The Joy of Hex

Drunken Monkey Coding Style with a hint of Carlin humor

- RSS

Search

Navigate…

- Blog
- Archives

Automated Database Changes Management With Ant and Dbdeploy

May 25th, 2012

In the last couple of months we started using Ant tool heavily to do server deployments, along with Jenkins server (although not in the role it was intended, but hey, it serves the purpose).

Most of the things in setup work well and it became a pretty smooth operation, but one big hurdle that remained was the db changes management.

Lack of time prevented me from exploring the options, but as luck would have it, we were attending the awesome phpDay 2012 conference, and an exellent talk by Jeremy Coates titled An introduction to Phing the PHP build system where on one of the slides two tools were mentioned, dbdeploy and Liquidbase.

I did not hear about them up till then, and they were mentioned briefly as db change automation tools, and those words rang in my ears, I could not believe my luck, so few days after, I started looking into them.

What I wanted to have is a way to incrementaly update the database, or just dump all the alters in, a nice side effect would be to be able to generate one flat deploy file that could be either user for some remote deployment, or submitted to the DBA if needed.

Both tools look excellent, and they seem to do their job well, but in the end I selected the dbdeploy because it was closer to the way we structure our projects and files, and would be less intrusive to our process.

dbdeploy relies on deltas to update the db, each delta is one file that should ideally have one CREATE/ALTER/DROP command and it's undo counterpart, the files should be named in form XXX_something_or_other.sql, where XXX is numeric that identifies the order of execution of the script, with that in mind, it goes along nicely with the way we work (granted naming of directories is different, but it works, for now…). We would have to adapt a bit to break down the initial db draft into individual alter files, and add their undos, but that is really a non-issue.

The setup consists of

1. Ant
2. dbdeploy, which you can get from here
3. MySql Connector JDBC driver, which you can get from here, or some other JDBC driver (I am targeting MySql so YMMV)

The tasks/targets are defined in the build.xml file, more on that later, for the moment I'd like to focus on the build.properties file. This file is not supposed to go into the VCS apart in it's build.properties.dist form (which you should rename and mod for your local setup). This file will contain some paths and db credentials that vary from project to project and/or server to server. This file is a standard ant properties file which consits of key/value pairs

build.properties:

```
1  # Property files contain key/value pairs
2  #key=value
3
4  # Credentials for the database migrations
5  db.host=localhost
6  db.port=3306
7  db.user=root
8  db.pass=root
9  db.name=dbdeploy
10
11 # Path to mysql
12 progs.mysql=/usr/bin/mysql
```

This is pretty straight forward stuff, but point of interest is line 12 – path to mysql, you may want to drop this, I used it to execute the flat sql file with mysql, more like a proof that it works than actual use

With that the variable properties are covered and the build.xml file can be built. My personal preference is that it goes in the VCS, but some prefer to keep only the build.xml.dist in the VCS.

build.xml:

```
1   <?xml version="1.0" encoding="UTF-8"?>
2
3   <project name="dbDeploy_example" basedir="." default="default">
4       <description>This is an ant build.xml file for the example dbdeploy project.</description>
5
6       <!-- Define the timestamp format for the generated files -->
7       <tstamp>
8           <format property="current.time" pattern="yyyy-MM-dd-HH-mm-ss"/>
9       </tstamp>
```

```
10
11      <!-- Load our configuration -->
12      <property file="./build.properties"/>
13
14      <!-- Define the sources dir -->
15      <property name="src" value="."/>
16
17      <!-- Define the path to the dbdeploy dir -->
18      <property name="build.dbdeploy.dbdeploy_dir" value="${src}/../dbdeploy"/>
19
20      <!-- Define the path to the deltas/alters dir -->
21      <property name="build.dbdeploy.alters_dir" value="${src}/../db/alters"/>
22
23      <!-- Define the path to the deploy flat file dir -->
24      <property name="build.dbdeploy.deploy_dir" value="${src}/../db/deploy"/>
25
26      <!-- Define the path to the undo flat file dir -->
27      <property name="build.dbdeploy.undo_dir" value="${src}/../db/undo"/>
28
29      <!-- Last change number to apply, useful for preventing the unchecked delta to mess things up -->
30      <property name="build.dbdeploy.lastChangeToApply" value="20"/>
31
32      <!-- Define the db driver -->
33      <property name="db.driver" value="com.mysql.jdbc.Driver"/>
34
35      <!-- Define the url to the database -->
36      <property name="db.url" value="jdbc:mysql://${db.host}:${db.port}/${db.name}"/>
37
38      <!-- Define the target DBMS -->
39      <property name="db.dbms" value="mysql"/>
40
41      <!-- Define tha path to the changelog table sql file -->
42      <property name="build.dbdeploy.changelog_file" value="${build.dbdeploy.deploy_dir}/scripts/createSchemaVersionTable.mysql.sql"/>
43
44      <!-- these two filenames will contain the generated SQL to do the deploy and roll it back-->
45      <property name="build.dbdeploy.deployfile" value="deploy-${current.time}.sql"/>
46      <property name="build.dbdeploy.undofile" value="undo-${current.time}.sql"/>
47
48      <property name="use-verbose" value="false"/>
49
50      <!-- Define the classpath for the db driver -->
51      <path id="mysql.classpath">
52          <fileset dir="${build.dbdeploy.dbdeploy_dir}">
53              <include name="mysql*.jar"/>
54          </fileset>
55      </path>
56
57      <!-- Define the classpath for the dbdeploy -->
58      <path id="dbdeploy.classpath">
59          <!-- include the dbdeploy-ant jar -->
60          <fileset dir="${build.dbdeploy.dbdeploy_dir}">
61              <include name="dbdeploy-ant-*.jar"/>
62          </fileset>
63
64          <!-- The dbdeploy task also needs the database driver jar on the classpath -->
65          <path refid="mysql.classpath"/>
66      </path>
67
68      <!-- Declare the dbdeploy task -->
69      <taskdef name="dbdeploy" classname="com.dbdeploy.AntTarget" classpathref="dbdeploy.classpath"/>
70
71      <!-- Target to generate the changelog table in the database -->
72      <!-- This should be run only the first time db is created (and if it is ever recreated), so ugly, but works -->
73      <target name="create-changelog-table">
74          <sql driver="${db.driver}" url="${db.url}"
75              userid="${db.user}" password="${db.pass}" classpathref="mysql.classpath">
76              <fileset file="${build.dbdeploy.changelog_file}"/>
77          </sql>
78      </target>
79
80      <!-- Target to generate two scripts, one for deploy, the other for rollback to the version specified in the build properties file,
81          useful when you want to submit to DBA for review -->
82      <target name="dbdeploy-generate-sql">
83
84          <!-- Generate the directories for the deploy and undo files -->
85          <mkdir dir="${build.dbdeploy.deploy_dir}" />
86          <mkdir dir="${build.dbdeploy.undo_dir}" />
87
88          <!-- generate the deployment scripts -->
89          <dbdeploy
90                 driver="${db.driver}"
91                 url="${db.url}"
92                 userid="${db.user}"
93                 password="${db.pass}"
94                 dir="${build.dbdeploy.alters_dir}"
95                 outputfile="${build.dbdeploy.deploy_dir}/${build.dbdeploy.deployfile}"
96                 undooutputfile="${build.dbdeploy.undo_dir}/${build.dbdeploy.undofile}"
97                 dbms="${db.dbms}"
98                 lastChangeToApply="${build.dbdeploy.lastChangeToApply}"
99                 />
100     </target>
101
102     <!-- Target to generate two scripts, one for deploy, the other for rollback, useful when you want to submit to DBA for review -->
103     <target name="dbdeploy-generate-sql-all">
104
105         <!-- Generate the directories for the deploy and undo files -->
106         <mkdir dir="${build.dbdeploy.deploy_dir}" />
107         <mkdir dir="${build.dbdeploy.undo_dir}" />
108
109         <!-- generate the deployment scripts -->
```

```
110         <dbdeploy
111                 driver="${db.driver}"
112                 url="${db.url}"
113                 userid="${db.user}"
114                 password="${db.pass}"
115                 dir="${build.dbdeploy.alters_dir}"
116                 outputfile="${build.dbdeploy.deploy_dir}/${build.dbdeploy.deployfile}"
117                 undooutputfile="${build.dbdeploy.undo_dir}/${build.dbdeploy.undofile}"
118                 dbms="${db.dbms}"
119                 />
120     </target>
121
122     <!-- Target to actually do the migration to the version specified in the build properties file -->
123     <target name="dbdeploy-migrate">
124
125         <!-- generate the deployment scripts -->
126         <dbdeploy
127                 driver="${db.driver}"
128                 url="${db.url}"
129                 userid="${db.user}"
130                 password="${db.pass}"
131                 dir="${build.dbdeploy.alters_dir}"
132                 dbms="${db.dbms}"
133                 lastChangeToApply="${build.dbdeploy.lastChangeToApply}"
134                 />
135
136     </target>
137
138
139     <!-- Target to actually do the migration to the latest version -->
140     <target name="dbdeploy-migrate-all">
141
142         <!-- generate the deployment scripts -->
143         <dbdeploy
144                 driver="${db.driver}"
145                 url="${db.url}"
146                 userid="${db.user}"
147                 password="${db.pass}"
148                 dir="${build.dbdeploy.alters_dir}"
149                 dbms="${db.dbms}"
150                 />
151
152     </target>
153
154     <!-- Target to import the geenrated deploy sql file into db via mysql -->
155     <target name="dbdeploy-execute-sql" depends="dbdeploy-generate-sql">
156         <!-- execute the SQL - Use mysql command line to avoid trouble with large files or many statements and PDO -->
157         <exec
158                 command="${progs.mysql} -h${db.host} -u${db.user} -p${db.pass} ${db.name} &lt; ${build.dbdeploy.deployfile}"
159                 dir="${build.dir}"
160                 checkreturn="true"/>
161     </target>
162 </project>
```

Points of interest:

- Line 18 – the path to the dbdeploy and MySql JDBC connector jars, because this will be standardized structure it makes more sense to set it here, than in build.properties
- Lines 20 – 27 – the paths for the deltas, deploy and undo scripts, they are defined relatively to the build.xml file or basedir definition
- Line 30 – build.dbdeploy.lastChangeToApply key – this is basically a limiter, if you have new deltas you are working on, and do not want them in the db, you can prevent their execution by setting the number to lower than the order number they have.
- Line 33 – the package for your db driver, to decrease the typos possible when generating a new target
- Line 36 – the url of the database
- Line 42 – the path to the changelog table script, it comes with dbdeploy package, and is **IMPORTANT**, dbdeploy uses it to track changes
- Lines 45 – 46 – definiton for the deploy and undo files, I chose to use timestamp value after the name
- Lines 51 – 55 – define classpath for the MySql Connector and the location of the jars for it
- Lines 58 – 66 – define classpath for the dbdeploy and the location of the jars for it
- Line 69 – declared the dbdeploy task to Ant
- Lines 73 – 78 – This is the target that actually creates the changelog table in your db, should be used once or when recreating the db. Without this table the dbdeploy will fail
- Lines 155 – 161 – This is more a proof of concept than it actually works kinda thing, it can be used do import the generated deploy file into the db via shell command

You might notice that the actual targets to execute are doubled, their syntax is almost the same, they differ in one important aspect. The *-all targets will execute **ALL** the deltas present indiscriminantly.

The targets without the -all will rely on the **build.dbdeploy.lastChangeToApply** from build.properties file to execute the deltas only upto, and including the number given.

This is important to us, because this way we can make sure that only the deltas that are verified as good be executed in the db, or to shoot ourselves in the foot and lose a couple of hours trying to figure out why delta with order number x is not executed while not realising that the build.dbdeploy.lastChangeToApply value y is less than x, **SO BEWARE OF THE LIMITER**

The targets dbdeploy-generate-sql/dbdeploy-generate-sql-all will generate two files (flat sql files), one for deploy on location set by build.dbdeploy.deploy_dir, and one for undo on location set by build.dbdeploy.undo_dir. These files can either be executed on remote server or submitted to your DBA for review, or some other purpose you may need.

The targets dbdeploy-migrate/dbdeploy-migrate-all will update your db schema with the deltas that were not yet executed.

You will notice that I do not mention the undo/rollback process, from what I understand the dbdeploy currently does not support rolling back to a specific script, and it's undo will happen only if the update is broken. Dbdeploy will assume you will fix the problem by making another delta and executing it on the db.

For our scenarios that is currently a non-issue because we test things locally and on dev server before we deploy to live. Chances of immediate rollback are small, yet still present.

The files are available on Github

Posted by Vranac Srdjan May 25th, 2012 posted in automation, linux, mysql, ubuntu tagged with ant, automation, dbdeploy, mysql

« Getting MAMP to play nice with vhosts Pear Packages installation under Vagrant with Puppet »

About me

I have been threading the murky depths of software development for a far too long a time, read as I spurt obscene code.

This blog is about the technology I use everyday and problems I encounter. There will be an occasional rant and rave too. All of this will be wrapped in some tongue-in-cheek humor, or at least I will try.

You can find me on Twitter as @vranac

Recent Posts

- On Speakers and Speaking
- Those That Do Not Heed the Star Are Bound to Execute It Repeatedly...
- Sublime Text Psql Build System
- How Was Your Weekend?
- Adding Mockery to Symfony 2.3

Tags

AS3 Actionscript CodeCamp DI DIC DVCS Event File Upload Flash Gem Git Gitosis IDE IOC IntelliJ Idea PHP Plugin ROR Rails Rant Rave Ruby SDK Silverlight Sonata Symfony 2 Symfony2 Windows ant apache automation bash ci command prompt conference cygwin dbdeploy doctrine events jenkins knpmenu linux mac mamp mockery mongo mysql osx pear php phpunit puppet sed sesmic shell ssh symfony talk terminal testing ubuntu vagrant vhost windows 7

GitHub Repos

- ansible-role-mailcatcher

  Ansible role to install mailcatcher

- vagrant-ansible-symfony

  Vagrant box, with Ansible provisioning to setup new Symfony project.

- revealjs-template
- octopress-blog
- ansible-helper

  Simple bash script to help with Ansible directory structure generation

- Sublime-psql-build-system

  Sublime Text psql build system

- link-checker

  quick and dirty link checker for web page

@vranac on GitHub