden emin olunuz.

Şekil 3-1 Maven'ın Eclipse 'ye modifiye edilebilmesi için gerekli olan URL ayarlamasını yapınız.

Bu ayarın yapılması ile birlikte genel depodaki plugin'lere ulaşılabilmektedir. Maven her proje çalışmasında güncellemeleri ve gerekli olan plugin'leri burada belirtilen yolu izleyerek yerel depoya indirilmesini sağlar. Yerel depoya indirilmesi ile birlikte genel depoya tekrar tekrar ulaşılması ve bundan doğacak zaman kaybını engellemiş oluruz. Maven'ın merkezi deposunu http://repo1.maven.org/maven2/ adresinde bulabilirsiniz.



Şekil 3-2 Global Repository (Genel Depo).

yukarı

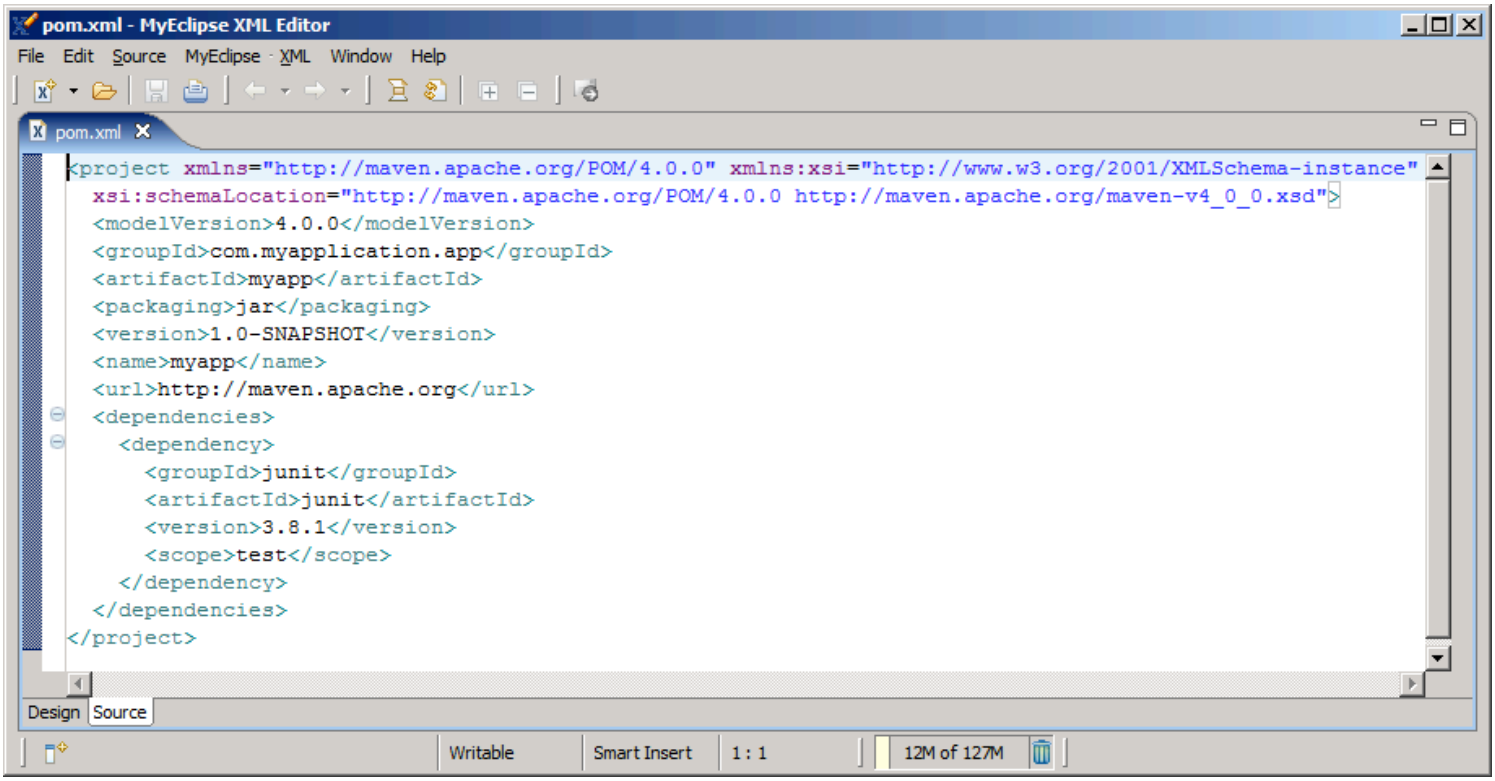
4. pom.xml Nedir?

POM (Project Object Model) “Nesne Modelli Proje” de diyebileceğimiz İngilizce açılımının baş harflerinden oluşmuştur. Dosyanın ismi ise *pom.xml* olarak program içerisinde yer alır.

Dosya konfigürasyonundan, organizasyon, projenin ihtiyaç duyduğu yapılandırma bilgileri ve lisansa kadar pek çok ön bilgi burada yer alır.

Aşağıda görünen kod bütünü projeyi yaratmak için komut satırında kullandığımız tek bir satır komut ile otomatik olarak oluşmuştur. **Şekil 4-1** ‘de Eclipse ‘deki kaynak dosyanın görünümüne ulaşabilirsiniz. pom.xml projenin çekirdeğini oluşturur. Bir nevi projenin tanıtım kartı gibidir. Maven öncelikle bu dosyaya bakar.

Geri



Şekil 4-1 pom.xml ‘in içeriği şekilde görüldüğü gibidir.

yukarı

Örneğin ; aşağıdaki POM parçası projenin gerekli ön bilgilerini içeriyor.

Liste 4-1 Project bölümünün ön bilgileri.

```
<project>
  <modelVersion>4.0.0</modelVersion>          ; Kullanılan maven 'ın versiyonu
  <groupId>org.codehaus.mojo</groupId>        ; Dahil olduğu grup,bir dizin oluşturur.
  <artifactId>my-project</artifactId>         ; Projeye verilen başlık.
  <version>1.0</version>                      ; Projeye verilen versiyon bilgisi.
</project>
```

Liste 4-2 Paketleme için kullanılacak değerin belirlenmesi.

```
<project>
  …
  <packaging>war</packaging>
  …
</project>
```

Yukarıdaki kod parçası ile paketleme işleminin yapısı belirleniyor. Packaging'in değeri : *POM, jar, maven-plugin, ejb, war, ear, rar, par* 'dan her hangi biri olabilir. Yukarıdaki örnekte packaging'in değeri "*war*" olarak verilmiştir.

Liste 4-3 Project bölümünde tanımlanan paketleme için kullanılacak değerin belirlenmesi.

```
<project>
  …
  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>4.0</version>
      <type>jar</type>
      <scope>test</scope>
      <optional>true</optional>
    </dependency>
    …
  </dependencies>
  …
</project>
```

yukarı

Liste 4-3 'deki kod parçası ile projenin bağlı olduğu başka bir projenin dahil edilmesi işlevi gerçekleştirilmiştir. Maven'ı tanımlarken de söz ettiğimiz kalıtım özelliği burada görülebilmektedir. Başka bir proje burada temel bilgileri verilerek yaratılan yeni proje için veri olarak gösterilmiştir.

5. Proje Tanımlayıcısı Oluşturmak

Maven proje tanımlı bir programdır ve POM maven'ın temel bir bileşenidir. Maven projeyi yapılandırırken ihtiyaç duyduğu konfigürasyon ve temel bilgilerini pom.xml aracılığı ile elde eder. Bünyesinde varsayılan olarak bulundurduğu yapılar da mevcuttur örneğin, target dizininde uygulama ve test dosyalarının nerede bulunacağı otomatik olarak belirlenir.

pom.xml Maven 1'de project.xml olarak adlandırılıyordu. Bu isim Maven 2 ile birlikte pom.xml olarak değiştirildi. Daha önce maven.xml içerisinde gerçekleştirilmesi gereken amaçlar ve işlenmesi gereken plugin'ler artık *pom.xml* içerisinde yapılandırılmaktadır. Maven artık temel uygulanması gereken her türlü işlev için pom.xml'den bilgi alıyor.

Hazırlanan işin her bir parçası proje olarak adlandırılır. Kalıtım özelliğini destekliyor olması parçaların bir bütünü olmasından kaynaklanır. Bu nedenle maven'la bir çalışmaya başlamadan önce bir proje tanımlayıcısı hazırlanması gerekir. Hazırlanacak olan bu yapı bir *.xml* çalışması olacak ve proje tanımlayıcısı olarak da adlandırılacaktır. Maven'ın projede kullanılabilmesi için pom.xml oluşturulmuş olması gerekir ve konumu hiyerarşinin en üst seviyesinde olmalıdır.

pom.xml'e bakıldığında yapının 4 ana kısımdan oluştuğu görülür :

- Management (Yönetim)
- Dependency section (Bağımlı Kısım)
- Build (Yapılandırma)
- Reports (Raporlama)

Liste 4-1 POM.xml .

```
01.  <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/maven-v4_0_0.xsd">
02. <modelVersion>4.0.0</modelVersion>
03. <groupId>com.myapp.app</groupId>
04. <artifactId>myapp</artifactId>
05. <packaging>jar</packaging>
06. <version>1.0-SNAPSHOT</version>
```

yukarı

```
07. <name>Maven Projesi</name>
08. <url>http://maven.apache.org</url>
09.   <!--Yönetim -->
10.   <!-- Bağımli Kesim -->
11.   <!-- Yapılandırma -->
12.   <!-- Raporlama -->
```

01. *maven* hakkında site kaynakları tanımlanmıştır, proje tanımlamasının temelidir.
02. *Maven* versiyonunun bilgisi.
03. Aynı groupId 'sini paylaşan projeler için oluşturulmuş dizine *com.myapp.app* değeri verilmiştir.
04. Projenin ismi.
05. Paketlemek için kullanılacak değerin belirlenmesi.*war,ear* v.b. olabilir.
06. Versiyon belirlemesi.
07. Projenin ismi.
08. Organizasyonun sahip olduğu ayrıntılardan biri.

5.1. Yönetim Kesimi

Yönetim kesimi adından da anlaşılacağı üzere organizasyonun genel bilgisini içerir. Organizasyonun web sitesi, projeye ait web sitesi, *SCM* 'nin bulunduğu yer, geliştiricilerin listesi, mail listeleri gibi bilgiler yer alır. Bu yapıda eklenebilecek bir çok kısım var, daha açık söylenmesi gerekirse *pom.xml* 'in şekillenmesi geliştiricinin isteğine bağlıdır.

Liste4.1-1 Pom'un yönetim kesimi.

```
<project>
  <modelVersion>4.0.0</modelVersion>
  <groupId>${groupId}</groupId>
  <artifactId>${artifactId}</artifactId>
  <packaging>jar</packaging>
  <version>${version}</version>
  <name>A custom project</name>
  <url>http://www.myorganization.org</url>
</project>
```

yukarı

groupId : Projenin dahil olduğu grubun tanımladığı kısımdır. Her ne kadar biz nokta koyarak tanımlamayı yapsak da *Maven* bu yapıyı bir hiyerarşi olarak algılar ve *org.myapplication.app* olarak yazılmış olan yapıyı *org/myapplication/app* şekline getirir.

artifactId : Dahil olduğunuz bir grup var ve siz kendinize özel bir isim belirliyorsunuz. Burada belirlenen isim sizin projenizin ismi oluyor.

packaging: Projeniz tamamlandıktan sonra onu taşıyabileceğiniz başka platformlarda kullanabileceğiniz bir yapıya sahip olmasını sağlamalısınız. Bu işlevin değeri bu kısımda belirleniyor.Örneğin, *jar,war* ya da *ear* olarak bu değeri belirleyebilirsiniz.

Liste 4.1-2 <packaging>Value</packaging> ile paketleme formatını belirlemek.

```
<project>
  ...
  <packaging>jar</packaging>
  ...
</project>
```

version : Projenizin belirli bir versiyonu olmalı bu bilgi burada belirleniyor. Varsayılan olarak 1.0-SNAPSHOT değeri belirlenir.

5.2. Bağımlı Kesim

Her bir proje için anahtar görevi üstleniyor. Projenin bağımlı olduğu değerlerin tanımlandığı kısımdır. **Liste 4.2-1** 'de de görüldüğü gibi projenin ihtiyaç duyduğu anahtar burada junit 'dir. Birden fazla tanımlama yapılabilir.

Liste 4.2-1 POM.xml'in bağımlı kesimi.

```
<project>
  ...
  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>4.0</version>
      <type>jar</type>
      <scope>test</scope>
      <optional>true</optional>
    </dependency>
    ...
  </dependencies>
  ...
</project>
```

yukarı

5.3. Yapılandırma

Yapılandırmada belirleyeceğiniz tanımlamalar proje için bir trafik polisi gibi hizmet eder. Kaynak kod, test, kaynak kütüklerin bulunduğu yeri tarif eder.

Liste 4.2-2 pom.xml'in bağımlı kesimi.

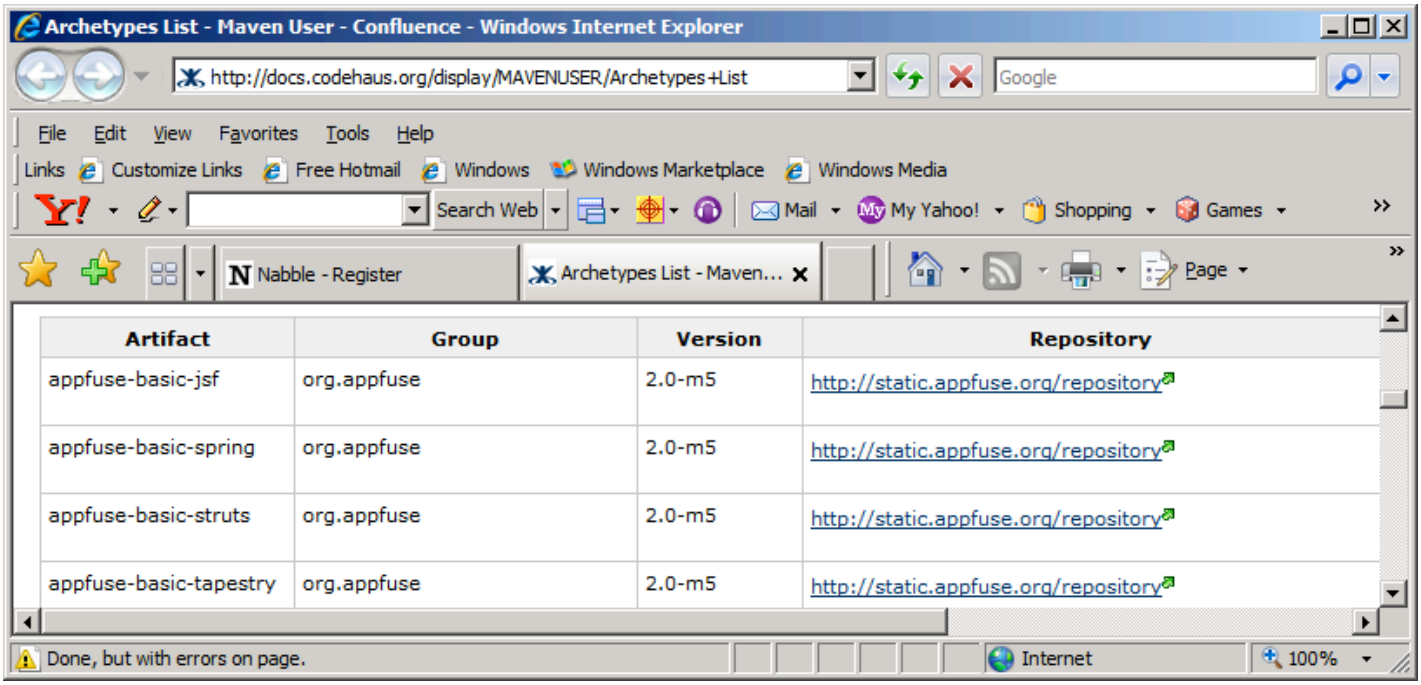
```
<build>
   <directory>target</directory>
   <outputDirectory>target/classes</outputDirectory>
   <finalName>${artifactId}-${version}</finalName>
   <testOutputDirectory>target/test-classes</testOutputDirectory>
   <sourceDirectory>src/main/java</sourceDirectory>
   <scriptSourceDirectory>src/main/scripts</scriptSourceDirectory>
   <testSourceDirectory>src/test/java</testSourceDirectory>
   <resources>
     <resource>
       <directory>src/main/resources</directory>
     </resource>
   </resources>
   <testResources>
     <testResource>
       <directory>src/test/resources</directory>
     </testResource>
   </testResources>
 </build>
```

yukarı

6. Proje Yaratmak

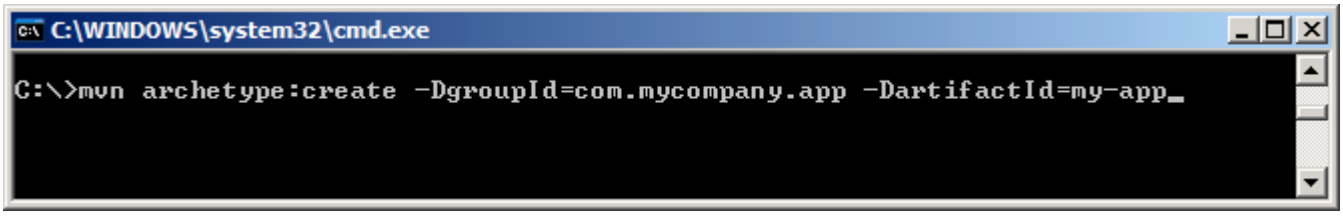
```
mvn archetype:create -DgroupId=com.mycompany.app -DartifactId=my-app
```

Farklı proje yapıları için http://docs.codehaus.org/display/MAVENUSER/Archetypes+List adresine bakabilirsiniz . Farklı proje yapıları oluşturma hakkındaki bilgiye **Archetype Yaratmak** ilgili konu başlığında ulaşabilirsiniz.



Şekil 6-1 Farklı archetype yapıları ile ilgili site.

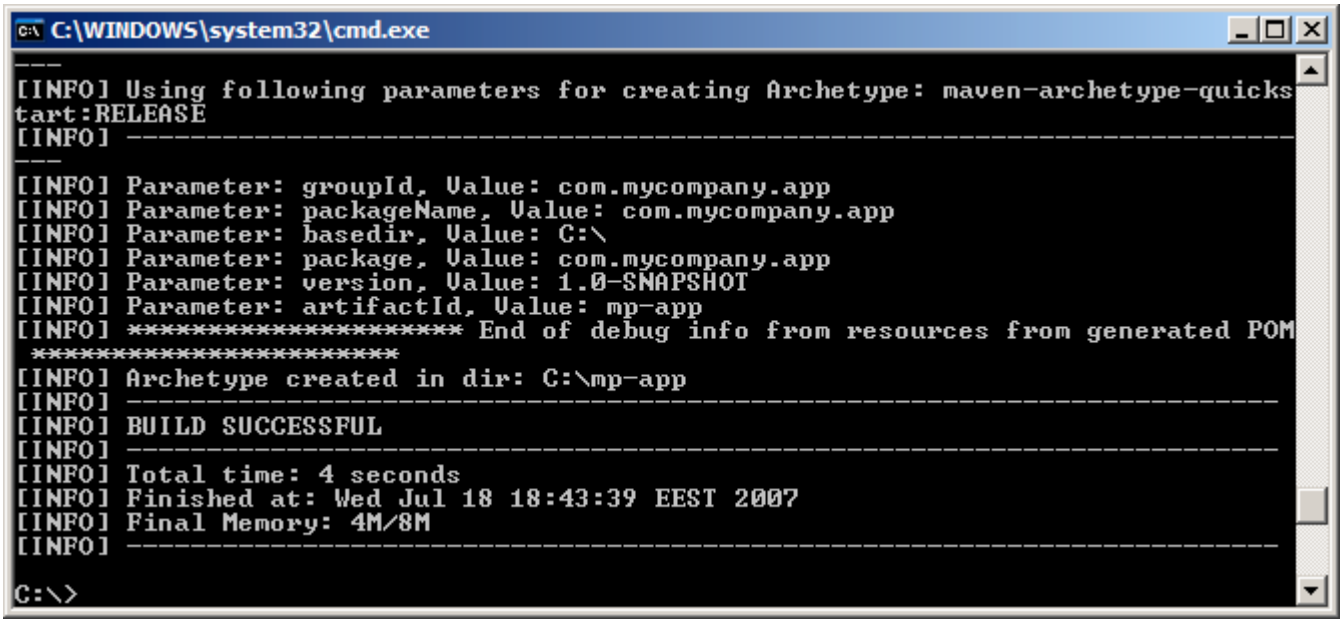
Komutunun çalıştırılması ile birlikte bir proje yaratılır.



Şekil 6-2 Yeni bir proje yaratmak.

yukarı

Komutun çalıştırıldıktan sonra **Şekil 6-3** 'teki gibi bir ekran ile karşılaşılır. Görüldüğü gibi işlem başarı ile gerçekleştirilmiştir.



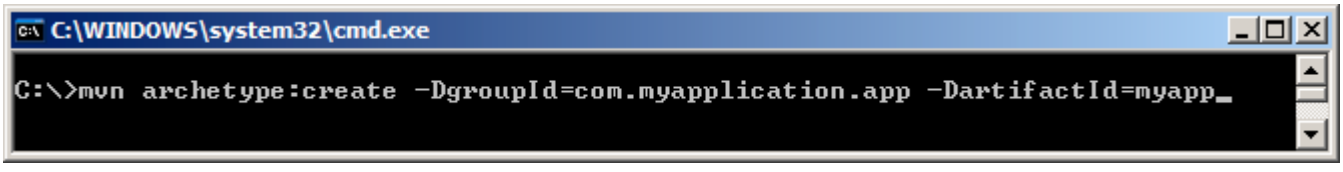
Şekil 6-3 Proje yaratma işleminin başarıyla sonuçlanması ile alınacak çıktı örneği.

7. Proje Örneği

Şimdiye kadar genel olarak değindiğimiz temel bilgilerle basit bir proje oluşturarak adım adım yapılması gereken temel işlemleri uygulayalım.

1. Adım : Projeye öncelikle bir isim ve paket belirlenir.
Projemizin ismi **myapp**
Paketin ismi ise **com.myapplication.app**
olsun.

2. Adım : **mvn archetype:create –DgroupId=com.myapplication.app –DartifactId=myapp** komutu ile birlikte otomatik bir proje yapısı oluşturulmaktadır.

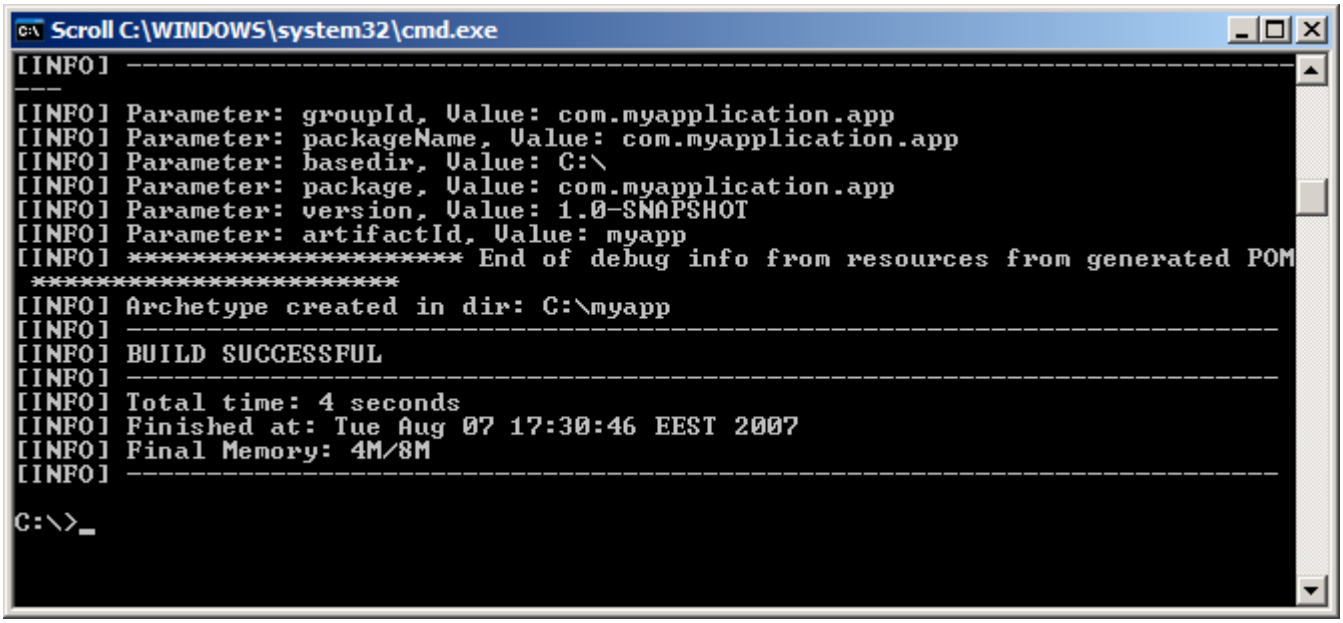


Şekil 7-1 Proje yaratman için kullanılan komut satırı.

yukarı

Projenin yapılandırması hakkındaki sonucun **Şekil 7-2** 'teki gibi olması gerekiyor. **[INFO] BUILD SUCCESSFUL** yazmalı. Maven'ı ilk kez kullanıyorsanız bu işlem gerekli olan plugin ve diğer dosyaların yerel depoya indirme işleminin gerçekleştirilmesi için daha fazla zaman alacaktır.

NOT: Daha önce aynı isimde bir proje yaratmışsanız işleminiz başarı ile gerçekleşmeyecektir.

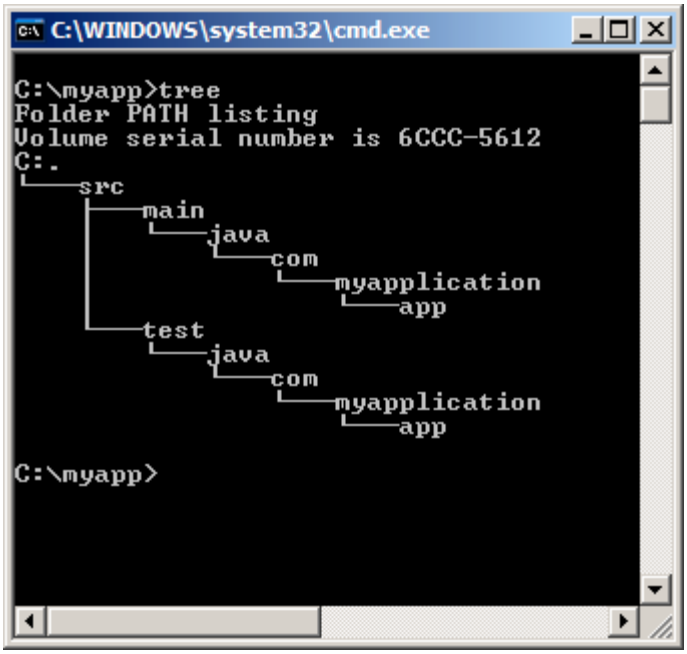


Şekil 7-2 Proje yaratma işleminin başarıyla sonuçlanması ile alınacak çıktı.

::Komutun çalıştırılması ile birlikte Şekil 7-3 'teki değişimler gözlemlenmektedir.

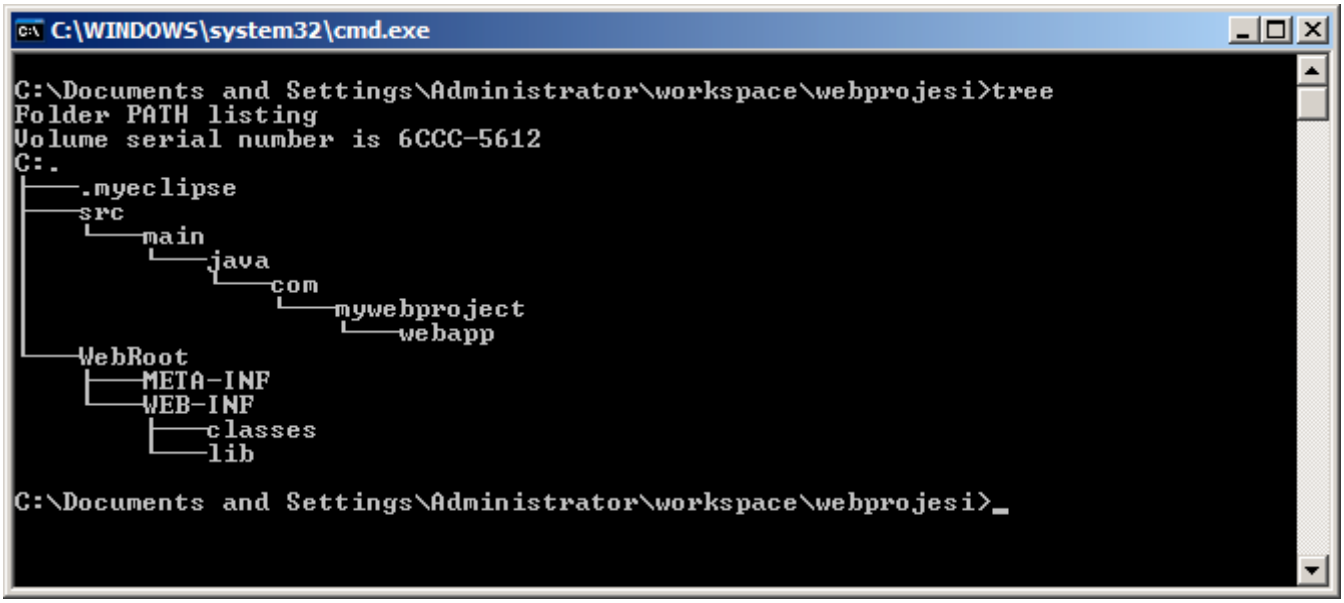
Maven'ın standart bir klasör yapısı vardır ;ancak, bu yapı archetype plugin'ninde yapılacak düzenlemeler aracılığı ile değiştirilebilmektedir. Yapılan işlemle birlikte elde edilen klasör yapısı **Şekil 7-3** 'te olduğu gibidir.

yukarı



Şekil 7-3 Oluşan projenin dizin yapısı.

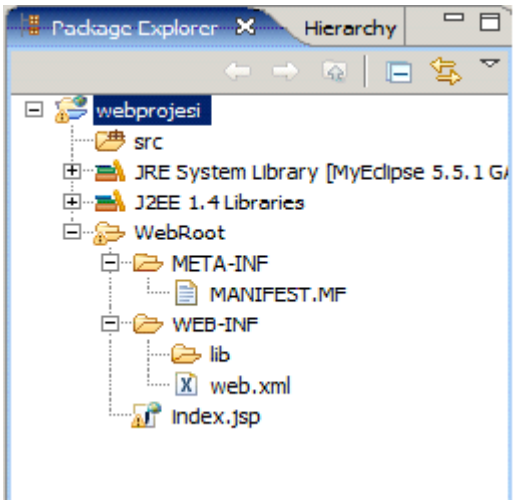
Görüldüğü gibi **src\main\java** uygulamanın kaynak dosyalarını, **src\test\java ise** test kaynak dosyalarını içermektedir. **pom.xml** *Maven*'ın proje yönetim dosyasıdır. **pom** (Project Object Model) dosyası sayesinde sadece birkaç satırla proje build işlemizi yapmış olursunuz. *pom.xml* projenin çekirdeğini oluşturur. *POM*'la ilgili ayrıntılı bilgi için **pom.xml Nedir?** linkinden ayrıntılı bilgiyi edinebilirsiniz.



Şekil 7-4 Web projesi seçildiğinde oluşan dizinin yapısı görüldüğü gibidir.

yukarı

Yaratmış olduğunuz proje web projesi olmuş olsaydı dizinin görünümü **Şekil 7-4** 'deki çıktıdaki gibi olacaktı. Bir WebRoot dizini görüldüğü gibi eklenmiştir.



Şekil 7-5 'de bulunan yapıyı incelerseniz normal bir *maven* projesinden farklı olarak pom.xml yerine bir web.xml 'in oluştuğunu görebilirsiniz. Bütününe bakıldığında ise bir WebRoot görülüyor. Bu root META-INF adı verilen bir dosya içerir. Bu dosya proje ile ilgili temel bilgileri içerir.

Şekil 7.5 Windows'da web projesinin hiterarşik yapısı.

3. Adım : **C:\>cd myapp ,** verdiğimiz komutla oluşturulan **myapp** dizinini seçelim.

4. Adım : Maven geliştirme süreci birkaç aşamadan meydana gelmektedir.

mvn compile : Proje dosyalarını compile eder.
mvn test : src/test/java klasörü altındaki test dosyalarını çalıştırır.
mvn package : Compile edilmiş kodu jar ya da seçilmiş değere göre paketler.
mvn install : Projenin başka projeler için kullanılır olması için yerel depoya indirir.

mvn package diyerek projeminizi deploy edelim.

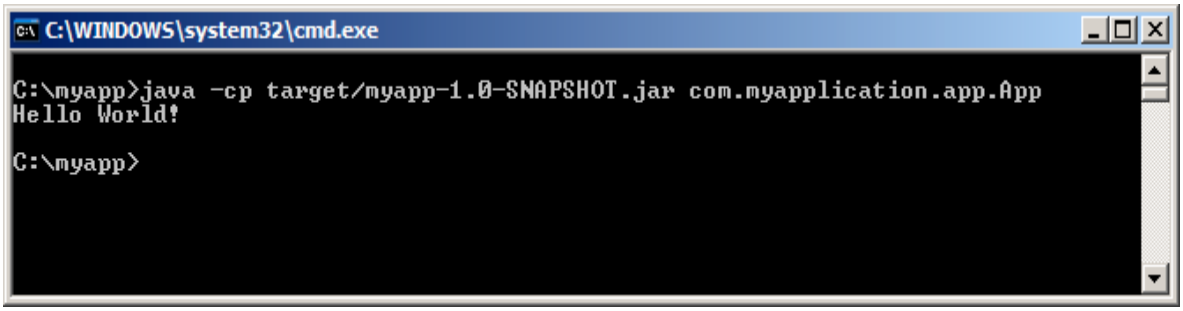
Proje mvn package ile deploy edildikten sonra artık uygulamayı çalıştırabilirsiniz. Oluşturduğunuz java kodunda package'i doğru belirlediğinizden emin olunuz.

```
Java –cp target/myapp-1.0-SNAPSHOT.jar com.myapplication.app.App
```

komutuyla projemizi çalıştıralım.

yukarı

Çıktı :

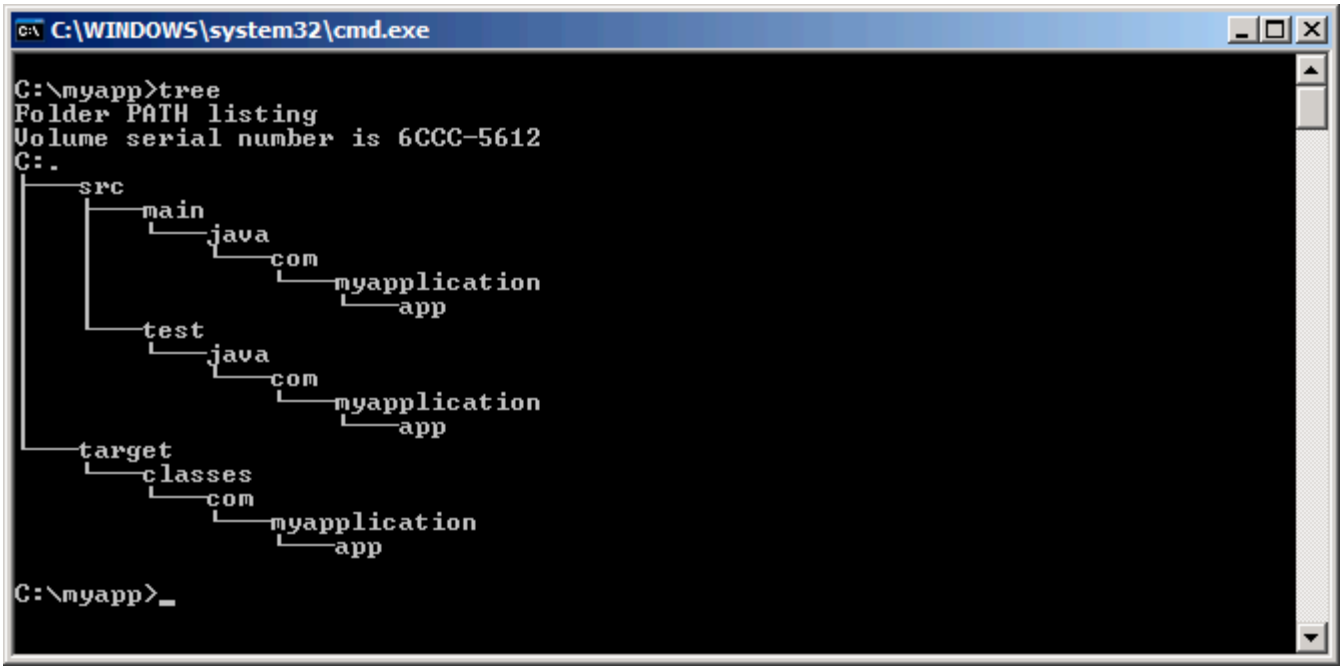


Şekil 7-6 Uygulamanın çıktısı yazılan komut satırı ile bu şekilde görünmektedir.

Yukarıda belirtilmiş olan adımlarla birlikte *maven* projesin'nin temel yapısı otuşturulmuştur. Projenin yapılandırma işlemleri (lifecycle) için phase'ler kullanılacaktır. Bu konu hakkındaki bilgiyi Proje Yapılandırma Adımları ' nda bulabilirsiniz.

7.1 Proje Yapılandırma Adımları

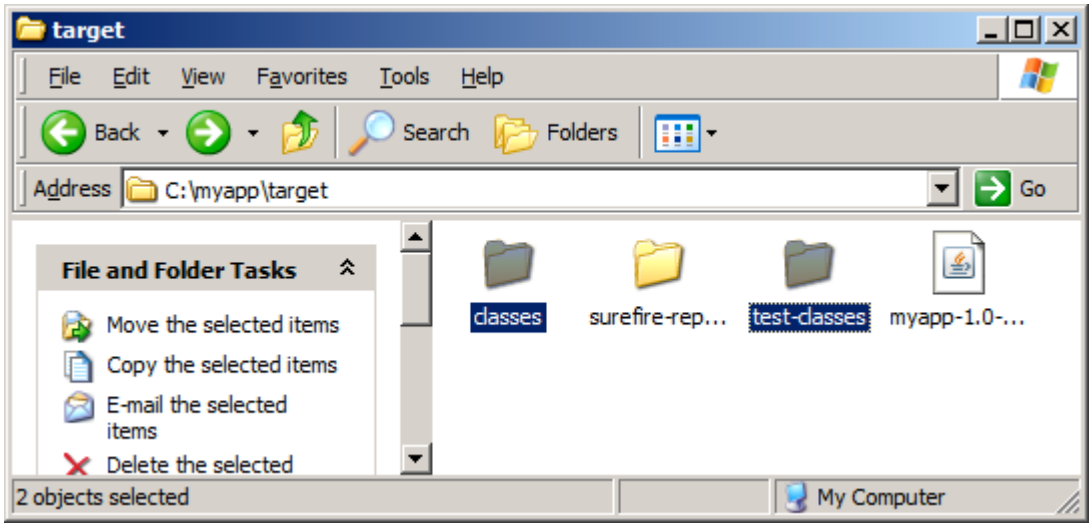
~>mvn compile işleminden sonraki klasör yapısı aşağıda görülmektedir. Gördüğünüz gibi target adında bir başka dizin eklenmiştir.



Şekil 7.1-1 *mvn compile* komutunun çalıştırılması ile *target* adında bir alt dizin oluşturulmaktadır.

yukarı

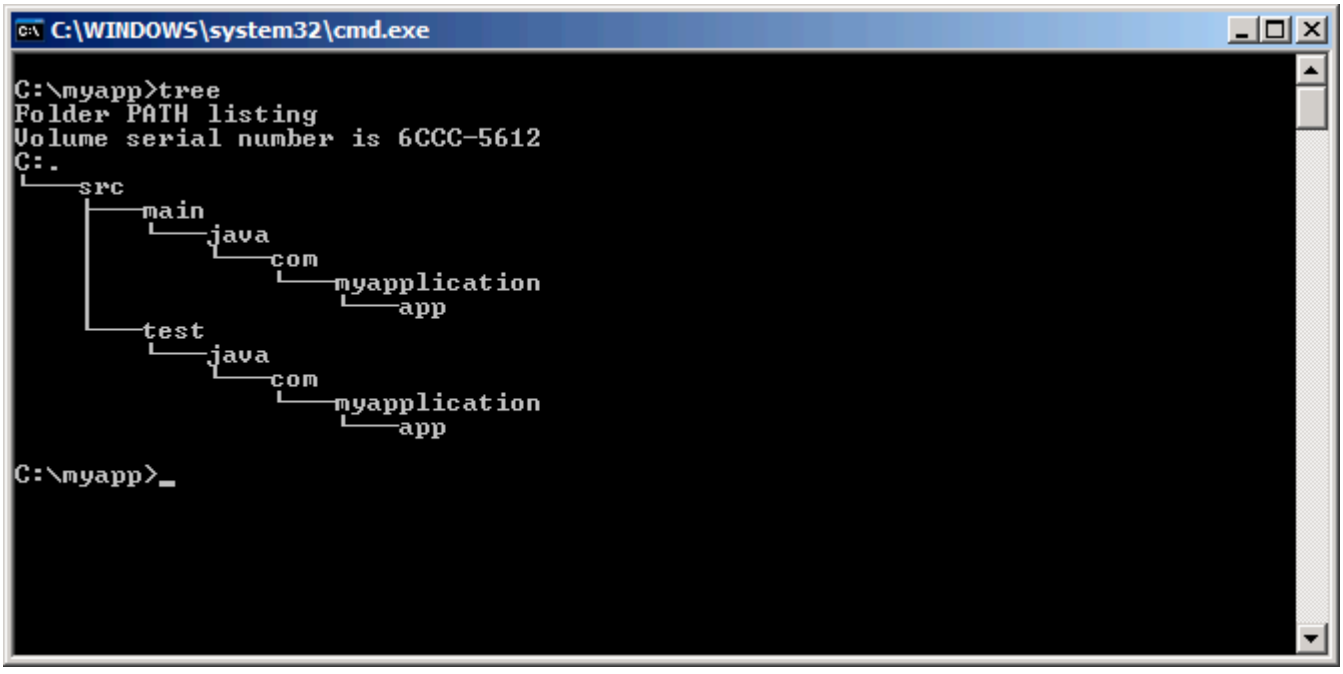
Maven derleme işlemi ile birlikte oluşan class dosyalarını **C:\myapp\target** dizini altına kaydeder.



Şekil 7.1-2 Oluşan class dosyaları *target* altında toplanmıştır.

Maven verileri nereden alacağını ve çıktıları nereye koyacağını bilmektedir. Bu hiyerarşik yapı proje geliştirme sürecinde geliştiricilere büyük bir kolaylık sunmaktadır. Bu yapı maven'ın sahip olduğu standart klasör yapısı ile elde edilmektedir.

~>mvn clean komutu ile *target* adında oluşturulmuş olan yapı silinmektedir. Bu şekilde aynı yapı içerisinde yeni verilerle işleminizin başından itibaren başlayabilirsiniz.

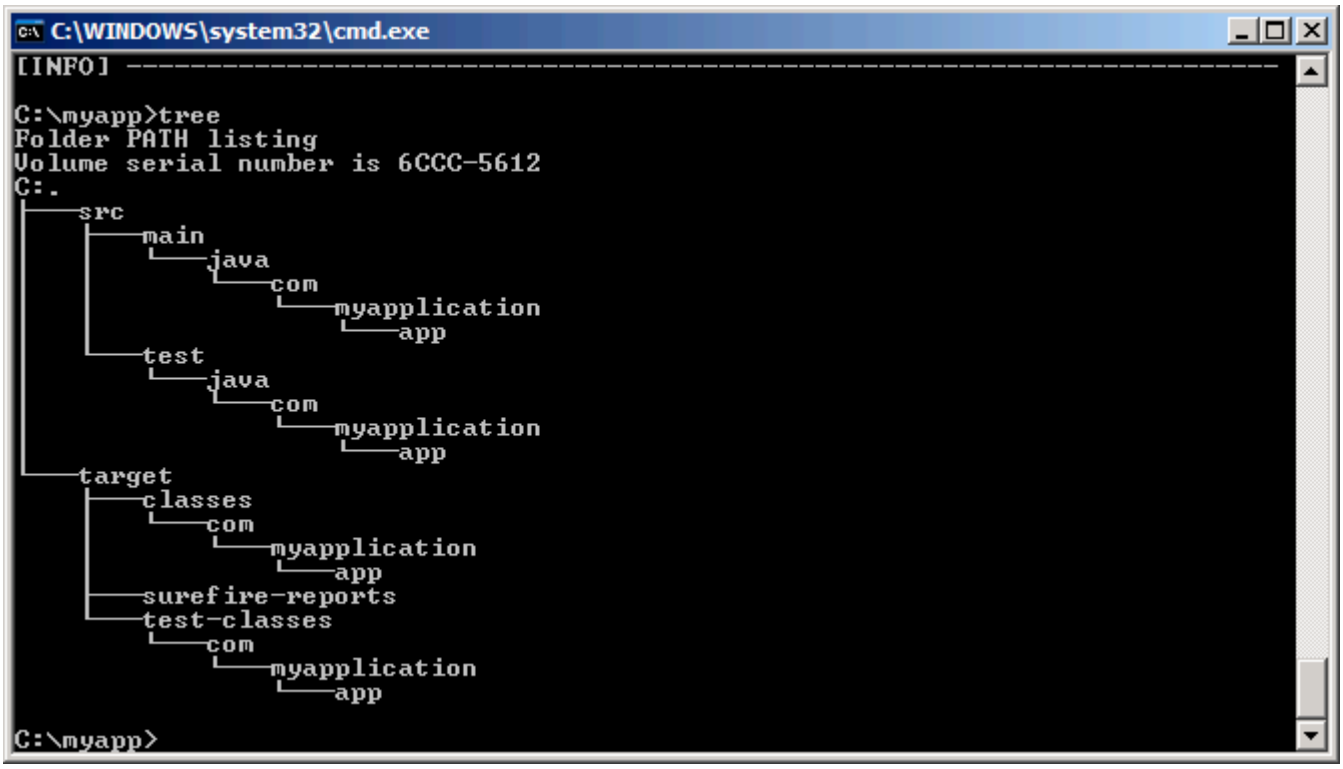


Şekil 7.1-3 *mvn clean* komutu kullanıldıktan sonra *mvn compile* ile oluşturulan target alt dizini artık görülmemektedir. Bu bize baştan başlama olanağı sağlar. yukarı

~>mvn package komutu ile projenin dağıtılabilir bir paket haline gelmesini sağlıyoruz.

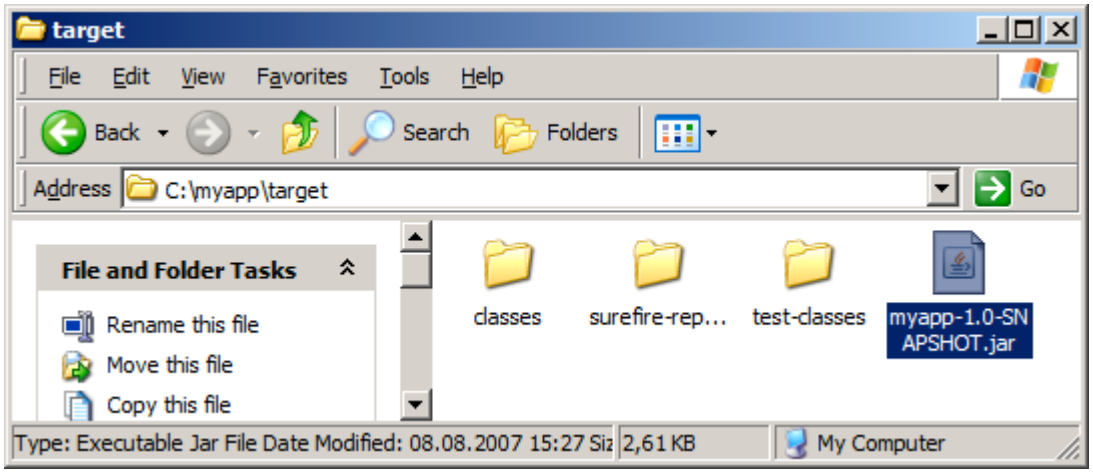
Aşağıda komutun uygulanması ile oluşan yapıyı görebilirsiniz. *POM.xml* de package in kullanacağı yapıyı **jar** seçebileceğiniz gibi **war** ya da **ear** olarak da şeçebilirsiniz.

<packaging>jar</packaging>



Şekil 7.1-4 *mvn package* komutu ile target altında bir *surfire-reports* dizini oluştu.

JAR dosyası **C:\myapp\target>** dizini altına oluşturulur.

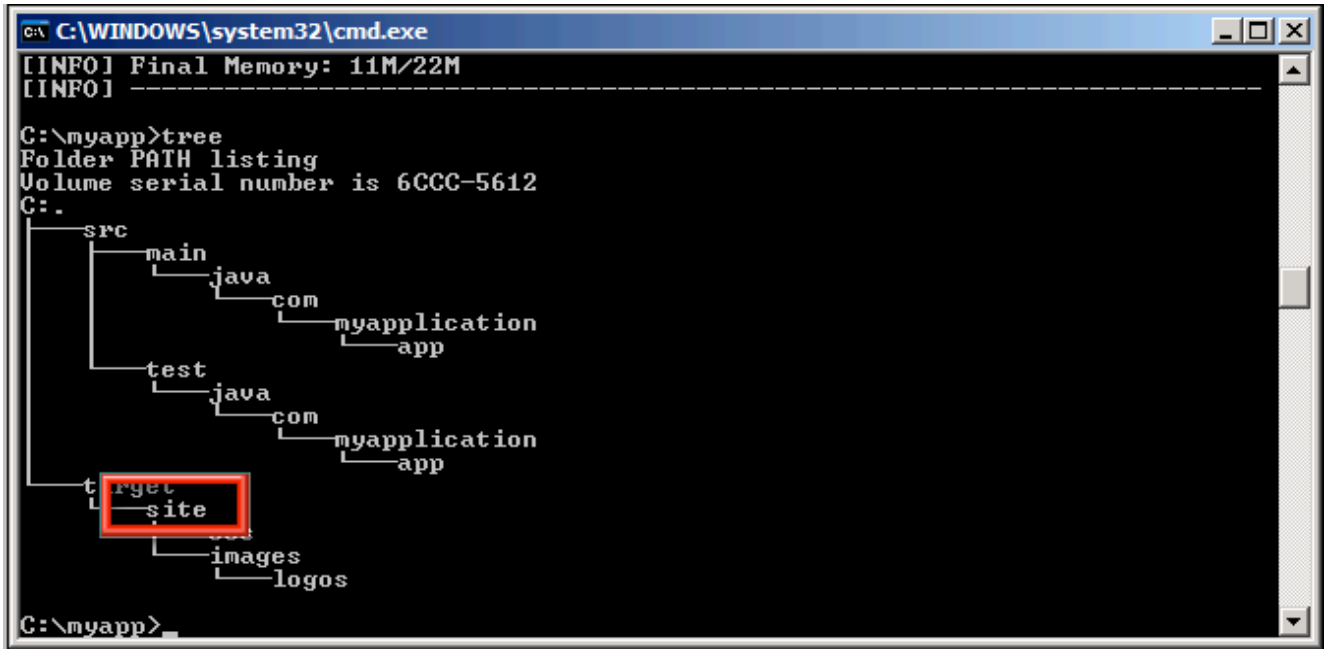


Şekil 7.1-5 jar dosyaları *mvn package* komutunun komut satırında çalıştırılması ile yukarıda belirtilen dizin oluşmuş olur.

yukarı

Oluşturulan bu JAR dosyasının başka projeler için de kullanılabilir olması için mvn install komutu kullanılır.

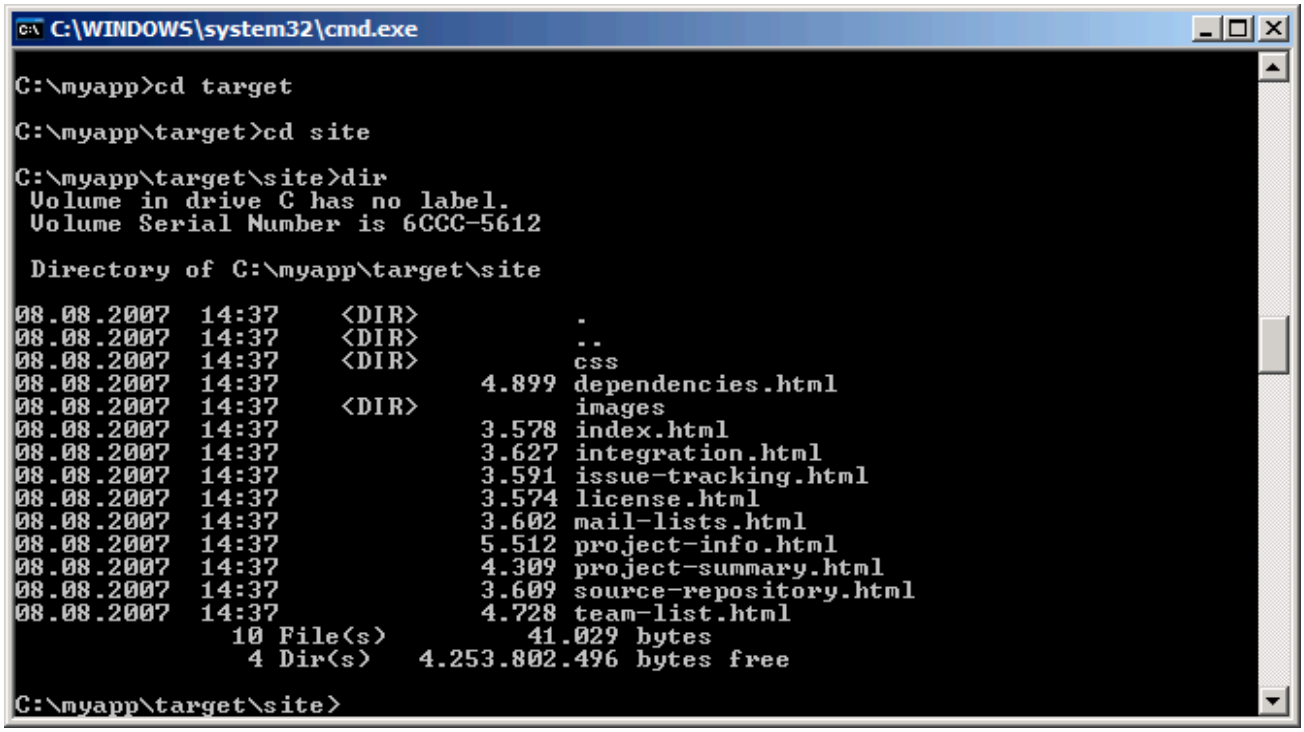
~>mvn site komutu ile basit bir web sitesi oluşturulabiliyor.



Şekil 7.1-6 *mvn site* komutu ile target altında bir site alt dizini oluşmuştur.

Site dizininin içeriği **Şekil 7.1-7** 'deki gibidir. Gördüğünüz gibi otomatik bir şekilde bir web sitesi için gerekli temel yapı oturtulmuş oldu. Bu yapı proje hakkında elde edilebilecek tüm genel bilgiye sahiptir.

yukarı



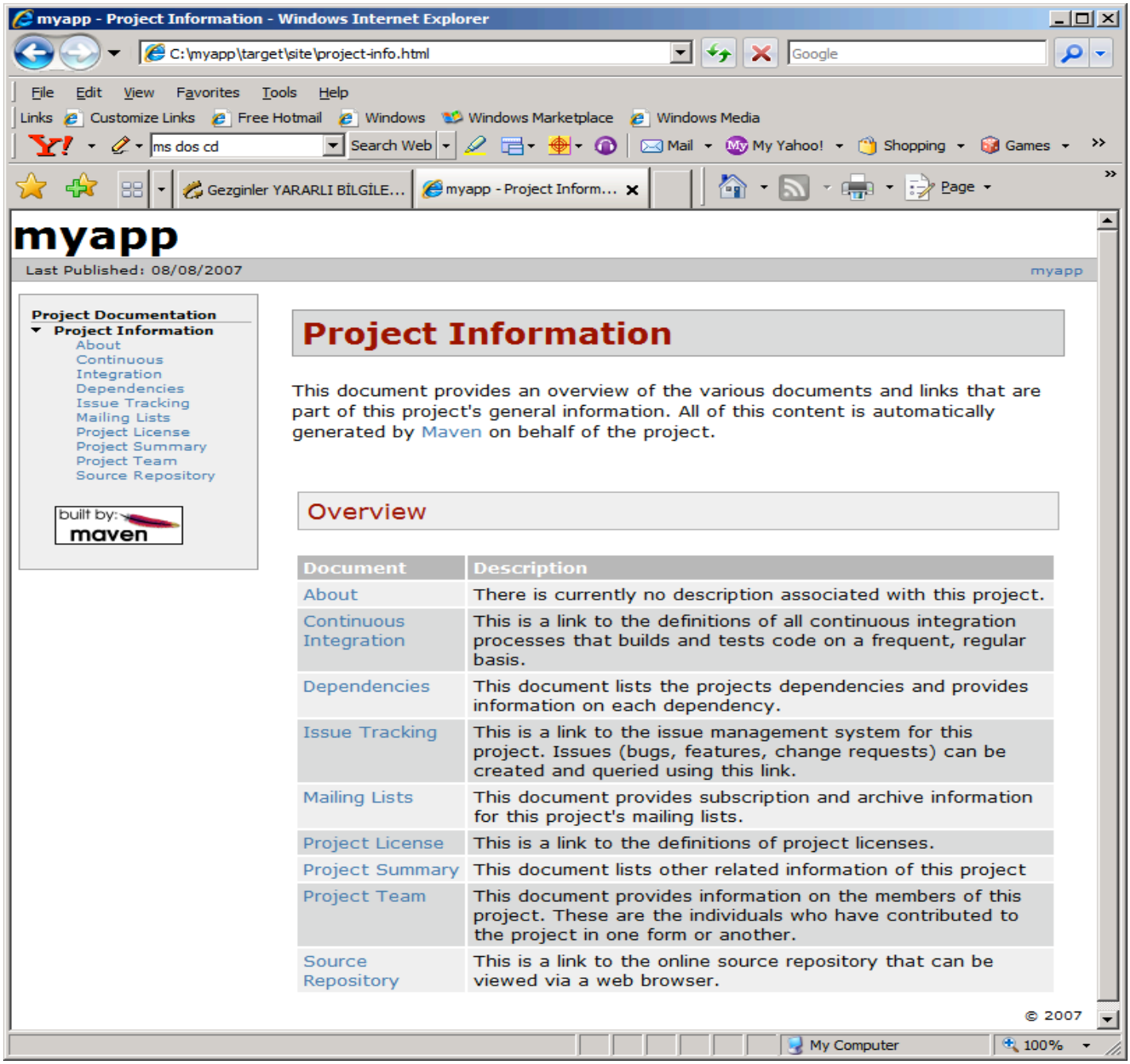
Şekil 7.1-7 *site* alt dizininin içeriği görüldüğü gibi olur.

Bu işlemlerin ardından artık projemizi yönetebileceğimiz bir siteye sahip oluyoruz.

Aşağıda oluşturulan web sitesinin kimi sekmelerinin gösterildiği örneklere ulaşabilirsiniz.

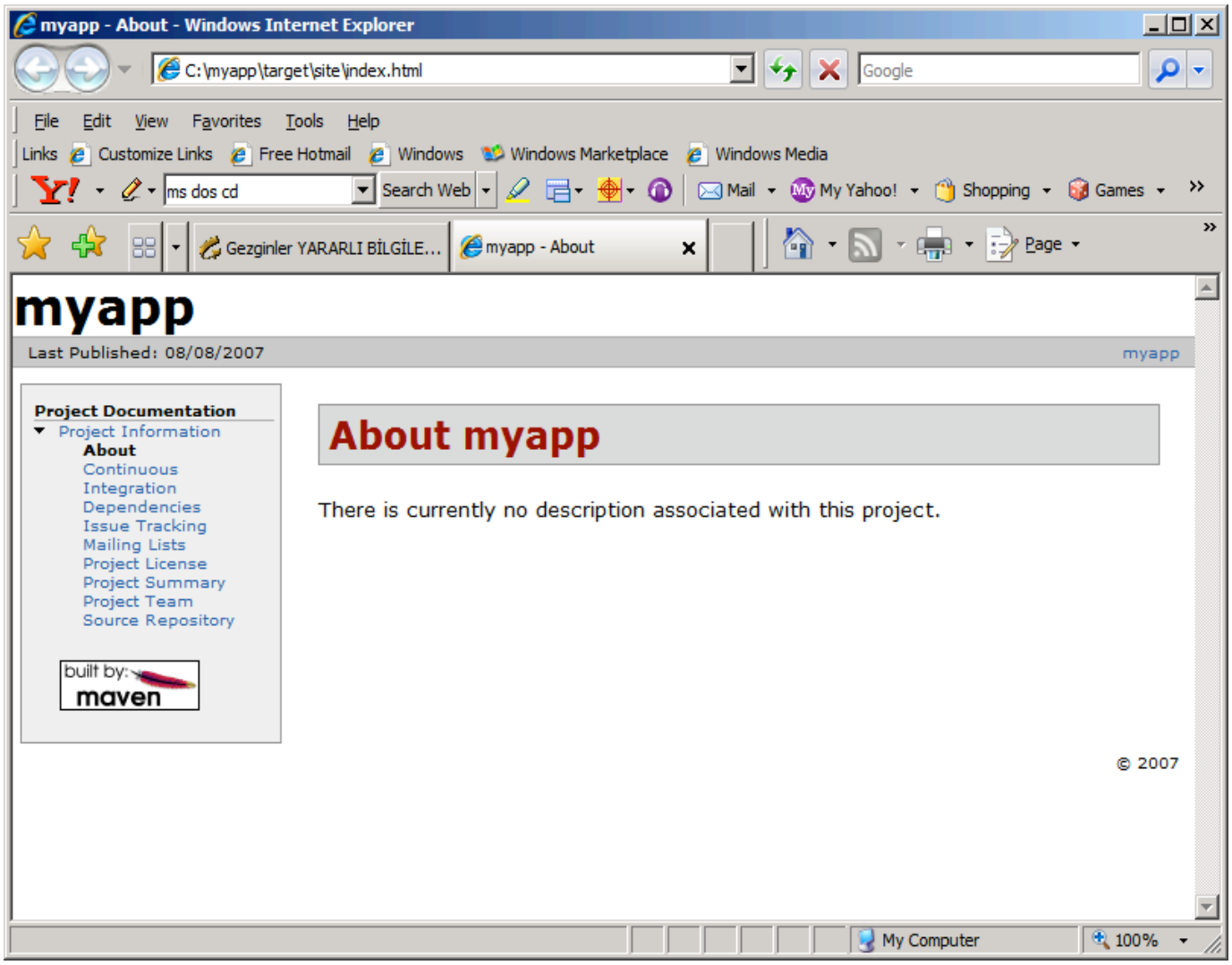
C:\myapp\target\site>index.html komutu yazıldığında **Şekil 7.1-8** 'deki sayfa explorer'da karşınıza gelecektir.

yukarı



Şekil 7.1-8 Otomatik olarak oluşturulan sitenin içeriğinden bir görünüm.

yukarı

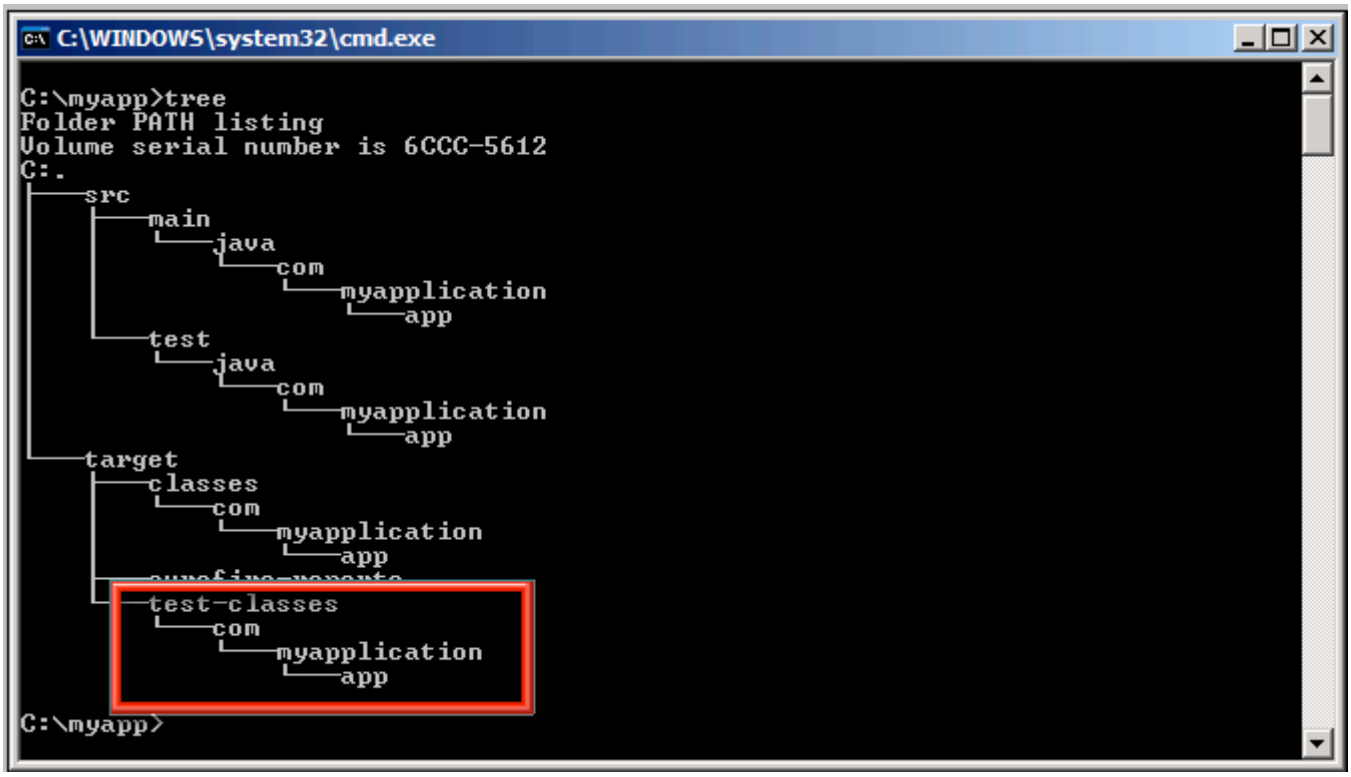


Şekil 7.1-9 Otomatik olarak oluşturulan sitenin içeriğinden bir görünüm.

Eğer sayfayı incelediyseniz proje hakkında verilen ayrıntılı bilgiyi farketmişsinizdir. Proje hakkında, proje için atanmış geliştiriciler, kaynak deposu, projenin genel gidişatı, lisansı hakkında geniş bilgiye ulaşabilirsiniz. Farklı bir ***site.xml*** sayfası oluşturularak otomatik oluşturulan bu site şekillendirilebiliyor.

yukarı

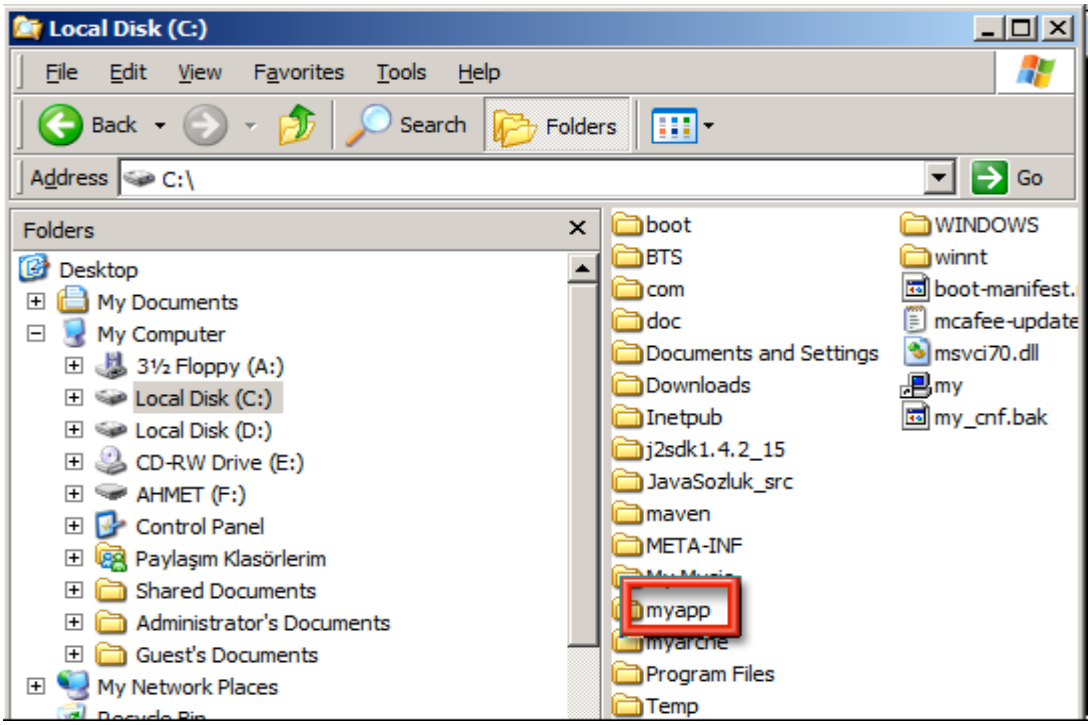
~>mvn test komutu ile test dosyaları çalıştırılmaktadır. **Şekil 7.1-10** 'da komutun çalıştırılması ile meydana gelen değişikliği görebilirsiniz. *Main* altında yer alan yapı test'e kopyalanıyor.



Şekil 7.1-10 *mvn test* komutu ile target dizini altında test dosyaları oluşturulmuştur.

Artık basit yapıda bir maven projemiz var ve **Şekil 7.1-11** 'deki C sürücüsünde oluşan yeni klasörü görebilirsiniz.

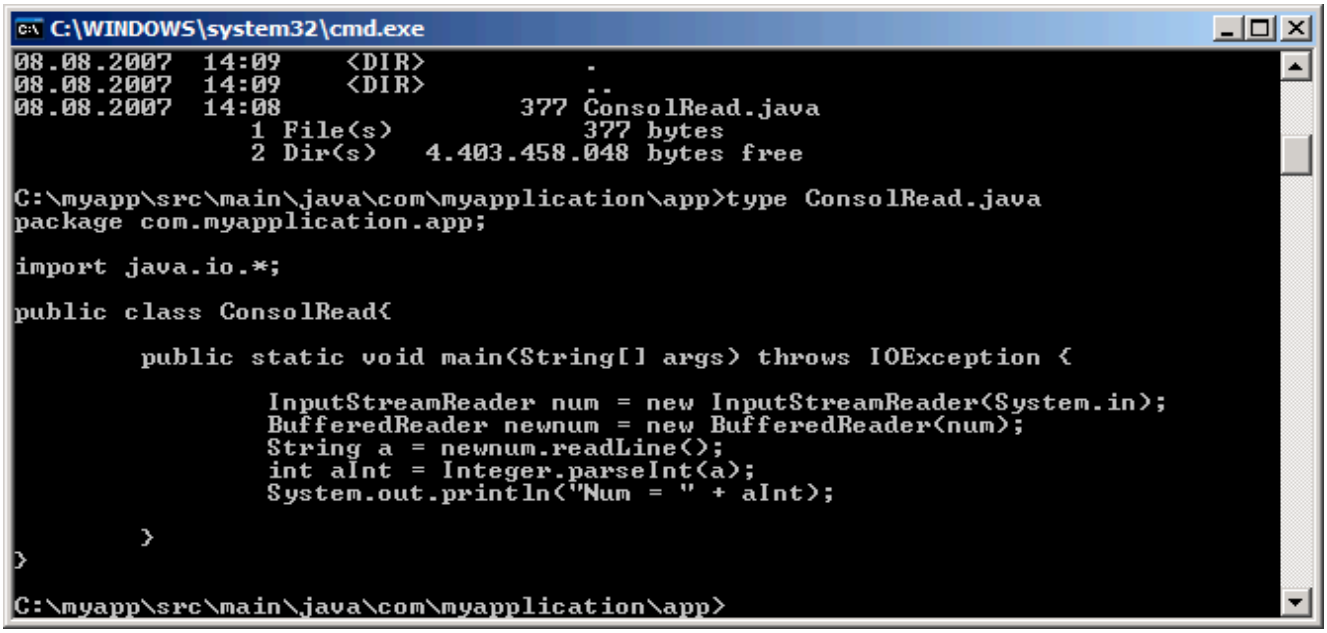
yukarı



Şekil 7.1-11 Oluşturulan projenin windows işletim sisteminde bulunacağı konum.

Aynı örneği farklı bir java programını hiyerarşiye ekleyerek gerçekleştirdim. Şekil 7.1-13 'de çıktıyı görebilirsiniz: Konsoldan girilen sayıyı yazdıran program örneği.

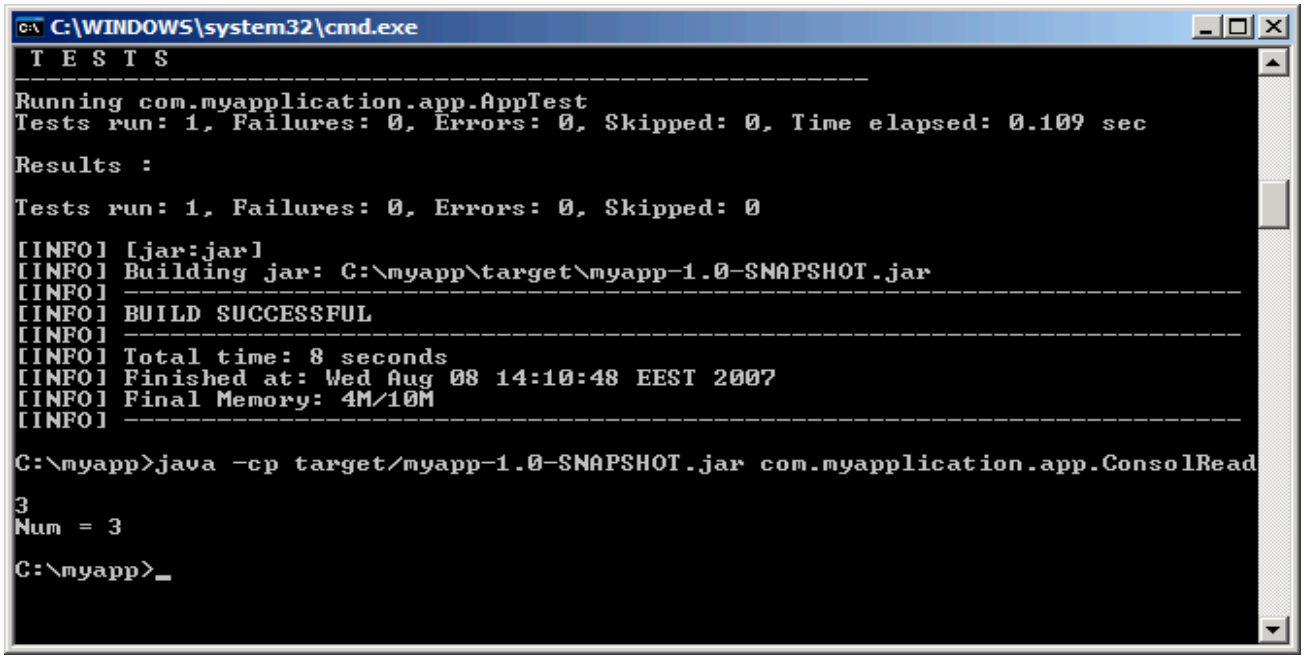
ConsolRead program parçası :



Şekil 7.1-12 Bir başka java programı.

yukarı

Çıktı :



Şekil 7.1-13 Yeni bir java pprogramı ile gerekli adımların gerçekleştirilmesinden sonra elde edilen çıktı.

8. Plugin ve Phase

Şimdiye kadar çeşitli plugin'leri kullanarak projenin yapılandırılması ile ilgili temel sayılabilecek işlemlerini gerçekleştirmiş olduk. plugin'ler sadece bu kadar olmayıp bir çok anlamda projenin yapılandırılmasını sağlayacak farklı plugin'ler de bulunmaktadır. plugin'lerle bir phase'in işleme şeklini değiştirebilirsiniz. Phase'ler plugin'lerin tetiklenmesini sağlayan komutlardır.

Örneğin:

clean → Komut Satırında Kullanımı : mvn clean

9. Built-Life Cycle Nedir?

Bir proje geliştirilirken belirli bir yol izlenir bu izlenen adımlara *built-life cycle* deniyor.Bu izlenen adımları sıralayacak olursak : *hazırlık,derleme,test,package,yükleme* gibi .
Maven'da *built-life-cycle* phase'lerle yapılmaktadır. Bu phase'ler goal'ları gerçekleştirmek için kullanılır. Maven'da bu phase'lerin çalışma şekilleri plugin'lerle değiştirilir.

Örneğin:

~>mvn compile

yukarı

10. Archetype

Archetype yaratma konusuna girmeden önce archetype'ın ne olduğunu, neden yaratma işlevinin gerçekleştirildiğini, bize sağladığı yararları gözden geçirelim.

Geri

10.1. Archetype Nedir?

Archetype basit bir plugin'dir. Bu plugin projenin temel yapısını hakkında bilgi içerir. Archetype plugin'le bir archetype konteynır yapılandırabilirsiniz, bu tıpkı bir çöp tenekesine ya da deniz taşımacılığına kullanılan bir nakliye aracına benzer. Galiba en mantıklısı bir nakliye aracı olacaktır. Bu nakliye aracı *POM, varsayılan depo, test, kaynaklar, test ve site kaynağı, özel dizin yapısını* içerir. Her geliştirmeci kendine özgü bir archetype yapısı oluşturarak projesini geliştirebilir.

Şekil 10.1-1 Archetype yapısı konteynır olarak da adlandırılmaktadır.

Maven 10 farklı archetype yapısını sunar :

- maven-archetype-archetype
- maven-archetype-j2ee-simple
- maven-archetype-mojo
- maven-archetype-portlet
- maven-archetype-profiles (*currently under development*)
- maven-archetype-quickstart
- maven-archetype-simple (*currently under development*)
- maven-archetype-site
- maven-archetype-site-simple, and
- maven-archetype-webapp

yukarı

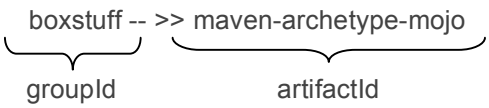
10.2. Archetype Yaratmak

Kendinize ait bir Archetype yapısına mı sahip olmak istiyorsunuz? O zaman şu adımları gerçekleştirelim.

10.2.1. archetype:create Ne İşe Yarar?

Yeni bir Archetype yaratmak için kullanılan phase'dir.

Artifact	Group	Repository	Description
maven-archetype-mojo	org.apache.maven.archetypes	http://repo1.maven.org/maven2	A Maven Java plugin development project



Yukarıdaki yapıyı incelerseniz aşağıdaki yapı için ne gerektiği konusunda bir fikir sahibi olabilirsiniz. Yeni bir archetype yapısı oluşturarak kendinize ait bir yapı ile aşağıdaki komut bütününü kullanarak projelerinizi oluşturabilirsiniz.

```
mvn archetype:create -DgroupId=<your group>                    \
                     -DartifactId=<your artifact>              \
                     -DarchetypeArtifactId=<wanted artifact>   \
                     -DarchetypeGroupId=<wanted artifact group> \
```

Proje örneği bölümünü incelerseniz kullanılan aşağıda görünen yapının nasıl adım adım şekillendiği konusunda bilgi edinebilirsiniz.

```
mvn archetype:create -DgroupId=com.mycompany.app -DartifactId=my-app
```

yukarı

10.2.2. archetype.xml

POM içerisinde belirlenen yapının archetype tanımlayıcısı **archetype.xml** 'dir. Bu dosya aşağıda belirtilen klasör hiyerarşisinin içerisinde yer alır.

~ src/main/resources/META-INF/

Archetype tanımlayıcı uygulamanın ve test dosyalarının bulunması gereken klasör yapısını belirler.

<sources> = src/main/java
<resources> = src/main/resources
<testSources> = src/test/java
<testResources> = src/test/resources
<siteResources> = src/site

```
<archetype>
  <id>quickstart</id>
  <sources>
    <source>src/main/java/App.java</source>
  </sources>
  <testSources>
    <source>src/test/java/AppTest.java</source>
  </testSources>
</archetype>
```

Yukarıdaki kod parçasına bakıldığında "**<id>quickstart </id>"** etiketi ile belirtilen kısımla projeye kısa kullanılabilir bir isim verilir. **<source></source>** etiketi tbelirlenmiş olan dizin içerisinde kodun derlenmesini sağlar.

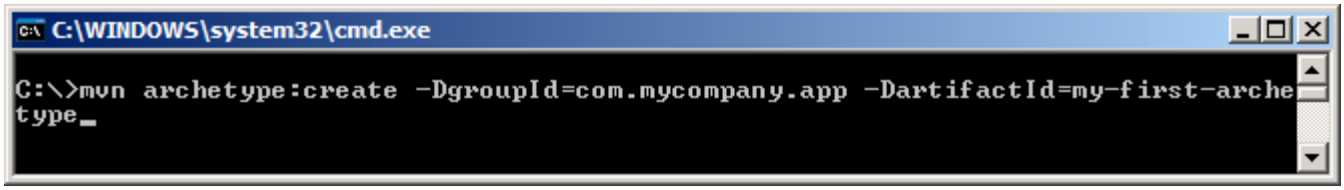
Liste 10.2.2-1 Projenin yapısı.

```
archetype
|-- POM.xml
`-- src
    `-- main
        `-- resources
            |-- META-INF
            |   `-- archetype.xml
            `-- archetype-resources
                |-- POM.xml
                `-- src
                    |-- main
                    |   `-- java
                    |       `-- App.java
                    `-- test
                        `-- java
                            `-- AppTest.java
```

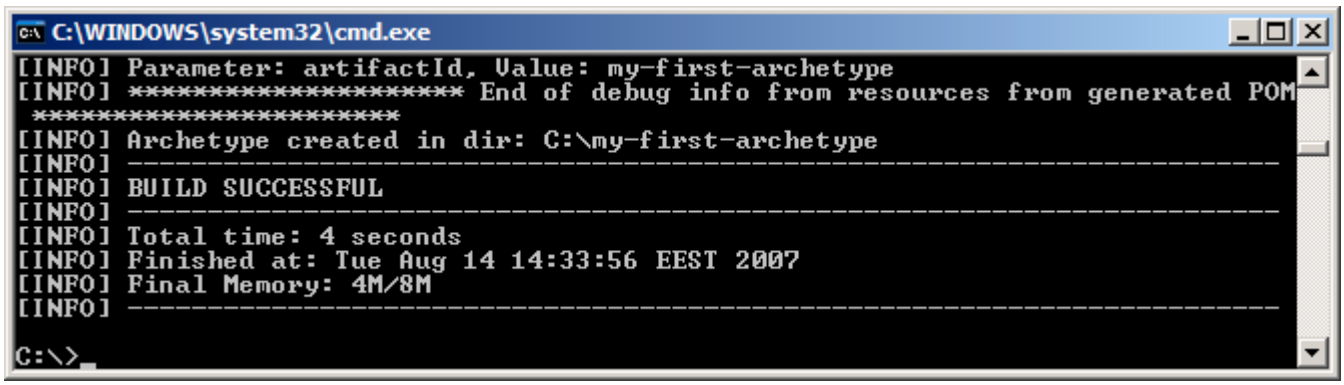
Belirlenen archetype yapılandırıcısı dizinin bu bölümünde yer almaktadır.

Uygulama görüldüğü gibi dizinin bu bölümünde.

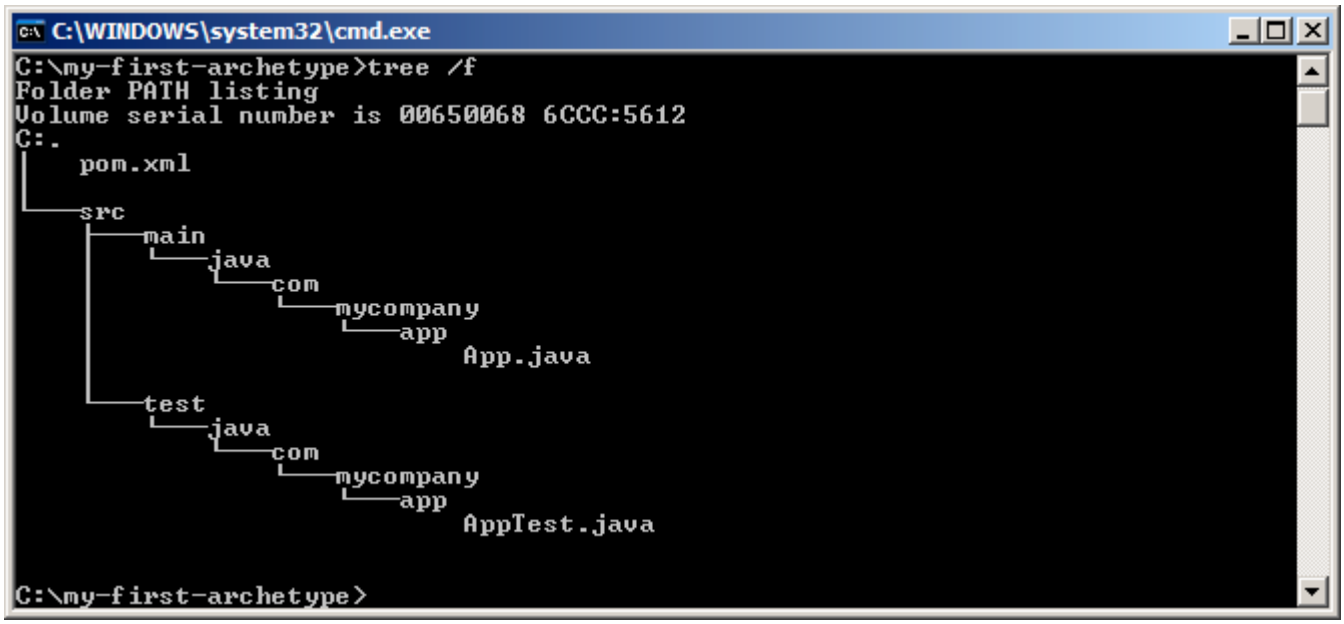
yukarı



Şekil 10.2.2-1 bir projeyi oluşturan komut, komut satırında yazıldı.



Şekil 10.2.2-2 Proje başarı ile yaratıldı.



Şekil 10.2.2-3 Otomatik oluşan dizin yapısı.

yukarı

Liste 10.2.2-2 maven-quickstart-archetype'la oluşan dizin yapısı.

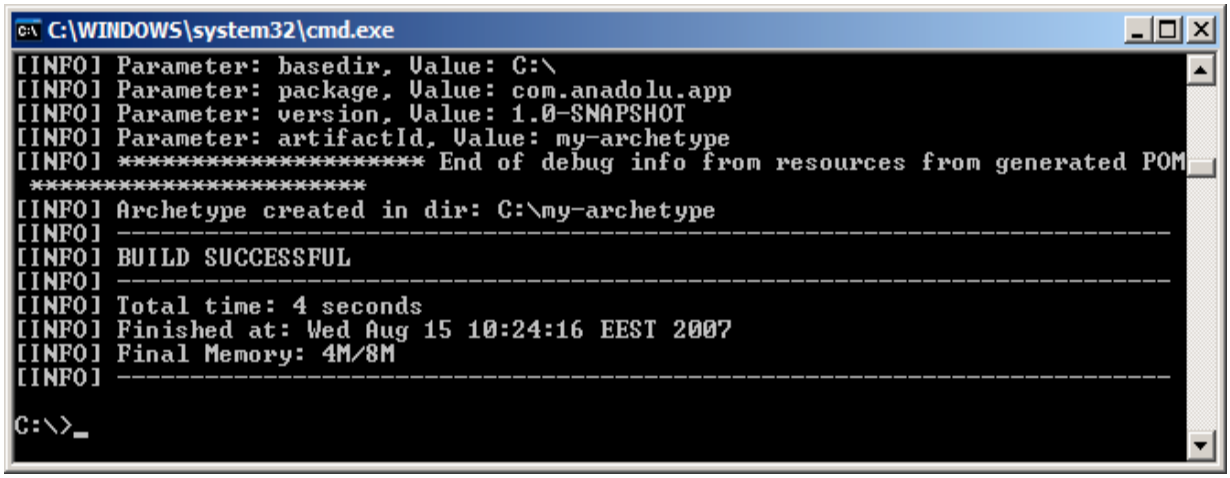
```
|-- POM.xml
`-- src
    `-- main
        `-- resources
            |-- META-INF
            |   `-- maven
            |       `-- archetype.xml
            `-- archetype-resources
                |-- POM.xml
                `-- src
                    |-- main
                    |   `-- java
                    |       `-- App.java
                    `-- test
                        `-- java
                            `-- AppTest.java9.2.2. Archetype.xml
```

Şimdi kendimize ait bir Archetype yaratalım :

1. Adım : Aşağıda belirlenmiş olan komut parçası yeni bir archetype yaratmak için gereken ilk adımdır. *groupId, artifactId* ve *versiyon* bilgisi verilen archetype komutun çalıştırılması ile birlikte size ait bir yapı oluşturmaya başlamış oluyoruz. Bir Archetype yaratmanın en kolay yolu **org.apache.maven.archetypes:maven-archetype-archetype** plugin'ini kullanmaktır.

Şekil 10.2.2-4 Kendimize ait my-archetype adında bir archetype oluşturuyoruz.

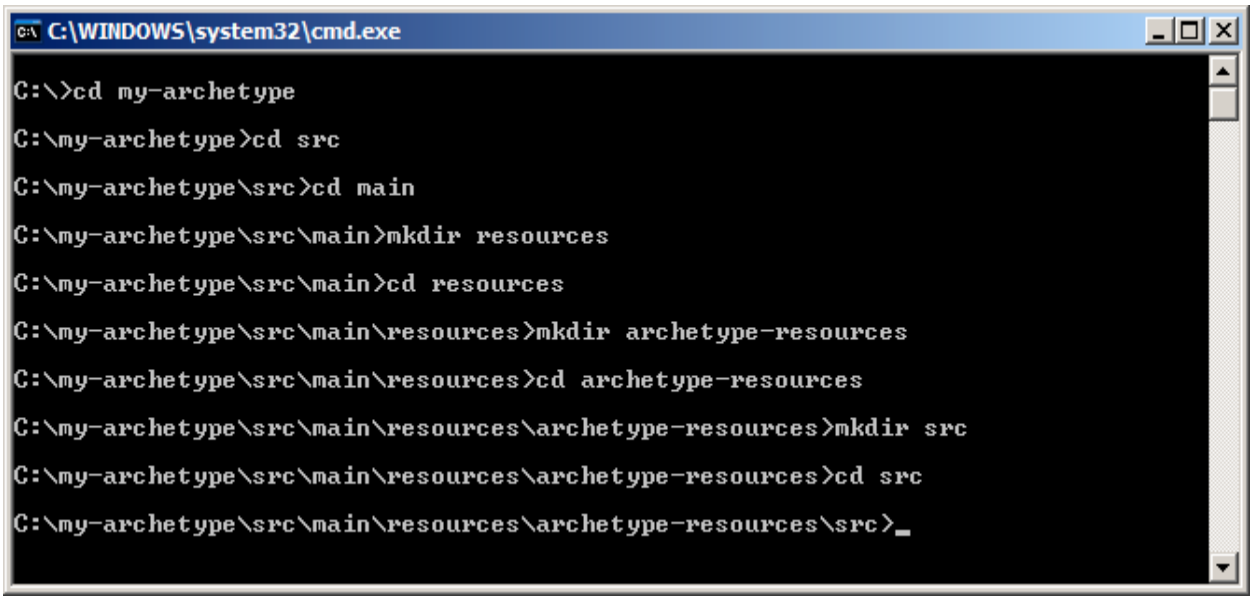
yukarı



Şekil 10.2.2-5 Proje başarı ile oluşturuldu burada dikkat etmemiz gereken archetype'ın yapısının yine var olan bir archetype ile oluşuyor olduğudur.

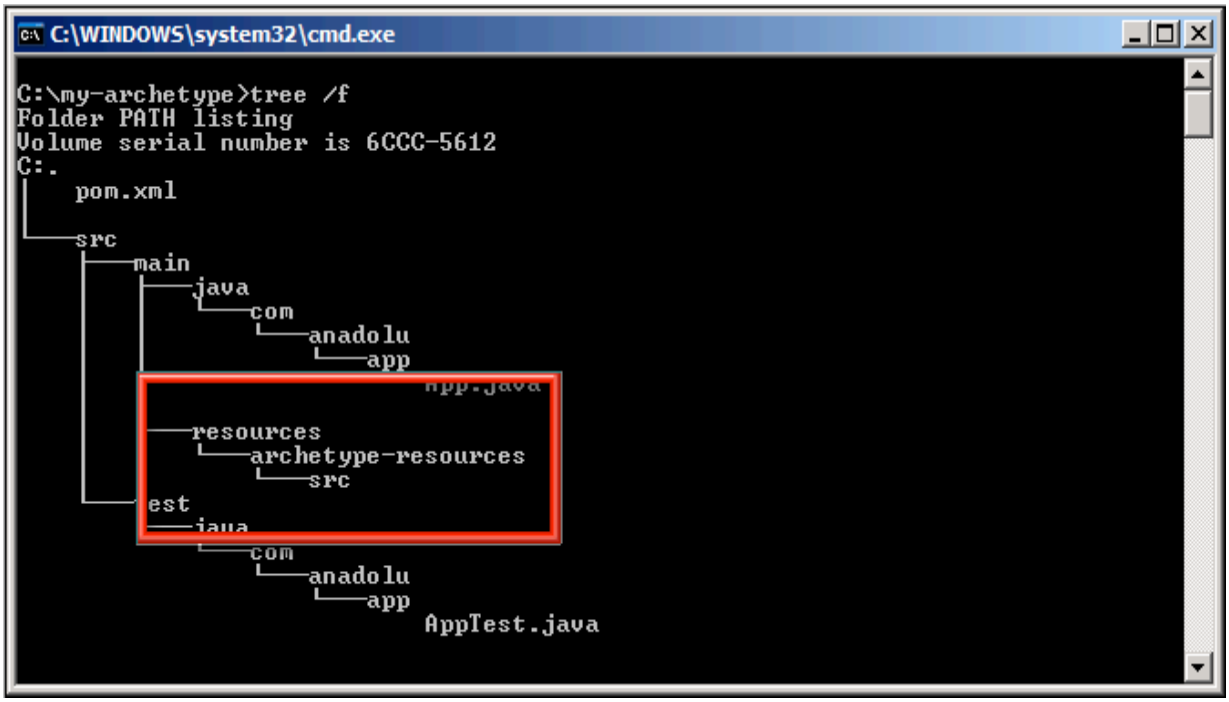
2. Adım : Biliyoruz ki şu ana kadar farklı bir yapı kullanmadık hatta var olan bir *archetype* ile yine bir proje elde ettik. Şimdi sırada bu projeye farklı bir yapı vermeye başlamakta,bunu archetype.xml ve MANIFEST.MF ile gerçekleştireceğiz.

Öncelikle **src/main/resources/archetype-resources/src** dizinini yaratalım. Bu dizinde archetype'ın kaynakları yer alacak.



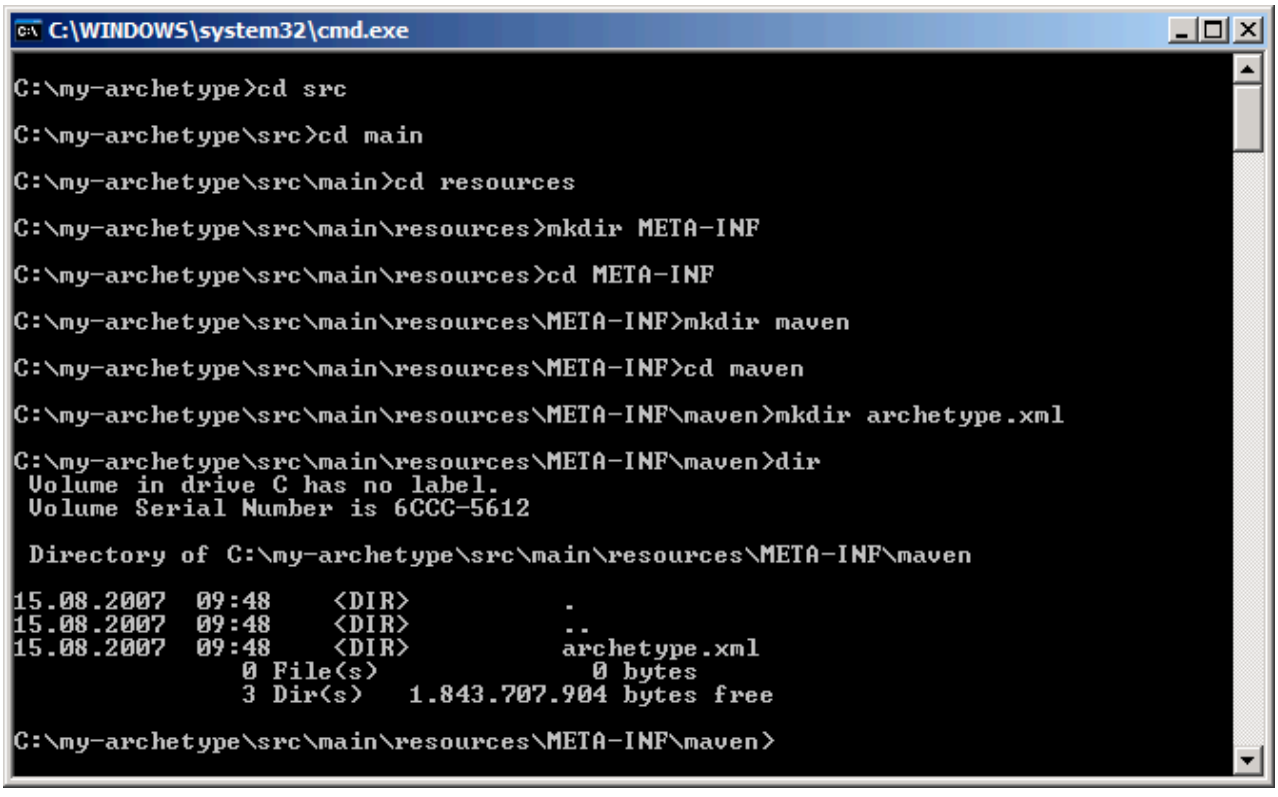
Şekil 10.2.2-6 Belirtilen dizin oluşturuldu.

yukarı



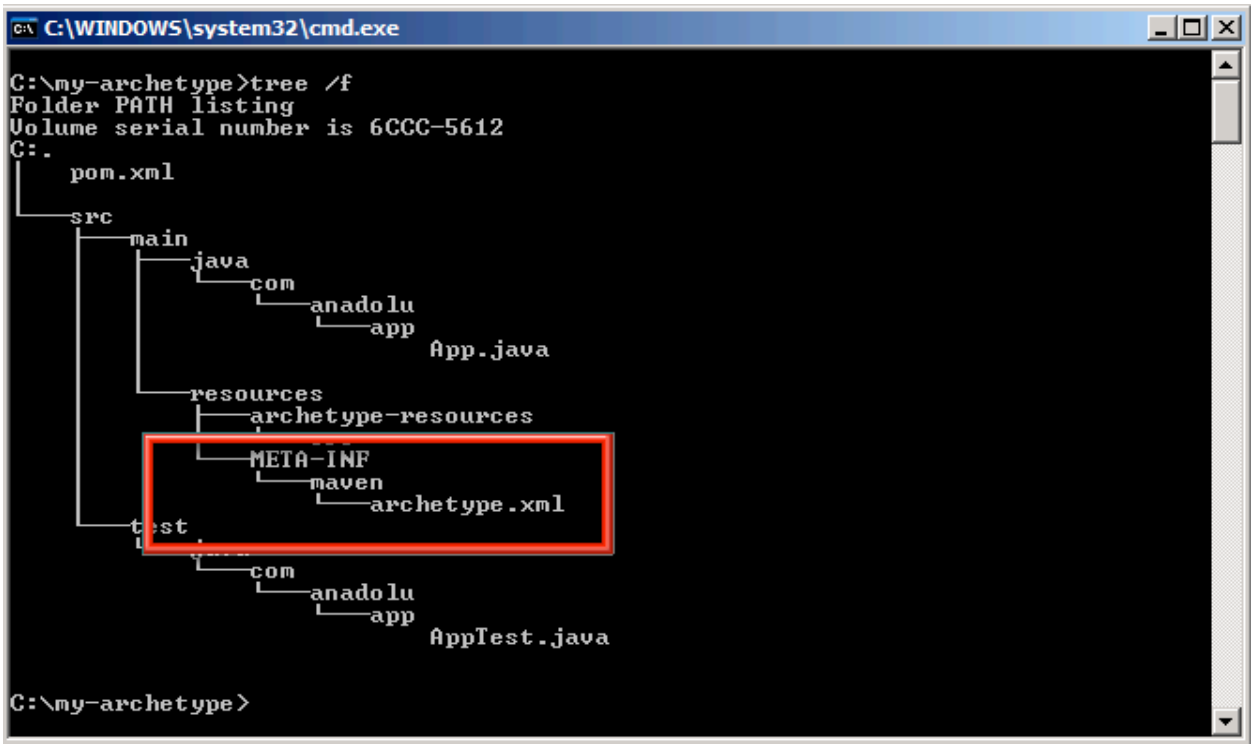
Şekil 10.2.2-7 Belirtilen dizinin görünümü.

3. Adım : src/main/resources/META-INF/maven/archetype.xml dizini oluşturulacak.



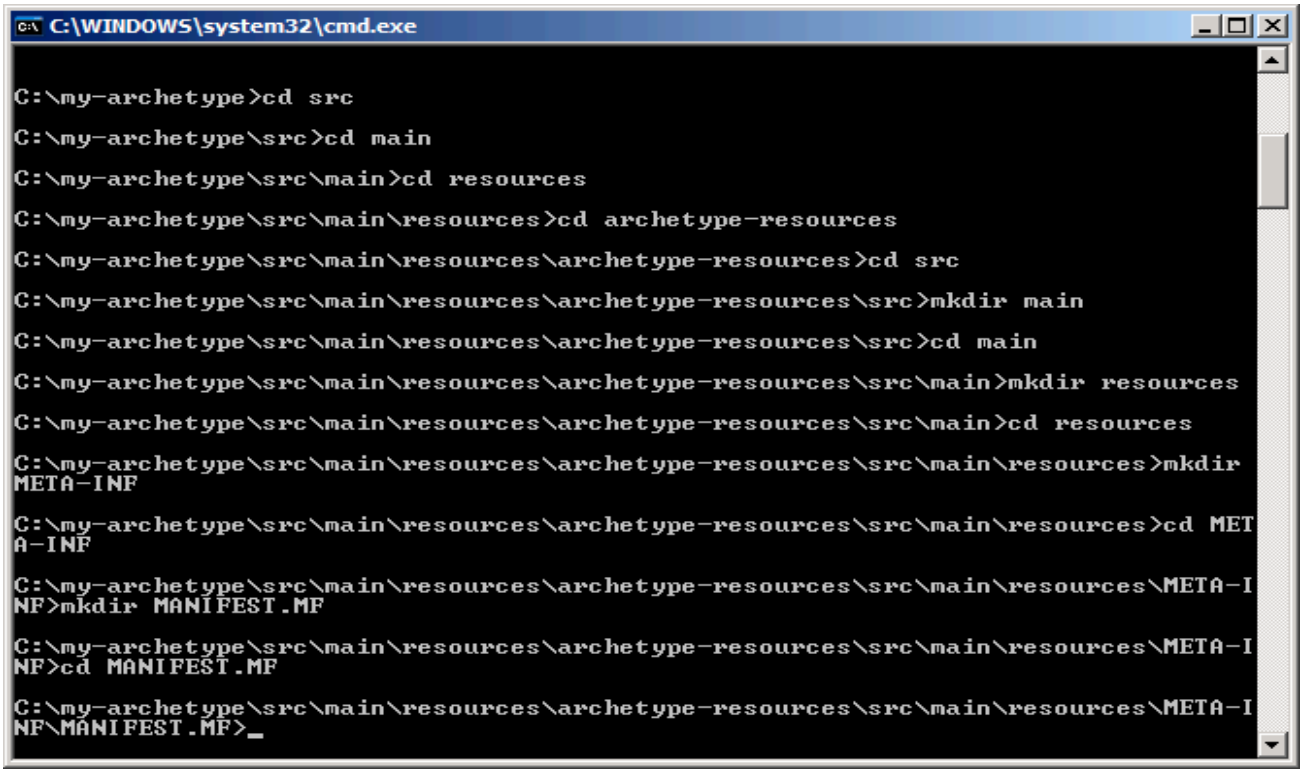
Şekil 10.2.2-8 Belirtilen dizinin oluşturulması.

yukarı



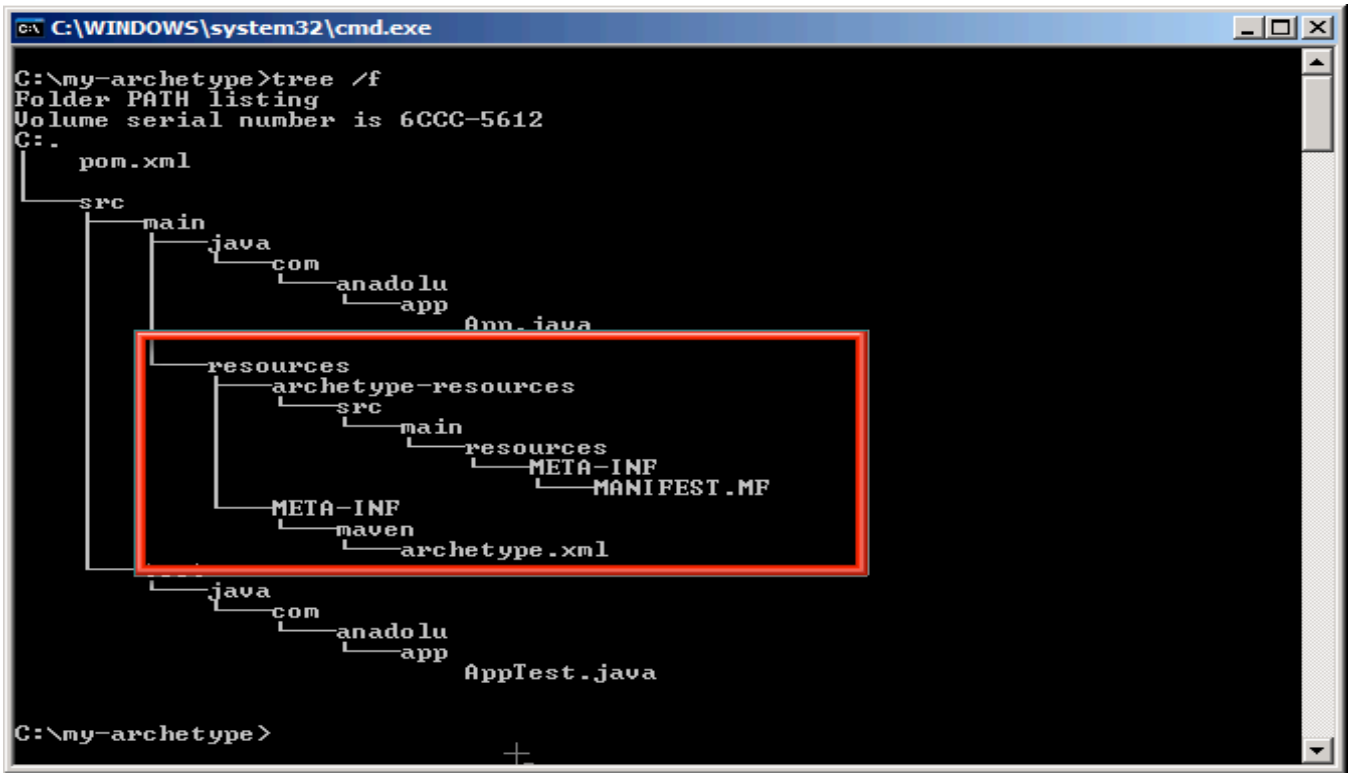
Şekil 10.2.2-9 Belirtilen dizinin görünümü.

4. Adım : **src/main/resources/archetype-resources/src/main/resources/META-INF/MANIFEST.MF** dizini oluşturulacak.



Şekil 10.2.2-10 Belirtilen dizinin yapısı.

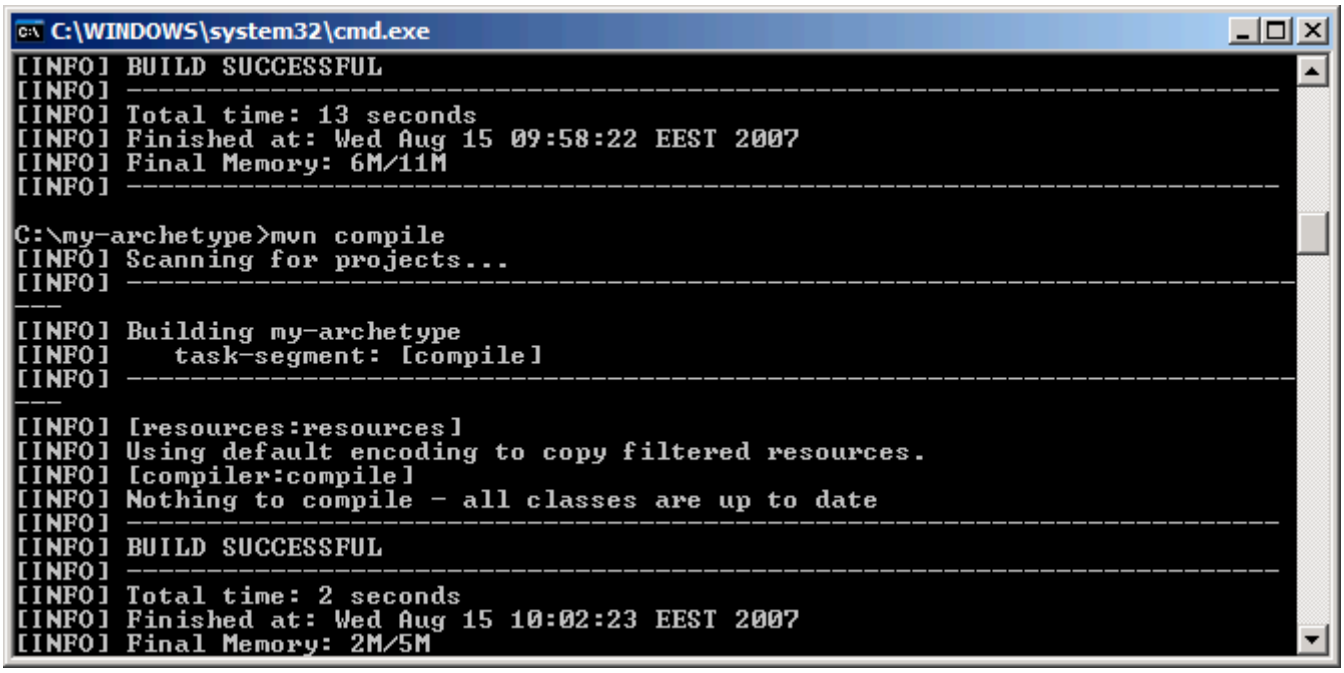
yukarı



Şekil 10.2.2-11 (Şekil 9.2.2-10)'da belirtilen dizinin görünümü.

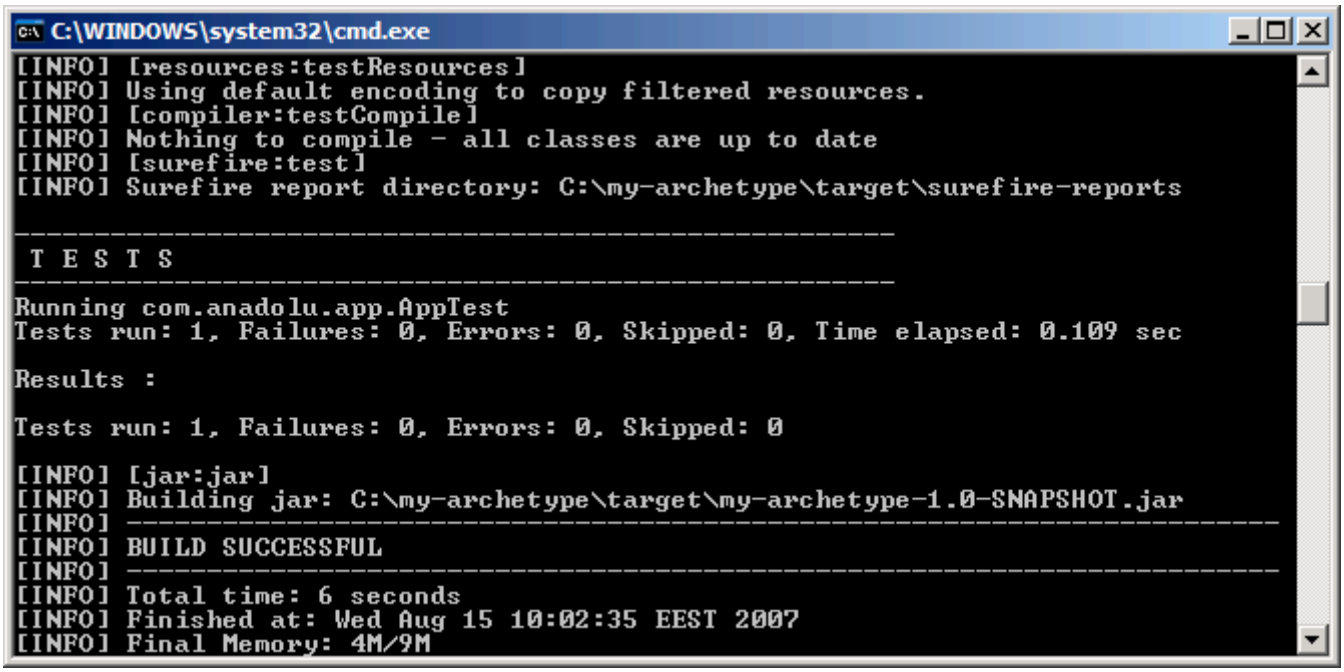
5. Adım : *mvn compile* komutu ile projemizi derleyelim, *mvn package* ile taşınabilir bir yapı kazandırarak, diğer projeler için de kullanılabilir olması için *mvn install* komutunu kullanalım.

Sırası ile yapılan işlemlerin çıktıları :

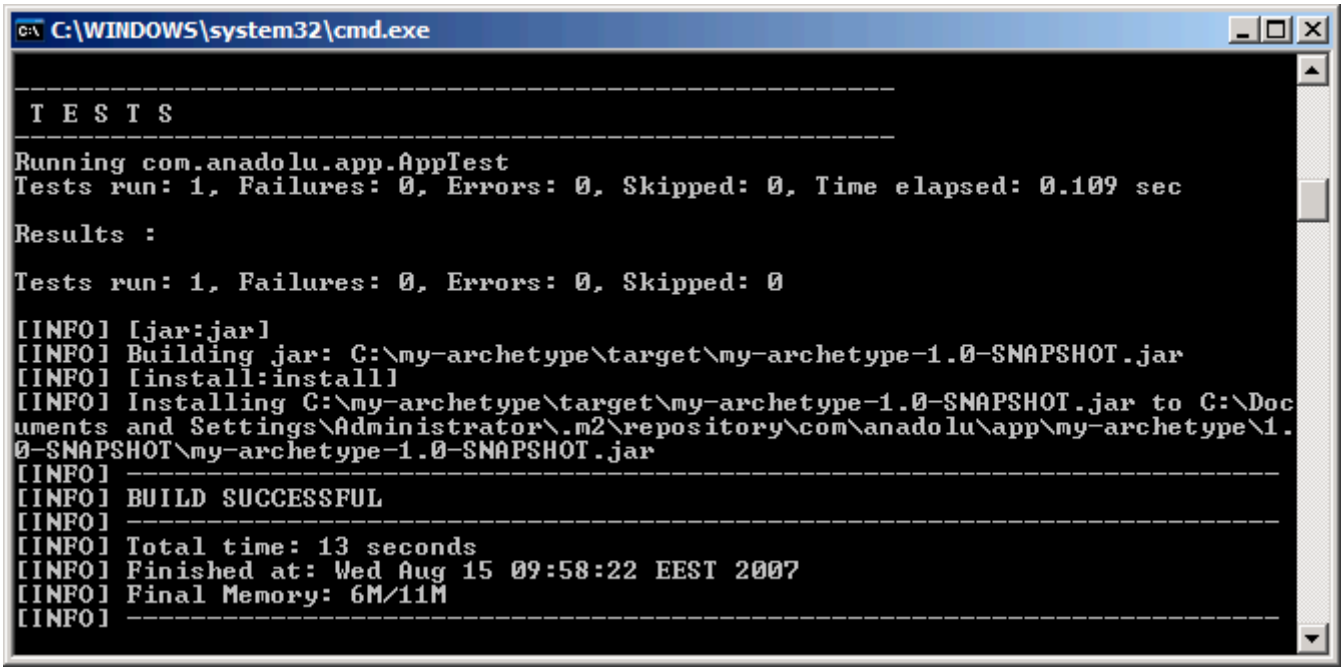


Şekil 10.2.2-12 *mvn compile* sonrası görünüm.

yukarı



Şekil 10.2.2-13 *mvn package* sonrası görünüm.



Şekil 10.2.2-14 *mvn install* sonrası görünüm.

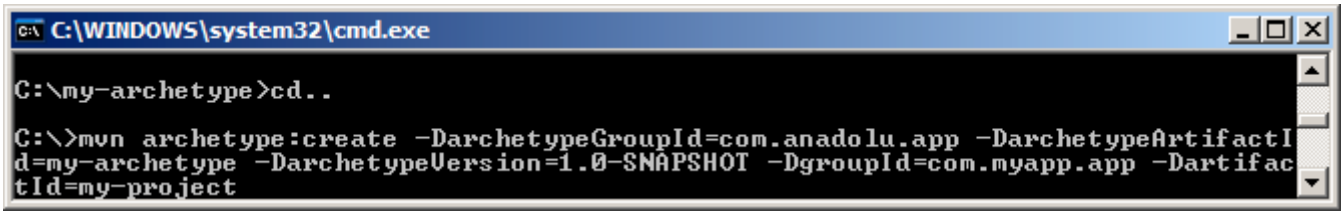
yukarı

10.2.3. Size Ait Archetype ile Proje Örneği

Şimdi artık kendi archetype'ımıza sahibiz. Bir proje oluşturarak deneyelim.

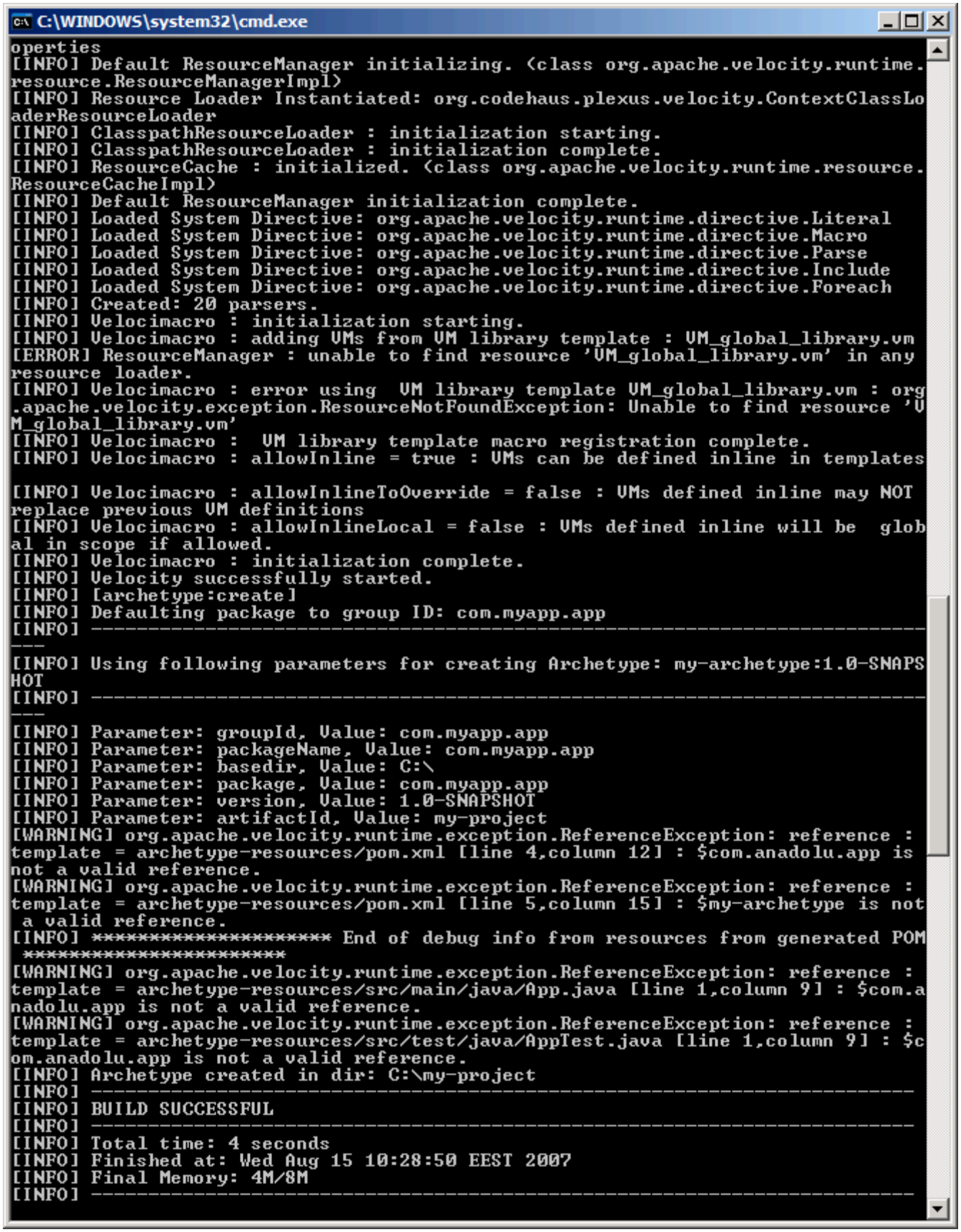
Liste 10.2.3-1 archetype'ınızı kullanmak için komut satırında yazılması gereken komutun yapısı.

```
mvn archetype:create -DarchetypeGroupId=[archetype'ınızın grup Id'si]  \
                     -DarchetypeArtifactId=[archetype'ınızın artifact Id'si]  \
                     -DarchetypeVersion=1.0-SNAPSHOT                          \
                     -DgroupId=[projenizin grup Id'si]                        \
                     -DartifactId= [projenizin artifact Id 'si]               \
```

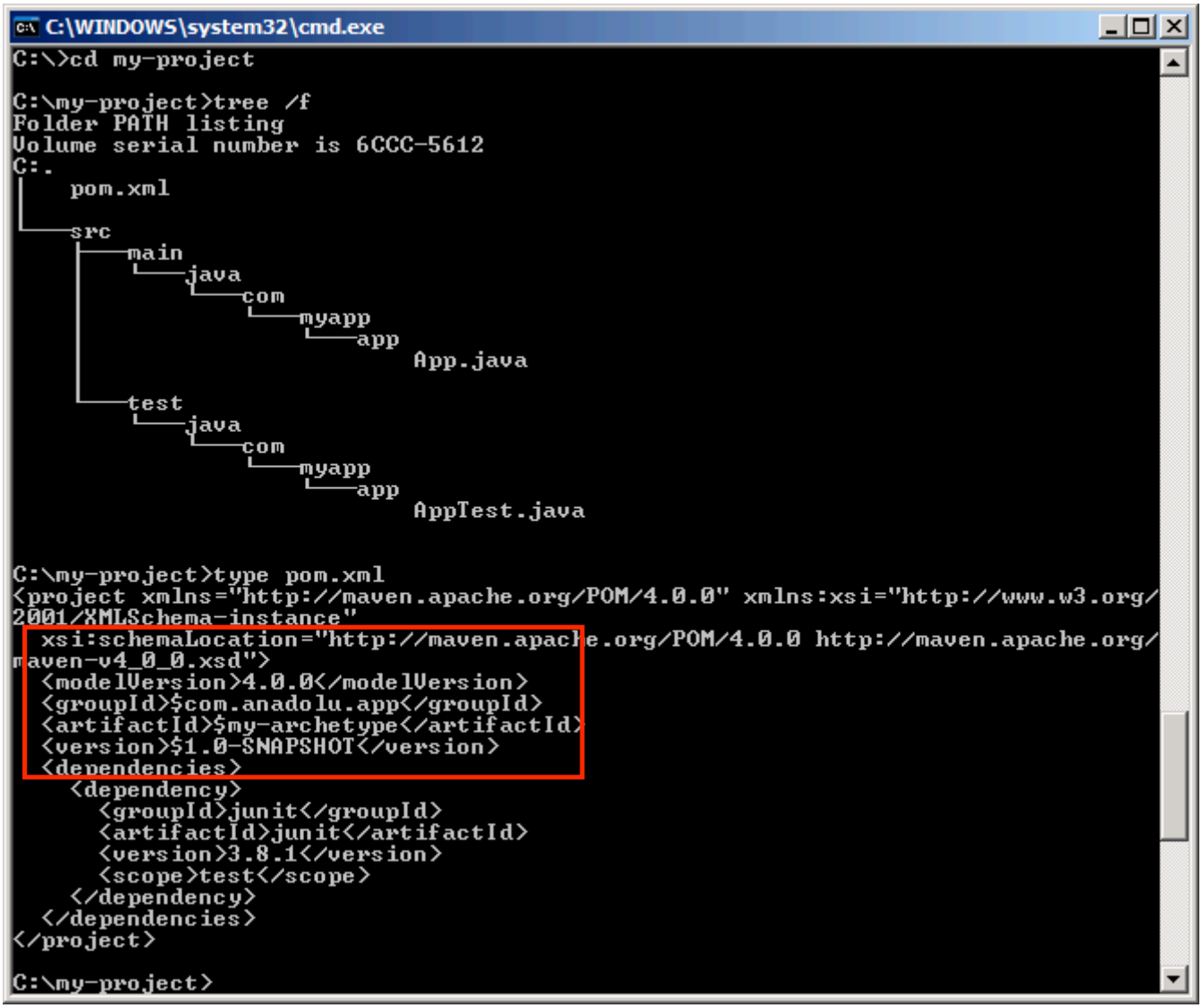


Şekil 10.2.3-2 Kendinize ait archetype'ınızla proje yaratmak için kullanılan komut.

yukarı



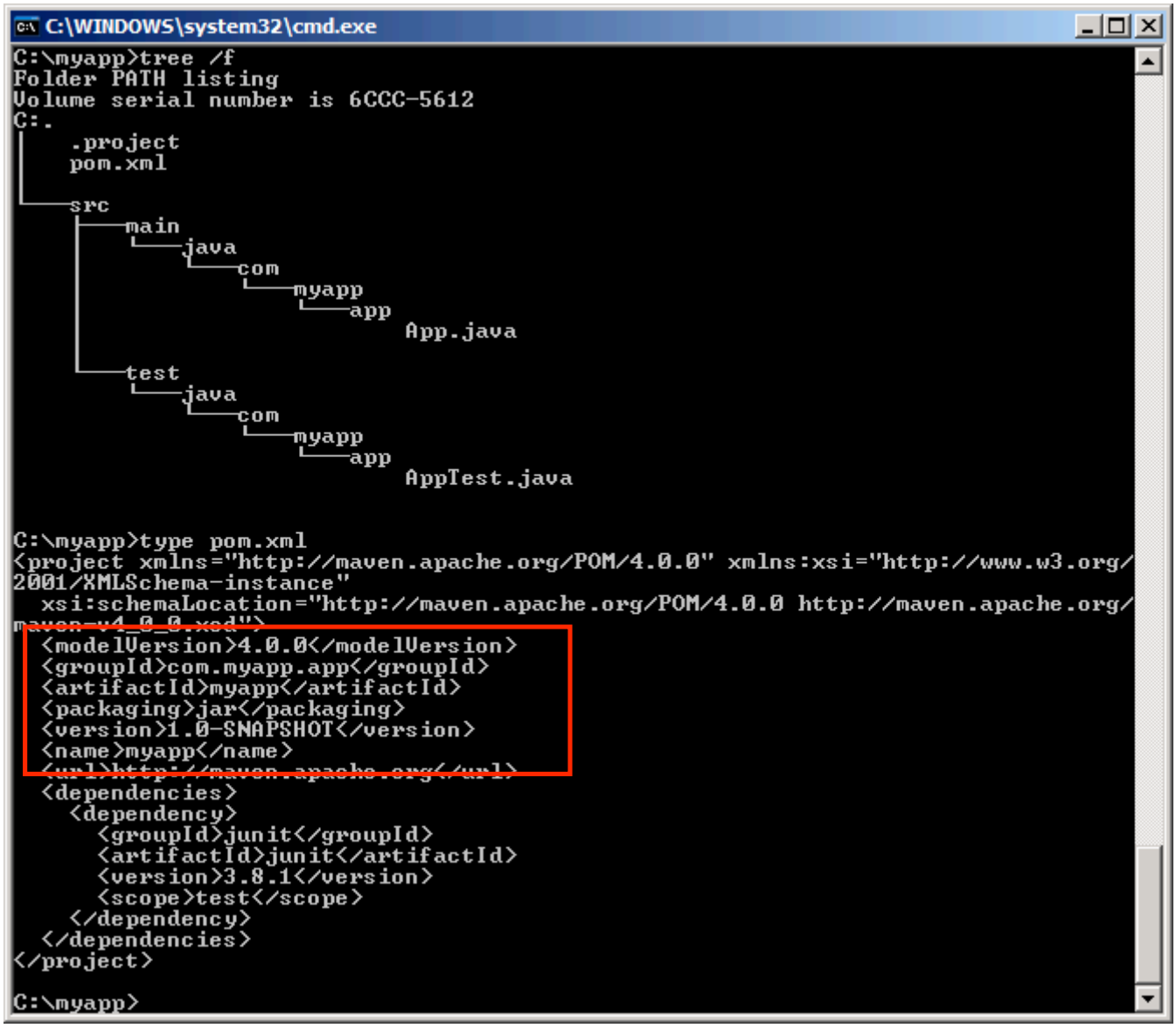
Şekil 10.2.3-3 Kendinize ait archetype'ınızla başarılı bir şekilde proje oluşturdunuz..



Şekil 10.2.3-4 Kendinize ait archetype'ınızla oluşturulan pom.xml'in içeriği.

yukarı

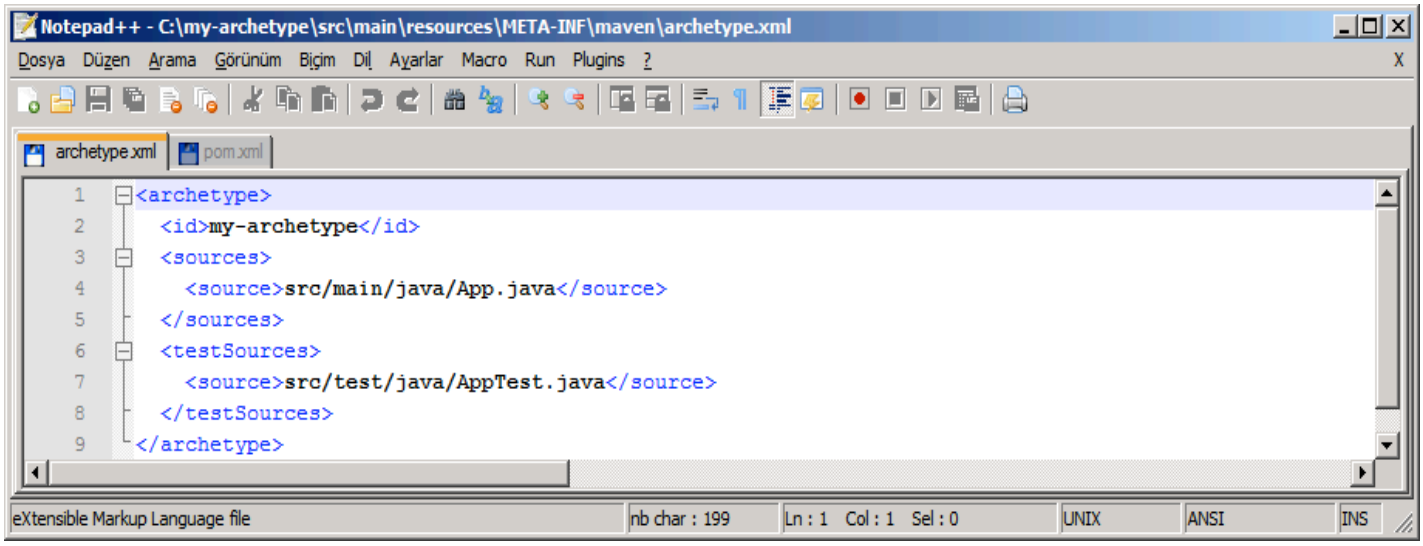
Şimdi daha önce oluşturduğumuz projenin *pom*'u ile karşılaştıralım.



Şekil 10.2.3-5 Daha önce oluşturduğumuz *myapp* projesinin POM bilgileri.

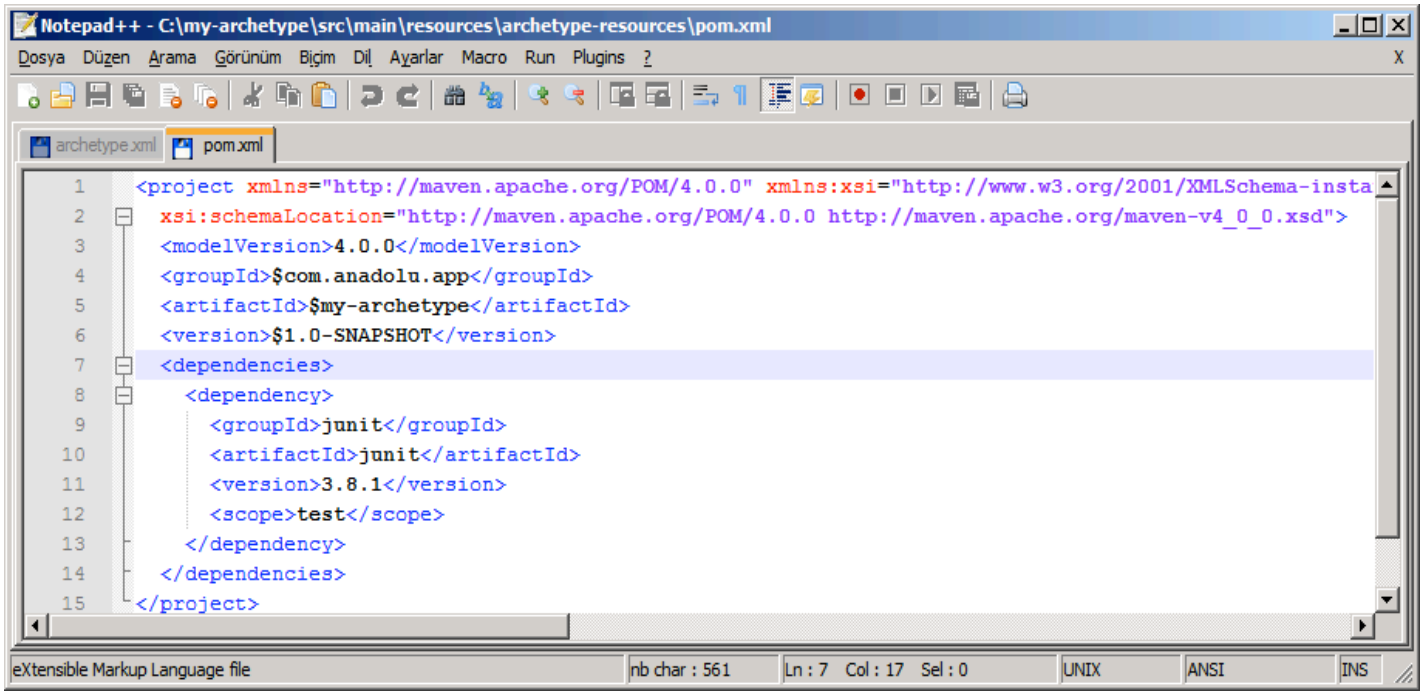
yukarı

- C:\my-archetype\src\main\resources\META-INF\maven dizini içerisinde *archetype.xml* bulunuyor.



Şekil 10.2.3-6 archetype.xml'in içeriği

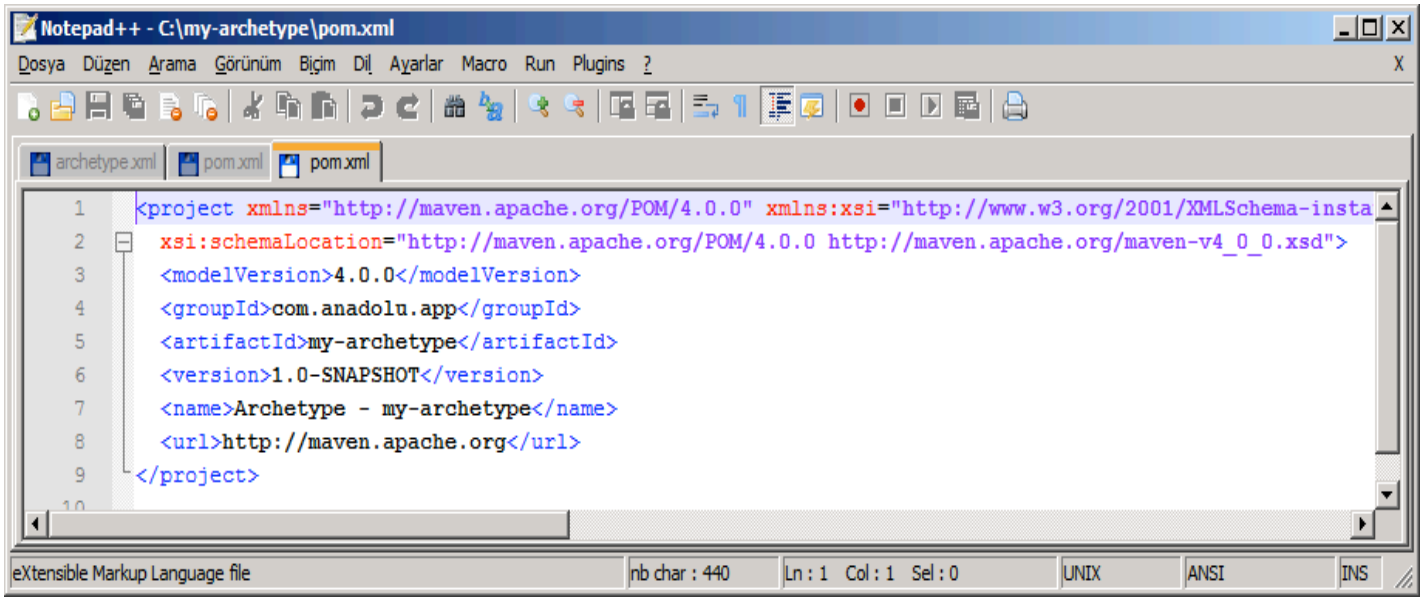
- C:\my-archetype\src\main\resources\archetype-resources dizini içerisinde archetype 'a ait *pom.xml* bulunuyor.



Şekil 10.2.3-7 archetype kaynağına ait POM.xml'in içeriği

yukarı

- C:\my-archetype dizini içerisinde projeye ait *pom.xml* bulunuyor.

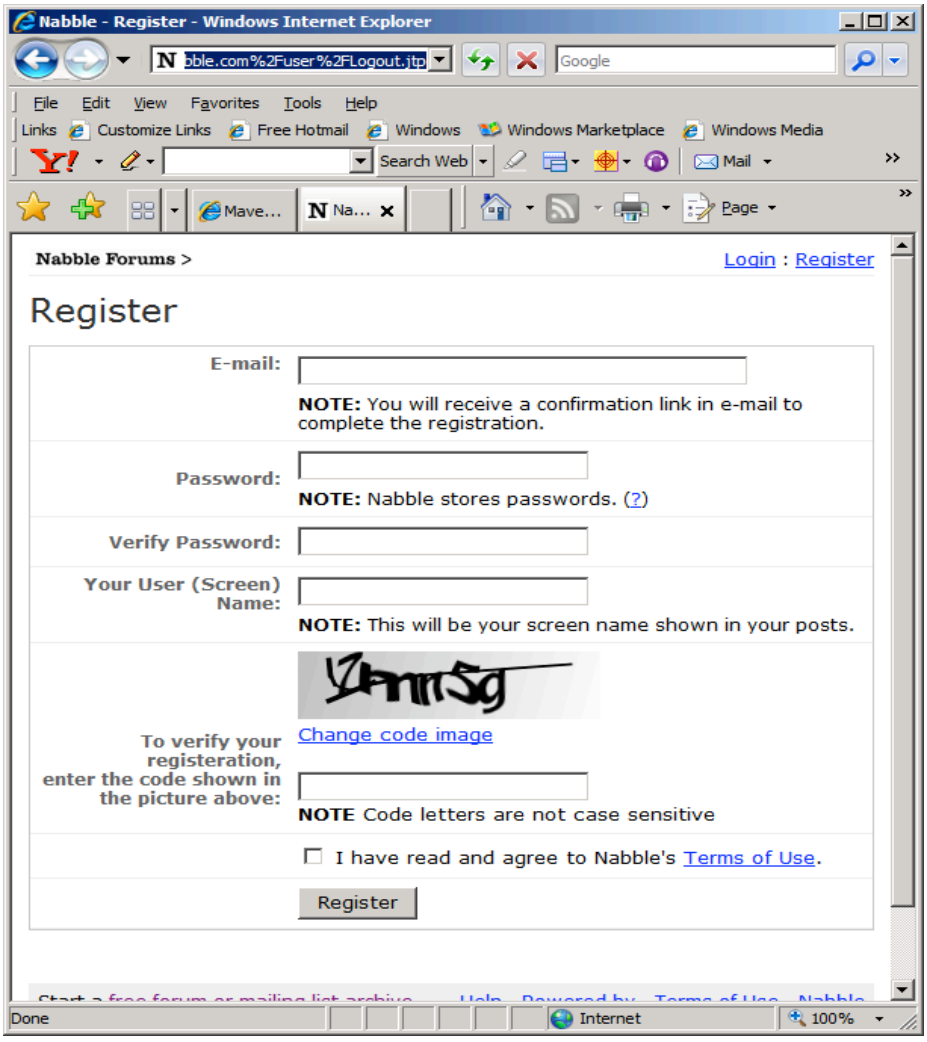


Şekil 10.2.3-8 Oluşturulan projenin POM.xml'in içeriği

yukarı

11. Geliştirici Mail Listesi (Developer Mailing List)

Adından da anlayacağınız gibi maven hakkında sorup öğrenebileceğiniz bir soru cevap ortamı sunar. Bu ortama dahil olmanız için aşağıda **Şekil 11-1** 'deki arayüze http://www.nabble.com/user/Register.jtp?nextUrl=http%3A%2F%2Fwww.nabble.com%2Fuser%2FLogout.jtp adresinden ulaşarak gerekli kısımları doldurup kayıt işleminizi tamamlayınız.



Şekil 11-1 İlgili kısımlar doldurularak kayıt işlemini tamamlayınız.

Geri

yukarı

12. JDK,SDK ve JRE arasındaki fark nedir?

12.1. JDK Nedir?

Java Development Kit. Program yazmak için gereklidir.

12.2. SDK Nedir?

Software Developmen Kit.Program yazmak ve test etmek için gereklidir.

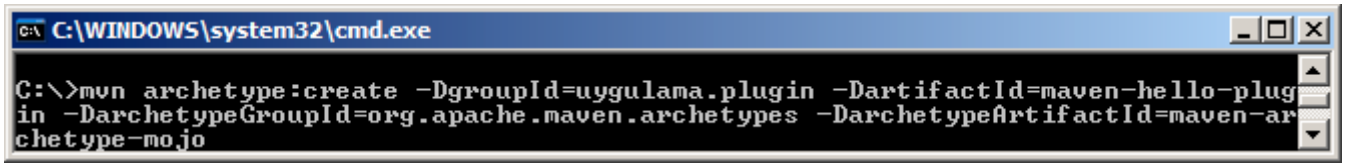
12.3. JRE Nedir?

Java kodlarını çalıştıran sistemdir.Derleyicisi yoktur.Derlenmiş programları çalıştırır.

Geri

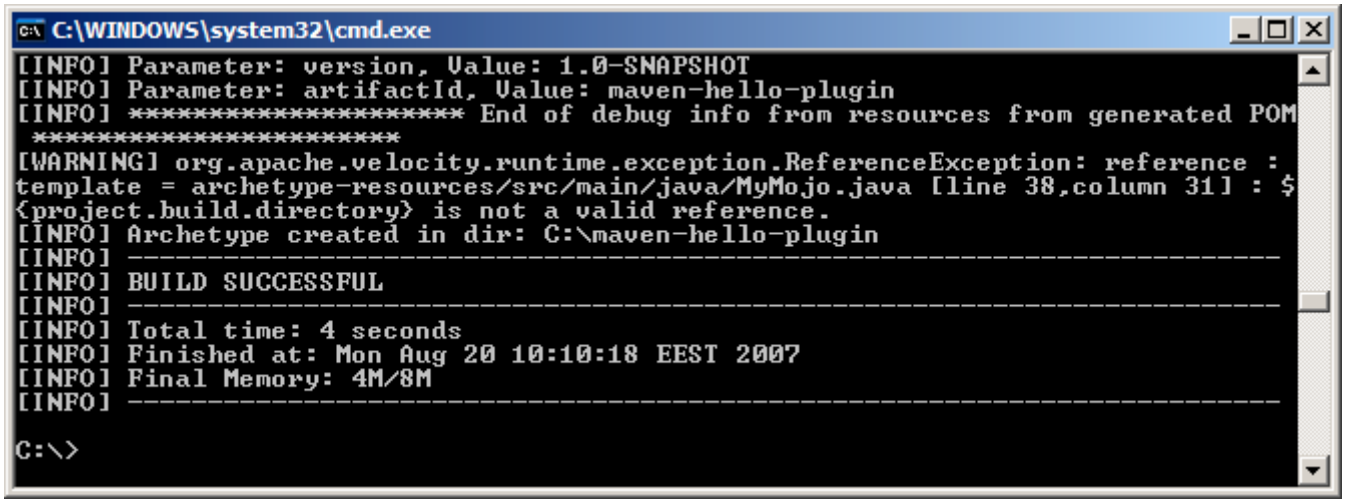
13. Mojo (Maven plain Old Java Object) Yazılımı

Mojo yazılımı ile plugin nasıl oluşturulur sorusunu yanıtlayacak ve basit parametresiz çalıştığında ekrana sadece mesaj yazan bir uygulama yazmış olacağız. Aşağıdaki gibi bir komut, komut satırında yazıldığında basit bir mojo template'i elde edilmiş oluruz.

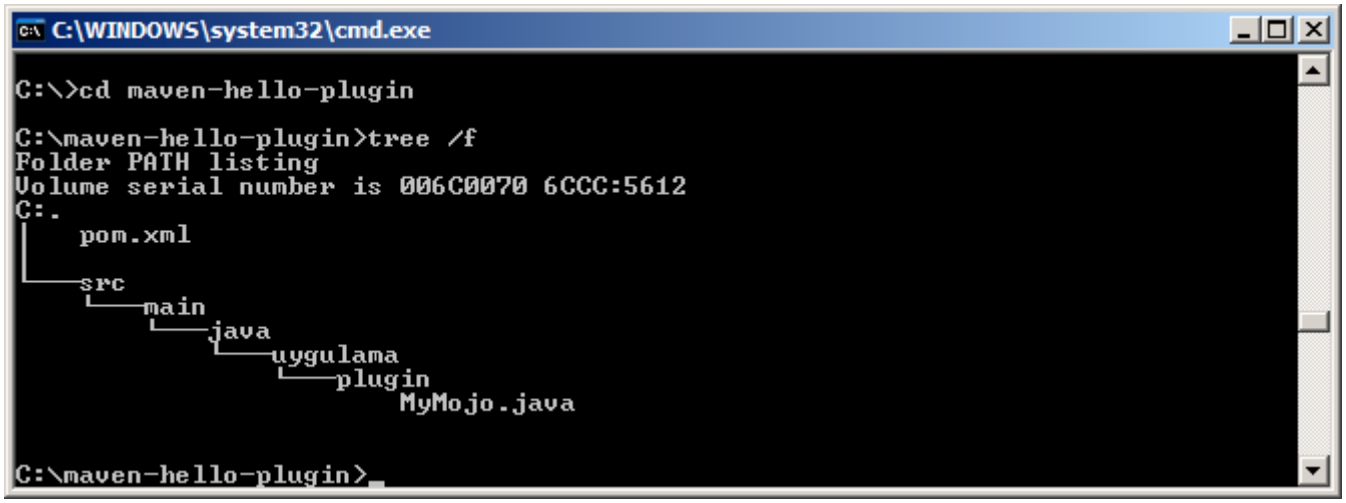


Şekil 13-1 Temel yapıda bir mojo template elde etmek için yazılan komut bütünü.

yukarı



Şekil 13-2 Temel yapıda bir mojo template elde etmek için yazılan komut bütünü başarı ile sonuçlandı.



Şekil 13-3 maven-isim-plugin'nin içeriği

Aşağıda görünen *pom.xml* komut satırının çalıştırılması ile otomatik oluşturulmaktadır.

Liste 13-1 pom.xml

```
<project>
  <modelVersion>4.0.0</modelVersion>
  <groupId>sample.plugin</groupId>
  <artifactId>maven-hello-plugin</artifactId>
  <packaging>maven-plugin</packaging>
  <version>1.0-SNAPSHOT</version>
  <name>Sample Parameter-less Maven Plugin</name>
  <dependencies>
    <dependency>
      <groupId>org.apache.maven</groupId>
      <artifactId>maven-plugin-api</artifactId>
```

```
  <version>2.0</version>
   </dependency>
  </dependencies>
</project>
```

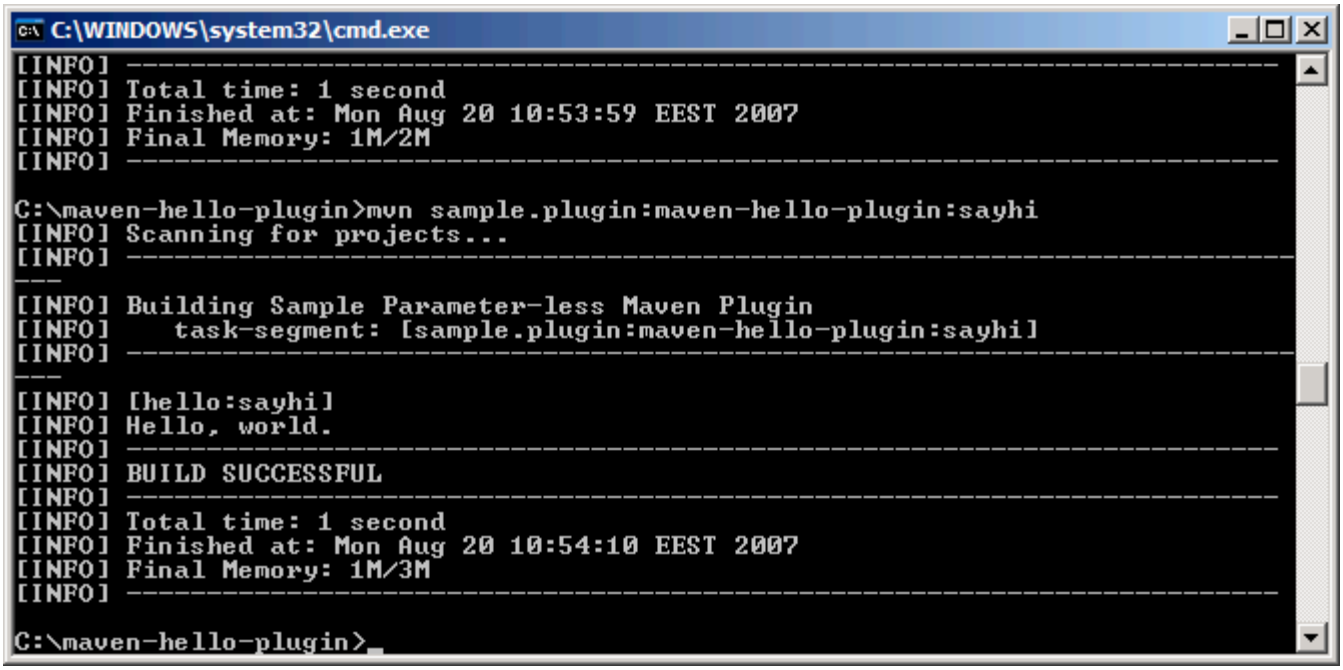
Aşağıda görünen kod parçası ise goal 'ün komut penceresinde çalışmasını sağlamak için *pom.xml* 'e eklenmelidir.

Liste 13-2 Build yapısı eklenerek komut satırında goal'ın çalışması sağlanmaktadır.

```
...
   <build>
   <plugins>
     <plugin>
       <groupId>sample.plugin</groupId>
       <artifactId>maven-hello-plugin</artifactId>
       <executions>
         <execution>
           <phase>compile</phase>
           <goals>
             <goal>sayhi</goal>
           </goals>
         </execution>
       </executions>
     </plugin>
   </plugins>
 </build>
…
```

Goal'ü çalıştırmak için gerekli olan komut satırı aşağıda verilmiştir.

mvn sample.plugin:maven-hello-plugin:1.0-SNAPSHOT:sayhi ya da mvn hello:sayhi



Şekil 13-4 INFO kısmında Hello çıktısını görebilirsiniz.

14. Maven 'la Çalışmak

Büyük projeler üzerinde çalışırken proje büyüdükçe ve geliştirici sayısı birden fazlaysa projenin değişen kısımları arttığından işlemler karmaşıklaşmaya başlar. Maven'ın standart bir klasör yapısının var olması, yerel bir deposunun olması ile işlemlerde hızı desteklemesi, versiyon yönetimi gibi özelliklerin bulunuyor olması grup çalışmalarında büyük bir kolaylık sunmaktadır.

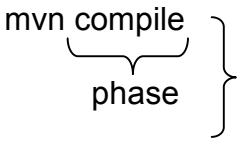
Maven Continuum ile organizasyona ait POM oluşturularak takım bağımlılıkları tanımlanır. Bu takım çalışmaları için proje yönetimini kolaylaştırmaktadır.

Maven projeleri küçük parçaların bir bütünüdür. Yapısında yapabileceğiniz değişikliklerle size ait olduğunu hissettiren bir özelliğe sahiptir. Kendinize ait hazırlayabileceğiniz bir archetype ile projenin dizin yapısını kendinize göre dizayn edebilir derleneme işleminden sonra kaynakları nerde görmek istiyorsanız oraya yönlendirebilirsiniz.

Maven ile bulacağınız kaynakların çoğu yabancı içerikli olup henüz yeni ismini duyuran kullanılır bir program olması ile gelecek vaat ediyor. Özellikle grup çalışmalarında tercih ediliyor, her şeyin sizin kontrolünüzde olduğunu hissettiren bir yönetim kolaylığı sunması, *mvn site* komutu ile kolaylıkla otomatik oluşabilen sitenin yönetim desteği gibi, maven'ın çekici tarafı olsa gerek. Kullanılan phase (komut)'lerin pluginler aracılığı ile işlevleri değiştirilebiliyor. Depodan alarak kullanılan jar dosyaları farklı pluginler altında da yer alsa bir tek plugin altında toplayarak kullanılabiliyor.

yukarı

Örneğin :



Plugin'ler phase'ler ile tetiklenir.

Gerekli programların kurulmasına gerek kalmadan sadece gerekli kısımlarının repository (depo)'ye indirilmesinden sonra Maven aracılığı ile çalıştırabilirsiniz.

Örneğin :

tomcat

Kaynaklar

- www.maven.apache.com
- www.devx.com
- http://www.jajakarta.org/turbine/en/turbine/maven/
- http://www.javaci.net/
- http://maven.apache.org/
- http://maven.apache.org/guides/plugin/guide-java-plugin-development.html
- http://docs.codehaus.org/display/MAVENUSER/Archetypes+List
- http://mirrors.ibiblio.org/pub/mirrors/maven2/
- http://repo1.maven.org/maven2/
- http://www.sonatype.com/book/lifecycle.html
- http://www.bilgidata.com/localhost/bilgidata/pdfs/maven.pdf

yukarı