Practical Java Tips

Maximilian Berger

Table of Contents

Project Management

Maven

- Download from http://maven.apache.org/
- Complete Project Management Tool
- Describe WHAT rather than HOW
- Convention over configuration
- Maven: The definite guide (English) [http://www.sonatype.com/books/maven-book/reference/]
- Maven: The definite guide (German) [http://www.sonatype.com/books/maven-book/reference_de/public-book.html]

Important Conventions

- Project is described in `pom.xml`
- Java source is in `src/main/java`
- Java tests are in `src/test/java`
- Webpage is in `src/site`
- Artifacts go to `target/`

Example 1. Example pom.xml

```
<?xml version="1.0"?>
<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/maven-v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>at.uibk.ac.at.dps</groupId>
  <artifactId>sample</artifactId>
  <name>Sample Project</name>
  <version>0.1.0-SNAPSHOT</version>
  <inceptionYear>2009</inceptionYear>
  <licenses>
    <license>
      <name>GNU General Public License, Version 3.0</name>
      <url>http://www.gnu.org/licenses/gpl-3.0-standalone.html</url>
      <distribution>manual</distribution>
    </license>
  </licenses>
  <build>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-compiler-plugin</artifactId>
        <version>2.0.2</version>
        <configuration>
          <source>1.5</source>
          <target>1.5</target>
          <debug>false</debug>
          <optimize>true</optimize>
        </configuration>
      </plugin>
    </plugins>
  </build>
  <dependencies>
    <dependency>
      <groupId>org.testng</groupId>
      <artifactId>testng</artifactId>
      <version>5.8</version>
      <type>jar</type>
      <classifier>jdk15</classifier>
      <scope>test</scope>
    </dependency>
    <dependency>
       <groupId>commons-logging</groupId>
       <artifactId>commons-logging</artifactId>
      <version>1.1.1</version>
    </dependency>
  </dependencies>
</project>
```

Important Maven commands

mvn test	Compile and run tests
mvn package	Compile, run tests, and package
mvn install	Compile, run tests, package, and install locally
mvn site	Run reports and generate website

For a list of available plugins use http://www.mvnrepository.com/ .

Other items that can be configured are

- description
- developers
- contributors
- scm

Regression Tests

TestNG

- Successor of JUnit
- Same Idea
- Easier to use
- TestNG Homepage [http://testng.org/]
- Eclipse plugin available

What you need to do:

- Write a normal class
- Test functions throw Exception
- Test functions are annotated with @Test
- Assert class for basic tests
- Assert.fail()

Maven Use:

- Put class in `src/test/java`
- Load TestNG plugin in `pom.xml` (see Example 1, “Example pom.xml”)

Example 2. Example TestNG Tests

```
import org.testng.Assert;
import org.testng.annotations.Test;
```

```
/**
 * Tests for {@link Blabla}.
 */
public class VirtualFileTest {

    /**
     * Tests the add function.
     *
     * @throws Exception
     *             if the test fails.
     */
    @Test
    public void addTest() throws Exception {
        Blabla b = new Blabla(4);
        b.add(5);
        Assert.assertEquals(b.getValue(), 9);
    }

    @Test
    public void divideTest() throws Exception {
       Blabla b = new Blabla(4);
       try {
          b.divide(0);
          Assert.fail();
       catch (IllegalArgumentException e) {
          // Good!
       }
    }
}
```

Other Test Frameworks

The following framework may help in GUI testing:

- FEST Swing [http://fest.easytesting.org/swing/wiki/pmwiki.php]

IDEs

VI(M) and Emacs (Kate / gedit) are only good for viewing and small edits!

Use a modern, IDE and use its features!

Since we use Maven for project management we can switch IDEs at any time.

It is important to always use the IDE which is fit for a particular task!

Eclipse

- Get from Eclipse Homepage [http://www.eclipse.org/]
- Current: 3.5 SR 1
- Summer: 3.6 or 4.0

You can import my `bookmarks.xml` `[http://max.berger.name/oss/bookmarks.xml]` file into the eclipse update sites to get access to some frequently used plugins (Go to Help/Software Updates / Manage Sites / Import).

Some important plugins:

- Eclipse Maven plugin [http://m2eclipse.sonatype.org/index.html]

Important settings:

- Auto Refresh
- Auto Update
- Font: Use a font that clearly distinguished Zero from Oh (0,O), one from el (1,l). Good example: Consolas (Win Vista), Andale Mono (Win XP/2k), DejaVu Sans Mono (Unix).

Important Features:

- Code Cleanup
- Format
- Organize imports

All theses should be configured

- Per Project
- Automatically on save!

Additional plugins which may be of interest:

- Jigloo [http://www.cloudgarden.com/jigloo/] GUI Builder

Netbeans

- Get from NetBeans homepage [http://www.netbeans.org]
- Current: 6.7.1
- Summer: 6.8 or 7.0

Important Features

- GUI Builder
- Profiler
- Built-in Maven support (since 6.7)

IntelliJ

- Built-in support for Maven
- Released as open source 16. Oct!
- http://www.jetbrains.com/idea/nextversion/free_java_ide.html
- Very good code inspection and refactoring features
- No experience yet

Version Management

History:

Shared File System	RCS
Single Repository	CVS, Subversion
Multiple Repositories	Mercurial, Git, Bazaar

Actually in use:

Subversion	Legacy projects
GIT	Unix Projects
Mercurial	Java / Python projects DV

Mercurial

Common to all DVCS:

- Distributed Version System
- Every client has a full copy of the repository
- Efficient Micro-Checkins
- Can push / pull from any other client copy
- Allow efficient branching

Important commands

hg init	create repository
hg clone	clone existing repository. Automatically makes this the default repository.
hg push	push changes to default remote repository
hg pull	pull changes from default remote repository (warning: does NOT update. Use **hg pull -u**)
hg commit	Check changes into local repository
hg update	Update files in local repository
hg heads	shows if there are different head revisions
hg merge	merges multiple head revisions

Can be configured in .hg/hgrc

Supports Plugins (here: Windows / Unix LF Conversion, Keyword expansion)

Example 3. Example hgrc

```
[paths]

[extensions]
hgext.keyword=
hgext.win32text=

[keyword]
**.java =
[encode]
** = cleverencode:
```

Ignore files can be configured using `.hgignore` in your root directory.

- Files will not be added to the repository
- Will not show up as changed
- Normally: All generated directories
- Maven: target directory

Example 4. Example .hgignore

```
syntax: regexp

target/
^test
^.*patch$
```

Links:

- Mercurial Homepage [http://www.selenic.com/mercurial/]
- Distributed revision control with Mercurial [http://hgbook.red-bean.com/] (online book)

Style

- Define Code Conventions (Naming, spaces vs. tabs, etc.)
- Re-use existing conventions!
- Follow them!
- Use tools to check them

Maven use:

- Put in <reporting> section

Checkstyle

- Checkstyle Homepage [http://checkstyle.sourceforge.net/]
- Checkstyle Eclipse plugin [http://eclipse-cs.sourceforge.net/]
- SQE NetBeans plugin [http://kenai.com/projects/sqe/] (Use version from Kenai for 6.7 Support!)
- Checkstyle Beans [http://www.sickboy.cz/checkstyle/] (may work better than SQE)
- Also provides some Code quality checks

Example 5. Example Checkstyle Maven Configuration

```
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-checkstyle-plugin</artifactId>
        <version>2.3</version>
        <configuration>
```

```
          <configLocation>checkstyle.xml</configLocation>
        </configuration>
      </plugin>
```

An example checkstyle.xml can be found at: http://wmsx.googlecode.com/hg/build-tools/src/main/resources/wmsx/checkstyle.xml

Checkstyle 4 vs 5

- The checkstyle maven plugin until version 2.3 is based on checkstyle 4.
- Current eclipse checkstyle plugins is based on checkstyle 5
- both are incompatible (slight changes in checkstyle.xml)

Fix: Use newer checkstyle plugin. Problem: Not officially available yet! Must set plugin location!

Example 6. Example Checkstyle 5 Maven Configuration

```
...reporting...plugins...
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-checkstyle-plugin</artifactId>
        <version>2.5-SNAPSHOT</version>
        <configuration>
          <configLocation>checkstyle.xml</configLocation>
        </configuration>
      </plugin>
...
  <pluginRepositories>
    <pluginRepository>
      <id>apache-snapshots</id>
      <name>Apache Snapshot repository</name>
      <url>http://repository.apache.org/snapshots</url>
      <snapshots>
        <enabled>true</enabled>
      </snapshots>
    </pluginRepository>
  </pluginRepositories>
```

PMD

Common Programming checks for code style.

Yes, all the standard rules make sense!

Plugins available for Eclipse, Netbeans, Maven

- PMD Homepage [http://pmd.sourceforge.net/]
- PMD Eclipse plugin [http://pmd-eclipse.sourceforge.net/]
- SQE NetBeans plugin [http://kenai.com/projects/sqe/] (Use version from Kenai for 6.7 Support!)

Example 7. PMD Configuration for Maven

```
     <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-pmd-plugin</artifactId>
```

```
          <version>2.4</version>
          <configuration>
            <linkXref>true</linkXref>
            <targetJdk>1.5</targetJdk>
            <sourceEncoding>utf-8</sourceEncoding>
            <rulesets>
              <ruleset>/rulesets/basic.xml</ruleset>
              <ruleset>/rulesets/braces.xml</ruleset>
              <ruleset>/rulesets/codesize.xml</ruleset>
              <ruleset>/rulesets/clone.xml</ruleset>
              <ruleset>/rulesets/design.xml</ruleset>
              <ruleset>/rulesets/finalizers.xml</ruleset>
              <ruleset>/rulesets/imports.xml</ruleset>
              <ruleset>/rulesets/strings.xml</ruleset>
              <ruleset>/rulesets/migrating_to_15.xml</ruleset>
              <ruleset>/rulesets/optimizations.xml</ruleset>
              <ruleset>/rulesets/sunsecure.xml</ruleset>
              <ruleset>/rulesets/unusedcode.xml</ruleset>
            </rulesets>
          </configuration>
        </plugin>
```

Findbugs

- Another Software quality tool
- Yes, the rules also make sense!
- Findbugs Homepage [http://findbugs.sourceforge.net/]
- Findbugs Eclipse plugin [http://findbugs.cs.umd.edu/eclipse/]
- SQE NetBeans plugin [http://kenai.com/projects/sqe/] (Use version from Kenai for 6.7 Support!)

Example 8. Findbugs Maven configuration

```
        <plugin>
          <groupId>org.codehaus.mojo</groupId>
          <artifactId>findbugs-maven-plugin</artifactId>
          <version>2.1</version>
          <configuration>
            <threshold>Low</threshold>
            <effort>Max</effort>
            <omitVisitors>FindDeadLocalStores</omitVisitors>
          </configuration>
        </plugin>
```

NCSS

- Non Commented Source Statements
- Simplified: The number of ;
- Better comparable than LOC (lines of code)

Example 9. NCSS Maven configuration

```
    <reporting>
    <plugins>
      <plugin>
        <groupId>org.codehaus.mojo</groupId>
        <artifactId>javancss-maven-plugin</artifactId>
        <version>2.0-beta-2</version>
      </plugin>
    </plugins>
  </reporting>
```

Logging

- Do Not use System.out for logging
- Use a logging framework

How:

- acquire logger: private static final LOG = Log.getLogger(Blabla.class);
- LOG.debug("bla");
- LOG.info("bla");
- logger can be configured for different log levels, and separately for each package / subpackage / class

log4j

- Commonly used
- Only use in new projects if you have previous experience with it!

commons-logging

- apache project
- auto-detects logger from classpath
- use in libraries

java.util.log

- built-in since Java 1.4
- Use in all user tools / gui projects