ITRW324 Project 4: Webtionary
November 9th, 2006

This document describes the Webtionary system and its installation procedures.

Table of contents:

<u>Section 1: Introduction</u>

<u>1. Site overview:</u>

Webtionary is a dynamic website designed to serve as a repository for South African languages. Its name is a slight pun, due to its likeness to a web dictionary. The system can be classified along its main features:

User management:
- User registration and administration
- User notification via e-mail in the events of registration and vocabulary modification
- Continuous, per-page login status notification

Vocabulary:
- Browse, review, download and upload user-contributed words and wordlists
- Download FOSS (*Free and Open Source Software*) wordlists from server tree
- Browse, review and download user-contributed translations
- Downloadable in plain text, XLS (Excel spreadsheet), XML and PDF formats

Games:
- Anagram maker
- Crossword builder
- Hangman
- Random word generator

Accessibility:
- Extensible multilingual support with default Afrikaans and English interfaces
- WAP support: login, vocabulary and translation interfaces delivered via WML
- Browser compatibility: Microsoft Internet Explorer, Firefox and Opera

Technology:
- Dynamic content retrievals via Ajax
- SSI page view counter
- Realm-based MD5 authentication via .htaccess file
- Localization implemented with JavaScript and content negotiation

All the abovementioned features are accessible via a web browser or mobile device, where appropriate.

The system makes use of PHP and the PostgreSQL DBMS.

Webtionary was created by Theo Fitchat as a third year web design project at the North-West University in Vanderbijlpark, South Africa. He is currently studying towards his BSc in Information Technology, to be concluded in 2006.

Section 2: Server configuration

This section briefly highlights the components required to run Webtionary.

2.1. Apache:

The Webtionary interface is designed to run on a web browser (mobile or otherwise), which ultimately sends and receives data in the HTTP protocol. A web server is required to communicate with the browser at the opposite end. The Apache web server is known to be compatible with Webtionary, and its usage is highly recommended.

Apache v2.0.55 through v2.2.3 was used during the development of Webtionary.
Apache home page: http://httpd.apache.org.

2.2. PHP:

Webtionary is programmed in PHP, and runs with its Apache module, as opposed to its standalone CGI interpreter.

It is required that PHP load its PostgreSQL extension library at startup, for interfacing with the database. This can be configured in the PHP.INI file. This alteration is usually easy to accomplish; for more details however, refer to the PHP documentation.

In addition, Webtionary requires the following settings to be configured:
- Web server needs to know where to locate PHP
- PHP needs the address of a valid SMTP server

It is beyond the scope of this document to expand on these procedures; once again, the PHP documentation will serve as a good starting point.

PHP v5.1.5 through v5.2.0 was used during the development of Webtionary.
PHP home page: http://www.php.net.

2.3. PostgreSQL:

The PostgreSQL DBMS is used for data storage and retrieval. A standard software installation will suffice for Webtionary’s purposes.

The Webtionary database structure is discussed in the following section. First-time database initialization (after structure creation) is discussed in Section 4.

PostgreSQL v8.0.3 through v8.1.5 was used during the development of Webtionary.
PostgreSQL home page: http://www.postgresql.com.

Section 3: Database design

3.1. Database overview:

Webtionary uses the PostgreSQL DBMS for data storage and retrieval.

A schema called ITRW324_PROJECT4 was created to collectively group the system's tables. The schema contains eight tables; a discussion of each table follows in sub-section 3.2.

3.2. Table synopsis:

Tables are listed in a pseudo-chronological fashion.

USER and ADMIN_USER:
Each row in these tables identifies either a regular or administrative Webtionary user, respectively.

LANG:
Each row in this table describes one language used within Webtionary.

WORD:
Each row in this table describes one word in a specific language, and also refers to the word's contributor.

AFFIX_TYPE:
Each row in this table describes the properties of one affix type, including the code used to represent it, its relative position (prefix/suffix) and the replaceable component.

AFFIX_SUBST:
Each row in this table describes a possible substitution for one affix type.

TRANS:
Each row in this table describes the translation of a word, from one language to another. No reference to the contributing user is stored.

CROSSWORD:
Each row in this table describes one crossword puzzle submitted by a user.

The aforementioned table design was chosen for simplicity and practicality. No redundant information or excessive null values exist in the database, except where user-induced, and the design also allows for rapid addition of new languages.

3.3. Table specifications:

ADMIN_USER:

Col. No.:	Name:	Type:	Constraint:
1	id	Integer (serial)	Primary key
2	username	Text	Not null
3	password	Text	Not null

USER:

Col. No.:	Name:	Type:	Constraint:
1	id	Integer (serial)	Primary key
2	username	Text	Not null
3	password	Text	Not null
4	name	Text	None specified
5	surname	Text	None specified
6	email	Text	None specified

LANG:

Col. No.:	Name:	Type:	Constraint:
1	id	Integer (serial)	Primary key
2	lang	Text	Not null

WORD:

Col. No.:	Name:	Type:	Constraint:
1	id	Integer (serial)	Primary key
2	lang	Integer	Foreign key to LANG.id, not null
3	user	Integer	Foreign key to USER.id, null allowed
4	word	Text	Not null

AFFIX_TYPE:

Col. No.:	Name:	Type:	Constraint:
1	id	Integer (serial)	Primary key
2	lang	Integer	Foreign key to LANG.id, not null
3	code	Text	Not null
4	type	Integer	Not null
5	affix	Text	Not null

AFFIX_SUBST:

Col. No.:	Name:	Type:	Constraint:
1	id	Integer (serial)	Primary key
2	affix	Integer	Foreign key to AFFIX_TYPE.id, not null
3	subst	Text	Not null

TRANS:

Col. No.:	Name:	Type:	Constraint:
1	id	Integer (serial)	Primary key
2	from_word	Integer	Foreign key to WORD.id, not null
3	to_word	Integer	Foreign key to WORD.id, not null

CROSSWORD:

Col. No.:	Name:	Type:	Constraint:
1	id	Integer (serial)	Primary key
2	lang	Integer	Foreign key to LANG.id, not null
3	user	Integer	Foreign key to USER.id, null allowed
4	rows	Integer	Not null
5	cols	Integer	Not null
6	data	Text	Not null
7	hint	Text	None specified

3.4. Database properties:

The following database rule adjustments are required by Webtionary:

Referential integrity:
The following shall be different from the default settings:

- All foreign keys are set to CASCADE on an ON DELETE event
- All foreign keys are set to CASCADE on an ON UPDATE event

Exceptions to abovementioned rules:

- Column CROSSWORD.user is set to SET NULL on an ON DELETE event, since the removal of a user shouldn't induce the removal of their contributed crossword puzzles

- Column WORD.user is set to SET NULL on an ON DELETE event, since the removal of a user shouldn't induce the removal of their contributed words

Indexes (secondary keys):
The following additional column indexes shall be created:

- ADMIN_USER: unique index on *username* column
- AFFIX_SUBST: unique index on (*affix*, *subst*) column combination
- CROSSWORD: unique index on (*lang*, *rows*, *cols*, *data*) column combination
- LANG: unique index on *lang* column
- TRANS: unique index on (*from_word*, *to_word*) column combination
- USER: unique index on *username* column
- WORD:
  - indexes on *lang*, *user*, and *word* columns
  - unique index on (*lang*, *word*) column combination

Checks:
The following additional table check shall be created:

- TRANS: *from_word* <> *to_word*

Implicit, unenforced rules:
LANG: no *id* shall fall outside the range [1; N], where N is the number of rows.

Privileges:
The *public* group shall (optionally) be revoked of all rights to access the database, as only the schema owner and the superuser need directly perform statements on tables in the ITRW324_PROJECT4 schema.

Section 4: System configuration

4.1. Installation:

The system has no automated installation. The steps to installing and running Webtionary are roughly:

1. Set up servers (Section 2)
2. Extract Webtionary site from distribution files (see below)
3. Create database structure (Section 3 and sub-section 4.2)
4. Launch web server and DBMS (see Usage on next page)
5. Visit Webtionary site

Webtionary's standard distribution files are located under the **/data/share/bin** directory. Included in the directory are:

- Base site files;
- SQL dump(s) of the default database contents (also called boot files);
- Public, static data (FOSS wordlists);
- Anagram generator with wordlists; and
- System documentation

Extracting these archives to some directory under the web server tree, keeping the archived directory structure intact, will complete the traditional *installation* of the system – now, the database structure has to be created. Please refer to Section 3 and sub-section 4.2 for more information on this procedure.

Webtionary uses a free server-side anagram generator. It has no separate installation or configuration; it is fully functional after extraction.

4.2. Database structure:

The Webtionary structure can be created either manually or automatically. For manual creation, please refer to Section 3.

Regarding the manual creation of the database structure; you may choose among one or more of the following utilities to aid in its creation:

- pgAdmin;
- phpPgAdmin (web-based); and
- psql (command-line interface).

pgAdmin and phpPgAdmin are both intuitive; only resort to the command line interface if the former are unavailable.

Alternatively, a *SQL dump* file exists that may be loaded into the DBMS to automatically create the database structure. This file is distributed with Webtionary, located at **/data/boot/init/structure.sql**. The file consists of various SQL data definition statements that set up the database environment required by Webtionary.

4.3. Initialization and recovery:

Webtionary contains a **boot script** which can be run at any time, in order to reset the database to its original state. This can be used, for example, to initialize the system, or to recover from a corrupted database.

The boot script is activated when either:

- **Reset database** is selected from Webtionary's administration area; or
- **init.php** is executed from the **cgi-bin** directory.

The **/data/boot/init/data.sql** file contains the SQL statements that are executed during initialization. This file is said to be the *SQL dump* of the database contents. This file can be re-created by executing **/cgi-bin/export.php** in a shell or command line.

The database structure is not created or modified by this process – the database is merely populated. Sub-section 4.2 describes the automatic structure creation process.

Note, however, that it may take a long time to reset the database, as all the current data in the database is discarded, and the original data reloaded from the SQL dump.

4.4. Usage:

To activate Webtionary:

1. Start web server
2. Start PostgreSQL (possibly using restricted user account)
3. If the web server is running on the local machine, visit http://localhost and navigate to the virtual path under the server tree where Webtionary was installed
4. If the web server is **not** running on the local machine, visit the appropriate site on the remote machine instead

4.5. Troubleshooting:

If you are unable to perform the steps in sub-section 4.4, then one or more of the following may be the cause:

- Webtionary was not properly installed;
- Its database was not set up correctly; or
- There is a server configuration problem.

Study the server log files for more information, as they are the first sources that report on errors.