Hibernate 4.x/5.x uses Gradle - For reasons why Hibernate switched from Maven to Gradle see - http://community.jboss.org/wiki/Gradlewhy

Prerequesites

- JDK 6 for 4.x and JDK 7 for 5.x/HEAD
- Git
- Gradle (you don't have to install Gradle on your system prior to building Hibernate. You can use the gradle wrapper *gradlew* to bootstrap Gradle and the build. You find *gradlew* in the root directory of your Hibernate git clone)

Get the sources

```
1. > git clone git://github.com/hibernate/hibernate-orm.git
```

You can also browse the code on GitHub: https://github.com/hibernate/hibernate-orm

Build tasks

Build all

```
1. ./gradlew clean build
```

Running tests

```
1. ./gradlew test
```

Running single tests

Single tests are executed via specifying the property *test.single* - see also Gradle Cookbook

```
1. ./gradlew -Dtest.single=ThisUniquelyNamedTest test
```

Skipping tests

```
1. ./gradlew build -x test
```

Other tasks

You get a full list of available tasks via

```
1. ./gradlew --tasks --all
```

Some importnant tasks are listed below:

- *build* - Assembles (jars) and tests this project
- *buildDependents* - Assembles and tests this project and all projects that depend on it. So think of running this in hibernnate-entitymanager, Gradle would assemble and test hibernate-entitymanager as well as hibernate-envers (because envers depends on entitymanager)
- *classes* - compiles the main classes
- *clean* - Deletes the build directory
- *jar* - Generates a jar archive with all the compiled classes
- *testClasses* - Assembles the test classes
- *test* - Runs the unit tests
- *uploadArchives* - think Maven deploy
- *install* - installs the project jar to your local maven cache (aka ~/.m2/repository)
- *idea* or *eclipse* - creates the project files for the respective IDE

Gradle wrapper

The above examples use the gradle wrapper (*gradlew*) to run the different build tasks. Using the wrapper you don't have to install Gradle prior to building the source. It also ensures that you are using the right version of gradle as specified in *build.gradle*. It requires, however, that you specify the path to *gradlew*. For example if you are in the hibernate-core module you need to use *../gradlew*. One way to avoid this is to add some custom functions or aliase to your *~/.bashrc* (assuming you are using Unix). Something like this might do:

```
1. function gradleProjectRoot()
2. {
3.       x=`pwd`; while [ "$x" != "/" ] ; do if [ `find "$x" -
maxdepth 1 -name gradlew` ]; then echo "$x"; break; fi;
x=`dirname "$x"`; done
4. }
5.
```

```
function gradlew()
{
    `gradleProjectRoot`/gradlew $@
}
```

This way you can just type *gradlew* in any subdirectory of the project root and the right wrapper script will be executed. If you have a better script, let us know :-)

IDE settings

Eclipse

To create the Eclipse project files you can run
./gradlew eclipse
After changes to the dependencies you need to clean the project files and recreate them:
./gradlew cleanEclipse eclipse

See also Contributing to Hibernate using Eclipse

Idea

To create the Idea project files you can run

```
./gradlew idea
```

After changes to the dependencies you need to clean the project files and recreate them:

```
./gradlew cleanIdea idea
```

See also general Idea setup information - IntelliJ Information

FAQ

Why does the build fail with unresolved dependencies

It could be that you tried building with a different gradle version as the one specified in build.gradle. It is probably better to use the gradlew wrapper script. Try cleaning out the gradle caches and try again:

```
> rm -rf ~/.gradle
> rm -rf <your-orm-checkout-dir>/.gradle
```