Build Procedure for OpenCms based Projects using Maven 2

May 06, 2008

Felix Noz
Overview

- Before Maven
- Introducing Maven
- Maven and OpenCms: Put them in touch
- Example Application
- The Open Future
Before Maven: only version control

- Archiving
- Recovery
- Conflict management
- Change history
- Manual building and packaging
Our old versioning process

OpenCms Installation
- Web Application
- DB

Sources
- Java Sources
- OpenCms VFS

Version Control
- Web Application
- Database
- Module 1
  - Java Sources
  - VFS-Resources
- Module 2
  - Java Sources
  - VFS-Resources
- Module n
  - Java Sources
  - VFS-Resources
Our old versioning process: CVS repository
Problems with our old versioning process

- Error prone because of the “human” factor
- Cumbersome
- Complex
- Time consuming
- Unreliable
- Too much binary data
- Redundant
- Lack of standardization
- Takes long time to get familiar with
- No process automation
Motivation

- A solution had to be found to
  - Enable plain versioning of VFS data
  - Automate the build procedure
Why Maven?

- Open-Source
- Good reputation
- Widespread use
- Plug-in Architecture
- Optional integration of ANT
Introducing Maven

- Tool for building and managing Java-based projects
- Evolved from former Apache Jakarta projects

Maven 1
- Based on ANT
- Used legacy property files

Maven 2
- Complete rewrite of Maven 1
- Redesign
Basic features of Maven

- Declarative approach
- Project Object Model: POM
- Convention over Configuration
- Dependency Management
- Customizable
- Easy to Use
Project Object Model

```
<parent>
  <groupId>com.comundus.opencms</groupId>
  <artifactId>parent</artifactId>
  <version>7.0.3</version>
  <relativePath>../parent/pom.xml</relativePath>
</parent>

<modelVersion>4.0.0</modelVersion>
<groupId>com.comundus.opencms</groupId>
<artifactId>timecheck</artifactId>
<packaging>jar</packaging>
<name>Timecheck module</name>
(version>1</version>
<description>Example module for OpenCms Days</description>

<dependencies>
  <dependency>
    <groupId>com.comundus</groupId>
    <artifactId>opencms</artifactId>
    <version>7.0.3-comundus</version>
  </dependency>
  <dependency>
    <groupId>javax.servlet</groupId>
    <artifactId>servlet-api</artifactId>
    <version>2.4</version>
    <scope>provided</scope>
  </dependency>
</dependencies>
```

- Inherits from parent POM
- result: timecheck-1.jar
- 1 primary artifact per POM
- Dependency to OpenCms library
- Dependency to servlet library of container
Downloading dependencies

```xml
<dependency>
  <groupId>javax.servlet</groupId>
  <artifactId>servlet-api</artifactId>
  <version>2.4</version>
  <scope>provided</scope>
</dependency>
```
Maven build life cycle

- Maven core
  - Life cycles
    - validate
    - compile
    - test
    - ... (default)
    - deploy
    - phases
    - pre-clean
    - clean
    - post-clean
    - site
    - site-deploy
    - phases
Maven Plug-ins

- Object orientated approach by Mojos
  - Parameter injection by Java Annotations
  - Not restricted to Java Objects
- Define goals
- Bound to phases
Maven Plug-ins

```
<plugins>
  <plugin>
    <groupId>com.comundus</groupId>
    <artifactId>test-plugin</artifactId>
    ...
    ...
    <executions>
      <execution>
        <id>execution1</id>
        <phase>clean</phase>
        <goals>
          <goal>doSomething1</goal>
        </goals>
        <configuration>
          ...
          ...
        </configuration>
      </execution>
      <execution>
        <id>execution2</id>
        <phase>package</phase>
        <goals>
          <goal>doSomething2</goal>
        </goals>
        <configuration>
          ...
          ...
        </configuration>
      </execution>
    </executions>
  </plugin>
</plugins>
```

plug-in client

plug-in

```
<groupId>com.comundus</groupId>
<artifactId>test-plugin</artifactId>
<packaging>maven-plugin</packaging>
<version>1.0</version>
<br name="Test Plugin"/>
```

![Diagram](image)
The Maven VFS Plug-in

- OpenCms from scratch
- Transforms VFS to RFS resources and vv
  - #synclist.txt
  - VFS data
  - VFS metadata
- Creates a running OpenCms installation
VFS Plug-in Goals

- clean
- setup
- importusers
- module
- sync
- publish
- createorgunits
- exportusers
“Maven Style” Projects: Mandatory subprojects

- **parent**
  - Contains the parent POM
  - Includes modules
- **webapp**
  - Goals: setup, importusers
- **system**
  - Goals: module, sync, publish
- **content**
  - Goals: sync, publish
“Maven style” projects: Additional subprojects

- Additional subprojects
  - Defined by packaging
  - Java Subprojects
    - Java source code
    - Resources
  - VFS Subprojects
    - VFS Resources
    - VFS Resources Metadata
    - Module descriptors
A simple example application

Open Project Workspace
The open future

- Maven VFS-Plug-in will go Open Source
  - http://www.comundus.com

- Ideas
  - Integrated SVN support
  - Packaging for OpenCms modules
  - Incremental builds
  - Integration into the OpenCms core
Conclusion

- OpenCms projects can be completely versioned
- Well defined repository
- Real concurrent versioning
- Automated build process
Thank you very much.

http://www.comundus.com