MAVEN

- Building a web application typically consists of tasks such as downloading dependencies, putting additional jars on a classpath, compiling source code into binary code, running tests, packaging compiled code into deployable artifacts such as JAR, WAR, and ZIP files, and deploying these artifacts to an application server or repository.

- Apache Maven is a **project management tool** that automates these tasks, minimizes the risk of human errors while building the application and separating the work of compiling and packaging our code from that of code construction.

MAVEN FEATURES

- Dependency management – it includes automatic updating, downloading and validating the compatibility, as well as reporting the dependency closures (also known as transitive dependencies)

- Central repository system – project dependencies can be loaded from the local file system or public repositories, such as Maven Central

DEPENDENCIES

- The external libraries that a project uses are called dependencies.
- The dependency management feature in Maven ensures automatic download of those libraries from a central repository, so you don't have to manually download and store them locally.
- Uses less storage by significantly reducing the number of downloads from remote repositories
- Makes checking out a project quicker
- Provides an effective platform for exchanging binary artifacts within your organization and beyond without the need for building artifact from source every time

DEPENDENCIES

- In order to declare a dependency on an external library, you need to provide the *groupId, artifactId*, and the *version* of the library

```
<!-- Servlet Spec -->
<dependency>
    <groupId>javax.servlet</groupId>
    <artifactId>servlet-api</artifactId>
    <version>2.4</version>
    <scope>provided</scope>
</dependency>
<dependency>
    <groupId>javax.servlet.jsp</groupId>
    <artifactId>jsp-api</artifactId>
    <version>2.1</version>
    <scope>provided</scope>
</dependency>
```

- As Maven processes the dependencies, it will download Servlet and JSP library into your local Maven repository – .m2 folder

PROJECT IDENTIFIER

- Maven uses a set of identifiers, also called coordinates, to uniquely identify a project and specify how the project artifact should be packaged
  - *groupId* – the unique base name of the company or group that created the project
  - *artifactId* – the unique name of the project
  - *version* – the version of the project
  - *packaging* – the packaging method (e.g. *WAR*/*JAR*/*EAR*)
- eg. *com.beaconfire.sample.1.0-SNAPSHOT*

PROJECT OBJECT MODEL(POM)

- The configuration of a Maven project is done via a *Project Object Model (POM)*, represented by a *pom.xml* file. The *POM* describes the project, manages dependencies, and configures plugins for building the software.

- The *POM* also defines the relationships among modules of multi-module projects

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XM
    <modelVersion>4.0.0</modelVersion>
    <parent>
        <!-- Your own application should inherit from spring-boot-starter-parent -->
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>1.0.2.RELEASE</version>
    </parent>
    <artifactId>SpringBootTest</artifactId>
    <groupId>com.springboot</groupId>
    <name>Spring Boot Web Secure Sample</name>
    <description>Spring Boot Web Secure Sample</description>
    <version>0.0.1-SNAPSHOT</version>
    <url>http://projects.spring.io/spring-boot/</url>
    <organization>
        <name>BeaconFireSolution</name>
        <url>http://www.spring.io</url>
    </organization>
    <properties>
        <main.basedir>${basedir}/../..</main.basedir>
    </properties>
    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-security</artifactId>
        </dependency>
        <dependency>…
        <dependency>…
    </dependencies>
    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
            </plugin>
        </plugins>
    </build>
</project>
```

REPOSITORIES

- A repository in Maven is used to hold build artifacts and dependencies of varying types.The default local repository is located in the *.m2/repository* folder under the home directory of the user.

- If an artifact or a plug-in is available in the local repository, Maven will use it. Else it will be downloaded from the central repository and stored in the local repository.The default central repository is Maven Central.

- Some libraries, such as JBoss server, are not available at the central repository but are available at an alternate repository. For those libraries,

  you need to provide the URL to the alternate repository inside *pom.xml* file

REPOSITORIES

```
<repositories>
    <repository>
        <id>springsource-milestones</id>
        <name>SpringSource Milestones Proxy</name>
        <url>https://oss.sonatype.org/content/repositories/springsource-milestones</url>
    </repository>
</repositories>
```

- Please note that you can use multiple repositories in your projects.
- In most case, in the enterprise environment, you will not being using the Maven Central Repo for security reasons.
    - That is, companies will have their own repositories setup and you will download dependencies from there.

REPOSITORIES

```
<profile>
        <id>ebay</id>
        <!-- add a repository property to ease the archetype:generate useage -->
        <properties>
                <archetypeCatalog>https://ebaycentral.corp.ebay.com/content/repositories/re
                </archetypeCatalog>
        </properties>
        <repositories>
                <!-- Raptor Repository -->
                <repository>
                        <id>raptor-releases</id>
                        <url>https://ebaycentral.corp.ebay.com/content/repositories/release
                        <releases>
                                <enabled>true</enabled>
                        </releases>
                        <snapshots>
                                <enabled>false</enabled>
                        </snapshots>
                </repository>
                <repository>
                        <id>raptor-snapshots</id>
                        <url>https://ebaycentral.corp.ebay.com/content/repositories/snapsho
                        <releases>
                                <enabled>false</enabled>
                        </releases>
                        <snapshots>
                                <enabled>true</enabled>
                        </snapshots>
                </repository>
```

PROPERTIES

- Custom <properties> can help make your *pom.xml* file easier to read and maintain.
- Maven properties are value-placeholders and are accessible anywhere within a *pom.xml* by using the notation *${name}*, where *name* is the property.

- In the classic use case, you would use custom properties to define versions for your project's dependencies.

```
<properties>
    <spring.version>4.3.5.RELEASE</spring.version>
</properties>

<dependencies>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-core</artifactId>
        <version>${spring.version}</version>
    </dependency>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-context</artifactId>
        <version>${spring.version}</version>
    </dependency>
</dependencies>
```

PLUGINS AND GOALS

- A Maven *plugin* is a collection of one or more *goals*.
- There are several commonly used goals
  - clean – Clean up after the build (Clean up the target folder)
  - install – Install the built artifact into the local repository.

ANY QUESTION?