# Home

This is the home page for the Maven users space. Where users can contribute documentation for others users and for incorporation into the main site. Any users with outside documentation should share it with others here. This is our sandbox and it's only as good as we make it.

Just for fun.... A song for Maven's developers: Got My Mojo Working

# NOTICE

Read This First

## Table of Contents

This wiki space is not for final documentation, but a workspace for creating it. If you think a document can be contributed back to the main site, please record that in JIRA. For discussions, ideas and design documents related to Maven, see the Maven space.

# Introduction

If you're a Java software developer that has worked in any type of team environment, you know that having a proper process is important to developing a quality product. You also know that rote methods can help maintain consistency and standardization among developers. Apache ANT provides most of this however Maven improves upon it in several areas. Maven takes some of that classpath and jar dependency frustrations out of the hands of the day-to-day developer. Most would agree that maintaining the project structure, dependencies, J2EE packaging, etc. is very difficult and when things are not structured correctly, applications cannot be built properly. Jars could be in the wrong place, be the wrong version, an XML file might be missing a descriptor, and a host of other project infrastructure issues can bring a project to stop. When this happens its sometimes difficult to determine where things went wrong.  Maven is an excellent tool that solves most of these problems.

While the typical project manager might disagree, Maven meets the project manager and the development staff somewhere in the middle. Apache's ANT is a powerful tool, but there's no standardization in targets. In fact, it may be too customizable. Maven did right by holding to a distinct set of build lifecycle goals. If you're the do-it-yourself type, you can create your own plugins to extend the functionality of Maven. Project managers want control and Maven injects this technical control into the build process. Maven assures everyone that the build and deploy process is done the same every time. Musicians practice scale repetition for this very reason. Rote processes are very effective because their consistency and effectiveness save time and assure quality.

The documentation listed in the table of contents is meant to be a place to begin to learn how to use Maven. Many development shops and software groups may be using Maven1 now. If you are, please take a look at Maven2. For anything to become a standard, it must be proven to be the standard by our use. Let's make Maven(2) the standard in our build process. Let's go write quality software and let Maven do our heavy lifting.

# Help Wanted!

Here are some ways you can contribute:

- Write or adjust the docs in this wiki space
- Add placeholder pages for docs you'd like, with an outline about what you think is needed (preferably file in JIRA and link to the page too)
- Help organise the content on these pages, and contribute it as a patch to the Maven site documentation. See http://maven.apache.org/guides/development/guide-helping.html for more information.

# TODO-CU

Looking for some documentations you can't seem to find? - Request for a wiki documentation by adding a list here.

Want to write a wiki documentation but don't know where to put it? - Write it first here, then let's transfer it later.

# Other material

Material on Maven 2, outside the maven site.

- Working with Maven 2: bullet-type course notes
- Testing strategies with Maven 2: Notes on testing strategies
- Maven related materials: Articles, books, extensions, IDEs, tutorials.
- Development by Example: Maven & Archiva