Java Hook Tutorial

One common use case for OrientDB Hooks (a.k.a. database triggers) is to manage created and updated dates for any or all classes (a.k.a. database tables). For example, it is nice to be able to set a CreatedDate field whenever a record is created and set an UpdatedDate field whenever a record is updated, and do it in a way where you implement the logic once at the database layer and never have to worry about it again at the application layer.

The following tutorial will walk you through exactly how to accomplish this use case using an OrientDB Hook and we'll do it from the perspective of a novice Java programmer working on a Windows machine.

Assumptions

It is assumed that you have already downloaded and installed a Java JDK. In my case I downloaded Java JDK version 8 for Windows 64 bit and installed it to folder C:\Program Files\Java\jdk1.8.0_40.

It is also assumed that you have downloaded, installed, and configured a working OrientDB Server. In my case I installed it to folder C:\Program Files\orientdb-community-2.0.5.

Exact instructions for these two steps are outside the scope of this tutorial.

Initial Server Configuration File

My OrientDB server configuration file is located at C:\Program Files\orientdb-community-2.0.5\config\orientdb-server-config.xml and is configured like this:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<orient-server>
    <handlers>
        <handler class="com.orientechnologies.orient.graph.handler.OGraphServerHandler">
            <parameters>
                <parameter value="true" name="enabled"/>
                <parameter value="50" name="graph.pool.max"/>
            </parameters>
        </handler>
```

```
        <handler class="com.orientechnologies.orient.server.hazelcast.OHazelcastPlugin">
            <parameters>
                <parameter value="false" name="enabled"/>
                <parameter value="${ORIENTDB_HOME}/config/default-distributed-db-config.j
                <parameter value="${ORIENTDB_HOME}/config/hazelcast.xml" name="configurat
            </parameters>
        </handler>
        <handler class="com.orientechnologies.orient.server.handler.OJMXPlugin">
            <parameters>
                <parameter value="false" name="enabled"/>
                <parameter value="true" name="profilerManaged"/>
            </parameters>
        </handler>
        <handler class="com.orientechnologies.orient.server.handler.OAutomaticBackup">
            <parameters>
                <parameter value="false" name="enabled"/>
                <parameter value="4h" name="delay"/>
                <parameter value="backup" name="target.directory"/>
                <parameter value="${DBNAME}-${DATE:yyyyMMddHHmmss}.zip" name="target.file
                <parameter value="9" name="compressionLevel"/>
                <parameter value="1048576" name="bufferSize"/>
                <parameter value="" name="db.include"/>
                <parameter value="" name="db.exclude"/>
            </parameters>
        </handler>
        <handler class="com.orientechnologies.orient.server.handler.OServerSideScriptInte
            <parameters>
                <parameter value="false" name="enabled"/>
                <parameter value="SQL,JAVASCRIPT" name="allowedLanguages"/>
            </parameters>
        </handler>
        <handler class="com.orientechnologies.orient.server.token.OrientTokenHandler">
            <parameters>
                <parameter value="true" name="enabled"/>
                <parameter value="INSERT YOUR OWN HERE" name="oAuth2Key"/>
                <parameter value="60" name="sessionLength"/>
                <parameter value="HmacSHA256" name="encryptionAlgorithm"/>
            </parameters>
        </handler>
    </handlers>
    <network>
        <sockets>
            <socket implementation="com.orientechnologies.orient.server.network.OServerSS
                <parameters>
                    <parameter value="false" name="network.ssl.clientAuth"/>
```

```
                    <parameter value="config/cert/orientdb.ks" name="network.ssl.keyStore
                    <parameter value="password" name="network.ssl.keyStorePassword"/>
                    <parameter value="config/cert/orientdb.ks" name="network.ssl.trustSto
                    <parameter value="password" name="network.ssl.trustStorePassword"/>
                </parameters>
            </socket>
            <socket implementation="com.orientechnologies.orient.server.network.OServerSS
                <parameters>
                    <parameter value="false" name="network.ssl.clientAuth"/>
                    <parameter value="config/cert/orientdb.ks" name="network.ssl.keyStore
                    <parameter value="password" name="network.ssl.keyStorePassword"/>
                    <parameter value="config/cert/orientdb.ks" name="network.ssl.trustSto
                    <parameter value="password" name="network.ssl.trustStorePassword"/>
                </parameters>
            </socket>
        </sockets>
        <protocols>
            <protocol implementation="com.orientechnologies.orient.server.network.protoco
            <protocol implementation="com.orientechnologies.orient.server.network.protoco
        </protocols>
        <listeners>
            <listener protocol="binary" socket="default" port-range="2424-2430" ip-addres
            <listener protocol="http" socket="default" port-range="2480-2490" ip-address=
                <commands>
                    <command implementation="com.orientechnologies.orient.server.network.
                        <parameters>
                            <entry value="Cache-Control: no-cache, no-store, max-age=0, r
                            <entry value="Cache-Control: max-age=120" name="http.cache:de
                        </parameters>
                    </command>
                    <command implementation="com.orientechnologies.orient.graph.server.co
                </commands>
                <parameters>
                    <parameter value="utf-8" name="network.http.charset"/>
                </parameters>
            </listener>
        </listeners>
    </network>
    <storages/>
    <users>
        <user resources="*" password="INSERT YOUR OWN HERE" name="root"/>
        <user resources="connect,server.listDatabases,server.dblist" password="guest" nam
    </users>
    <properties>
        <entry value="1" name="db.pool.min"/>
```

```
        <entry value="50" name="db.pool.max"/>
        <entry value="true" name="profiler.enabled"/>
        <entry value="info" name="log.console.level"/>
        <entry value="fine" name="log.file.level"/>
        <entry name="server.database.path" value="/data/orientdb" />
    </properties> [
</orient-server>
```

Step 1 - Install Apache Maven

Apache Maven is a useful tool for people wishing to write and compile Java programs, which you will need to do in order to create an OrientDB Java Hook.

You can download Apache Maven fromhttps://maven.apache.org/download.cgi. Follow the installation instructions until you can open a command prompt and successfully run the command

```
mvn --help
```

Step 2 - Create a new Maven project

Open a command prompt and change directory to some root folder where you like to code. Feel free to use a directory inside the a repository you already have.

Before you create a Maven project, it is useful to think about how to you want to name your code package. Package organization is usually a heated discussion but I just like to keep it unique but simple. I choose to put all of my OrientDB java hooks in a package called river.hooks. Therefore, I might call my first hook river.hooks.hook1 and my next hook river.hooks.hook2.

Now create a new Maven project in the folder location you have selected by running the following command:

```
mvn -B archetype:generate -DarchetypeGroupId=org.apache.maven.archetypes -DgroupId=river
```

You'll notice that the result of this command is a brand new directory structure created underneath the folder in which you ran the command. Take special note that Maven has created a file called .\hooks\pom.xml and a folder called .\hooks\src\main\java\river\hooks.

Edit pom.xml

The thing you need to pay attention to in this file is the section called dependencies. As your OrientDB Java Hook will leverage OrientDB code, you need to tell Maven to download and cache the OrientDB code libraries that your hook needs. Do this by adding the following to your pom.xml file:

```
<dependencies>
  ...
  <dependency>
    <groupId>com.orientechnologies</groupId>
    <artifactId>orientdb-core</artifactId>
    <version>2.0.5</version>
  </dependency>
</dependencies>
```

Create hook file(s)

Now that Maven knows that your code will build upon the oriendb-core code libraries, you can start writing your Hook file(s). Go to folder .\hooks\src\main\java\river\hooks. This is the folder where you will put your .java hook files. Go ahead and delete the placeholder App.java file that Maven created and which you don't need.

Let's start out by adding a HookTest.java file as follows:

```
package river.hooks;
import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.StringReader;
import java.util.ArrayList;
import java.util.List;
import java.util.concurrent.locks.ReentrantLock;
import com.orientechnologies.orient.core.hook.ODocumentHookAbstract;
```

```
import com.orientechnologies.orient.core.hook.ORecordHook;
import com.orientechnologies.orient.core.hook.ORecordHookAbstract;
import com.orientechnologies.orient.core.db.ODatabaseLifecycleListener;
import com.orientechnologies.orient.core.db.ODatabase;
import com.orientechnologies.orient.core.record.ORecord;
import com.orientechnologies.orient.core.record.impl.ODocument;

public class HookTest extends ODocumentHookAbstract implements ORecordHook {
  public HookTest() {
    setExcludeClasses("Log"); //if comment out this one line or leave off the constructor
  }

  @Override
  public DISTRIBUTED_EXECUTION_MODE getDistributedExecutionMode() {
    return DISTRIBUTED_EXECUTION_MODE.BOTH;
  }

  public RESULT onRecordBeforeCreate( ODocument iDocument ) {
    System.out.println("Ran create hook");
    return ORecordHook.RESULT.RECORD_NOT_CHANGED;
  }

  public RESULT onRecordBeforeUpdate( ODocument iDocument ) {
    System.out.println("Ran update hook");
    return ORecordHook.RESULT.RECORD_NOT_CHANGED;
  }

}
```

What this sample code does is print out the appropriate comment every time you create or update a record of that class.

Let's add one more hook file setCreatedUpdatedDates.java as follows:

```
package river.hooks;
import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.StringReader;
import java.util.ArrayList;
import java.util.List;
import java.util.concurrent.locks.ReentrantLock;
```

```
import com.orientechnologies.orient.core.hook.ODocumentHookAbstract;
import com.orientechnologies.orient.core.hook.ORecordHook;
import com.orientechnologies.orient.core.hook.ORecordHookAbstract;
import com.orientechnologies.orient.core.db.ODatabaseLifecycleListener;
import com.orientechnologies.orient.core.db.ODatabase;
import com.orientechnologies.orient.core.record.ORecord;
import com.orientechnologies.orient.core.record.impl.ODocument;

public class setCreatedUpdatedDates extends ODocumentHookAbstract implements ORecordHook
  public setCreatedUpdatedDates() {
    setExcludeClasses("Log"); //if comment out this one line or leave off the constructor
  }

  @Override
  public DISTRIBUTED_EXECUTION_MODE getDistributedExecutionMode() {
    return DISTRIBUTED_EXECUTION_MODE.BOTH;
  }

  public RESULT onRecordBeforeCreate( ODocument iDocument ) {
    if ((iDocument.getClassName().charAt(0) == 't') || (iDocument.getClassName().charAt(0
      iDocument.field("CreatedDate", System.currentTimeMillis() / 1000l);
      iDocument.field("UpdatedDate", System.currentTimeMillis() / 1000l);
      return ORecordHook.RESULT.RECORD_CHANGED;
    } else {
      return ORecordHook.RESULT.RECORD_NOT_CHANGED;
    }
  }

  public RESULT onRecordBeforeUpdate( ODocument iDocument ) {
    if ((iDocument.getClassName().charAt(0) == 't') || (iDocument.getClassName().charAt(0
      iDocument.field("UpdatedDate", System.currentTimeMillis() / 1000l);
      return ORecordHook.RESULT.RECORD_CHANGED;
    } else {
      return ORecordHook.RESULT.RECORD_NOT_CHANGED;
    }
  }

}
```

What this code does is look for any class that starts with the letters r or t and sets CreatedDate and UpdatedDate when the record gets created and sets just UpdatedDate every time the record gets updated.

Step 3 - Compile your Java hooks

In a command prompt, go to your .\hooks file and run the following commands:

```
mvn compile
```

This compiles your hook source code into java .class files.

```
mvn package
```

This zips up your compile code files with the needed directory structure into a file called .\target\hooks-1.0-SNAPSHOT.jar.

Step 4 - Move your compiled code to where OrientDB Server can find it

Finally, you need to copy your finished .jar file to the directory where your OrientDB server will look for them. This means the .\lib folder under your OrientDB Server root directory like this:

```
copy /Y .\target\hooks-1.0-SNAPSHOT.jar "\Program Files\orientdb-community-2.0.5\lib"
```

Step 5 - Enable your test hook in the OrientDB Server configuration file

Edit C:\Program Files\orientdb-community-2.0.5\config\orientdb-server-config.xml and add the following section near the end of the file:

```
    <hooks>
        <hook class="river.hooks.HookTest" position="REGULAR"/>
    </hooks>
    ...
</orient-server>
```

Step 6 - Restart your OrientDB Server

Once you restart your OrientDB Server, the hook you defined in orientdb-server-config.xml is now active. Launch an OrientDB console, connect it to your database, and run the following command:

```
insert into V set ID = 1;
```

If you review your server output/log you should see the message

```
Ran create hook
```

Now run command:

```
update V set ID = 2 where ID = 1;
```

Now your server output should say

```
Ran update hook
```

Step 7 - Enable your real hook in the OrientDB Server configuration file

Edit C:\Program Files\orientdb-community-2.0.5\config\orientdb-server-config.xml and change the hooks section as follows:

```
    <hooks>
        <hook class="river.hooks.setCreatedUpdatedDates" position="REGULAR"/>
    </hooks>
    ...
</orient-server>
```

Step 8 - Restart your OrientDB Server

Now create a new class that starts with the letter r or t:

```
create class tTest extends V;
```

Now insert a record:

```
insert into tTest set ID = 1;
select from tTest;

----+-----+------+----+-----------+-----------
#   |@RID |@CLASS|ID  |CreatedDate|UpdatedDate
----+-----+------+----+-----------+-----------
0   |#19:0|tTest |1   |1427597275 |1427597275
----+-----+------+----+-----------+-----------
```

Even though you did not specify values to set for CreatedDate and UpdatedDate, OrientDB has set these fields automatically for you.

Now update the record:

```
update tTest set ID = 2 where ID = 1;
select from tTest;

----+-----+------+----+-----------+-----------
#   |@RID |@CLASS|ID  |CreatedDate|UpdatedDate
----+-----+------+----+-----------+-----------
0   |#19:0|tTest |2   |1427597275 |1427597306
----+-----+------+----+-----------+-----------
```

You can see that OrientDB has changed the UpdatedDate but let the CreatedDate unchanged.

Conclusion

OrientDB Java Hooks can be an extremely valuable tool to help automate work you would otherwise have to do in application code. As many DBAs are not always Java experts, hopefully the information contained in this tutorial will give you a head start in feeling comfortable with the technology and empower you to successfully create database triggers as the need arises.

Good luck!