Apache Maven

Apache Maven is a build tool like Apache Ant (but with different goals) that manages dependencies and can automatically resolve and obtain required software from a repository. It also can manage dependencies multiple levels deep (transitive dependencies).

Eclipse Integration

m2eclipse: http://m2eclipse.codehaus.org/ - provides Eclipse integration for Maven. Requires Maven installed.

New > Project > Maven Project creates a new Maven project.

Developing with Eclipse and Maven is a free online book:
http://www.sonatype.com/books/m2eclipse-book/reference/

To create a web project, use New > Project > Maven Project and do not select create simple project. Select maven-archetype-webapp as the archetype. Then, create a new source folder src/main/java – this is where you place your serlvets. You can also create src/test/java and src/test/resources. You will be able to publish this project to your web server just like any regular web project.

Convention over Configuration

Maven, like many other projects adopts the convention over configuration idea which means it provides sensible defaults for many configuration parameters, letting users get away with doing less, while not sacrificing flexibility.

Project Object Model (POM)

Projects contain a `pom.xml` file that describes the software being built, its dependencies and the build order. Example file:

```
<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/xsd/maven-4.0.0.xsd">

    <modelVersion>4.0.0</modelVersion><!--for Maven 2 -->

    <!--application information -->
    <groupId>org.hibernate.tutorials</groupId>
    <artifactId>hibernate-tutorial</artifactId>
    <version>1.0.0-SNAPSHOT</version>
    <name>First Hibernate Tutorial</name>

    <build>
         <!-- we dont want the version to be part of the generated war
file name -->
```

```
         <finalName>${artifactId}</finalName>
    </build>

    <dependencies>
        <dependency>
            <groupId>org.hibernate</groupId>
            <artifactId>hibernate-core</artifactId>
        </dependency>

        <!-- Because this is a web app, we also have a dependency on the
servlet api. -->
        <dependency>
            <groupId>javax.servlet</groupId>
            <artifactId>servlet-api</artifactId>
        </dependency>

        <!-- Hibernate uses slf4j for logging, for our purposes here use
the simple backend -->
        <dependency>
            <groupId>org.slf4j</groupId>
            <artifactId>slf4j-simple</artifactId>
        </dependency>

        <!-- Hibernate gives you a choice of bytecode providers between
cglib and javassist -->
        <dependency>
            <groupId>javassist</groupId>
            <artifactId>javassist</artifactId>
        </dependency>
    </dependencies>

</project>
```

Search engines exist where one can find the coordinates of a required library. Example: http://mvnrepository.com/

Standard Directory Layout

The `pom.xml` file lies in the top level directory of the project. Projects may consist of multiple subprojects, each with their own POM and a master POM to build everything.

src/main/java	Contains the deliverable Java sourcecode for the project.
src/main/resources	Contains the deliverable resources for the project, such as property files.
src/main/filters	Resource filter files
src/main/assembly	Assembly descriptors
src/main/config	Configuration files
src/main/webapp	Web application sources
src/test/java	Contains the testing classes (JUnit or TestNG test cases, for example) for the project.
src/test/resources	Contains resources necessary for testing.
src/test/filters	Test resource filter files

src/site	Site
target	Output directory for builds
LICENSE.txt	Project license
README.txt	Project readme

Compiling and Building

Run the command “mvn package” – this will compile all Java files, run all tests and produce a jar file in the target directory.

To run an executable Java class with arguments:

```
mvn exec:java -Dexec.mainClass="org.hibernate.tutorial.EventManager" -
Dexec.args="store"
```

Maven Plug-ins

The syntax to run a plug-in is:

```
mvn plugin-name:goal-name
```

The above executes goal-name in a plug-in called plugin-name. Plug-ins exist for building, testing, managing source code, running a web server, generating Eclipse project files, etc. Some plug-ins are included in all projects by default (e.g. to compile, test and create jar)

Build Cycle

The mvn command takes the following possible parameters:

- process-resources
- compile
- process-test-resources
- test-compile
- test
- package
- install - builds a project and places its binaries in the local repository. Then other projects can utilize this project by specifying its coordinates in their POMs.
- deploy