ZeeGuu
Step-by-Step Installation Guide

Supplementary Documentation

Simon Marti

marti.simon@students.unibe.ch

Software Composition Group
Institute of Computer Science and Applied Mathematics
University of Berne

Supervisors:
Dr. Mircea F. Lungu
Prof. Dr. Oscar Nierstrasz

September 2. 2013

Abstract

This document provides a step-by-step guide on how to install the server-side part of the *ZeeGuu* platform introduced in *ZeeGuu - A Platform for Second Language Acquisition Through Free Reading and Repetition*[1]. This includes the *Language Progress Model*, the accompanying *Application Programming Interface* as well as the *Language Gym*.

This guide provides instructions for development environments on *OS X*[1] and *Linux* operating systems as well as productive environments on *Linux*[2].

[1]`http://apple.com/osx`

[2]While *OS X* is capable of acting as a productive web server, it is rarely actually used for this.

Contents

Chapter 1

Introduction

ZeeGuu is implemented as a *Python*[1] *flask*[2] application. It is packed into a single *Python* package called `zeeguu` and runs on *Python* versions `2.7.x`.

The following chapters guide through the installation of all dependencies, the application itself and its configuration.

The instructions for *OS X* are targeted at *OS X Mountain Lion* (`10.8`) as well as the yet to be released *OS X Mavericks* (`10.9`). The installation process is very similar, if not identical, on older versions of *OS X*. All provided screenshots are made on the preview version `13A558` of *OS X Mavericks*, released on August 21, 2013.

For *Linux* the instructions are shown at the example of *Ubuntu* version `13.04`. On *Debian*[3]-based systems the process is identical and on other distribution it is very similar. Some distributions might provide a different packet manager, such as *yum*[4], instead of *Debian*'s *APT*[5]; the names of the needed packages are however close to identical in all of them.

The methods described in this document assume that the user has root privileges on the system. While it is possible to install all dependencies in user space, this is not recommended and should only be attempted by experienced users.

[1] `http://python.org/`
[2] `http://flask.pocoo.org/`
[3] `http://www.debian.org/`
[4] Yellowdog Updated, Modifier: `http://yum.baseurl.org/`
[5] Advanced Packaging Tool

Chapter 2

Installing Dependencies

This chapter describes the process of installing *Python* `2.7.x` and *git*[1] on the target system. If these dependencies are already met this chapter can be skipped.

2.1 OS X

Python is preinstalled on machines running *OS X*, it might however be outdated. Both the newest version of *Python* and *git* can be installed through *Apple*'s own *Integrated Development Environment* (IDE) *Xcode* which is available for free in the *App Store*: `https://itunes.apple.com/ch/app/xcode/id497799835`.

When launched for the first time, *Xcode* might need to download and install additional components. *Python* and *git* are available as a package called *Command Line Tools* which can be downloaded from the *Components* section of the *Downloads* tab in *Xcode*'s preferences window (which can be accessed by pressing `cmd+,`). The preferences window is shown in figure 2.2. It is recommended to restart the computer after the installation of this package.

2.2 Linux

Before beginning the installation the *APT* package lists should be updated with this command:

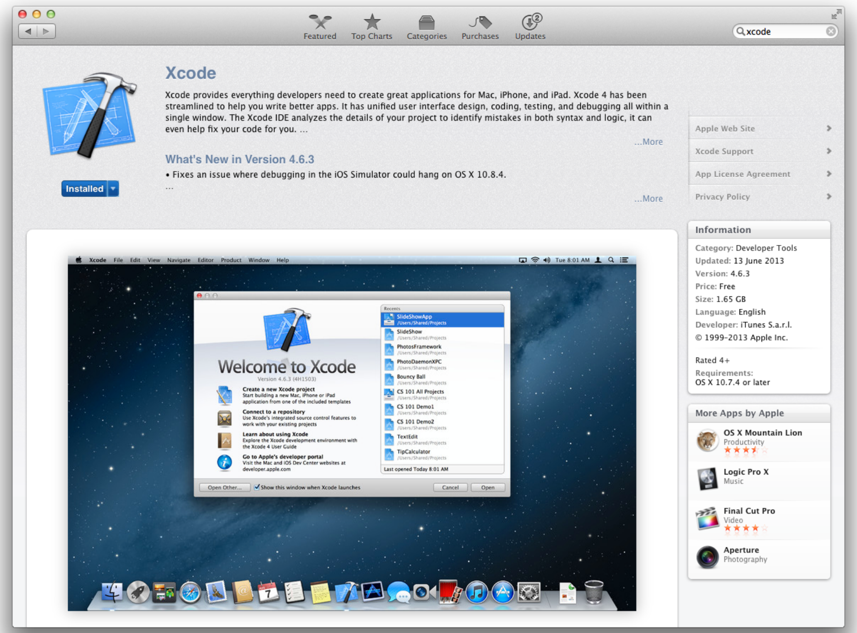
```
$ sudo apt-get update
```

All dependencies are then easily installed with the following command:

```
$ sudo apt-get install git python-setuptools gcc
```

APT might prompt for confirmation which can be done by pressing the return key.

[1] `http://git-scm.com/`



Xcode

Xcode provides everything developers need to create great applications for Mac, iPhone, and iPad. Xcode 4 has been streamlined to help you write better apps. It has unified user interface design, coding, testing, and debugging all within a single window. The Xcode IDE analyzes the details of your project to identify mistakes in both syntax and logic, it can even help fix your code for you. ...

...More

What's New in Version 4.6.3

• Fixes an issue where debugging in the iOS Simulator could hang on OS X 10.8.4.

...

...More

Apple Web Site
Xcode Support
App License Agreement
Privacy Policy

Information

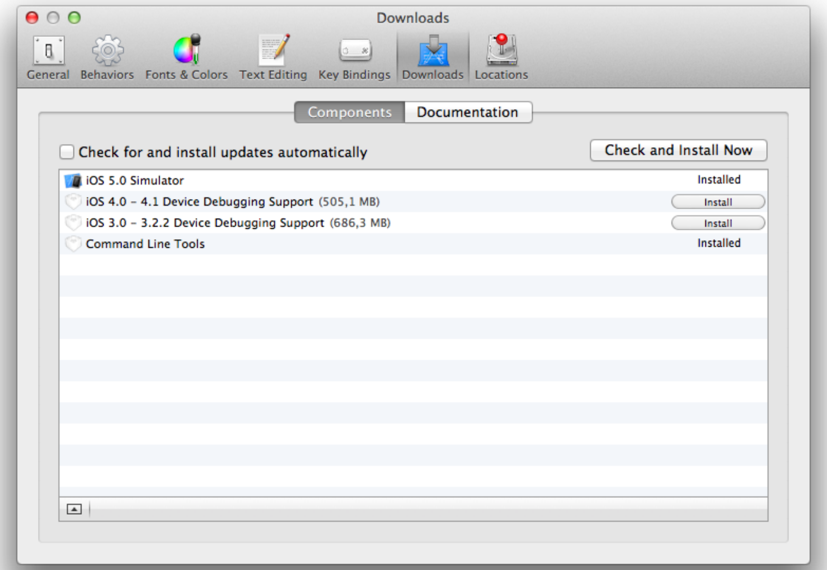
Category: Developer Tools
Updated: 13 June 2013
Version: 4.6.3
Price: Free
Size: 1.65 GB
Language: English
Developer: iTunes S.a.r.l.
© 1999-2013 Apple Inc.

Rated 4+
Requirements:
OS X 10.7.4 or later

More Apps by Apple

OS X Mountain Lion — Productivity
Logic Pro X — Music
Final Cut Pro — Video
Aperture — Photography

Figure 2.1: *Xcode* in the *App Store*



☐ Check for and install updates automatically

iOS 5.0 Simulator	Installed
iOS 4.0 – 4.1 Device Debugging Support (505,1 MB)	Install
iOS 3.0 – 3.2.2 Device Debugging Support (686,3 MB)	Install
Command Line Tools	Installed

Figure 2.2: The *Command Line Tools* package in *Xcode*'s preferences window

Chapter 3

Installing ZeeGuu

ZeeGuu is available on *GitHub*[1] and can be cloned into whatever dictionary the user prefers. *OS X* users have to open the *Terminal* application from `/Applications/Utilities` before continuing.

Downloading the source code and installing all needed *Python* modules can be done in the following steps:

1. Navigate to the preferred directory; for example:
   ```
   $ cd ~/dev
   ```
2. Clone *ZeeGuu*'s *git* repository:
   ```
   $ git clone https://github.com/ZeeGuu/API.git
   ```
3. Rename the repository to something more appropriate:
   ```
   $ mv API ZeeGuu
   ```
4. Change into the *ZeeGuu* directory:
   ```
   $ cd ZeeGuu
   ```
5. Install *ZeeGuu* and all needed *Python* packages:
   ```
   $ sudo python setup.py develop
   ```

The *ZeeGuu* module is installed in linked mode, meaning that the python files inside the repository can be edited directly; without the need to reinstall after every change. This makes sense in a development environment as *flask* even includes a feature which automatically restarts the application when changes to the source files are detected.

[1] `https://github.com/`

Chapter 4

Configuration

The default configuration of *ZeeGuu* is stored in the file `zeeguu/config.py`. This file should not be edited as it is under version control. Instead create a new configuration file in the app's `instance` folder:

```
$ mkdir -p instance
$ touch instance/config.cfg
```

The `config.cfg` is a regular *Python* file which can define a number of global variables. Some examples include:

HOST
: The host to bind to. (Default: localhost)

PORT
: The port to bind to. (Default: 9000)

DEBUG
: Enable debugging mode. (Default: True)

ASSETS_DEBUG
: Disable asset compression module. (Default: False)

SQLALCHEMY_DATABASE_URI
: Database URI. (Default: In-memory *SQLite* database)

In a development environment it might make sense to use an *SQLite* file as the database. The corresponding database URI looks as follows:

```
sqlite:////tmp/db.sqlite
```

The database URI syntax is further explained in *sqlalchemy*'s documentation[1].

The configuration file might look like this:

[1] `http://docs.sqlalchemy.org/en/rel_0_8/core/engines.html#sqlalchemy.create_engine`

```
PORT = 8080
SQLALCHEMY_DATABASE_URI = "sql:////tmp/db.sqlite"
```

After setting up the database the needed tables have to be created using the `zeeguu.populate` script. Simply run it like this:

```
$ python -m zeeguu.populate
```

This command also sets up a few languages and a testing user with the email address `user@localhost.com` and the password `password`.

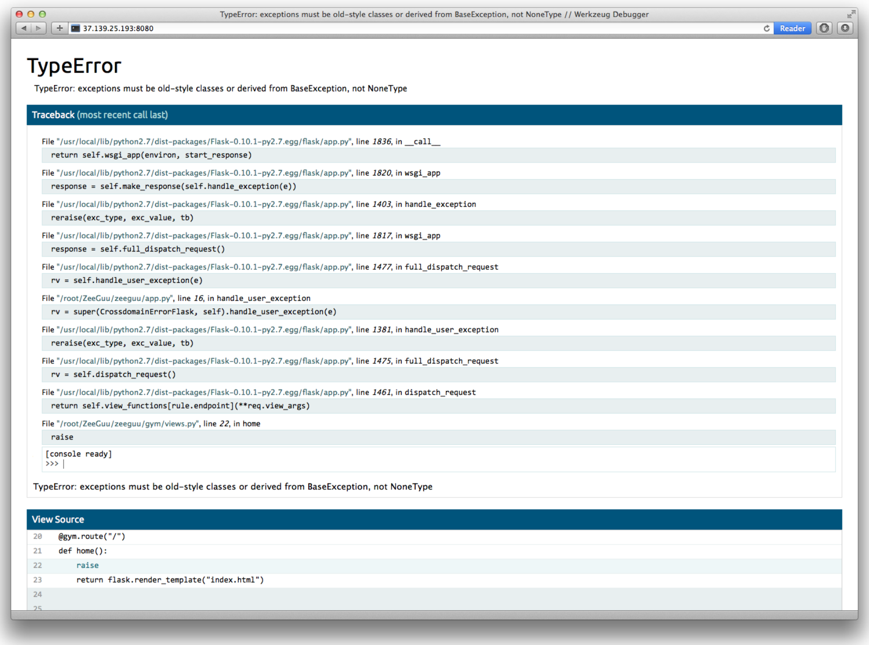
Chapter 5

Running ZeeGuu

To run *ZeeGuu* in development mode simply execute the `zeeguu` *Python* module directly:

```
$ python -m zeeguu
```

If not configured otherwise, *ZeeGuu* will now run on port `9000` and allow only connections from the local machine. As mentioned in chapter 3, running a *flask* application in this mode causes it to automatically restart after every change to the source files. While running in development mode *flask* also displays an elaborate debugger in case of application errors instead of a simple `500 Internal Server Error` page (see figure 5.1).



TypeError: exceptions must be old-style classes or derived from BaseException, not NoneType // Werkzeug Debugger
37.139.25.193:8080
Reader

TypeError

TypeError: exceptions must be old-style classes or derived from BaseException, not NoneType

Traceback (most recent call last)

```
File "/usr/local/lib/python2.7/dist-packages/Flask-0.10.1-py2.7.egg/flask/app.py", line 1836, in __call__
  return self.wsgi_app(environ, start_response)
File "/usr/local/lib/python2.7/dist-packages/Flask-0.10.1-py2.7.egg/flask/app.py", line 1820, in wsgi_app
  response = self.make_response(self.handle_exception(e))
File "/usr/local/lib/python2.7/dist-packages/Flask-0.10.1-py2.7.egg/flask/app.py", line 1403, in handle_exception
  reraise(exc_type, exc_value, tb)
File "/usr/local/lib/python2.7/dist-packages/Flask-0.10.1-py2.7.egg/flask/app.py", line 1817, in wsgi_app
  response = self.full_dispatch_request()
File "/usr/local/lib/python2.7/dist-packages/Flask-0.10.1-py2.7.egg/flask/app.py", line 1477, in full_dispatch_request
  rv = self.handle_user_exception(e)
File "/root/ZeeGuu/zeeguu/app.py", line 16, in handle_user_exception
  rv = super(CrossdomainErrorFlask, self).handle_user_exception(e)
File "/usr/local/lib/python2.7/dist-packages/Flask-0.10.1-py2.7.egg/flask/app.py", line 1381, in handle_user_exception
  reraise(exc_type, exc_value, tb)
File "/usr/local/lib/python2.7/dist-packages/Flask-0.10.1-py2.7.egg/flask/app.py", line 1475, in full_dispatch_request
  rv = self.dispatch_request()
File "/usr/local/lib/python2.7/dist-packages/Flask-0.10.1-py2.7.egg/flask/app.py", line 1461, in dispatch_request
  return self.view_functions[rule.endpoint](**req.view_args)
File "/root/ZeeGuu/zeeguu/gym/views.py", line 22, in home
  raise
[console ready]
>>> |
TypeError: exceptions must be old-style classes or derived from BaseException, not NoneType
```

View Source

```
@gym.route("/")
def home():
    raise
    return flask.render_template("index.html")
```

Figure 5.1: **DON'T PANIC**, *Werkzeug*'s debugger

Chapter 6

Production Environment

In a production environment a few things should be configured and set up differently.

First of all the web server built into *Werkzeug* is rather slow and not intended to be used for serving static files such as images. Instead *flask* applications can be run behind a high-performance web server such as *Apache*[1] or *nginx*[2] which accesses it through a *Web Server Gateway Interface*[3] (WSGI) or *FastCGI*[4]. For a more detailed list of deployment options for *flask* applications see also its documentation[5]. In this guide *Apache* and *WSGI* are used as they're the more popular option and work very well in most environments.

Secondly the *SQLite* database should be replaced by a system like *MySQL*[6] or *PostgreSQL*[7] to allow for faster and concurrent access to the stored data. This guide again uses the more popular option which is *MySQL*.

Further, the `zeeguu` module should be set up to run in production mode, which can easily be achieved through the `config.cfg` file created in chapter 4.

This chapter only addresses the *Linux* operating system, the same software is however also capable of running on *OS X*.

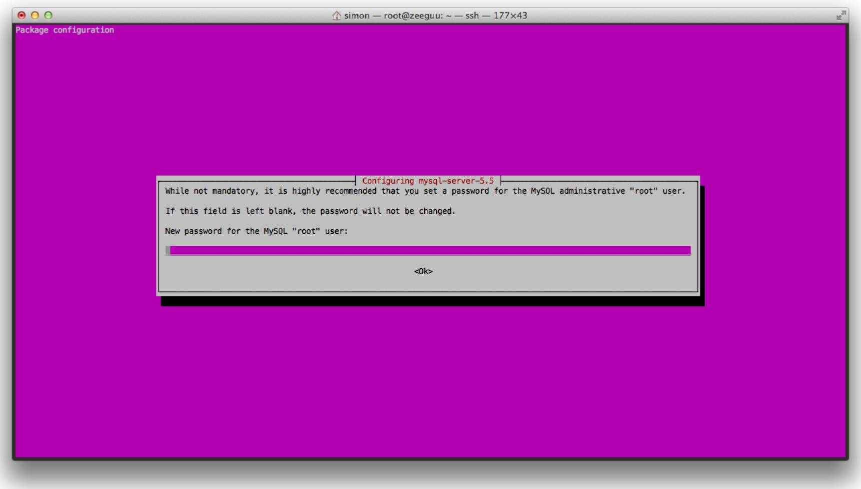
6.1 Installing MySQL Server

A MySQL server (and a few related packages) can be installed through *APT*:

```
$ sudo apt-get install mysql-server mysql-client
```

[1]`http://httpd.apache.org/`
[2]`http://nginx.org/`
[3]`http://wsgi.org/`
[4]`http://www.fastcgi.com/`
[5]`http://flask.pocoo.org/docs/deploying/`
[6]`http://mysql.com/`
[7]`http://www.postgresql.org/`

The installation script prompts for a root password, which is optional but should be set to something as secure as possible (figure 6.1).



```
Package configuration

Configuring mysql-server-5.5
While not mandatory, it is highly recommended that you set a password for the MySQL administrative "root" user.

If this field is left blank, the password will not be changed.

New password for the MySQL "root" user:

<Ok>
```

Figure 6.1: The *MySQL* installer prompting for a password

After the installation a user and database should be set up for *ZeeGuu*. To do this enter the mysql shell with the following command:

```
$ mysql -u root -p
```

Use the password set during the installation to get access. Now generate a new user (change the password):

```
> CREATE USER 'zeeguu'@'localhost' IDENTIFIED BY 'examplepassword';
```

And a new database:

```
> CREATE DATABASE zeeguu;
```

And finally grant the new user all privileges on his database:

```
> GRANT ALL ON zeeguu.* TO 'zeeguu'@'localhost';
```

Exit the shell by pressing `ctrl+D`.

To allow *sqlalchemy* to connect to a *MySQL* database, another *Python* module is needed. It has to be installed manually through pip[8] and has some dependencies of its own. Execute these commands:

```
$ sudo apt-get install libmysqlclient-dev python-dev
$ sudo easy_install pip
$ sudo pip install mysql-python
```

[8]Installing it directly through `easy_install` does not work.

6.2 Configuration

Before further configuring the `zeeguu` module, it should be installed properly into the intended locations. To do this invoke the `setup.py` script again, this time with a different argument:

```
$ sudo python setup.py install
```

In production mode a few settings of *flask* should be adjusted:

DEBUG
Needs to be disabled to prevent arbitrary code execution by users.

SQLALCHEMY_DATABASE_URI
Has to be adjusted to use the newly installed *MySQL* database.

SECRET_KEY
Should be set to a random string to prevent users from manipulating their session.

For more available options and more detailed descriptions see the *flask* documentation[9]. Setting these three values should be enough in most environments though. The configuration file might look something like this:

```
DEBUG = False
SQLALCHEMY_DATABASE_URI = ("mysql://zeeguu:example"
                           "password@localhost/zeeguu")
SECRET_KEY = "aabIFrkaAG14tBcqNg36wwoErAcWMgCNFO17UrsG"
```

To figure out where to save it the *flask* application needs to be run from somewhere else than its repository. Execute:

```
$ cd
$ python -m zeeguu
```

The path to the application's instance folder should be printed to the console. Terminate the application using `ctrl+C` and put your configuration file into the instance folder. This can be done with your favorite text editor, on most systems *nano* is preinstalled:

```
$ nano /usr/var/zeeguu.app-instance/config.cfg
```

The instance folder is not only used for the configuration but also to store combined and compressed assets, it is therefore important to grant the web

[9] `http://flask.pocoo.org/docs/config/#builtin-configuration-values`

server full access to it. This is best achieved by transferring ownership to the user `www-data`:

```
$ chown -R www-data /usr/var/zeeguu.app-instance
```

Don't forget to populate the database:

```
$ python -m zeeguu.populate
```

6.3 Installing Apache

Apache and its *WSGI* module can be installed through *APT*:

```
$ sudo apt-get install libapache2-mod-wsgi
```

Apache should instantly start running and display a "It works!" page when accessed in a browser.

To access the `zeeguu` module *Apache* needs a *WSGI* script which looks as follows and should be stored in `/etc/apache2/wsgi/zeeguu.wsgi`:

```
#!/bin/env python
from zeeguu import app as application
```

The last piece of the puzzle is the configuration for *Apache* itself. If *ZeeGuu* will be the only website running on the server the default configuration located at `/etc/apache2/sites-available/default` can simply be overwritten by this:

```
<VirtualHost *:80>
    WSGIDaemonProcess zeeguu user=www-data group=www-data
   threads=5
    WSGIScriptAlias / /etc/apache2/wsgi/zeeguu.wsgi

    <Directory /etc/apache2/wsgi>
        WSGIProcessGroup zeeguu
        WSGIApplicationGroup %{GLOBAL}
        Order deny,allow
        Allow from all
    </Directory>
</VirtualHost>
```

If *ZeeGuu* should only be accessible through a single domain name, store the following configuration in `/etc/apache2/sites-available/zeeguu` (replacing `example.com` with the desired domain):

```
Listen 80
NameVirtualHost *:80
```

```
<VirtualHost *:80>
 ServerName example.com

    WSGIDaemonProcess zeeguu user=www-data group=www-data
   threads=5
    WSGIScriptAlias / /etc/apache2/wsgi/zeeguu.wsgi

    <Directory /etc/apache2/wsgi>
        WSGIProcessGroup zeeguu
        WSGIApplicationGroup %{GLOBAL}
        Order deny,allow
        Allow from all
    </Directory>
</VirtualHost>
```

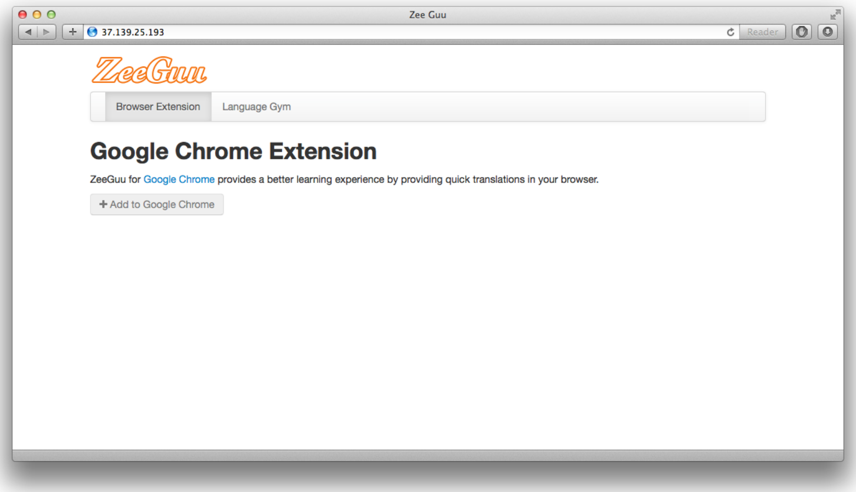
In the latter case the configuration must then be linked into the `sites-enabled` directory:

```
$ cd /etc/apache2/sites-enabled
$ sudo ln -s ../sites-available/zeeguu 001-zeeguu
```

After all these steps are complete *Apache* can be restarted:

```
$ sudo apachectl restart
```

And if everything went well, accessing the machine in a *Web Browser* shows the *ZeeGuu* web-interface (figure 6.2).



Google Chrome Extension

ZeeGuu for Google Chrome provides a better learning experience by providing quick translations in your browser.

+ Add to Google Chrome

Figure 6.2: The *ZeeGuu* web-interface

Bibliography

[1] Simon Marti. Zeeguu - a platfotm for second language acquisition through free reading and repetition. Bachelor thesis, University of Berne, August 2013.