HadoopUnderIDEA

Outdated

This wiki page is well outdated as Hadoop 2.x uses Mavenized builds. IntelliJ IDEA can import Maven projects via the root pom.xml file.

Building/Testing Hadoop under IDEA

Hadoop builds with Ant, but you can set it up to work under IDEA for testing and some iterative development. This does not take away the need to run Ant; you just run it side by side.

Before you begin

- Be on Java6. That's where Hadoop is going.
- Have the build work on the command line.

Building

- Create and edit a 'build.properties' file; add the following line

```
build.webapps=build/classes/webapps
```

- The above line is required. The default location, '`build/webapps`', will not work. `StatusHttpServer` locates 'webapps' via

  ```
   classloader.getResource("webapps")
  ```

  and it must be in classpath.

- Do a command line build first, to set up all the output dirs.

Creating a new module

Create a new IDEA module for Hadoop.

Source folders

```
build/src
conf
src/ant
src/contrib/streaming/src/java
src/core
src/examples
src/hdfs
src/mapred
src/native/src
src/tools
```

- Even if you are not working on an area, adding it to the source tree makes refactoring and class/method use queries more reliable.
- Everything under `build/` goes away on a clean build, and needs to be picked up again by resynchronizing IDEA (if it is not automatic)
- By default, the webapp metadata goes into `build/webapps`, which is not the right place to be picked up by the IDE. Moving it under `build/resources/` is needed to place it somewhere manageable.
- `build/src` is required for compiled jsp files. Unfortunately, there is no separated ant task to regenerate them. The best is running ant command line.
- `conf` is required for `hadoop-default.xml` to be copied to `build/classes`. `Configuration` will load `hadoop-default.xml` as a resource via classloader.

test source directories

```
src/test
build/test/src
```

- Exclude stuff under there that you do not need.
- Like 'Source folders', everything under `build/` goes away on a clean build. You can re-create it by running 'generate-test-records' ant task.

Build Paths

Set these to the full path of where Hadoop's Ant build sticks things, such as :

- Output: `/home/user/hadoop-core/build/classes`
- Test output: `/home/user/hadoop-core/build/test/classes`

Libraries

- everything in the `lib/` directory.
- If you have global libraries set up for Log4J, JUnit and Jetty (for example), use them and omit the versions in `lib/`. Do keep in sync with library versions, especially that of Jetty.
- For tests, you need to add everything in the `src/test/lib/` directory.

Setting up a test run

To run JUnit tests under the IDE, create a new test configuration pointing to the chosen tests.

- Use the classpath and JDK of the Hadoop module.
- Select the package, class or method containing the chosen tests
- VM Parameters: you must set up the logging directory

```
-Dhadoop.log.dir=/home/user/hadoop-core/build/test/logs
```

JRockit users: consider editing `conf/log4j.properties` to

```
log4j.appender.console.layout.ConversionPattern=%-4r %-5p %c %x - %m%n
```

This may seem odd, but it eliminated deadlocks in the logging.