EclipseEnvironment

Working with Hadoop under Eclipse

Here are instructions for setting up a development environment for Hadoop under the Eclipse IDE. Please feel free to make additions or modifications to this page.

This document assumes you already have Eclipse downloaded, installed, and configured to your liking. It also assumes that you are aware of the HowToContribute page and have given that a read.

Quick Start

We will begin by downloading the Hadoop source. The hadoop-common source tree has three subprojects underneath it that you will see after you pull down the source code: hadoop-common, hdfs, and mapreduce.

Let's begin by getting the latest source from Git (Note there is a a copy mirrored on github but it lags the Apache read-only git repository slightly).

```
git clone git://git.apache.org/hadoop-common.git
```

This will create a hadoop-common folder in your current directory, if you "cd" into that folder you will see all the available subprojects. Now we will build the code to get it ready for importing into Eclipse.

From this directory you just 'cd'-ed into (Which is also known as the top-level directory of a branch or a trunk checkout), perform:

```
$ mvn install -DskipTests
$ mvn eclipse:eclipse -DdownloadSources=true -DdownloadJavadocs=true
```

Note: This may take a while the first time, as all libraries are fetched from the internet, and the whole build is performed.

In Eclipse

After the above, do the following to finally have projects in Eclipse ready and waiting for you to go on that scratch-itching development spree:

For Common

- File -> Import...
- Choose "Existing Projects into Workspace"
- Select the hadoop-common-project directory as the root directory
- Select the hadoop-annotations, hadoop-auth, hadoop-auth-examples, hadoop-nfs and hadoop-common projects
- Click "Finish"
- File -> Import...
- Choose "Existing Projects into Workspace"
- Select the hadoop-assemblies directory as the root directory
- Select the hadoop-assemblies project
- Click "Finish"
- To get the projects to build cleanly:
- * Add target/generated-test-sources/java as a source directory for hadoop-common
- * You may have to add then remove the JRE System Library to avoid errors due to access restrictions

For HDFS

- File -> Import...
- Choose "Existing Projects into Workspace"
- Select the hadoop-hdfs-project directory as the root directory
- Select the hadoop-hdfs project
- Click "Finish"

For MapReduce

- File -> Import...
- Choose "Existing Projects into Workspace"
- Select the hadoop-mapreduce-project directory as the root directory
- Select the hadoop-mapreduce-project project
- Click "Finish"

For YARN

- File -> Import...
- Choose "Existing Projects into Workspace"
- Select the hadoop-yarn-project directory as the root directory
- Select the hadoop-yarn-project project

- Click "Finish"

Note: in the case of MapReduce the `testjar` package is broken. This is expected since it is a part of a testcase that checks for incorrect packaging. This is not to be worried about.

To run tests from Eclipse you need to additionally do the following:

- Under project Properties, select Java Build Path, and the Libraries tab
- Click "Add External Class Folder" and select the `build` directory of the current project

Footnotes

- With the new release of the m2e plug-in, this doesn't work anymore as pretty much all targets are not supported by the new 'connector framework' - Yes, it is a giant mess. This means falling back to m2eclipse or just doing the eclipse generation via mvn as mentioned in this page.