Dzo 1.2.0 quick introduction

1 Introduction

The goal with the utility dzo is to treat a application database objects the same way the applications source code is treated, with respect to development, revision control, and deployment. Dzo uses a text file (sql source code) that contains native create statements for all database objects and compares them against the actual database-schema. As a result, dzo creates the SQL statements needed to update the database schema (or you can let dzo execute the SQL statements directly).
When a application is deployed the sql source code should be a part of the deployment package, so the deployment package is self-contained. A stage in the deployment process is to execute dzo to change/update the database to the structure the new revision of the application needs.
When sql source code is a part of the deployment package there is no longer a need to know the state of the database, it doesn't matter if it is a upgrade or a downgrade, if the database was updated yesterday, two weeks or two minutes ago. Dzo will change the database to the state which is represented by the sql source code in the application.
If the application lives in a Tomcat or JBoss container, dzo has a servlet that controls the deployment process, undeploys the old application, inspect and execute the needed database changes, and finally deploys the new application.

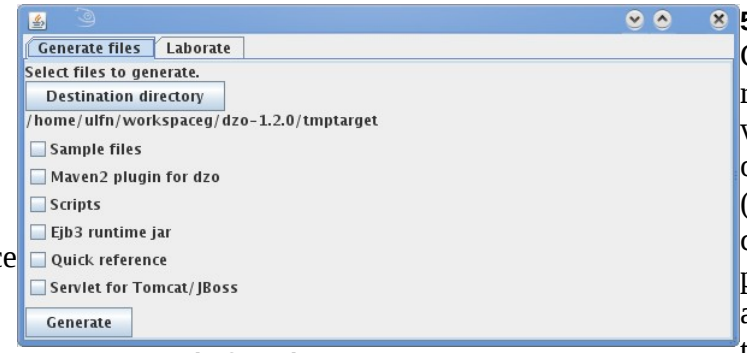
Dzo currently works with database engines MySQL, Oracle and Sql Server.

2 Getting started

This document and the servlet and some other files can be generated from dzo:s jar-file.
Run the command:

```
java -jar dzo-1.2.0.jar
```

and a simple swing-application is started where the files can be generated:



3 Dzo:s main functions:

1. Compare a databaseschema to sql source code and show or execute the needed changes.
2. Create sql source code from a database schema.
3. Generate annotated ejb3-entity classes or use annotated ejb3 entity beans as sql source code.

4 Compare

The most common use is to compare a database schema to sql source code to find out the needed database changes, but it is possible to compare two different sql source code (two different versions for examples), or to compare two different database schemas.

Compare can be done from:

1. A ant task (see 9 and 20)
2. A maven goal (see 10 and 21)
3. command line(see 11 and 19)
4. dzo:s servlet in Tomcat or JBoss container (see 22 ,23, 24)

Ant is mostly used by developers in the development process to keep the developers local database synchronized to the sql source code. Command line and servlet is used when deploying to controlled environments.
The servlet needs a central `dzo.xml` (see 22) which have settings for different applications and the applications database schemas.
The deployable `war`/`ear` should be suffixed by `.dzo` and needs a `DZO-INF` directory with a local `dzo.xml` file and one sql source code-file for every database schema in the application.

5 Create sourcecode

Create sourcecode is normally done when starting to revisioning database objects. Very often the only place where the complete knowledge of which database objects the application consists of is the database itself (the production database). A good starting point to take control is to generate sql source code from the database, put the script under revision control together with the application so it is tagged/labelled and deployed together with the application. When database changes is needed from now on, change sql source code and let dzo take care of database changes.

6 Generate ejb3 entity classes

Create and maintain annoted entity beans is a boring and error-proning task for developers. It is easy to forget a column or to have the wrong java datatype. In many ways the entity beans is only a mirror of the sql source code. Dzo can generate annoted ejb3 entity beans so it always will reflect the actual sql source code. To generate entity beans use the special annotation comment `/*@Annotate: ejb3...*/` before tables and columns. Is supports the 3 different inheritance strategies. It is also possible to create own utility methods (which of course not will be overwritten when the entity beans is regenerated).

7 Note

MySql:
Create user with:

```
grant all on *.* to dzodba@localhost
  identified by 'password' with grant option
```

Always treat sql code as case-sensitive, when parsing sql source code dzo dont know if the database is case-sensitive or case-insensitive.

Oracle:
username2 (see 14) should have the following grants:

```
resource, connect, dba
```

Sql Server:

Dzo 1.2.0 quick reference card

8 Sql directives

--DB:databastype

If used it must be the first line in file. Databasetype is one of `mysql,oracle or sqlserver`.

#convert tablename convname cond=(condition)

`sql-statement`
`convert` is only available in dzo professional.
Condition can use logical operators `NOT, AND, OR`
`Functions:`
`- exists(:new-or-old.tablename)`, `exists(:new-or-old.tablename.columnname)`
`- created(tablename)`, `created(tablename.columnname)`
`- dropped(tablename)`, `dropped(tablename.columnname)`
`:new-or-old.tablename.columnname = ”sql-datatype”`

#ejb3include

`#ejb3include 'package'`
Expands classes in or in a subpackage which is annotated with @Entity.

#include

`#include 'filename'`
Expands filename.

#protect databaseobject

Dont do any changes on databaseobject.

#referencedata

Keep the content in a table synchronized with `insert`-statements in sql-file.

```
#referencedata
    tablename key=(columnname)
    discriminate=(sql-condition)
    history=(valuecol,historytable,stampcol,changecol)
insert into tab(id,col1,col2)values(id1,val1,val2);
```

`history` is only available in dzo professional.

#revision

Persists a key-value in table zdzo.

```
#revision 'key' 'value'
#revision 'value'
```

#sql

```
#sql begin onError=(continue/break,fail/abort)
sql-statements1
#sql success onError=(continue/break,fail/abort)
sql-statements2
#sql fail onError=(continue/break,fail/abort)
sql-statements3
#sql finally onError=(continue/break,fail/abort)
sql-statements4
#sql aborted
sql-statements5
#sql end
```

/@Annotate: annotationtype: annotationdata*/*

9 Ant

<dzo>

subelement `<schema>`
attributes:
`showSql` boolean
`maxThreads` integer
`execute` boolean
`destFile` filename for result.
`type` see Resulttype.
For type `java` see Ejb3.Attribute.

<schema>

attributes:
`fromDatabase`, `toDatabase` see Sql directives.--DB:databasetype
`fromDriver`, `toDriver` see database.
`fromUrl`, `toUrl` see database.
`fromUser`, `toUser` see database.
`fromFile`, `toFile` filename with sql-statements.
`fromIncludePath`, `toIncludePath` directorys to search when `#include`.
Use either database-attributes or file-attribute.

10 Maven

Have the same names as maven.

11 Command line

```
dzo opts fromSchema toSchema
dzo opts toSchema
opts:
-f,--result output file.
-g,--type See Resulttype.
-h,--stopstage Use ”chgdb” to execute sql.
-o,--object Process only object. Can be repeated.
-t,--maxthreads max number of threads.
-v,--revision Show revision and exits.
fromSchema, toSchema:
-I includePath -J includeEjb3Path -G key=value src;[dbtype;]filename
db;dbtype;[driver;]url;dbuser
dbtype See Sql directives.--DB:databasetype

-I Includepath when #include
-J Includepath when #ejb3include, jar or directorys
-G For type java see Ejb3.Attribute and remove java-prefix.
driver See database.
url See database.
dbuser See database.
```

12 Resulttype

When comparing 2 schemas:
- `alter`, creates sql-statements to alter database.
- `diff` create a short diff

Only one schema:
- `xml`, creates a xml-document
- `src`, creates source-code
- `java`, creates ejb3 annoted java classes

13 Propertys

`dzo.fullStacktrace`, for debugging
`dzo.include.path`, directorys to search when #include.
`dzo.ejb3include.path`, directorys to search when #ejb3include.
`dzo.log4j`, when null dont use log4j
`dzo.SAXParser`, when null use parser in JRE.

14 Database

Mysql

Default driver: `com.mysql.jdbc.Driver`
Url: `jdbc:mysql://`**`host:port/db`**
User: **`username/password`**

Oracle

Default driver: `oracle.jdbc.OracleDriver`
Url: `jdbc:oracle:thin:@`**`host:port:db`**
User: **`username/password`**
or **`username/password(username2/password2)`**
`username` is the schema to compare.
`username2` is a user with higher permissions (needed for example if the schema should be created) or system grants is needed.

Sql Server

Default driver: `com.microsoft.sqlserver.jdbc.SQLServerDriver`
Url: `jdbc:sqlserver://`**`host:port`**`;databaseName=`**`dbName`**
User: **`username/password`** or **`username/password(schema)`**

15 dzo.xml

All elements with name-attribute and `<dzo>`-element can have attribute `acceptChange`, to deny or allow a local `dzo.xml` to override the settings from a central `dzo.xml`.

<dzo>

Root-element.
subelement `<application>`, `<login>`

<login>

subelement `<user>`

<application>

subelement `<inform>`, `<database>`
attributes: name, deploy; ”manual”, ”auto”

<inform>

subelement `<email>`, `<file>`

<database>

subelement `<driver>`, `<url>`, `<user>`
attributes: name, type

<url>

<driver>

<user>

attributes:name (only in login), password (only in login)

<email>

attributes: name, level; ”full” or ”small”, format; ”text” or ”html”

<file>

attributes: name, file; filename, level; ”full”, ”small”

16 Ejb3

Attribute:

- `javaDir`, sourcecode directory.
- `javaTempDir`, generated source code (no revision control).
- `javaPackage`, packagename for util,proxy, interface and package-classes.
- `javaAnnotedPackage`, packagename for annoted classes.
- `javaRelationCode`, creates code to update the non-owning side of relations.

Annotations:

(column) ejb3:tablename [rolename] @OneToOne
(column) ejb3:- [rolename2] @OneToOne
(column) ejb3:tablename [rolename] @ManyToOne
(column) ejb3:- collectiontype [rolename2] @OneToMany
(column) ejb3:- Map[:keycolumn] [rolename2] @OneToMany
(column) ejb3:@Id
(column) ejb3Datatype:java-datatype
(table) ejb3:@Inheritance
(table) ejb3Extends baseclass
(table) ejb3Implements interface1[,interface2 ...]
(table) ejb3Inheritance: tablename
When JOINED or TABLE_PER_CLASS.
(table) ejb3Inheritance: value tablename
When SINGLE_TABLE.
(table) ejb3Inheritance: value tablename basetablename
When SINGLE_TABLE.

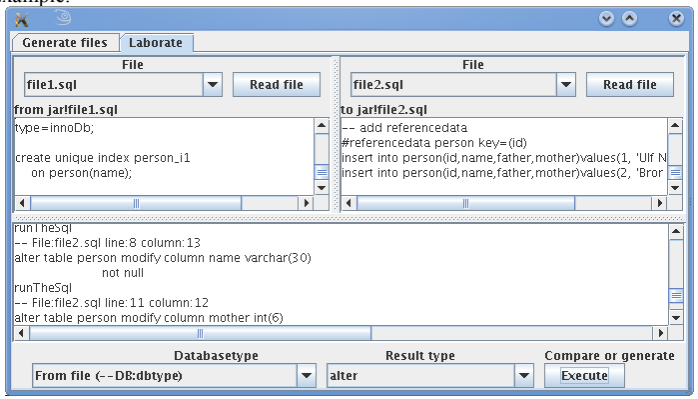
(column) ejb3DiscriminatorColumn: [datatype]
When SINGLE_TABLE.or JOINED.
(column) ejb3Discriminate: tablename
When SINGLE_TABLE.
(table) ejb3Constant: java-datatype name = value
(insert) ejb3Constant: constantname
(table) ejb3Utility: import-or-methoddeclarations

17 Install & laborate

```
java -jar dzo-1.2.0.jar -vh -d destDir -t fileType -f size
```

`-d` destination directory for created files.
`-f` fontsize
`-h` prints help and exits.
`-t` creates selected filetype, can be ”`servlet`”, ”`sample`”, ”`script`”.
`-v` prints revision and exits.

If no `filetype`:s is given a simple swing-application starts with 2 tabs, one for creating files, and one for laborating with dzo.
Example:



18 Sql source code

example.sql

```
1. --DB:mysql
2. #revision '$Id:$'
3.
4. #ejb3include 'se.abgd.ejb3'
5.
6. create table person
7. (
8.   id        int primary key
9.   ,name     varchar(20)
10.  /*@Annotate:ejb3: role @ManyToOne*/
11.  /*@Annotate:ejb3: - Collection @OneToMany*/
12.  ,role_id    int references role(id)
13.) type=innodb;
14.
15.--Executes when column fname is renamed to name.
16.#convert person conv1 cond=(created(person.name) AND
   dropped(person.fname))
17.#sql begin onError(abort)
18.update person set name = fname
19.runTheSql
20.#sql aborted
21.alter table person drop column name
22.runTheSql
23.#sql end
24.
25.create table role
26.(
27.  id        int primary key
28.  ,rolename varchar(20)
29.) type=innodb;
30.
31.#referencedata role key=(id)
32./*@Annotate:ejb3Constant: MANAGER*/
33.insert into role(id,rolename)values(1,'Manager');
34.insert into role(id,rolename)values(2,'Employee');
```

19 Command line

Compare and update database:

```
1. dzo -h chgdb
   ”db;mysql;jdbc:mysql://localhost/voff;dzodba/dzodba”
   -J target ”src;example.sql”
```

To compare and only show differences remove flag ”`-h chgdb`” above.

Generate source code from database:

```
1. dzo ”db;mysql;jdbc:mysql://localhost/voff;dzodba/dzodba”
```

20 Ant build file:

Ant targets to show needed database changes (`showDbChanges`), update database (`execDbChanges`) and generate ejb3 entity beans (`genEjb3`):

build.xml

```
1. <project name=”example”>
2.   ...
3.   <taskdef classname="se.grunddata.dzo.ant.Dzo" name="dzo">
4.     <classpath>
5.       <path location="${dzojar}" />
6.       <path location="${jdbc.mysql}" />
7.     </classpath>
8.   </taskdef>
9.   ...
10.  <target name="showDbChanges">
11.    <dzo showSql="true">
12.      <schema fromDatabase="mysql"
13.        fromUrl="jdbc:mysql://localhost/voff"
14.        fromUser="dzodba/dzodba"
15.        toEjb3Include=”target”
16.        toFile="example.sql" />
17.    </dzo>
18.  </target>
19.
20.  <target name="execDbChanges">
21.    <dzo showSql="true" execute="true">
22.      <schema fromDatabase="mysql"
23.        fromUrl="jdbc:mysql://localhost/voff"
24.        fromUser="dzodba/dzodba"
25.        toEjb3Include=”target”
26.        toFile="example.sql" />
27.    </dzo>
28.  </target>
29.
30.  <target name="genEjb3">
31.    <delete dir="tempSrc"/>
32.    <delete dir="tempSrcPack"/>
33.    <dzo type="java"
34.      javaDir="src"
35.      javaTempDir="tempSrc"
36.      javaPackage="se.abgd.dao"
37.      javaAnnotedPackage="se.abgd.dao.mysql" >
38.      <schema toFile="example.sql" />
39.    </dzo>
40.  </target>
41.  ...
42.</project>
```

21 Maven build file:

Available goals:

Goal	Description
`compareShow`	show needed database changes
`compareExec`	perform needed database changes
`xml`	generate xml-file
`source`	generate sql source-file
`ejb3`	generate sql source-file

Example build file:

pom.xml

```
1. <project>
2.   <modelVersion>4.0.0</modelVersion>
3.   <groupId>se.grunddata.dzo.test</groupId>
4.   <artifactId>test</artifactId>
5.   <version>1.0-SNAPSHOT</version>
6.   <build>
7.     <plugins>
8.       <plugin>
9.         <groupId>se.grunddata.dzo</groupId>
10.        <artifactId>dzo-maven2</artifactId>
11.        <version>1.2.0</version>
12.        <configuration>
13.          <showSql>true</showSql>
14.          <header>full</header>
15.          <schemas>
16.            <schema>
17.              <database>mysql</database>
18.              <fromUrl>jdbc:mysql://localhost/voff</fromUrl>
19.              <fromUser>dzodba/secret</fromUser>
20.              <ejb3Include>target</ejb3Include>
21.              <file>example.sql</file>
22.            </schema>
23.          </schemas>
24.          <!-- javaXX is only needed for mojo ejb3 -->
25.          <javaDir>tmp/src</javaDir>
26.          <javaTempDir>tempSrc</javaTempDir>
27.          <javaPackage>se.abgd.dao</javaPackage>
28.          <javaAnnotedPackage>se.abgd.dao.mysql</javaAnnoted
   Package>
29.        </configuration>
30.        <dependencies>
31.          <dependency>
32.            <!-- needed jdbc-driver(s) -->
33.          </dependency>
34.        </dependencies>
35.      </plugin>
36.    </plugins>
37.  </build>
38.</project>
```

22 Central dzo.xml

Update database by dzo-servlet.
The central `dzo.xml` is in directory `$CATALINA_HOME/conf` or `$JBOSS_HOME/conf`.

dzo.xml

```
1. <dzo acceptChange="false">
2.   <login>
3.     <user name="ulfn" password="secret"/>
4.   </login>
5.   <application name="dzoapp">
6.     <inform>
7.       <email name="emailer"
   level="full">your.email@address.com</email>
8.       <email name="deployer" acceptChange="true"/>
9.     </inform>
10.    <database name="example" type="mysql">
```

```
11.      <url>jdbc:mysql://localhost/dzoappdb</url>
12.      <user>dzodba/dzodba</user>
13.    </database>
14.  </application>
15.
16.  <application name="anotherApp">
17.    <inform>
18.      <email name="emailer"
   level="full">another.email@address.com</email>
19.    </inform>
20.    <database name="DB" type="mysql">
21.      <url>jdbc:mysql://localhost/app2</url>
22.      <user>dzodba/dzodba</user>
23.    </database>
24.  </application>
25.</dzo>
```

23 Structure war/ear-file

Suffix the deploy file with `.dzo`.

Structure in war-file (for a tomcat-application) or ear-file (for Jboss):

```
META-INF/
DZO-INF/
        example.sql
        dzo.xml (local dzo.xml)
```

24 Local dzo.xml

The local `dzo.xml` under DZO-INF directory in `war` or `ear` file.

dzo.xml

```
1. <dzo>
2.   <application name="dzoapp">
3.     <inform>
4.       <email name="deployer"
   level="full">deploy.email@address.com</email>
5.     </inform>
6.   </application>
7. </dzo>
```

25 Code example

How to use the generated code.

Example.java

```
1. package se.abgd.prog;
2. import se.abgd.dao.Role;
3. ...
4. {
5.   try {
6.     EntityManager em = ...;
7.     Class rolePack = Role.Factory.findAnnotedClass(
8.       ”se.abgd.dao.mysql”);
9.     // Create and save role
10.    Role role = Role.Factory.newInstance(rolePack);
11.    role.set... // set attributes
12.    em.persist(role);
13.    // Read manager role
14.    Role manager = em.getReference(
15.      rolePack, Role.Factory.MANAGER);
```