

eHealth Framework

How to use the eHF Masterbuild Maven Plugin

Notice

The wording in this document applies equally to women and men. The masculine form was selected to ease the comprehensibility and legibility of the text.

All company logos are a registered trademark of InterComponentWare AG.

The product names mentioned in this documentation are either trademarks or registered trademarks of the respective owners and are stated for identification purposes only.

This documentation and the software components are protected by copyright © 2006-2009 InterComponentWare AG.



Note:

The current version of this document has a draft status and various chapters are still in review.

The document is collaboratively built with the use of the Darwin-Information-Typing-Architecture (DITA) and has therefore a draft status concerning styles and layout. The necessary adaptations are currently also in a developmental stage.

All rights reserved.

Contents

1 Overview	1
2 What is the eHF Masterbuild Maven Plugin	2
3 Example	3
4 Installation	5
5 Goals	6
6 Properties	7
7 Dependencies	9
8 Future	10

1 Overview

Purpose

This how-to explains how to configure and use the eHF Masterbuild Maven Plugin.

Scope

The how-to is especially interesting to people who are involved in Build Management and Continuous Integration topics. In particular to those who need to build several interdependent modules.

2 What is the eHF Masterbuild Maven Plugin

The *eHF Masterbuild Maven Plugin* was created because the concept of multiple artifacts per Maven module used by the eHF Multiartifact Maven Plugin does not stick to the Maven 1 artifact naming conventions. Thus the standard Maven 1 Reactor build employed by the Maven Multiproject Plugin cannot be used since it will not correctly resolve module dependencies for modules that use dependencies created by the eHF Multiartifact Maven Plugin.

The *eHF Masterbuild Maven Plugin* provides a dependency resolver that was inspired by the default implementation provided by Maven 1.

Please also see:

- [DependencyResolverInterface](#) ↗
- [DependencyResolver](#) ↗
- [WerkzDependencyResolver](#) ↗

Our implementation though is optimized (speedup by the factor 32) and more intelligent (e.g. checks for naming collisions before calculating the build sequence).

3 Example

Please consider a folder with the following content:

```
drwxr-xr-x 3 build users 4096 2009-07-28 08:34 ehf-build-tools
drwxr-xr-x 3 build users 4096 2009-07-28 08:34 ehf-generator
drwxr-xr-x 3 build users 4096 2009-07-28 08:34 ehf-test-tools
-rw-r--r-- 1 build users  27 2009-07-28 08:34 project.properties
```

The folders contained are each a self-contained Maven1 project.

The project.properties have the following content:

```
ehf.masterbuild.goals=clean
```

This results in the clean goal being executed for all modules.

Now lets execute the eHF Masterbuild Maven Plugin and have a look at the result:

```
%> maven masterbuild:masterbuild

|__/\_|__ _Apache__ __
| |\V| / _` \ V / -_) ' \ ~ intelligent projects ~
|_|  |_\_,_| \_\/\_____|_|  v. 1.1

Running reactor...

Calculated sequence:
ehf-build-tools
ehf-test-tools
ehf-generator

Writing sequence file to target folder.

Building projects...

Building: 'ehf-build-tools'
Took: 2 second(s)
Building: 'ehf-test-tools'
Took: 1 second(s)
Building: 'ehf-generator'
Took: 2 second(s)

Building 3 project(s) completed in 7 second(s).

build:start:

masterbuild:masterbuild:
-----
BUILD SUCCESSFUL
-----
Total time   : 8 seconds
Finished at  : Tuesday, July 28, 2009 1:58:42 PM UTC
Final Memory : 3M/1016M
-----
```

As you can see first the sequence is calculated and then the projects are build accordingly. Now lets have a look at the target folder content:

How to use the eHF Masterbuild Maven Plugin

```
%> ll target/  
total 20  
-rw-r--r-- 1 build users 521 2009-07-28 13:58 ehf_ehf-build-tools.log  
-rw-r--r-- 1 build users 784 2009-07-28 13:58 ehf_ehf-generator.log  
-rw-r--r-- 1 build users 521 2009-07-28 13:58 ehf_ehf-test-tools.log  
-rw-r--r-- 1 build users  45 2009-07-28 13:58 folder-sequence.txt  
-rw-r--r-- 1 build users  57 2009-07-28 13:58 id-sequence.txt
```

You see a log file for each project build and two sequence files (one containing the project ids and one containing the project folders).

4 Installation

For installing the *eHF Masterbuild Maven Plugin* you simply need to copy the plugin JAR (e.g. `masterbuild-maven-plugin-0.0.3.jar`) into `$MAVEN_HOME/plugins`.

5 Goals

The *eHF Masterbuild Maven Plugin* has the following goals:

Goal	Description
masterbuild:masterbuild	Needs to be executed in a folder that contains Maven 1 projects. It will discover all projects, calculate a build sequence according to the projects dependencies and then it will try to build all the projects along the sequence executing the goal masterbuild:all (unless you specify otherwise).

6 Properties

Property	Description	Default Value
ehf.masterbuild.filters	Defines the naming extensions used by the eHF Multiartifact Maven Plugin. These string delimited by comma are used to discover multiartifacts by filtering the projects artifactIds.	-config,-runtime,-api,-bootstrap,-webapp,-test,-doc
ehf.masterbuild.filterexclusions	By setting this property you can explicitly define which artifactIds to exclude from name filtering. In general this property should not be used since the plugin discovers exclusions on its own.	-
ehf.masterbuild.includes	This comma delimited list defines what POMs to include when discovering Maven1 projects.	*/project.xml
ehf.masterbuild.excludes	This comma delimited list defines what POMs to exclude when discovering Maven1 projects.	By default nothing is excluded.
ehf.masterbuild.executebuild	If true the projects will also be build after calculating the correct build order.	true
ehf.masterbuild.goals	By setting this property one can change the goal that is executed for the project builds.	masterbuild:all
ehf.masterbuild.mavenparameter	Here you can give additional command line parameters to Maven itself (e.g. - <i>Dmaven.repo.local=/tmp/some-path</i>).	-
ehf.masterbuild.targetdir	Directory to write output to. The plugin will create <ul style="list-style-type: none"> • <i>maven1_masterbuild.log</i>: Overall log file. • <i>folder-sequence.txt</i>: File containing the build sequence represented by the root directories of the projects. • <i>id-sequence.txt</i>: File containing the build sequence represented by the project IDs (groupId + artifactId). • <i><PROJECT_ID>.log</i>: One file for each Maven1 project 	\${basedir}/target

How to use the eHF Masterbuild Maven Plugin

Property	Description	Default Value
	containing the log output of the given project's build.	
ehf.masterbuild.basedir	Directory to look for the Maven1 projects.	\${basedir}

7 Dependencies

The *eHF Masterbuild Maven Plugin* has the following runtime dependencies you need to consider:

```
<dependency>
  <groupId>commons-exec</groupId>
  <artifactId>commons-exec</artifactId>
  <version>1.0</version>
</dependency>
```

It will also only work with Maven 1.1 and unmodified \$MAVEN_HOME/libs.

8 Future

For the Maven 2 migration of the eHF remains to say, that we then will stick to the Maven naming conventions by using Maven Qualifiers.