



**LAITEC**

Learn All Information Technology

## Apache Maven

امیر سام بهادر

امیر نوروزی فرد

## خلاصه

بسیاری از برنامه نویسان Maven را یک ابزار Build میدانند. ابزاری که برای ساختن محصولات ( Artifacts ) از Source Code مورد استفاده قرار میگیرد. نگاه مدیران پروژه و مهندسان نرم افزار به Maven گسترده تر است : یک ابزار مدیریت پروژه. تفاوت این دو دیدگاه چیست ؟ یک ابزار Build مانند Ant فقط بر روی پیش پردازش (Preprocessing) ، کامپایل ، بسته بندی ( Packaging ) ، آزمایش و توزیع متمرکز میباشد. یک ابزار مدیریت پروژه مانند Maven یک سری ویژگی های مافوق ابزار Build را در اختیار شما قرار میدهد. Maven علاوه بر این که قابلیت های Build را داراست ، همچنین در تهیه گزارشات ، تولید Web Site و تسهیل ارتباط بین اعضای تیم توسعه نرم افزار استفاده میشود. به عبارت دیگر ، هر چقدر که پروژه های متن باز به Maven به عنوان یک سکوی مدیریت منتقل میشوند ، توسعه دهندگان نیز به این نتیجه رسیده اند که Maven فقط ابزار ساده سازی مدیریت Build نیست ، بلکه یک رابط مشترک بین توسعه دهندگان و پروژه های نرم افزاری می باشد. هدف از این مقاله یادگیری مفاهیم بنیادی Apache Maven در قالب چند مثال خواهد بود. ما در این مقاله IntelliJ IDEA را به عنوان IDE انتخاب کرده ایم. امیدواریم شما با پایان رساندن این مقاله درک درستی از ایده Maven برای ساخت پروژه پیدا کنید.

**کلمات کلیدی :** Apache Maven ، Project Object Model (POM) مدل شیء پروژه ، Dependency ، (وابستگی) ، Plugin ، Repositories (مخازن)

## ۱- معرفی

تعریف Apache Maven به این صورت است که : Maven یک ابزار مدیریت پروژه است که شامل یک Project Object Model (POM) ، یک سری از استانداردها ، طول عمر پروژه (Lifecycle) ، سیستم مدیریت وابستگی (Dependency) و منطقی برای اجرای اهداف Plugin در فازهای تعریف شده طول عمر می باشد.

اجازه ندهید این حقیقت که Maven یک ابزار مدیریت پروژه است شما را بترساند. اگر شما فقط در جستجوی یک ابزار Build هستید Maven این کار را به خوبی انجام می دهد و همچنین آن را به صورت کامل توزیع می کند. برجسته ترین مشخصه Maven این است قادر است به صورت خودکار وابستگی های کتابخانه ها را از مخازن پیش فرض بارگذاری دانلود نماید. Maven برای نگهداری اطلاعات مربوط به وابستگی ها از یک فایل XML استفاده میکند. البته مواردی مانند کامپایل کد و بسته بندی (Packaging) نیز به صورت پیش فرض در این فایل قرار داده میشوند که در قسمت های بعدی به آن اشاره میکنیم.

## ۲- مفهوم مدل یک پروژه ( Model of a Project )

Maven یک مدل از پروژه را ایجاد مینماید. شما فقط Source Code را به Byte Code کامپایل نمی کنید بلکه شما تعریف یک پروژه نرم افزاری را توسعه می دهید و یک سری از مختصات واحد و یکتا را به آن اختصاص می دهید شما در واقع خصوصیات نرم افزار را توصیف می کنید : مجوز این پروژه چیست ؟ چه کسی آن را توزیع کرده است ؟ چه پروژه ای به این پروژه وابستگی دارد ؟ Maven فقط یک پیشرفت در ابزاری مانند ant و make نیست. بلکه یک Platform است که معانی جدیدی از توسعه نرم افزار را در بر میگیرد. این تعریف از مدل برای هر پروژه نرم افزاری تعدادی از ویژگی ها را بوجود می آورد :

#### ۱- مدیریت وابستگی ها (Dependency Management)

به این دلیل که یک پروژه به وسیله یک سری از مختصات که شامل شناسه گروه (Group ID) ، شناسه محصول (Artifact ID) و ورژن تعریف می شود ، پروژه ها می توانند از این مختصات برای شناساندن وابستگی ها استفاده کنند.

#### ۲- مخازن راه دور (Remote Repositories)

در ارتباط با مدیریت وابستگی ، ما میتوانیم از مختصاتی که در مدل شیء پروژه (POM) ، Maven تعریف شده است برای بوجود آوردن مخازن محصولات Maven استفاده کنیم.

#### ۳- باز استفاده عمومی از منطق ساخت (Universal Reuse Of Build Logic)

Plugin ها شامل منطقی می شوند که با داده های توصیفی و پارامترهای پیکر بندی شده در مدل شیء پروژه (POM) کار می کنند. آن ها به صورتی طراحی نشده اند که فقط با فایل هایی که در موقعیت های مشخص قرار دارند کار کنند.

#### ۴- Easy Searching and Filtering of Project Artifacts (فیلترینگ و جستجوی راحت در محصولات پروژه)

ابزاری مانند Nexus به شما اجازه می دهد که محتویات یک مخزن را index کرده و search نمایید . که این کار با استفاده از اطلاعات ذخیره شده در POM انجام میشود.

#### ۵- Integration / Tool Portability (قابلیت حمل ابزار ، یکپارچگی )

امروزه ابزارهایی مانند Eclipse ، Netbeans ، IntelliJ IDEA دارای مکان های مشترکی برای پیدا کردن اطلاعات پروژه هستند. در واقع Maven توانسته است به وسیله مدل شیء پروژه (POM) یک استاندارد را برای IDE های مختلف ایجاد نماید.

## نصب و پیکربندی Maven

### نصب Maven بر روی Windows

نیازی نیست که Apache Maven را به عنوان یک سرویس در ویندوز نصب کنیم ، شما فقط باید فایل zip شده Maven را دانلود و آن را extract کرده و Windows Environment Variable را پیکربندی کنید.

ابزار مورد نیاز :

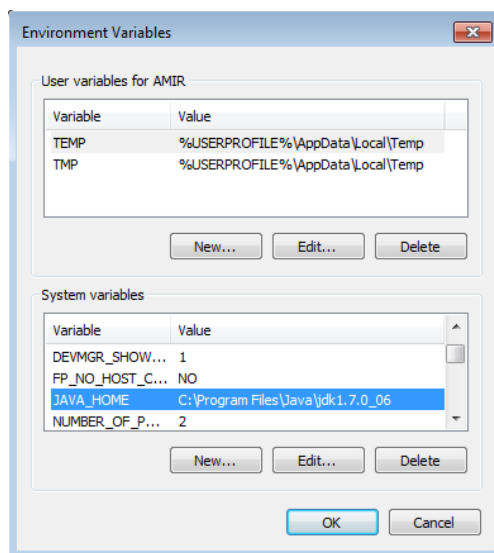
1-JDK 1.6 or Higher

2-Maven 3.0.4

3-Windows 7

### 1- JDK and JAVA\_HOME :

با رفتن به Windows Environment Variable مطمئن شوید که JAVA\_HOME اضافه شده است و محل نصب JDK را نشان میدهد.



### ۲- دانلود Apache Maven

فایل فشرده Maven را از آدرس <http://maven.apache.org/download.html> دانلود نمایید . ورژن Maven مورد نظران را میتوانید از این سایت انتخاب کرده و دانلود کنید ما در این مقاله از apache-bin.zip-3.0.4 maven استفاده میکنیم.

### ۳- Extract It

فایل دانلود شده در پوشه دلخواه Extract نمایید ما آن در درایو C ، Extract کرده و نام آن را به عنوان مثال به C:\maven تغییر داده ایم.



```
Java version: 1.7.0_06
Java home: C:\Program Files\Java\jdk1.7.0_06\jre
Default locale: en_US, platform encoding: Cp1252
OS name: "windows 7" version: "6.1" arch: "x86" Family: "windows"
```

## نصب Maven بر روی Linux

نیازی نیست که Apache Maven را به عنوان یک سرویس در لینوکس نصب کنیم ، شما فقط باید فایل zip شده Maven را دانلود و آن را مثلا در /home/amir ، extract کرده و .bash\_profile کاربر amir را پیکربندی کنید.

ابزار مورد نیاز :

1-JDK 1.6 or Higher

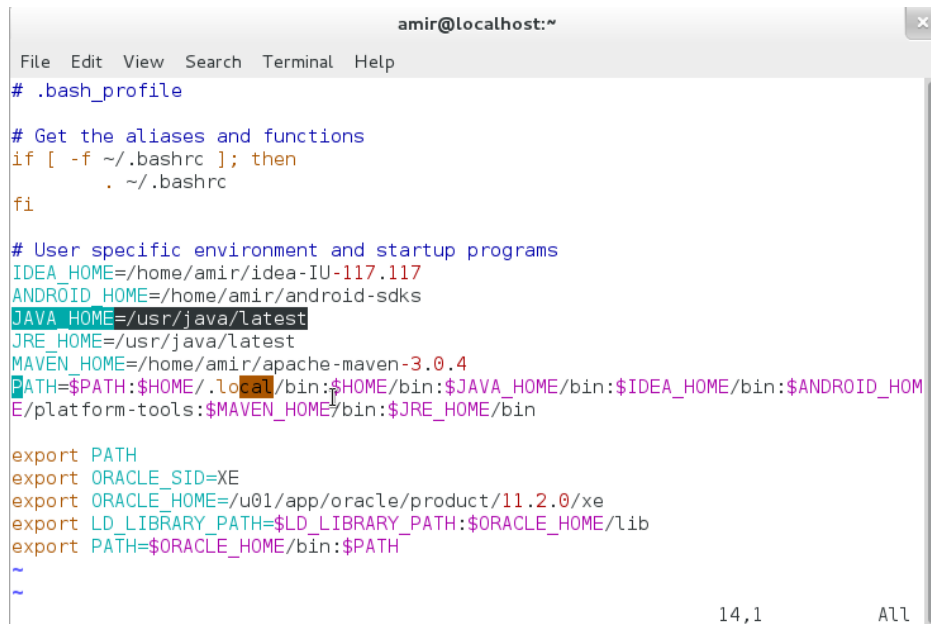
2-Maven 3.0.4

3-Fedora Linux

با رفتن به .bash\_profile کاربر amir مطمئن شوید که JAVA\_HOME و MAVEN\_HOME اضافه شده است و محل نصب JDK و Maven را نشان میدهد.

با استفاده از متغیر PATH مانند شکل زیر دقیقا محل پوشه bin را نیز مشخص می کنیم.

```
[amir@localhost ~]$ vim .bash_profile
```



```
amir@localhost:~  
File Edit View Search Terminal Help  
# .bash_profile  
  
# Get the aliases and functions  
if [ -f ~/.bashrc ]; then  
    . ~/.bashrc  
fi  
  
# User specific environment and startup programs  
IDEA_HOME=/home/amir/idea-IU-117.117  
ANDROID_HOME=/home/amir/android-sdks  
JAVA_HOME=/usr/java/latest  
JRE_HOME=/usr/java/latest  
MAVEN_HOME=/home/amir/apache-maven-3.0.4  
PATH=$PATH:$HOME/.local/bin:$HOME/bin:$JAVA_HOME/bin:$IDEA_HOME/bin:$ANDROID_HOM  
E/platform-tools:$MAVEN_HOME/bin:$JRE_HOME/bin  
  
export PATH  
export ORACLE_SID=XE  
export ORACLE_HOME=/u01/app/oracle/product/11.2.0/xe  
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$ORACLE_HOME/lib  
export PATH=$ORACLE_HOME/bin:$PATH  
~  
~  
14,1 All
```

برای اطمینان از اجرای صحیح Maven دستور زیر را خط فرمان Linux صادر کنید نتیجه باید مانند شکل زیر باشد.

```
[amir@localhost ~]$ mvn -version  
  
Apache Maven 3.0.4 (r1232337; 2012-01-17 12:14:56+0330)  
  
Maven home: /home/amir/apache-maven-3.0.4  
  
Java version: 1.7.0_10, vendor: Oracle Corporation  
  
Java home: /usr/java/jdk1.7.0_10/jre  
  
Default locale: en_US, platform encoding: UTF-8  
  
OS name: "linux", version: "3.6.8-2.fc17.x86_64", arch: "amd64", family: "unix"
```

## پیکربندی Maven در Proxy

اگر شما در محل کار پشت یک Proxy Server قرار دارید و نمیتوانید به صورت مستقیم به اینترنت متصل شوید Maven هم نمیتواند وابستگی های مربوط به پروژه را از مخازن دانلود نماید. برای اینکه این مشکل را برطرف کنیم باید Proxy Server را در فایل پیکربندی Maven که در `{MAVEN_HOME}/conf/settings.xml` قرار دارد تعریف کنیم.

فایل `{MAVEN_HOME}/conf/settings.xml` را پیدا کرده و جزییات Proxy را در آن وارد میکنیم.

`{MAVEN_HOME}/conf/settings.xml`

```

<!-- proxies
| This is a list of proxies which can be used on this machine to connect to the network.
| Unless otherwise specified (by system property or command-line switch), the first proxy
| specification in this list marked as active will be used.
|-->
<proxies>
<!-- proxy
| Specification for one proxy, to be used in connecting to the network.
|
<proxy>
  <id>optional</id>
  <active>true</active>
  <protocol>http</protocol>
  <username>proxyuser</username>
  <password>proxypass</password>
  <host>proxy.host.net</host>
  <port>80</port>
  <nonProxyHosts>local.net|some.host.com</nonProxyHosts>
</proxy>
-->
</proxies>

```

مانند شکل زیر مقادیر Proxy را از comment در آورده و جزئیات Proxy Server را در آن اضافه میکنیم به عنوان مثال :

```

<!-- proxies
| This is a list of proxies which can be used on this machine to connect to the network.
| Unless otherwise specified (by system property or command-line switch), the first proxy
| specification in this list marked as active will be used.
|-->
<proxies>
<proxy>
  <id>optional</id>
  <active>true</active>
  <protocol>http</protocol>
  <username>amir</username>
  <password>password</password>
  <host>proxy.amir.com</host>

```



```
<port>8888</port>  
  
<nonProxyHosts>local.net|some.host.com</nonProxyHosts>  
  
</proxy>  
  
</proxies>
```

بعد از انجام تغییرات فایل را ذخیره میکنیم.

## مخازن در Apache Maven

### مخزن محلی Maven (Maven local repository)

مخزن محلی Maven یک پوشه است که برای ذخیره سازی تمام وابستگی های پروژه ( Plugin Jars و فایل های دیگری که توسط Maven دانلود شده اند) مورد استفاده قرار می گیرد. به عبارت ساده تر هنگامی که شما یک پروژه Maven را Build میکنید همه ی وابستگی ها در مخزن محلی ذخیره می شود.

به صورت پیش فرض مخزن محلی Maven در پوشه `.m2` قرار دارد.

۱. Unix/Mac OS X – `~/m2`

۲. Windows – `C:\Documents and Settings\{your-username}\m2`

### بیکربندی مخزن محلی Maven (Maven local repository)

ما نام این محل ذخیره سازی پیش فرض (`.m2`) را به یک نام شناخته شده تر مثل `c:/maven_repo` تغییر می دهیم.

فایل `{MAVEN_HOME}/conf/settings.xml` را باز کرده و مقدار `localRepository` را به `c:/maven_repo` تغییر میدهیم.

```
{MAVEN_HOME}\conf\setting.xml
<settings>
  <!-- localRepository
  | The path to the local repository maven will use to store artifacts.
  |
  | Default: ~/.m2/repository
  <localRepository>/path/to/local/repo</localRepository>
  -->
  <localRepository>c:/maven_repo</localRepository>
```

بعد از انجام تغییرات فایل را ذخیره میکنیم.

### مخزن مرکزی (Maven central repository) Maven

هنگامی که شما یک پروژه Maven را Build میکنید، Maven فایل pom.xml را برای شناسایی وابستگی ها به منظور دانلود چک مینماید. ابتدا Maven وابستگی ها را از مخزن محلی (Maven local repository) میگیرد و اگر آن ها را پیدا نکرد، وابستگی ها را از مخزن پیش فرض مرکزی (Maven central repository) دانلود مینماید:

<http://search.maven.org>

### مخزن راه دور (Maven Remote repository) Maven

به صورت پیش فرض، Maven تمام وابستگی ها را از مخزن مرکزی Maven دانلود میکند اما تمام کتابخانه ها در مخزن مرکزی وجود ندارند و فقط در مخزن های راه دور مثل Java.net یا مخزن JBoss (JBoss repository) وجود دارند. برای اضافه کردن این مخازن از روش زیر استفاده میکنیم.

#### ۱- مخزن Java.net

جزئیات مربوط به مخزن Java.net را در فایل pom.xml به صورت زیر وارد میکنیم:

pom.xml

```
<project ...>
  <repositories>
    <repository>
      <id>java.net</id>
      <url>https://maven.java.net/content/repositories/public/</url>
    </repository>
  </repositories>
</project>
```

## ۲- مخزن JBoss (JBoss repository)

جزئیات مربوط به مخزن JBoss را در فایل pom.xml به صورت زیر وارد میکنیم:

```
pom.xml
<project ...>
  <repositories>
    <repository>
      <id>JBoss repository</id>
      <url>http://repository.jboss.org/nexus/content/groups/public/</url>
    </repository>
  </repositories>
</project>
```

## سازو کار وابستگی ها در Maven

سازو کار وابستگی ها در Maven کمک میکند که تمام وابستگی ها به صورت خودکار دانلود شوند و ورژن کتابخانه ها نیز به روز باشند.

فرض کنید که شما میخواهید Log4J به عنوان سازو کار logging پروژه خود استفاده کنید دو حالت در پیش روی شما است:

### ۱- روش سنتی

۱- مراجعه به سایت <http://logging.apache.org/log4j>

۲- دانلود jar فایل Log4J

۳- اضافه کردن این jar فایل به پروژه

۴- اجرا و حل مشکلات احتمالی مربوط به وابستگی ها به صورت دستی

## ۲- روشی Maven

۱- شما نیاز دارید که مختصات Maven (Maven coordinates) Log4J را بدانید برای مثال :

```
<groupId>log4j</groupId>  
<artifactId>log4j</artifactId>  
<version>1.2.14</version>
```

Maven ورژن ۱.۲.۱۴ ، Log4J را به صورت خودکار دانلود میکند. اگر ورژن ذکر نشده باشد جدیدترین ورژن Log4J به صورت خودکار دانلود می شود.

۲- مختصات Maven را در فایل pom.xml معرفی می کنیم :

```
<dependencies>  
<dependency>  
<groupId>log4j</groupId>  
<artifactId>log4j</artifactId>  
<version>1.2.14</version>  
</dependency>  
</dependencies>
```

۳- در زمان کامپایل یا Build ، Maven به صورت خودکار Log4J.jar را دانلود نموده و آن را در مخزن محلی Maven قرار میدهد.

۴- تمام موارد به وسیله Maven مدیریت می شود.

## نتیجه گیری

تفاوت را مشاهده کردید ؟ چه اتفاقی در Maven افتاد ؟ زمانی که شما یک پروژه Maven را Build میکنید ، فایل pom.xml فراخوانی میشود ، اگر مختصات Log4J را مشاهده کند به ترتیب زیر عمل می کند :

۱- Log4J را در مخزن محلی Maven (Maven local repository) جستجو میکند.

۲- Log4J را در مخزن مرکزی Maven (Maven central repository) جستجو میکند.

۳- Log4J را در مخزن راه دور (Maven Remote repository) جستجو میکند. (اگر در داخل فایل pom.xml تعریف شده باشد).

همان طور که میبینید مدیریت وابستگی کتابخانه ها در Maven بسیار جالب است و به میزان زیادی در وقت برنامه نویسان صرفه جویی میکند.

حالا سوال این است که چطور مختصات Maven را بدست بیاوریم ؟ شما میتوانید با مراجعه به سایت <http://search.maven.org> ،

Jar فایل مورد نظر خود را جستجو نمایید.

### اضافه کردن کتابخانه به مخزن محلی (Maven local repository)

برای مثال [kaptcha](#) یک کتابخانه مهم برای جاوا است که برای ایجاد تصاویر captcha به منظور متوقف کردن spam مورد استفاده قرار میگیرد ، اما این فایل در مخزن مرکزی (Maven central repository) موجود نمی باشد. در این قسمت شما را با نحوه نصب این jar فایل در مخزن محلی Maven آشنا میکنیم.

#### ۱- mvn install

[kaptcha](#) را دانلود کنید و `kaptcha-version.jar` را در جایی مثلا درایو D کپی کنید. فرمان زیر را صادر کنید :

```
mvn install:install-file -Dfile=D:\kaptcha-{version}.jar -
DgroupId=com.google.code
-DartifactId=kaptcha -Dversion={version} -Dpackaging=jar
```

```
c:\>mvn install:install-file -Dfile=D:\kaptcha-2.3.jar -
DgroupId=com.google.code
-DartifactId=kaptcha -Dversion=2.3 -Dpackaging=jar
[INFO] Scanning for projects...
[INFO] Searching repository for plugin with prefix: 'install'.
[INFO] -----
-----
```

```
[INFO] Building Maven Default Project
[INFO] task-segment: [install:install-file] (aggregator-style)
[INFO] -----
[INFO] [install:install-file]
[INFO] Installing D:\kaptcha-2.3.jar to
C:\maven_repo\com\google\code\kaptcha\2.3\kaptcha-2.3.jar
[INFO] -----
[INFO] BUILD SUCCESSFUL
[INFO] -----
[INFO] Total time: < 1 second
[INFO] Finished at: Tue May 12 13:41:42 SGT 2009
[INFO] Final Memory: 3M/6M
[INFO] -----
```

اکنون kaptcha.jar در مخزن محلی Maven کپی شده است .

### ۲ - pom.xml

بعد از نصب ، مختصات این jar فایل را در pom.xml معرفی میکنیم.

```
<dependency>
<groupId>com.google.code</groupId>
<artifactId>kaptcha</artifactId>
<version>2.3</version>
</dependency>
```

### ۳ - Build

اکنون kaptcha.jar از مخزن محلی (Maven local repository) قابل دریافت است.

### ایجاد یک پروژه JAVA با استفاده از Maven

در این قسمت شما با نحوه ایجاد یک پروژه جاوا با استفاده از Maven و معرفی آن به IntelliJ IDEA و بسته بندی آن به صورت یک jar فایل آشنا میشوید.

۱. Maven 3.0.۴

۲. IntelliJ IDEA 12

۳. JDK 6 or Higher

## ۱- ایجاد یک پروژه از Maven Template

در ترمینال linux یا خط فرمان ویندوز به شاخه ای که میخواهید پروژه دانلود شده در آن ذخیره شود بروید و فرمان زیر را

صادر کنید :

```
mvn archetype:generate -DgroupId={project-packaging} -DartifactId={project-name} -DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false
```

این دستور به Maven میگوید که یک پروژه از `maven-archetype-quickstart` ایجاد کند که این یک Plugin پیش فرض در Maven میباشد. اگر شما از وارد کردن آرگومان `archetypeArtifactId` صرف نظر کنید یک لیست از الگوها (Template) برای انتخاب به شما نشان داده میشود.

برای مثال :

```
$ mvn archetype:generate -DgroupId=com.amir -DartifactId=NumberGenerator -DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false
[INFO] Scanning for projects...
[INFO]
[INFO] -- omitted for readability
[INFO] -----
[INFO] Using following parameters for creating project from Old (1.x) Archetype:
    maven-archetype-quickstart:1.0
[INFO] -----
[INFO] Parameter: groupId, Value: com.amir
[INFO] Parameter: packageName, Value: com.amir
[INFO] Parameter: package, Value: com.amir
[INFO] Parameter: artifactId, Value: NumberGenerator
[INFO] Parameter: basedir, Value: /Users/amir/Documents/workspace
[INFO] Parameter: version, Value: 1.0-SNAPSHOT
[INFO] project created from Old (1.x) Archetype in dir:
    /Users/amir/Documents/workspace/NumberGenerator
[INFO] -----
```

```
[INFO] BUILD SUCCESS

[INFO] -----

[INFO] Total time: 3.917s

[INFO] Finished at: Mon Dec 17 18:53:58 MYT 2012

[INFO] Final Memory: 9M/24M

[INFO] -----
```

با اجرای این دستور یک پروژه به نام NumberGenerator در شاخه ای که هستیم با تمام ساختار آن کپی میشود.

## ۲ - ساختار پروژه Maven

Maven یک ساختار استاندارد را برای هر پروژه ایجاد می کند که به صورت زیر است :

```
NumberGenerator
  |--src
  |   |--main
  |       |--java
  |           |--com
  |               |--amir
  |                   |--App.java
  |                       |--test
  |                           |--java
  |                               |--com
  |                                   |--amir
  |                                       |--AppTest.java
  |                                           |--pom.xml
```

به عبارت ساده تر تمام source code در پوشه /src/main/java/project-package / کدهای تست نیز در /src/test/java/project-package قرار میگیرد.

و همانطور که میبینید یک فایل pom.xml نیز ایجاد میشود که این فایل شبیه فایل Build.xml ، Ant می باشد که اطلاعات داخلی پروژه ، ساختار پروژه ، Plugin های پروژه برای دسترسی به وابستگی ها و .... در آن قرار دارد.

```
pom.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/maven-v4_0_0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.amir</groupId>
    <artifactId>NumberGenerator</artifactId>
    <packaging>jar</packaging>
    <version>1.0-SNAPSHOT</version>
    <name>NumberGenerator</name>
    <url>http://maven.apache.org</url>
    <dependencies>
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>3.8.1</version>
            <scope>test</scope>
        </dependency>
    </dependencies>
</project>
```

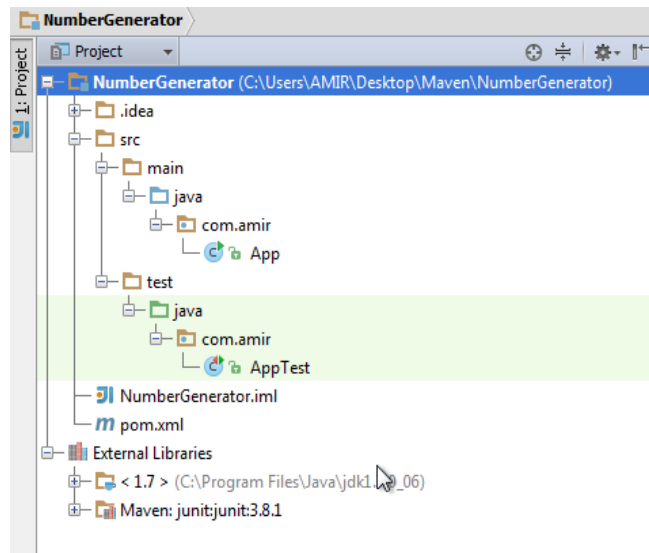
### ۳- متناسب سازی پروژه با IntelliJ IDEA

ابتدا وارد پوشه پروژه شوید (جایی که pom.xml در آن قرار دارد) و برای تبدیل پروژه Maven به پروژه قابل اجرا در

IntelliJ IDEA از این دستور استفاده کنید :

```
mvn idea:module
```

این دستور تمام فایل های مورد نیاز برای IntelliJ IDEA را ایجاد مینماید.



#### ۴- بروز کردن POM

pom.xml که توسط Maven دانلود شده است بسیار ساده و قدیمی است در اغلب موارد شما نیاز دارید یک Compiler Plugin را اضافه کنید تا Maven متوجه شود که از کدام ورژن JDK برای کامپایل پروژه استفاده کند. ( ورژن پیش فرض در JDK در فایل pom که ۱.۴ است بسیار قدیمی است).

```
<plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-compiler-plugin</artifactId>
    <version>2.3.2</version>
    <configuration>
        <source>1.7</source>
        <target>1.7</target>
    </configuration>
</plugin>
```

همچنین ورژن junit را نیز از ۳.۸.۱ به ۴.۱۱ ارتقا میدهیم.

```
<dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.11</version>
    <scope>test</scope>
```

```
</dependency>
```

pom.xml – full version.

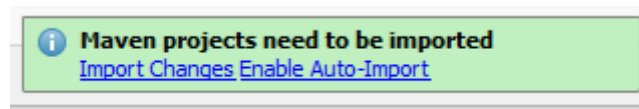
```
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
    http://maven.apache.org/maven-v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.amir</groupId>
  <artifactId>NumberGenerator</artifactId>
  <packaging>jar</packaging>
  <version>1.0-SNAPSHOT</version>
  <name>NumberGenerator</name>
  <url>http://maven.apache.org</url>

  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>4.11</version>
      <scope>test</scope>
    </dependency>
  </dependencies>

  <build>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-compiler-plugin</artifactId>
        <version>2.3.2</version>
        <configuration>
          <source>1.7</source>
          <target>1.7</target>
        </configuration>
      </plugin>
    </plugins>
  </build>
```

</project>

بعد از بروز سازی فایل pom.xml ، IntelliJ IDEA ، پیامی را به منظور اعمال تغییرات و دانلود مجدد وابستگی ها صادر میکند که با زدن Import Changes شروع به دانلود کردن وابستگی ها میکند بدیهی است هر زمان که فایل pom.xml تغییر میکند این پیام نشان داده میشود و بعد یک بار پروژه reload میگردد تا تغییرات در ساختار پروژه اعمال شود اگر Import Changes را نزنیم باید در خط فرمان دوباره دستور mvn idea:module را در پوشه NumberGenerator صادر کنیم.



## ۵ - بروزرسانی Business Logic

برای که اطمینان پیدا کنیم که شیء app که دارای متد تولید یک کلید یکتا میباشد دقیقا ۳۶ حرف الفبا ایجاد میکند تغییرات زیر را unit test ایجاد میکنیم. (Test driven development)

```
AppTest.java
package com.amir;

import org.junit.Assert;
import org.junit.Test;

public class AppTest {

    @Test
    public void testLengthOfTheUniqueKey() {

        App obj = new App();

        Assert.assertEquals(36, obj.generateUniqueKey().length());

    }
}
```

کامل کردن business logic

App.java

```
package com.amir;

import java.util.UUID;

/**
 * Generate a unique number
 */
public class App
{

    public static void main( String[] args )
    {

        App obj = new App();

        System.out.println("Unique ID : " + obj.generateUniqueKey());

    }

    public String generateUniqueKey(){

        String id = UUID.randomUUID().toString();

        return id;

    }

}
```

#### ۶ - بسته بندی (Packaging) Maven

اکنون از Maven برای کامپایل و تولید jar فایل پروژه استفاده میکنیم . با مراجعه به فایل pom.xml و قسمت packaging ساختار خروجی jar فایل قابل مشاهده میباشد.

pom.xml – full version.

```
<project ...>

    <modelVersion>4.0.0</modelVersion>

    <groupId>com.amir</groupId>

    <artifactId>NumberGenerator</artifactId>
```

```
<packaging>jar</packaging>
<version>1.0-SNAPSHOT</version>
<!--
```

در پوشه NumberGenerator یک بار دستور mvn package را صادر میکنیم.

```
C:\NumberGenerator>mvn package
```

با اجرای این دستور یک jar فایل در پوشه target پروژه ایجاد میگردد. که برای اجرای آن از دستور زیر استفاده میکنیم:

```
C:\NumberGenerator> java -cp target\NumberGenerator-1.0-SNAPSHOT.jar com.amir.App
Unique ID : f1947107-2deb-4926-a635-ea3db61453e8
```

## ایجاد یک پروژه تحت وب JAVA با استفاده از Maven

در این مثال چگونگی ایجاد یک پروژه تحت وب جاوا با استفاده از Spring MVC و مطابق سازی آن با IntelliJ IDEA را به شما نشان میدهیم.

ابزار مورد استفاده :

۱. Maven 3.0.5
۲. IntelliJ IDEA ۱۲
۳. or Higher JDK 6
۴. Spring 3.2.0.RELEASED
۵. JBoss 6.1

۱- تولید یک پروژه تحت وب از Maven Template

در ترمینال linux یا خط فرمان ویندوز به شاخه ای که میخواهید پروژه دانلود شده در آن ذخیره شود بروید .

```
mvn archetype:generate -DgroupId={project-packaging} -DartifactId={project-name} -DarchetypeArtifactId=maven-archetype-webapp -DinteractiveMode=false
```

و فرمان زیر را صادر کنید :

```
C:\>mvn archetype:generate -DgroupId=com.amir -DartifactId=CounterWebApp  
DarchetypeArtifactId=maven-archetype-webapp -DinteractiveMode=false
```

یک پروژه به نام CounterWebApp دانلود میشود که دارای ساختار تحت وب استاندارد Maven است.

## ۲ - ساختار پروژه Maven

Maven به صورت خودکار یک ساختار تحت وب برای پروژه ایجاد میکند که دارای فایل های pom.xml و web.xml است.

ساختار پروژه به صورت زیر می باشد:

```
CounterWebApp  
  |--src  
  |---main  
  |-----resources  
  |-----webapp  
  |-----index.jsp  
  |-----WEB-INF  
  |-----web.xml  
  |--pom.xml
```

فایل pom.xml به صورت زیر است:

```
pom.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/maven-v4_0_0.xsd">  
  <modelVersion>4.0.0</modelVersion>  
  <groupId>com.amir</groupId>  
  <artifactId>CounterWebApp</artifactId>
```

```
        <packaging>war</packaging>
        <version>1.0-SNAPSHOT</version>
<name>CounterWebApp Maven Webapp</name>
        <url>http://maven.apache.org</url>
        <dependencies>
            <dependency>
                <groupId>junit</groupId>
                <artifactId>junit</artifactId>
                <version>3.8.1</version>
                <scope>test</scope>
            </dependency>
        </dependencies>
        <build>
            <finalName>CounterWebApp</finalName>
        </build>
    </project>
```

فایل web.xml به صورت زیر است:

همان طور که می بینید ورژن Servlet مورد استفاده ۲.۳ است که آن را در مرحله بعد به ۲.۵ ارتقا می دهیم:

```
web.xml -
    <!DOCTYPE web-app PUBLIC
        "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
        "http://java.sun.com/dtd/web-app_2_3.dtd" >
    <web-app>
        <display-name>Archetype Created Web Application</display-name>
    </web-app>
```

index.jsp – A simple hello world html file

```
<html>
<body>
    <h2>Hello World!</h2>
</body>
```



</html>

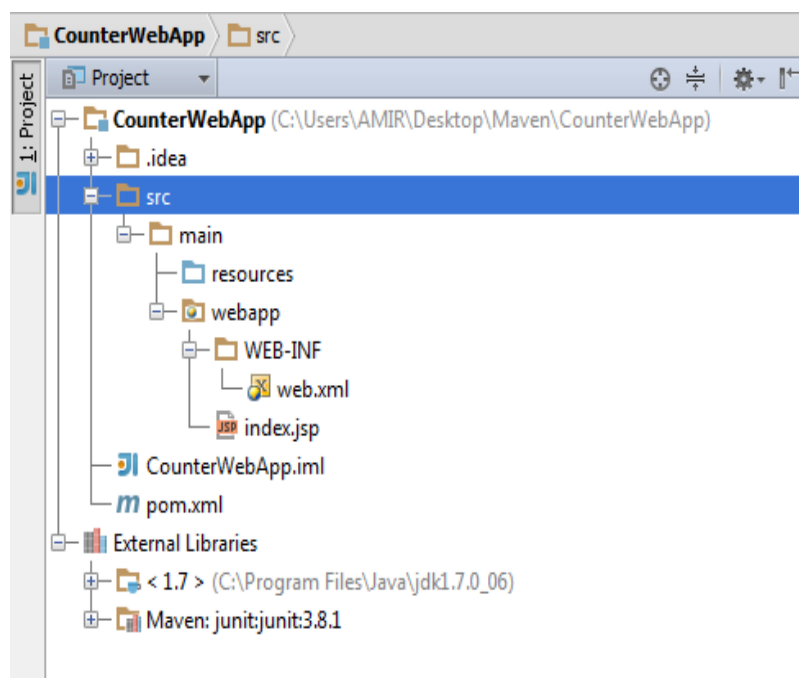
### ۳- متناسب سازی پروژه با IntelliJ IDEA

برای تبدیل فایل Maven تحت وب به یک پروژه متناسب با IntelliJ IDEA ابتدا به پوشه CounterWebApp رفته و بعد دستور زیر را صادر کنید :

```
C:\CounterWebApp>mvn idea:module -Dwtpversion=2.0
```

ما به یک پروژه تحت وب جاوا نیاز داریم نه یک پروژه جاوا به همین دلیل در آخر دستور `mvn idea:module` از آرگومان `Dwtpversion` استفاده میکنیم تا پروژه تبدیل به یک پروژه تحت وب جاوایی شود.

اکنون پروژه را با IntelliJ IDEA باز کرده و آن را یک بار با JBoss اجرا میکنیم. می بینیم که پیام Hello World چاپ میشود.



### ۴- بروزرسانی POM

- ۱ - اضافه کردن plugin کامپایلر تا مشخص شود JDK7 این پروژه را کامپایل میکند.
- ۲ - اضافه کردن وابستگی های Spring
- ۳ - بروزرسانی junit به نسخه ۴.۱۱

pom.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/maven-v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.amir</groupId>
  <artifactId>CounterWebApp</artifactId>
  <packaging>war</packaging>
  <version>1.0-SNAPSHOT</version>
  <name>CounterWebApp Maven Webapp</name>
  <url>http://maven.apache.org</url>

  <properties>
    <spring.version>3.0.5.RELEASE</spring.version>
    <junit.version>4.11</junit.version>
    <jdk.version>1.7</jdk.version>
  </properties>

  <dependencies>

    <!-- Spring 3 dependencies -->
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-core</artifactId>
      <version>${spring.version}</version>
    </dependency>

    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-web</artifactId>
```

```
<version>${spring.version}</version>
    </dependency>

    <dependency>
<groupId>org.springframework</groupId>
<artifactId>spring-webmvc</artifactId>
    <version>${spring.version}</version>
    </dependency>

    <dependency>
        <groupId>junit</groupId>
        <artifactId>junit</artifactId>
        <version>${junit.version}</version>
        <scope>test</scope>
    </dependency>
</dependencies>

    <build>

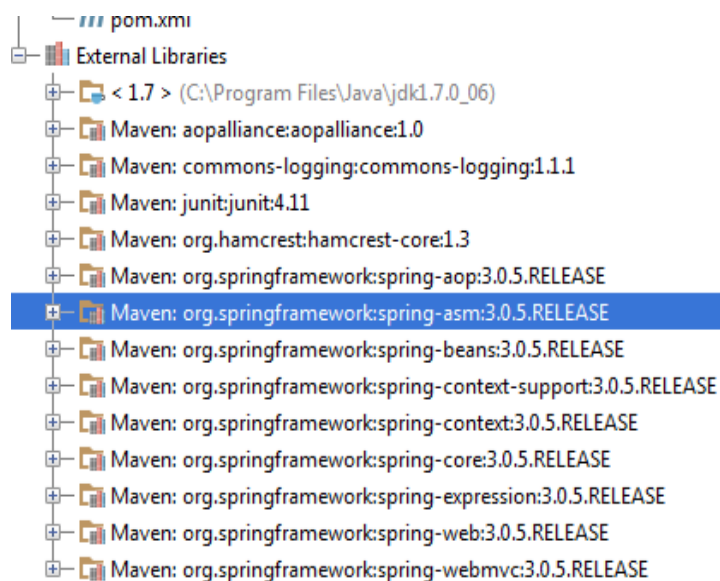
        <finalName>CounterWebApp</finalName>

        <plugins>

            <plugin>

                <groupId>org.apache.maven.plugins</groupId>
<artifactId>maven-compiler-plugin</artifactId>
                <version>3.0</version>
                <configuration>
                    <source>${jdk.version}</source>
                    <target>${jdk.version}</target>
                </configuration>
            </plugin>
        </plugins>
    </build>
</project>
```

بعد از بروز سازی فایل pom.xml ، IntelliJ IDEA ، پیامی را به منظور اعمال تغییرات و دانلود مجدد وابستگی ها صادر میکند که با زدن Import Changes شروع به دانلود کردن وابستگی ها میکند بدیهی است هر زمان که فایل pom.xml تغییر میکند این پیام نشان داده میشود و بعد یک بار پروژه reload میگردد تا تغییرات در ساختار پروژه اعمال شود اگر Import Changes را نزنیم باید در خط فرمان دوباره دستور mvn idea:module را در پوشه CounterWebbApp صادر کنیم.



## ۵ - Spring MVC REST (اضافه کردن کنترلر spring)

یک کلاس کنترلر به پروژه اضافه میکنیم که دارای دو متد ساده برای چاپ Message میباشد:

```
/src/main/java/com/amir/controller/BaseController.java
package com.amir.controller;

import org.springframework.stereotype.Controller;
import org.springframework.ui.ModelMap;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;

@Controller
@RequestMapping("/")
public class BaseController {
    @RequestMapping(value="/welcome", method = RequestMethod.GET)
    public String welcome(ModelMap model) {
```

```
model.addAttribute("message", "Maven Web Project + Spring 3 MVC - welcome ( " );

//Spring uses InternalResourceViewResolver and return back index.jsp

return "index";

}

@RequestMapping(value="/welcome/{name}", method = RequestMethod.GET)

public String welcomeName(@PathVariable String name, ModelMap model) {

model.addAttribute("message", "Maven Web Project + Spring 3 MVC - " + name);

return "index";

}

}
```

یک فایل پیکربندی spring به نام mvc-dispatcher-servlet.xml را در پوشه WEB-INF ایجاد میکنیم.

```
/src/main/webapp/WEB-INF/mvc-dispatcher-servlet.xml

<beans xmlns="http://www.springframework.org/schema/beans"

xmlns:context="http://www.springframework.org/schema/context"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="

http://www.springframework.org/schema/beans

http://www.springframework.org/schema/beans/spring-beans-3.0.xsd

http://www.springframework.org/schema/context

http://www.springframework.org/schema/context/spring-context-3.0.xsd">

<context:component-scan base-package="com.amir.controller" />

<bean

class="org.springframework.web.servlet.view.InternalResourceViewResolver">

<property name="prefix">

<value>/WEB-INF/pages/</value>

</property>

<property name="suffix">

<value>.jsp</value>
```

```
        </property>
    </bean>
</beans>
```

فایل web.xml را نیز به servlet ورژن ۲.۵ ارتقا میدهیم و همچنین Spring Framework را از طریق Spring Listener با پروژه تحت وب یکپارچه میسازیم:

```
/src/main/webapp/WEB-INF/web.xml
<web-app xmlns="http://java.sun.com/xml/ns/javaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
    http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
  version="2.5">
  <display-name>Counter Web Application</display-name>
  <servlet>
    <servlet-name>mvc-dispatcher</servlet-name>
    <servlet-class>
      org.springframework.web.servlet.DispatcherServlet
    </servlet-class>
    <load-on-startup>1</load-on-startup>
  </servlet>
  <servlet-mapping>
    <servlet-name>mvc-dispatcher</servlet-name>
    <url-pattern>/</url-pattern>
  </servlet-mapping>
  <context-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>/WEB-INF/mvc-dispatcher-servlet.xml</param-value>
  </context-param>
  <listener>
    <listener-class>
      org.springframework.web.context.ContextLoaderListener
```

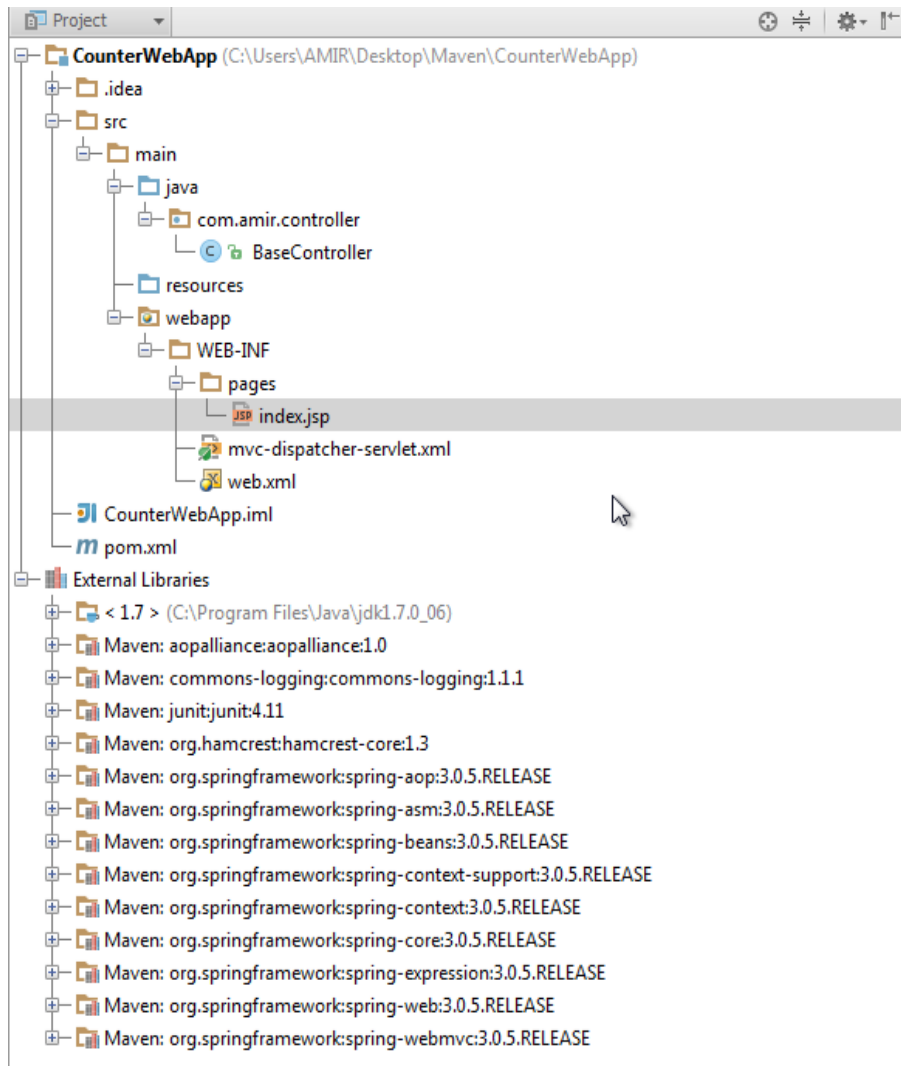
```
</listener-class>
</listener>
</web-app>
```

فایل index.jsp را به پوشه pages در داخل WEB-INF کپی کرده تا از دسترسی مستقیم کاربر به آن جلوگیری کنیم. فایل را ویرایش کرده و قسمت چاپ message را به آن اضافه می کنیم:

```
/src/main/webapp/WEB-INF/pages/index.jsp
```

```
<html>
<body>
<h2>Hello World!</h2>
<h4>Message : ${message}</h4>
</body>
</html>
```

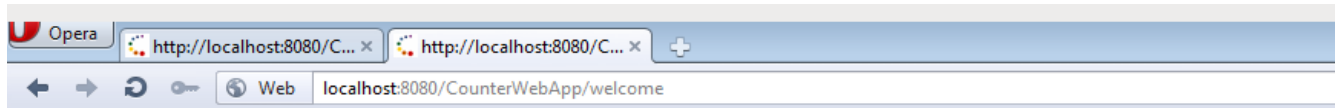
ساختار نهایی به این صورت خواهد شد :



اکنون پروژه را با استفاده از JBoss 6.1 اجرا میکنیم که نتیجه به صورت زیر است :

<http://localhost:8080/CounterWebApp/welcome>

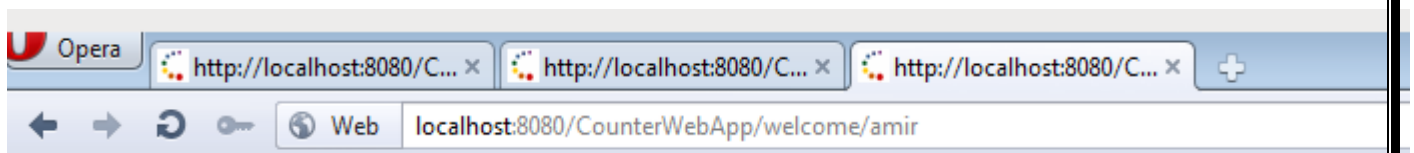




# Hello World!

**Message : Maven Web Project + Spring 3 MVC - welcome()**

<http://localhost:8080/CounterWebApp/welcome/amir>



# Hello World!

**Message : Maven Web Project + Spring 3 MVC - amir**

#### ۶ - بسته بندی (Packaging) Maven

برای ایجاد war file نیز مانند پروژه جاوا ابتدا به شاخه برنامه تحت وب میرویم و دستور mvn package را صادر می کنیم که بعد از اجرا یک war file در پوشه target برنامه ایجاد میکند که آن را میتوانیم به tomcat اضافه کرده و اجرا نماییم.

## خدمات لایتک



کودکان



خبرنامه



مقالات



فایل آموزشی



پرسش و پاسخ



بسته آموزشی



فرصت های  
شغلی



فیلم آموزشی



آموزش ایمیلی



باشگاه  
دانشجویان



لینوکس



جاوا



شبکه



توریسم



توسعه وب



گرافیک



امنیت



پایگاه داده