Maven

https://tech.lds.org/wiki/Introduction_to_Maven

Links

https://maven.apache.org/guides/getting-started/maven-in-five-minutes.html

https://spring.io/guides/gs/maven/

http://www.mkyong.com/maven/how-to-create-a-java-project-with-maven/

http://www.oracle.com/webfolder/technetwork/tutorials/obe/java/Maven_SE/Maven.html

https://examples.javacodegeeks.com/enterprise-java/maven/create-java-project-with-maven-example/

http://www.mkyong.com/maven/how-to-install-maven-in-windows/

http://www.mkyong.com/maven/how-to-create-a-java-project-with-maven/

Testavimas

git clone https://github.com/Mxas/debts.git

git clone https://github.com/kolorobot/spring-boot-thymeleaf.git

git clone https://karpinskas@bitbucket.org/karpinskas/maven-example.git

shopizer

https://github.com/shopizer-ecommerce/shopizer

Veikianti Internetinė pardotuvė.

Preiš bandant įsitikinkit, kad esate instaliave GIT (git --version) ir Mave (mvn --version).

Komandinėje eulutėje rašome:

- git clone https://github.com/shopizer-ecommerce/shopizer.git
- cd shopizer
- mvn clean install
- cd sm-shop
- mvn spring-boot:run

Access the deployed web application at: http://localhost:8080/

Acces the admin section at: http://localhost:8080/admin
#####username : admin
#####password : password

Features

- Dependency System
- Multi-module builds
- Consistent project structure
- Consistent build model
- Plugin oriented
- Project generated sites

The Maven Mindset

- All build systems are essentially the same:
  - Compile Source code
  - Copy Resource
  - Compile and Run Tests
  - Package Project
  - Deploy Project
  - Cleanup
- Describe the project and configure the build
  - You don’t script a build
  - Maven has no concept of a condition
  - Plugins are configured

Maven POM

- Stands for Project Object Model
- Describes a project
  - Name and Version
  - Artifact Type
  - Source Code Locations
  - Dependencies
  - Plugins
  - Profiles (Alternate build configurations)
- Uses XML by Default
  - Not the way Ant uses XML

Project Name

- Maven uniquely identifies a project using:
  - groupID: Arbitrary project grouping identifier (no spaces or colons)
    - Usually loosely based on Java package
  - artfiactId: Arbitrary name of project (no spaces or colons)
  - version: Version of project
    - Format {Major}.{Minor}.{Maintanenence}
    - Add '-SNAPSHOT ' to identify in development
- GAV Syntax: groupId:artifactId:version

Project Name

```
<?xml version="1.0" encoding="UTF-8"?>
<project>
    <modelVersion>4.0.0</modelVersion>
    <groupId>org.lds.training</groupId>
    <artifactId>maven-training</artifactId>
    <version>1.0</version>
</project>
```

Packaging

- Build type identified using the “packaging” element
- Tells Maven how to build the project
- Example packaging types:
  - pom, jar, war, ear, custom
  - Default is jar

```
<?xml version="1.0" encoding="UTF-8"?>
<project>
    <modelVersion>4.0.0</modelVersion>
    <artifactId>maven-training</artifactId>
    <groupId>org.lds.training</groupId>
    <version>1.0</version>
    <packaging>jar</packaging>
</project>
```

Project Inheritance

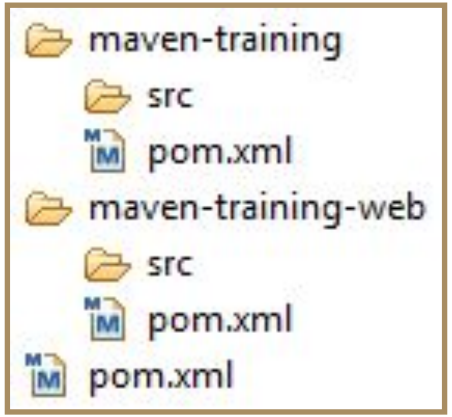
- Pom files can inherit configuration
  - groupId, version
  - Project Config
  - Dependencies
  - Plugin configuration
  - Etc.

```
<?xml version="1.0" encoding="UTF-8"?>
<project>
    <parent>
        <artifactId>maven-training-parent</artifactId>
        <groupId>org.lds.training</groupId>
        <version>1.0</version>
    </parent>
    <modelVersion>4.0.0</modelVersion>
    <artifactId>maven-training</artifactId>
    <packaging>jar</packaging>
</project>
```

Multi Module Projects

- Maven has 1st class multi-module support
- Each maven project creates 1 primary artifact
- A parent pom is used to group modules

```
<project>
    ...
    <packaging>pom</packaging>
    <modules>
        <module>maven-training</module>
        <module>maven-training-web</module>
    </modules>
</project>
```



```
maven-training
    src
    pom.xml
maven-training-web
    src
    pom.xml
pom.xml
```

Maven Conventions

- Maven is opinionated about project structure
- target: Default work directory
- src: All project source files go in this directory
- src/main: All sources that go into primary artifact
- src/test: All sources contributing to testing project
- src/main/java: All java source files
- src/main/webapp: All web source files
- src/main/resources: All non compiled source files
- src/test/java: All java test source files
- src/test/resources: All non compiled test source files

Maven Build Lifecycle

- A Maven build follow a lifecycle
- Default lifecycle
  - generate-sources/generate-resources
  - compile
  - test
  - package
  - integration-test (pre and post)
  - Install
  - deploy
- There is also a Clean lifecycle

Example Maven Goals

- To invoke a Maven build you set a lifecycle “goal”
- mvn install
  - Invokes generate* and compile, test, package, integration-test, install
- mvn clean
  - Invokes just clean
- mvn clean compile
  - Clean old builds and execute generate*, compile
- mvn compile install
  - Invokes generate*, compile, test, integration-test, package, install
- mvn test clean
  - Invokes generate*, compile, test then cleans

Summary

- Maven is a different kind of build tool
- It is easy to create multi-module builds
- Dependencies are awesome