How to seed DB with test data

For tests to run smoothly, it's important that each test, or at the minimum, each test suite, be able to restore the Tiki application to a given starting state. This starting state should contain the data necessary to carry out the test or test suite, and nothing more. In particular, it should not contain any side effects caused by tests that have been run priorly.

There are at least two possible approaches to this (actually, probably more, see below):

- Create an XML dump which, when restore will put the TikiWiki in the desired starting state for a given test. Then restore this dump at the beginning of each suite or test.

- Create an ordered list of profiles which, when applied to a fresh install of Tiki, will put the TikiWiki in the desired starting state for a given test. Then restore this profile at the beginning of each suite or test.

Both approaches seem reasonable. The criteria for choosing between them might be:

- Criteria 1: Which approach will allow easiest creation of tests by developer/tester, especially for non-technical testers.
  - Simplicity of creation is primordial for the following use case: Testing Use Case: Non-dev wants to create a test and share it with the dev community
  - It's not clear which of the two approaches is best w.r.t. Criteria 1. On the one hand, it's probably easier to understand a profile than a DB. On the other hand, it might be easier for a non-technical person to create DB dump than a profile. Creating a DB dump involves creating a new DB, then using the Tiki UI to configure it just the way you want it to be, then do the dump. Creating a profile requires you to understand the syntax of the profile files. Also, there might be some things that need to be done to the DB which simply cannot be done through existing profile instructions.

- Criteria 2: Which approach will be faster in terms of being able to restore the app to a given starting point.
  - Running speed of tests is critical to the following user cases:
    - Testing Use Case: Dev wants to ensure that a particular version of Tiki passes all tests
    - Testing Use Case: Dev wants to know if he broke something with his most recent change
  - It's not clear which of the two approaches will result in tests that run fastest.
  - Based on some tests done by Alain Désilets (see below), it seems that restoring a basic DB (as it was created right after install) took a good 30 seconds to restore!!! But that was using the "easy use" profile, which seems to have a lot of baggage installed. Will try with the Default profile which, I presume is more barebone.

- Criteria 3: Which approach will yield tests which are easiest to share, no matter what version of TW was used to record the test?
  - This is critical for the following use cases:
    - Testing Use Case: Non-dev wants to create a test and share it with the dev community
    - Testing Use Case: Dev wants to create a test and share it with the dev community
  - In that respect, profiles are probably more neutral to the TW version than DB dumps.

- Criteria 3: which approach will yield tests that are more amenable to version control?
  - This is crucial to use case Testing Use Case: Dev needs to be able to modify an existing test
  - I think both DB dump and profiles are text based. But profiles are probably easier to understand
and merge with SVN than DB dumps.

Approach 1: Restore a MySQL dump before each new test (synchronously).


Tried this command:

```bash
mysqldump -uroot tiki > tiki.sql
```

where 'tiki' was the name of the DB.

But got this error:

```
mysqldump: Couldn't execute 'SHOW TRIGGERS LIKE 'tiki\_freetags'': Can't create/write to file 'C:\DOCUME~1\DESILE~1\LOCALS~1\Temp\#sql_138c_0.MYI' (Errcode: 17)
```

As it turns out, according to this page:

http://www.poscribes.com/5_steps_to_mysql_replication_on_windows

This is caused by VirusScan. In order for mysqldump to work on windows, you need to set some exceptions in VirusScan:

- Right click on VirusScan icon in the system tray
- On-Access Scan Properties
- All Process (at the left)
- Detection Tab
- Exclusions > Add > Browse
- Select the temp directory mentioned in the error message.
- Check Also execute subfolders
- OK

Also:

- All Process
- Unwanted Programs
- Uncheck Detect unwanted programs
- OK

Once that was done, I was able to restore the DB as follows:

```bash
mysql -uroot tiki < tiki.sql
```

Note however that restoring took a good 30 seconds.

Ideas for decreasing the restore time:

- Manually edit the .sql file, and delete the tables that are not needed.
○ Have to be careful in doing it.
  ○ Increase MySQL buffer size in mysql.ini (Rick Sapir suggested that, but not clear to AD why that would help).
  ○ I edited the my.ini file and increased all variables that had something to do with buffer, by a factor of 10. Saved, restarted EasyPHP, rerun: still about 30 secs.
  ○ Increase MySQL max-allowed-packet in mysql.ini
    ○ tried --max_allowed_packet=16M, then 160M, then 1600M. Still about 30 secs each time.

Approach 2: Restore a MySQL dump before each new test (Asynchronously)

This one was suggested by by Chris Drake

> If you wanted an "instant" (0-seconds-restore) solution - that, I think, would be very easy:
> 1. Create *two* identical databases (eg: tiki1 and tiki2)
> 2. To "restore" - you...
  A. simply change the tiki settings to use whichever other one it's not already using (and, to be safe, restart apache)
  B. now - in parallel, you can drop and re-create the "stale" database that you just stopped using, in preparation for the next restore.

The only problem with that approach is that running the tests might take only say, 5 seconds, while creating the DB for the next test might take 30 seconds.

Approach 3: Delete data from tables at the end of each test

  - Instead of redefining the tables from scratch before each test, might do the following.
  - Create a test DB that has all the tables defined in it.
  - At the end of each test, simply delete all the data contained in all the tables (using mysql statements).
  - Then, at the start of each test, load profiles into this empty db.
  - Question is... how long does clearing the tables take? Might be just as long as creating the tables.

Approach 4: Copy all files related to a particular DB

  - In all likelyhood, the information contained in a particular DB is captured in the form of a series of files, and maybe some environment variables and registry entries.
  - Do we know what all those files en entries are?
  - Could we simply make copy of those files for the DB in its start state, then restore the DB to its start state by copying those files over?
  - I know this is not the way to backup and restore a DB that is in production. But it might be good enough for a test DB.
  - Question is... what are those files and entries?

According to Rick Sapir:

If you're talking about the actual *MySQL* database tables (*.MYD, *.MYI, etc.), they are located in
(assuming you are still using EasyPHP):

C:Program FilesEasyPHP 2.0b1mysqldata*YOUR DATABASE NAME*
(or something similar).

Simply copying these files to "rebuild/restore" your database each time should be much faster than rebuilding the database via MySQL each time. A good idea (wish I had thought of it)...

-R

Some info here:


"When it comes to backing up MySQL databases, don't ignore the most obvious solution: making copies of the database files themselves."

2009-02-20: As it turns out, just copying the DB files takes a good 15-30 seconds! So it's not really faster than doing a SQL DB dump and a restore.

Approach 5: MySQL test framework???

There might be some useful information here:


Approach 5: DbUnit

DBUnit:

- DbUnit is a JUnit extension (also usable with Ant) targeted at database-driven projects that, among other things, puts your database into a known state between test runs. This is an excellent way to avoid the myriad of problems that can occur when one test case corrupts the database and causes subsequent tests to fail or exacerbate the damage.

PHP port available here:

Recommended on this page:

by the developer of phunit.

Approach 6: SQLite

[http://www.sqlite.org/](http://www.sqlite.org/)

SQLite is a software library that implements a self-contained, serverless, zero-configuration, transactional SQL database engine.
Recommended on this page:


by the developer of phunit.