



Automated Portal

scott/tiger

Deployment with Apache

Ant

**sten.vesterli@
scott-tiger.dk**

The Problem

A portal application consists of:

- Database objects**
- Static files**
- Java applications**

scott/tiger

The Classic Solution

- A thick installation guide with lots of manual steps
- Developers install on production system.

Apache Ant

- Apache Ant is a general build tool
- Controlled by an XML build file
- Defines “tasks” and dependencies among them
- Possible to write your own tasks

The Better Solution

- **Developer packages component in a specific directory structure**
- **Developer writes a deployment file which is an Ant script**
- **A number of pre-built special tasks are available to call**
- **The Deployer can add implementation-specific details**

Directory Structure

- **STPS_DISUSS_FORUM**
 - 1.0.3
 - readme_stps_discuss_forum_1.0.3.txt
 - build.xml
 - sql
 - deploy
 - html
 - images
 - css

scott/tiger

Build file for SQL

```
<antcall target="sqlrunner">
  <param name="run_as"
value="<run_as_schema>" />
  <param name="db_password"
value="\${<db_password_property>}" />
  <param name="script_name"
value="<script_name>" />
</antcall>
```

The sqlrunner task

```
<target name="sqlrunner"
  depends="init">
  <fail message="..." unless="run_as"/>
  ...
  <concat destfile="...">
  ...
  <filterchain>
  <tokenfilter>
  <replacestring from="&portal_schema"
    to="\${portal_schema}"/>
```

The sqlrunner task, cont'd

```
<sql
  driver="oracle.jdbc.driver.OracleDriver"
  classpath="${env.CLASSPATH}:${env.ORACLE_HOME}/jdbc/lib/classes12.jar"
  url="${db_connect}" userid="${run_as}"
  password="${db_password}"
  onerror="continue" showheaders="no"
  print="yes" delimiter=";"
  delimitertype="normal"
  keepformat="yes"
  src="${sqlrunner.tempfile}"
```

Notes on running SQL

- Use `<tokenfilter>` to replace `&&portal_schema` etc.
- Use `<deletecharacters>` to clean up
- Wrap these in a `<filterchain>`
- Use `delimiter=;` and `delimitertype=normal` to execute each command as soon as a semicolon is met
- Consider if an SQL error is fatal `onerror=continue`

scott/tiger

Build file for PL/SQL

```
<antcall target="sqlrunner">  
  <param name="run_as"  
    value="<run_as_schema>" />  
  <param name="db_password"  
    value="$ {<db_password_property>}" />  
  <param name="script_name"  
    value="<script_name>" />  
</antcall>
```

scott/tiger

The plsqlrunner task

```
<target name="plsqlrunner"  
  depends="init">
```

```
<fail message="DB schema not provided"  
  unless="run_as"/>
```

...

```
<sql  
  driver="oracle.jdbc.driver.OracleDriver"  
  classpath="{env.CLASSPATH}:{env.ORACLE_HOME}/jdbc/lib/classes12.jar"  
  url="{db_connect}" userid="{run_as}"  
  password="{db_password}"
```

Notes on PLSQL

- Use `delimiter=/` og `delimitertype=row` to ensure that command is not processed until a `/` on a line by itself
- Important: `keepformat=true`
- Consider if a PL/SQL error is fatal `onerror=continue`

General about SQL and PL/SQL

- Use fail ... unless to ensure that developer has provided all parameters
- When running the SQL, set escapeProcessing=false to avoid the JDBC driver choking when meeting curly brackets {}

scott/tiger

Ant <sql> is not SQL*Plus!

- You can't use SQL*Plus commands like SPOOL
- You can't change connection with CONNECT (use another sqlrunner/plsqlrunner block with run_as)
- You can't call other scripts with @script.sql

Password handling

- Don't allow username=password in test and production systems!
- Solution: System-generated password at deploy time

The create_schema task

```
<target name="create_schema"
  depends="init">
  <antcallback
    target="get_schema_password"
    return="password">
  ...
  </antcallback>
  <if>
  <equals arg1="{password}" arg2="" />
  <then>
```

The create_schema task, ^{scott/tiger} cont'd

- Use an <sql> task (running as e.g. the PORTAL user) to do
 - drop user cascade
 - create user ... identified by ...
 - grant to the user

The `get_schema_password` task scott/tiger

Use a shell script that runs `ldapsearch`, for instance

```
ldapsearch -D cn=orcladmin -w $3 -p $2 -h  
$1 -b "orclReferenceName=$4,cn=IAS  
Infrastructure  
Databases,cn=IAS,cn=Products,cn=OracleContext" -s sub orclresourcename=$5  
orclpasswordattribute | grep  
orclpasswordattribute | cut -d = -f 2
```

The generate_password task

A little piece of Java, for instance

```
public class GeneratePassword extends
    Task {
    private Random randomGen = new
        Random();
    public void execute() {
        int digitOnePos =
            randomGen.nextInt(7) + 2;
        int digitTwoPos =
            randomGen.nextInt(7) + 2;
```

The store_password task

1. Create a sample .ldif file for a user entry.
Replace username and password with placeholders

2. Use a shell script to replace placeholders with actual values:

```
cat user_template.ldif | sed  
"s/##SCHEMA##/$1/" | sed  
"s/##PASSWORD##/$2/" >  
/tmp/temp.ldif
```

3. Use another shell script to add user with
`ldanadd -D cn=orcladmin -w $3 -n $2 -h`

•Build file for static files

```
<target name="filecopy" depends="<init_discuss_forum>">  
  <echo message="...Installing DB objects  
for component  
${component_short_name}"/>  
  <copy_files>  
    <param name="type" value="images"/>  
  </copy_files>  
</target>
```

The copy_files task

```
<target name="copy_files"
depends="set_environment">
  <exec executable="ssh">
    <arg value="{portal_server}"/>
    <arg value="mkdir
${vendor_base}/${vendor_short_name}"
/>
  </exec>
  <exec executable="ssh">
    <arg value="{portal_server}"/>
```

The add_httpd_line task

- Purpose is to add an extra line to httpd.conf
- One solution:
 - Get the httpd.conf file
 - Use Unix sed to extract customization section
 - Add the extra line to the customization section
 - Rebuild the httpd.conf file with the new customization section

Notes on copy_files

- Our solution added an Alias line to the httpd.conf file
- For simplicity, the virtual directory is hardwired to *<vendor_short_name>/<component_short_name>/<file type>*, e.g.
stps/discuss_forum/images
- The deployer decides on where to place the file on the server by setting $\${vendor_base}$

•Build file for Java applications

```
<target name="deploy" depends="init_discuss_forum,set_environment">
```

```
  <antcall
```

```
    target="nnit_portal_deployment_tasks.deploy"
```

```
  />
```

```
</target>
```

- Similar for undeploy, redeploy

•The deploy target

```
<target name="deploy" depends="init">
```

```
  <exec executable="scp">
```

```
    <arg
```

```
      value="\${component_home_dir}deploy/\${ear_
file_name}" />
```

```
    <arg value="\${j2ee_server}:/tmp" />
```

```
  </exec>
```

```
  <exec executable="ssh">
```

Notes on deploying EAR files

- Simply copy it to the destination server
- Use `dcmctl deployApplication` (available in 10.1.2)

•Other features

- Create_oc4j (using dcmctl)
- Add_datasouce (by XSL transformation of data-sources.xml)
- Import transport sets
- Missing: Allow extended characters (æøå) in SQL scripts...

Invoking

- The deployer simply executes
- **ant -Denvironment=<environment> -Doracleadmin_pwd=<password> install**
- Alternatively **db_install, deploy, redeploy, undeploy**

Lessons Learned

- SQL files normally run with SQL*Plus needs to be separated and cleaned up
- Difference between sqlrunner and plsqlrunner not obvious to developers (?)
- run_as feature allows one component to log on as e.g. UTIL user and grant privileges (but hard to keep track of in practice...)

Summary

- **Developer runs deployment on Test and fixes errors until deployment runs correctly**
- **Deployer can simply add a bit of environment-specific info and run deployment**
- **Comprehensive logging with Log4J facilitates debugging**