MAVEN REPOSITORIES

INTRODUCTION :

* In this module , we are going to cover repositories. Repositories are the place where Maven searches to download source codes and artifacts from so that we can use that inside our application.
* We are going to see the difference between a dependency repository and a plugin repository.
* We will also see how to override those repositories from our default location.
* Where it should look for releases or snapshots.

WHAT IS A REPOSITORY :

* We have covered local repositories in a previous module , Maven looks at the local repository first cause that’s where it stores everything. If we don’t have , then it will go and download from a remote repository.
* When it downloads it , it will install it in your home directory\.m2
* It is the default location , we can override it but most people leave it there itself.
* In the local repository , we store artifacts given the groupId , artifactId , version.
* Its defined by the people who created the artifact we want to use.
* Reason why this is good , is because it avoids duplication by copying it in every project and storing it in source code control module.
* Repositories are just a http accessible location where we can download our files from.
* Very often they don’t have any security wrapped around them , for internal repository you can definitely add it.
* Our default location is defined in the super pom.xml , its located in the Maven Installation. Please don’t edit that.
* The default location that its pointing to is repo.maven.apache.org/maven2
* This has 95% of whatever we want to download.
* It has all of the open source project , we wont upload any companies internal libraries.
* Multiple repositories are allowed and respected , its common practice.
* Corporate Repository is outside the scope of this scope , we are encouraged to use one though , caused denial of service attack . Nexus is one of the corporate repository which is built on Maven and Artifactory.

DEPENDENCY REPOSITORY & DEMO :

* A dependency repository is from where we download all of our dependencies from.
* It can contain releases or snapshots or both.
* We can specify the repository in the repositories section of our POM.xml  
  <repositories>  
  <repository>  
  <id>spring-snapshot</id>  
  <name>Spring Maven SNAPSHOT Repository</name>  
  <url>http://repo.springsource.org/libs-snapshot</url>  
  <snapshots>  
  <enabled>true</enabled>  
  </snapshots>  
  <releases>  
  <enabled>false</enabled>  
  </releases>  
  </repository>  
  </repositories>
* We can give a name or not , then the URL to the repository , at the end of the URL we have snapshot declaration.
* We want everyone in the organization to download snapshot and releases then we would say <enabled>true</enabled> for both.
* We can have multiple repositories and it would just be another element inside the repositories element.
* Lets see how to add a repository to our project :
* Open Chrome
* <http://repo.springsource.org/libs-snapshot>
* Its just a simple download site.
* We navigate through the groupId.
* Scroll down to ORG -> SPRINGFRAMEWORK -> SPRING-CORE -> 3.2.0.BUILD-SNAPSHOT
* They have taken their time to build javadocs and sources.
* Open up POM.xml
* Add a dependency :  
  <dependencies>  
  <dependency>  
  <groupId>org.springframework </groupId>  
  <artifactId>spring-core</artifactId>  
  <version>3.2.0.BUILD-SNAPSHOT</version>  
  </dependency>  
  </dependencies>
* Save the file.
* We have an error – I cant find this maven dependency.
* We have to add a repository for it to download that :   
  <repositories>  
  <repository>  
  <id>spring-snapshot</id>  
  <name>Spring Maven SNAPSHOT Repository</name>  
  <url>http://repo.springsource.org/libs-snapshot</url>  
  <snapshots>  
  <enabled>true</enabled>  
  </snapshots>  
  <releases>  
  <enabled>false</enabled>  
  </releases>  
  </repository>  
  </repositories>
* When we click Save -> Our error gets resolved and now we can verify this by looking at our project -> Maven dependencies -> Spring-core is present.

PLUGIN REPOSITORY :

* Plugin repositories are identical to dependency repositories but they deal with plugins.
* Its downloaded from and Identical to dependencies.
* Will only look for plugins , usually from a separate repository , just cause of code cleanliness.
* We have a different XML element   
  <pluginRepositories>  
  <pluginRepository>  
  <id>acme corp</id>  
  <name>Acme Internal Corporate Repository</name>  
  <url>http://acmecorp.com/plugins</url>  
  <snapshots>  
  <enabled>true</enabled>  
  </snapshots>  
  <releases>  
  <enabled>true</enabled>  
  </releases>  
  </pluginRepository>  
  </pluginRepositories>
* Same rules apply to plugin repository that works for dependency repository.

RELEASES vs SNAPSHOTS :

* Releases and Snapshots can come from the same repositories.
* Why wouldn’t we want to project to upload everything to the central repo.
* Snapshots , Milestones , Release Candidates , Release Policies
* Best not to upload all of these to a central repository.
* More work and headache to do so.
* We may just use a website to handle SNAPSHOTS and it may be on a website with a URL.
* When we looked at the example of spring-core , we saw that they had a separate repository for their SNAPSHOT and they didn’t put it up on repo.maven.apache.org.
* We get only snapshots from corporate repository of Spring.

SUMMARY :

* Dependency repository and the plugin repository can be the same repository or separate.
* Projects will not upload their SNAPSHOT to a central repo cause of the changes they have to do later on and the release process is a headache to upload in the central repo and everything has to be proper before uploading.
* Plugins are usually in the same repository as dependencies.
* They are often broken out for corporate repositories.
* Companies should use a corporate repository to lighten the load on the central repository.