# MyBatis Generator - How to Run the Examples

All examples are ready to be run using Apache Ant. They can be run from Eclipse or, if you prefer, from the command line.

To run an example you must start the database and prepare the tables and data. Then you can use HotRod to generate all DAO classes. Finally you can run your application that uses the DAO classes.

First, let's load the Ant build.xml file in Eclipse:

1. In Eclipse open the Ant view (Window -> Show View -> Ant). This opens the Ant window that is (for now) empty.
2. Find the build.xml file in the example folder and drag an drop it into the Ant window you just opened.
3. Expand the build.xml entry in the Ant window (click on the left arrow icon). Now you can see all the available tasks Ant can do for you (“targets” in Ant lingo). Don't click on them yet! Double-clicking will execute them.

All examples run against a local H2 database. H2 is a full featured open source RDBMS. It can be easily started and stopped and makes the examples run more easily.

## Start the H2 Database

To start the H2 database find the task start-h2-database and double-click on it. The console window will open and show something like:

Buildfile: examples/mybatis/helloworld/build.xml

start-h2-database:

-issue-h2-start:

[echo] [ H2 local database has been started ]

-h2-already-started:

BUILD SUCCESSFUL

Total time: 365 milliseconds

This means the database is now up and running! Yes running... but also empty. If you're not sure H2 is started, it's safe to run this task multiple times; if H2 is already up it won't start it again.

## Prepare the Database Tables and Sample Data

To initialize the database tables and sample data find the task initialize-database and double click on it. The console window will open and show something like:

Buildfile: examples/mybatis/helloworld/build.xml

initialize-database:

[sql] Executing resource: examples/mybatis/helloworld/prepare-database.sql

[sql] 0 rows affected

[sql] 0 rows affected

[sql] 1 rows affected

[sql] 1 rows affected

[sql] 1 rows affected

[sql] 5 of 5 SQL statements executed successfully

[echo] [ Database tables and data initialized ]

BUILD SUCCESSFUL

Total time: 895 milliseconds

All tables and data are now created.

## HotRod Generation

To run HotRod and generate all DAO classes find the task hotrod and double click on it. The console window will open and show something like:

Buildfile: examples/mybatis/helloworld/build.xml

hotrod:

[delete] Deleting directory examples/mybatis/helloworld/auto-generated

[mkdir] Created dir: examples/mybatis/helloworld/auto-generated/java

[mkdir] Created dir: examples/mybatis/helloworld/auto-generated/mappers

[hotrod] HotRod version 3.0 (build 20170124-131633)

[hotrod]

[hotrod] Database URL: jdbc:h2:tcp://localhost:12345/db001;IFEXISTS=TRUE

[hotrod] Database Name: H2 - version 1.3 (1.3.176 (2014-04-05))

[hotrod] JDBC Driver: H2 JDBC Driver - version 1.3 (1.3.176 (2014-04-05))

[hotrod]

[hotrod] HotRod Database Adapter: H2 Adapter

[hotrod] Database Catalog: DB001

[hotrod] Database Schema: PUBLIC

[hotrod]

[hotrod] Generating all facets.

[hotrod]

[hotrod] Table VEHICLE included.

[hotrod]

[hotrod] Generating MyBatis DAOs for 1 table, 0 views, 0 DAOs (0 sequences, 0 updates), and 0 select queries...

[hotrod]

[hotrod] MyBatis generation complete.

BUILD SUCCESSFUL

Total time: 882 milliseconds

HotRod connected to the database and generated all DAO classes for the tables specified in the configuration file.

## Run the Example Application

Now all the DAOs (and MyBatis mappers) are ready to run your example application. Find the task for the example with the name run-example-01 (change the number accordingly) and double click on it.

The console window will open and show the output of the example:

Buildfile: examples/mybatis/examples/build.xml

run-example-01:

[delete] Deleting directory examples/mybatis/examples/build

[mkdir] Created dir: examples/mybatis/examples/build

[javac] Compiling 4 source files to examples/mybatis/examples/build

[echo]

[java] List of all vehicles, before changes:

[java] ID, BRAND, MODEL, USED, CURRENT\_MILEAGE, PURCHASED\_ON

[java] [1, Kia, Soul, true, 28500, 2014-03-14]

[java] [2, Toyota, Tercel, false, 26, 2017-01-28]

[java] [3, DeLorean, DMC-12, true, 241689, 1982-11-17]

[java]

[java] New vehicle Skoda added. New ID=4. Rows inserted=1

[java]

[java] DeLorean found.

[java]

[java] DeLorean updated. Rows updated=1

[java]

[java] Toyota deleted.

[java]

[java] List of all vehicles, after all changes:

[java] ID, BRAND, MODEL, USED, CURRENT\_MILEAGE, PURCHASED\_ON

[java] [1, Kia, Soul, true, 28500, 2014-03-14]

[java] [3, DeLorean, DMC-12, true, 270500, 1982-11-17]

[java] [4, Skoda, Octavia, false, 7, 2017-01-24]

BUILD SUCCESSFUL

Total time: 1 second

That's it! The example application ran as expected.

## Stop the H2 database

You can run the example multiple times if needed. Once you're done with it, don't forget to shutdown the H2 database. Find the task stop-h2-database and double-click on it. The console window will open and show something like:

Buildfile: examples/mybatis/examples/build.xml

stop-h2-database:

-issue-h2-stop:

[java] Shutting down TCP Server at tcp://localhost:12345

[echo] [ H2 local database has been stopped ]

-h2-already-stopped:

BUILD SUCCESSFUL

Total time: 2 seconds

This means the database is now shut down. If you're not sure you stopped H2 it's safe to run this task multiple times; if H2 is already down it won't stop it again.

Also, as a bonus the database data with all changes produced by the example is actually kept on disk. If you start H2 again you will see the data you modified in the previous session, and will remain there until you re-initialize the database again.