Introduction to Maven

- What is Maven?
- Maven Fundamentals
- Running Maven
- Integrating Maven with Eclipse

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

What is Maven?

- Maven is a project management and build tool that uses a **project object model** and favors **convention over configuration**
- Maven is an Apache project and is hosted at maven.apache.org



The word "maven" is Yiddish and refers to a knowledgable person.

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

Maven Fundamentals

- Maven can satisfy dependencies - e.g. your project specifies that it needs Hibernate, Maven will fetch Hibernate's dependent libraries
- Maven emphasizes common project directory structure so that it's easy for new developers to understand a project
- The Maven core delegates work to **plugins** that do the actual work of building the project
- You describe a project in an XML file referred to as a **project object model**

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

Maven vs Ant

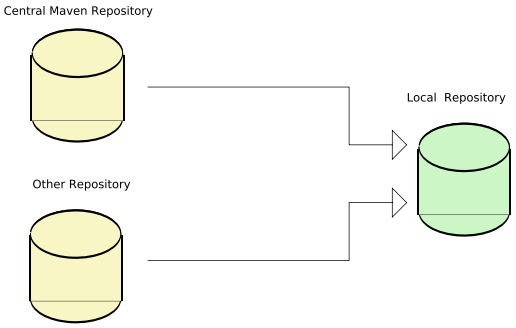
- **Ant** is another popular build tool that predates Maven
- Ant is more like a traditional build script, e.g. **make**, while Maven provides a more complete project management approach
- It is possible to run Ant scripts from Maven



Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

Maven Repositories

- Maven stores dependencies (i.e. JARs) in a **repository**
- When you install Maven, it creates a **local repository**, typically in a folder named **.m2** in your computer's home directory
- When you build, Maven will fetch dependencies, as needed, from a central, remote repository and cache them in your local repository



The default central repository is at: http://repo1.maven.org/maven2/

If the dependencies you need are not in the standard central repository, you can configure references to other remote repositories.

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

Project Lifecycle Phases

- Maven defines lifecycle **phases**, which represent a stage in a project's development, e.g. **compile** or **package**
- There are three built-in lifecycles: default, clean and site, each of which define one or more phases
- Some of the phases in the **default** lifecycle:

Phase	Description
validate	Validate the project is correct
compile	Compile the source code of the project
test	Test using a unit testing framework
package	Package the code into an archive, e.g. JAR or WAR
verify	Verify the package is valid
install	Install the package into the local repository

See the Maven documentation for a complete listing of phases for each lifecycle.

The site lifecycle typically builds documentation for a project.

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

Lifecycle Goals

- Phases themselves comprise **goals**, which are executed by plugins to accomplish the phase
- Maven names goals using a syntax like **plugin-name:goal-name**

```
Phase       Goal(s)
-----       -------

validate    pom:validate
compile     compile:compile
test        surefire:test
package     jar:jar
```

A goal can be bound to one or more phases in a lifecycle so it executes as part of that phase. If a goal is unbound, you can execute it directly from the command-line:

```
mvn archetype:generate
```

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

Maven Command Line

- You can run maven from the command-line or perhaps from within your IDE
- When you run maven, you can specify phase(s) and/or goal(s)
- When you specify a phase, maven first executes the prior phases - for example, if you specify **install**, maven first executes *validate*, *compile* and so forth

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

Maven Coordinates

- Maven projects define metadata, referred to as **coordinates**, that uniquely identify the project in a Maven repository

Metadata	Description
groupID	The project's organization, typically in reverse domain name order, e.g. com.goliath
artifactID	A dentifier unique within the organization that describes the project
version	Project release number
packaging	jar, war, etc.

This metadata is defined in the pom.xml.

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

Building a Starter Project

- Maven provides the **archetype:generate** goal which you can use to generate a pom.xml and starter project
- You specify the desired **coordinates** to this goal

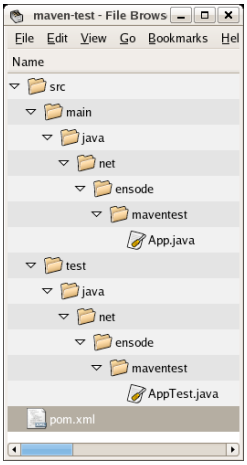
```
mvn archetype:generate \
  -DgroupId=com.goliath \
  -DartifactId=myproject \
  -Dpackage=com.goliath.myproject.first \
  -Dversion=1.0-SNAPSHOT
```


Here, we are directly running the archetype:generate goal and providing a new project's coordinates. This goal will create a pom.xml and the standard dproject irectory structure. Note that since we didn't specify a "packaging" coordinate, it will default to creating a JAR project.

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

Standard Project Structure

- Maven assumes a standard structure for the files and directories that comprise a project
- You can override this, but that's not recommended – this is an example of "convention over configuration"



Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

Writing a POM

- You can express project dependencies in the pom.xml - Maven will fetch the dependencies and their dependencies

pom.xml

Maven will fetch dependencies as needed into the local repository and will then setup the CLASSPATH for the project to reference all dependencies. So when you execute "mvm compile", your code can reference the dependencies.

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/20
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/
    <modelVersion>4.0.0</modelVersion>

    <groupId>com.descriptor.test</groupId>
    <artifactId>test</artifactId>
    <version>1.0-SNAPSHOT</version>
    <packaging>jar</packaging>

    <name>test</name>
    <url>http://maven.apache.org</url>

    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <spring.version>3.0.5.RELEASE</spring.version>
    </properties>

    <dependencies>
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>3.8.1</version>
            <scope>test</scope>
        </dependency>

        <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>

        <dependency>
            <groupId>org.hibernate</groupId>
            <artifactId>hibernate-core</artifactId>
            <version>3.5.1-Final</version>
        </dependency>

    </dependencies>
</project>
```

Integrating Maven with Eclipse

- Modern versions of Eclipse include Maven support, and you can install the **m2eclipse** Eclipse plugin into earlier versions
- You can generate a standard Eclipse project file by running **mvn eclipse:eclipse** and then importing the project's directory into an Eclipse workspace

Modern versions of Eclipse include tooling to let you run Maven from within Eclipse, work with pom.xml files and so forth.

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

Chapter Summary

In this chapter, you learned:

- The fundamentals of Maven
- How to create projects and run Maven from the command line

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum