The Standard Directory Layout

The Standard Directory Layout :

In the course of your work, as a Java developer, you have probably used a lot of Open Source projects (if you haven't, well you are now). What is always amazing but at the same time annoying is that all the projects out there seem to use their own project directory layout flavour. For instance, Apache projects have their own standard (when they respect it), as Sun™ projects do to, ... Not using some sort of a standard directory layout leads to some problems:

- For every new project, you need to learn a new way of structuring your files.
- You need to specify a lot of redundant configuration in order to make your build tool work.

Nevertheless, all the projects don't use the same type of files (Java, XML, HTML, JSP, Groovy, SQL scripts, ...) so it might be hard to standardize the directory layout.

Standard Directory Layout Pattern:

The Standard Directory Layout Pattern address the following problem: **How can I organize my different projects files in a standard way, knowing each project use different file types, except some basic cases (Java sources files, Java test source files, ...)?**

*[pattern -description to be added] *

Of course, there are many different strategies, or if you prefer many ways to implement this pattern but we are interested in the Maven one.

Maven standard directory layout strategy

At the top level, descriptive files of the project are found: a `pom.xml` file (and any properties, `maven.xml` or `build.xml` if using Ant). In addition, there are **textual documents meant for the user to be able to read immediately** on receiving the source: `README.txt`, `LICENSE.txt`, etc.

There are just two subdirectories of this structure: `src` and `target`. The only other directories that would be expected here are metadata like `CVS` or `.svn`, and any subprojects in a multiproject build (each of which would be laid out as above).

The `target` directory is used to house all output of the build.

The `src` directory contains all of the source material for building the project, its site and so on. It contains a **subdirectory for each type of use**: `main` for the main build artifact, `test` for the unit test code and resources, `site` and so on.

Within artifact producing source directories (ie. `main` and `test`), there is **one subdirectory for each type of file**. For instance, Java source files are stored under the subdirectory *java*, the resources files (the ones copied directly to the output directory) under the subdirectory *resources*, ... The name you give to each of those subdirectories usually depends on the plugins you use to manage those types of files (plugins usually assume some default names, so no need to reinvent the wheel there).

Here are the directories that Maven's standard set of plugins use:

Directory	Description
src/main/java	Application/Library sources
src/main/resources	Application/Library resources
src/main/filters	Resource filter files
src/main/assembly	Assembly descriptors
src/main/webapp	Web application sources
src/test/java	Test sources
src/test/resources	Test resources
src/test/filters	Test resource filter files
src/site	Site
src/site/apt	Documentation in apt format.

src/site/fml	Documentation in fml format.
src/site/resources	Resources for the site; will be copied into target/site as-is.
src/site/xdoc	Documentation in xdoc format.
src/site/site.xml	The site's descriptor.

And here is what your project directory layout should look like in most common cases :