Dzo Users Guide

Authors

Ulf Näslund

Dzo Users Guide: Authors

by Ulf Näslund

Published April 2010

Abstract

This book is a guide to using and configure dzo.

Table of Contents

List of Figures

List of Tables

Foreword

This documentation is my third attempt to make a useful documentation for dzo. The first try was a OpenOffice document in Swedish which of course have a limited audience in this global world. My second effort is a Quick-reference, also in OpenOffice. This quick-reference will probably continue to live. And this third (third time is lucky or "tredje gången gillt" as we say in Sweden) is done with DocBook using JBoss maven JDocBook plugin.

I'm sure that there will be flaws, things that I have missed or is unclear in this document, I would be very glad to know about this.

I hope that you will find some use for this utility.

Sollentuna, Sweden 4-may-2010, Ulf Näslund (ulf.naslund@grunddata.se)

Chapter 1. Introduction

The two main problems to handle databases in the deployment process is:

1. Changes in database schemas is made up of deltas (differences from the actual state). You write:

   ```
   alter table Person add shoesize decimal(3)
   ```

   to add the column `shoesize` in table `Person`.

   But what if column `shoesize` already exists in the table but have the datatype `decimal(2)`, or if the table `Person` does not exists at all, then this statement fails. There is a lots of conditions that must be fulfilled to be able to successfully execute these deltas. When changes is made up of deltas, it's not enough to know what you want, you must also know what you have.

2. Someone, a physical person (often a dba) must have the knowledge of the database objects a certain revision of a application requires. There is different solutions on how this knowledge is passed to this person, sometimes a paper of the needed changes or a script with deltas to be executed in the environment. The dependency to a physical person makes it impossible to fully automate this process.

The goal with the utility dzo is the ability to treat database objects the same way application source code is treated, with respect to development, version control, baseline control, and deployment. Dzo uses a text file (SQL source code) that contains SQL create statements for all database objects (`create table ...`, `grant ...`, never any `alter table ...`, or `revoke ...`), and compares them against the actual database schema. As a result, dzo creates the needed SQL statements to update the database schema (or you can let dzo execute SQL statements directly).

A dzo-enabled application includes the SQL source code (in some way) in the applications deployment package, so that the deployment package is self-contained. During the deployment of a dzo-enabled application, dzo compares the SQL source code (in the deployment package) with the actual database objects in the schema and update the database schema to reflect the SQL source code. When the SQL source code is part of the deployment package it is no longer needed to know the state of the database, it does not matter if you do a system upgrade or a downgrade, if the database was updated yesterday, two weeks or two minutes ago. Dzo will change the database to the state represented by the SQL source code in the application. For applications deployed in a web container, dzo has a servlet that controls the deployment process, inspects and executes the needed database changes, and finally deploys the new application.

All of us agree about the need to have control of a application source code. To have it in a repository, be able to build a specific revision of the application in a controlled and reproducible way and to deploy this revision in a controlled way to a environment. Dzo makes it possible to take control of database objects and to handle these in exactly the same way as applications source code. When a application is labelled the SQL source code is labelled. When the deployable artefact is build the SQL source code is packaged together with the application. When the application is deployed in a environment dzo compares the database schema(s) that belong to the application against to the SQL source code in the deployable artefact and makes the changes , this way the application always have the database it expects.

As a side-effect it will be much easier to inspect how the database evolves between different versions of the application (exactly the same way as the applications source code). Compare this to solutions where deltas is versioned and you must puzzle together all deltas from the beginning to get the picture of the database.

Dzo currently works with the following database management systems: MySQL, Oracle and Microsoft SQL Server.

This first version of the document is focused to describe some samples and how to get dzo working by examples.

Chapter 2. Getting started

Samples with detailed instructions on how to setup and run.

All directory paths in the samples uses '/' (a unix/linux slash) as separator.

Table 2.1. Variables used in samples

Variable	Description
DZO_HOME	Basedirectory for dzo, this is the directory where the installation program extracted dzo.
JBOSS_HOME	JBoss home directory.

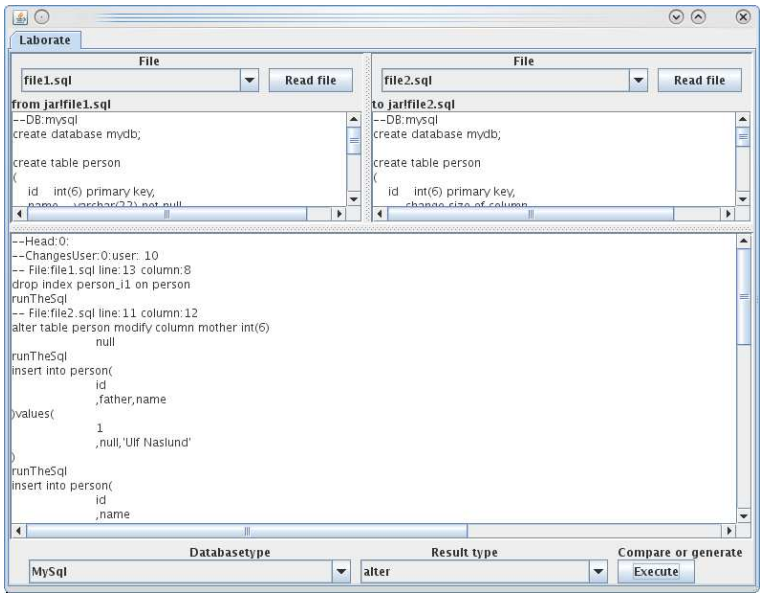
Laborate with dzo

Dzo is distributed with a very simple application to laborate with dzo in the file `dzo-1.5.1.jar`. This is the simplest way to get a feeling what dzo is all about. The application has no connection to a real database, instead it shows the sql that is needed to change from one state to another state.

To start this application you need a java JRE and run the command:

```
java -jar DZO_HOME/lib/dzo-1.5.1.jar laborate
```

Figure 2.1. Application for laborating with dzo



There are 2 textboxes on the top of the application labelled `from` and `to`. Think of the `from` textbox as if it is the database (current state), and the `to` textbox as the sql source code (the wanted state). Select databasetype in the combo box, press execute and the needed sql will be presented in the large textbox.

To make it a little bit easier, the jar-file is distributed with some sql-files. Select a jar-fil under file and press read and the textbox will be replaced with the selected file.

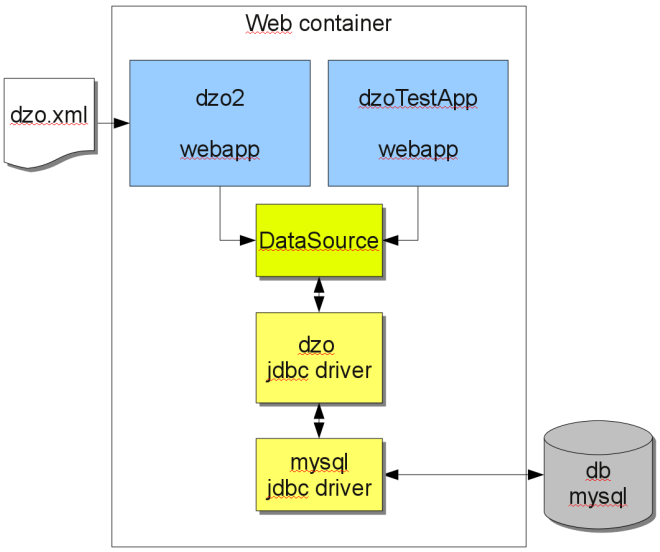
Sample for JBoss

How to setup dzo for a sample application in JBoss.

Overview of the setup

The setup of this sample results in the following figure. Two Web-applications, dzo2 and dzoTestApp. Dzo2 administrating databasechanges through a datasource by looking up a configured jndi-name. DzoTestApp is a simple application that maintain a table i the database.

Figure 2.2. Components in sample setup



The setup for the JBoss sample consists of the following steps.

1. Install dzo
2. Install mysql jdbc-driver
3. configure datasource for sample application
4. Install sample application
5. Start JBoss application server
6. Inspect database changes and deploy application

Setup in detail

1. Install dzo
   - Copy `DZO_HOME/lib/dzo-1.5.1.jar` to `JBOSS_HOME/server/default/lib`. Check that you dont have any old version of `dzo-X.jar` in the directory.
   - Copy `DZO_HOME/samples/dzo2.war` to `JBOSS_HOME/server/default/deploy`
   - Put the following in the file `JBOSS_HOME/server/default/conf/dzo.xml`

```
<dzo>
  <application name="dzoTestApp" deploy="manual">
    <database name="voffwar" type="mysql">
      <file>dzoTest.sql</file>
      <datasource>java:jdbc/dzoTestAppDS</datasource>
    </database>
  </application>
</dzo>
```

2. Install mysql jdbc-driver
   - Download mysql jdbc driver from `http://mysql.com/downloads/connector/j`
   - Extract and copy the jdbc driver to directory `JBOSS_HOME/server/default/lib`
3. configure datasource for sample application
   - Create database and a database user in mysql with the sql commands:

```
create database testapp;
grant all priviliges
  on testapp.* to dzodba@localhost
  identified by 'dzodba' with grant option;
```

   - Put the following in the file `JBOSS_HOME/server/default/deploy/dzoTestApp-ds.xml`

```
<datasources>
    <local-tx-datasource>
        <jndi-name>jdbc/dzoTestAppDS</jndi-name>
        <connection-url>jdbc:dzo:jdbc:mysql://localhost/testapp?
characterEncoding=UTF8&amp;dzoRealDriver=com.mysql.jdbc.Driver</
connection-url>
        <driver-class>se.grunddata.dzo.jdbc.Driver</driver-class>
        <user-name>dzodba</user-name>
        <password>dzodba</password>
    </local-tx-datasource>
</datasources>
```

Note

The content of element `connection-url` is on one line with no white space.

4. Install sample application

   Copy the file `DZO_HOME/samples/dzoTestApp.war` to `JBOSS_HOME/server/default/deploy`

5. Start JBoss application server

   And now you need to start or restart JBoss application server.

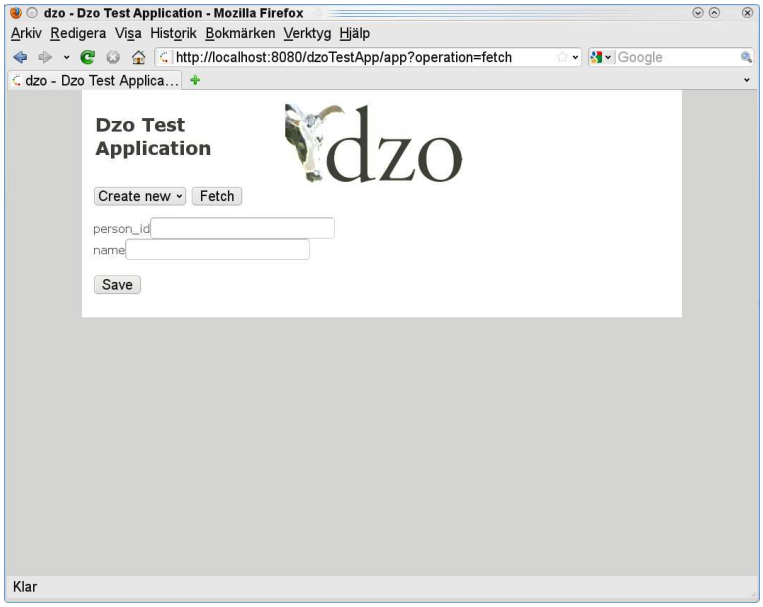
6. Inspect database changes and deploy application

   - Use your favourite browser and go to http://localhost:8080/dzo2
   - Select the application dzoTestApp and execute the needed database-changes.
   - Deploy the application
   - go to http://localhost:8080/dzoTestApp and laborate with the application.

Test Application.

The test application is no rocket science, it consists of a simple page which show the columns in table Person:

Figure 2.3. Dzo Sample Application



Select a person in the drop-down box and press Fetch. Select Create new to create a new person.

When the Person table changes the test application reflects this automatically and will show the actual columns in the table. The first column is assumed to be key/id and the second column the name that shows up in the drop-down box.

Make database changes

In the `DZO_HOME/samples` directory do:

- edit the file dzoTest.sql , and change Person table (for example add column `age`, and remember no `alter`, just add a line with " `,age decimal(2)`" before the closing ')' in `create table Person`).
- run `./ant dzoTestApp` to rebuild `dzoTestApp.war` with the new database objects
- Redo point 4, Install application
- Redo point 6, Inspect database changes and deploy application

Make a application dzo-enabled

To make a application dzo-enabled the following must be done:

1. The existence of DZO-INF directory and the sql sourcecode in the file DZO-INF/dzoTest.sql.

```
    0 Wed Apr 28 15:55:18 CEST 2010 META-INF/
  144 Wed Apr 28 15:55:16 CEST 2010 META-INF/MANIFEST.MF
    0 Wed Apr 28 15:14:22 CEST 2010 DZO-INF/
  168 Wed Apr 28 15:14:22 CEST 2010 DZO-INF/dzoTest.sql
    0 Wed Apr 28 15:14:22 CEST 2010 WEB-INF/
    0 Wed Apr 28 15:14:22 CEST 2010 WEB-INF/lib/
 7441 Wed Apr 28 15:55:16 CEST 2010 WEB-INF/lib/dzoTestApp.jar
   95 Wed Apr 28 15:14:22 CEST 2010 index.html
 1085 Wed Apr 21 11:21:02 CEST 2010 WEB-INF/web.xml
```

2. In web.xml there must be a servlet-definition with a load-on-startup number which must be the lowest in the application.

```
 <servlet>
  <servlet-name>dzo</servlet-name>
  <servlet-
class>se.grunddata.dzo.container.generic.DzoClientServlet</servlet-
class>
  <load-on-startup>1</load-on-startup>
 </servlet>
```

3. The use of dzo:s jdbc driver in the datasource.

And configure the application in the dzo.xml configuration file.

Sample for ant

How to setup dzo using a simple build-file. Ant i distributed in `dzo-1.5.1.jar` so there is no need to install ant for this sample.

Overview of the setup

This sample makes it possible to compare and execute changes in a database with the ant utility.

Setup in detail

- Install mysql jdbc-driver
  - Download mysql jdbc driver from `http://mysql.com/downloads/connector/j`
  - Extract and copy the jdbc driver to directory `DZO_HOME/lib`
- configure database for sample application
  - Create database and a database user in mysql with the sql commands:

```
create database testapp;
grant all priviliges
  on testapp.* to dzodba@localhost
  identified by 'dzodba' with grant option;
```

- Change SQL source code

  The file `DZO_HOME/samples/dzoTest.sql` contains the SQL source code. Edit the file.
- Compare and show the needed changes
  - Make a cd to `DZO_HOME/samples`
  - Compare and show the needed changes with

```
./ant compareDB
```

  - Execute the needed changes with

```
./ant changeDB
```

Chapter 3. Build tools

How to use dzo with ant

Get dzo with ivy.

To download dzo with ivy put the following line in ivy.xml:

```
<dependency org="se.grunddata.dzo" name="dzo" rev="1.5.1" />
```

Example `build.xml`

```
<!DOCTYPE null [
<!ENTITY version "1.5.1-SNAPSHOT">
<!ENTITY today "maj 24, 2010">

]>
<project default="usage">
  <property name="topdir" value=".."/>
  <property name="testwar" value="dzoTestApp.war" />
  <property name="db.type" value="mysql"/>
  <property name="db.driver" value="${topdir}/lib/mysql-connector-
java-5.1.6.jar"/>
  <property name="db.sqlfile" value="dzoTest.sql"/>
  <property name="db.url" value="jdbc:mysql://localhost/testapp"/>
  <property name="db.user" value="dzodba/dzodba"/>

  <target name="usage">
    <echo>The following targets exists:
  dzoTestApp Rebuilds dzoTestApp.war and replace DZO-INF/${db.sqlfile}
  compareDB  Compare db ${db.url} and show the needed changes
  changeDB  Like compare but executes the needed changes

compareDB and changeDB needs a jdbc-driver in file ${db.driver}
    </echo>
  </target>

  <target name="dzoTestApp">
    <echo>Update ${testwar} with new database objects.</echo>
    <mkdir dir="DZO-INF" />
    <copy file="${db.sqlfile}" todir="DZO-INF" />
    <zip destfile="${testwar}" basedir="." includes="DZO-INF/*"
 update="true" />
    <delete dir="DZO-INF" />
  </target>

  <target name="dzo">
    <taskdef classname="se.grunddata.dzo.ant.Dzo" name="dzo">
      <classpath location="${topdir}/lib/dzo-1.5.1.jar" />
```

```
      <classpath location="${db.driver}" />
    </taskdef>
  </target>

  <target name="compareDB" depends="dzo">
    <dzo>
      <schema database="${db.type}"
        toFile="${db.sqlfile}"
        fromUrl="${db.url}"
        fromUser="${db.user}"/>
    </dzo>
  </target>

  <target name="changeDB" depends="dzo">
    <dzo showSql="true" execute="true">
      <schema database="${db.type}"
        toFile="${db.sqlfile}"
        fromUrl="${db.url}"
        fromUser="${db.user}"/>
    </dzo>
  </target>
</project>
```

How to use dzo with maven2

Dzo is installed in maven central repositorys.

Note

The maven plugin for dzo needs at least version 2.2 of maven to work properly.

Table 3.1. Maven available goals

Goal	Description
`dzo:compareExec`	Compares one or more schemas and executes the needed changes
`dzo:compareShow`	Compares one or more schemas and shows the needed changes in the log
`dzo:generateJpa`	Creates java JPA-annotated entity beans
`dzo:source`	Generates sql sourcecode
`dzo:xml`	Generates xml document.

Table 3.2. Maven configuration propertys

Property	Mandatory	Type	Description
`schemas`	Yes	java.util.List	Contains one ore more of the sub-element schema
`destFile`	No	java.lang.String	Destination file for the selected mojo:s result, if omitted the result shows up in the log.

The sub-element schema have two groups of propertys prefixed with `from` and `to`. The `from` propertys is normally the database schema (what you have right now) and the `to` is the SQL source code (what you want). This means that you normally use the propertys `fromUrl` and `fromUser` and `toFile` (or just `file`).

Table 3.3. Maven configuration propertys for sub-element `schema`

Property	Mandatory	Type	Description
`database`	Yes	java.lang.String	Type of database
`fromDriver`	No	java.lang.String	Classname for the jdbc driver to use
`fromUrl`	No	java.lang.String	Jdbc url to schema to compare
`fromUser`	No	java.lang.String	Username and password delimited by '/'
`fromFile`	No	java.lang.String	A file containing SQL source code
`fromIncludepath`	No	java.lang.String	Directorys delimited by ':' when processing the `#include` directive.
`fromJpaIncludepath`	No	java.lang.String	Directorys delimited by ':' to search for JPA annotated classes when processing the `#jpainclude` directive. Basedirectory to the compiled classes. Normally the `target/classes` directory
`toDriver` or `driver`	No	java.lang.String	See fromDriver
`toUrl` or `url`	No	java.lang.String	See `fromUrl`
`toUser` or `user`	No	java.lang.String	See `fromUser`
`toFile` or `file`	No	java.lang.String	See `fromFile`
`toIncludepath` or `includepath`	No	java.lang.String	See `fromIncludepath`
`toJpaIncludepath` or `jpaIncludepath`	No	java.lang.String	See `fromJpaIncludepath`

A small example:

Example `pom.xml`

```
<!DOCTYPE null [
<!ENTITY version "1.5.1-SNAPSHOT">
<!ENTITY today "maj 24, 2010">

]>
<project xmlns="http://maven.apache.org/POM/4.0.0"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://
maven.apache.org/xsd/maven-4.0.0.xsd">

  <modelVersion>4.0.0</modelVersion>
  <groupId>se.grunddata.dzo.example</groupId>
  <artifactId>example</artifactId>
  <version>1.5.1-SNAPSHOT</version>
```

```
  <packaging>war</packaging>
  <name>Example using dzo in maven2</name>
  <build>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-war-plugin</artifactId>
        <configuration>
          <webXml>web.xml</webXml>
        </configuration>
      </plugin>
      <plugin>
        <groupId>se.grunddata.dzo</groupId>
        <artifactId>dzo-maven2</artifactId>
        <version>1.5.1</version>
        <configuration>
          <schemas>
            <schema>
              <database>mysql</database>
              <fromUrl>jdbc:mysql://localhost/testapp</fromUrl>
              <fromUser>dzodba/dzodba</fromUser>
              <file>dzoTest.sql</file>
            </schema>
          </schemas>
        </configuration>
       <executions>
          <execution>
            <id>makeXml</id>
            <phase>prepare-package</phase>
            <goals>
              <goal>xml</goal>
            </goals>
            <configuration>
              <destFile>target/example-${project.version}/DZO-INF/
dzoTest.xml</destFile>
            </configuration>
          </execution>
        </executions>
        <dependencies>
          <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
            <version>5.1.6</version>
          </dependency>
        </dependencies>
      </plugin>
    </plugins>
  </build>
</project>
```

The execution with id `makeXml` in `pom.xml` creates a xml-document with the SQL in `DZO-INF` directory the generated war.

To compare the database schema and show the needed changes:

```
mvn dzo:compareShow
```

Index