Bits & Books - User Manual

Authors: Ryan McGowan and Alex Notwell

Introduction

This document contains the basic information to accomplish several common tasks associated with database management. Specifically, this document describes how to accomplish these tasks with the given schema. These tasks include:

- Accessing the database
- Adding records to the database
- Running a saved query (or script)

NOTE: This document assumes you are using a UNIX like system. All the required tasks should be possible with windows, but instructions are not offered here.

Accessing the Database

The database is preloaded on the student CS&E server. To access this database there are two basic steps:

1. Logging in to the student server.
2. Logging into MySQL.

Part 1. Logging into stdsun

To run basic mysql commands it does not matter whether you log into stdsun, or stdlogin. However, if you want to use any python added functionality you must use stdlogin since stdsun does not have a recent enough version of python. To logon to stdsun use the command below:

```
ssh <username>@stdsun.cse.ohio-state.edu
```

NOTE: You can skip to Part 2. Logging into MySQL here if you are not going to install the python application.

If you want to use stdlogin the just change the subdomain to `stdlogin`.

```
ssh <username>@stdlogin.cse.ohio-state.edu
```

If you want to use python functionality (you logged into stdlogin), you must subscribe to PYTHON27. To do this use the `subscribe` command. Once you log out and back in you should have access to python–2.7. This version of python is required to run the python application included with this project.

Once you have python ready you need to install the application, but first we need the source. You can clone the source with `git` (you must `subscribe` to GIT first) using the following command:

```
git clone http://www.ryanmcg.com/repo/cse670.git
```

You can also use the packaged tar you should have received.

Instructions for installing the application are in the projects `README.md` file, which has been attached to this document for your convenience.

In the installation instructions `virtualenv` is used. It is not readily available on stdlogin so it must also be installed. To install `virtualenv` just grab it from github:

```
curl -O https://raw.github.com/pypa/virtualenv/master/virtualenv.py
```

And replace all calls to `virtualenv` with `python virtualenv.py` in the README.md file.

NOTE: For virtualenv to work well you should be using `bash`. stdlogin does not use `bash` as the default shell, but you can switch into it by executing `bash`.

Part 2. Logging into MySQL

Once you are logged in you can connect to mysql. If you successfully set up the python environment then you can start a python shell after you have activated your virtual environment with a simple call to `python`. Now, just import `play.py` to use application models to access the database:

```
>>> from play import * # This imports all models and the db instance
>>> # To get all the users in the database you can do something like this:
>>> User.query.all()
```

Of course, you do not have access the database via the web application. To access the database with mysql use the following command where the <`database name`> = `c670aa_mcgowanr` on the cse server.

```
mysql -u <your username> -p <database name>
```

You should get some prompt like the one below. The query shown will show you what tables are in the schema.

```
mysql> SHOW TABLES;
```

Adding Records

Once we have successfully logged in to mysql and/or setup the python app we can add some records! To do this in mysql we run the necessary SQL commands. For example, to create a User we would do:

```
mysql> INSERT INTO User (username, email, password, date_modified,
       date_created) VALUES ('Cooldude', 'cool@bro.com', '12345', NOW(),
       NOW();
```

We can do the same thing using the python app and sqlalchemy with the following commands in the python interpreter after we have imported `play.py`.

```
>>> ses.add(User(email="cool@bro.com", username="Cooldude", password="12345"))
>>> ses.commit()
```

To see extended examples of adding records see the Database Description Document.

Running A Saved Query (Or Script)

If you went through the python application setup then you will have probably already ran some saved queries in the form of `.sql` files. There are two ways to do this. We can either direct a file into mysql via stdin or we can source a sql file from within the mysql interpreter.

To run a mysql script through stdin you can do something like this:

```
mysql [whatever your connection information is] < some_sql_commands.sql
```

This is assuming that the `some_sql_commands.sql` file contains valid SQL queries. You can also log in to the mysql prompt (as before) and run either of the following commands:

```
mysql> source some_sql_commands.sql
mysql> # Or if you like shortcuts
mysql> \. some_sql_commands.sql
```

Bits & Books

Authors: Ryan McGowan and Alex Notwell

Description

Flask front-end for CSE 670 Database project.

Development

Technologies Used:

- Python
  - Flask
- Compass and SASS
  - blueprint/semantic
- HAML
- MySQL

Setup

Database Setup

Setting up the database is a multi step process. To start off you need to have the necessary permissions to create a database. Any value enclosed by angle brackets (e.g.) is a variable and should be replaced by the appropriate value.

If you are installing this on your own system and your MySQL super user is root then you can use the following values:

```
<super user> = root
<bitbook user> = bitbook
<bitbook database> = bitbook
```

If the database is preinstalled (e.g. if we are running this on stdsun), then the first command should not be used.

1. Create the database/schema and a user to access it. This only needs to be done if the database does not already exist.

   ```
   mysql -u <super user> -p < create-database.sql
   ```

2. Create the tables (The default password for the bitbook user is 'amazon'). If you are not running as the bitbook user (e.g. the database has already been created for you) then use whatever username has already been given.

   ```
   mysql -u <bitbook user> -p <bitbook database> < create.sql
   ```

3. To use the built in data loaders you must first have the python application setup properly. This can be done by following the instructions below.

4. After you have finished loading the data you may finalize the schema.

   ```
   mysql -u <bitbook user> -p <bitbook database> < finalize.sql
   ```

And that's it for creating the database. Once it is up and running you can try installing and running the Flask application with the instructions below.

Web Application (Python/Flask) Setup

This package is compatible with Heroku/cedar. See the Heroku website for instructions on how to deploy a Python - Flask app to Heroku.

To get the app running locally you just need to run a few commands. (Most of these are covered in the Heroku instructions referenced above).

1. Install *pip* on your computer (if it isn't already installed). This may change depending on your distribution, so you might want to look this up on your own.

   Since Flask is not compatible with Python 3 you should make sure that your *pip* excutable is using Python 2.x not Python 3.x. On Arch Linux this means you have to run 'pip–2.7' instead of 'pip'.

2. Install *virtualenv*

   ```
   pip install virtualenv
   ```

   You might need to run the above command as sudo depending on your setup.

3. Setup and initialize *virtualenv*.

```
cd /path/to/this/project/
virtualenv --no-site-packages env
source env/bin/activate
```

If you are using bash you can replace the last command with:

```
. env/bin/activate
```

4. Install Flask and other dependencies.

```
pip install -r requirements.txt
```

5. Before we can run the application we must make it aware of the database. Copy the `config-example.yml` file to `config.yml` with the following commands and edit it accordingly. *NOTE: Pay attention to what python to mysql adapter you use. You might not get pymysql to work on your machine, but something else might do the job.*

```
cp config-example.yml config.yml
$EDITOR config.yml
```

6. That's it! You probably want to load the data now. You can do this by running the following commands:

```
$ python load-users.py
$ python load-books.py
$ python load-inventory-orders.py
```

Once that's done you can now either use the application interactively:

```
$ python
....
>>> from play import *
>>> #Now I can use the models and application to do whatever I want
>>> User.query.all() # This returns all the users in the database
```

Or you can start the web app:

```
python web.py
```

Or if you have foreman installed (`gem install foreman`):

```
foreman start web
```