Silpa Documentation

Release 0.1

Santhosh Thottingal

February 27, 2014

Contents

1 Install Instructions **3**
- 1.1 VirtialEnv Instructions . . . 3

2 Silpa-Flask **5**
- 2.1 Writing a module for Silpa-Flask . . . 5

3 Modules **7**
- 3.1 the modules . . . 7
- 3.2 flask-webfonts . . . 8

4 SILPA Webservice APIs **9**
- 4.1 Introduction . . . 9
- 4.2 Concepts . . . 9
- 4.3 Usage . . . 10

5 Frequently Asked Questions **11**
- 5.1 I am interested in contributing to this application. To whom should I contact? . . . 11
- 5.2 Can I use this application in my machine without having internet access? . . . 11
- 5.3 Can I use this application in windows? . . . 11
- 5.4 Can I host this application in my domain? . . . 11
- 5.5 Who is sponsoring the development of this project? . . . 11
- 5.6 When did this project development start? . . . 11
- 5.7 Is this application available in any GNU/Linux distributions? . . . 12
- 5.8 I found a bug in one module. How can I report? . . . 12

6 Credits **13**

7 Indices and tables **15**

SILPA is an acronym of Swathanthra(Mukth, Free as in Freedom) Indian Language Processing Applications. Its a web framework written using Flask micro framework for hosting various Indian language computing algorithms written in python. It currently provides JSONRPC support which is also used by web framework itself to input data and fetch result.

The modules work as standalone python packages which will serve their purpose and also they plug into the silpa-flask webframewok so that they can be accessed as web services also, or become just another webapp like the dictionary module.

Contents:

CHAPTER 1

Install Instructions

Note that this is currently a work in progress. So things may not work as you expected it to work. If you want to test this you need to have following installed on your system

- pip
- Virtualenv

Test deployment

```
$ git clone https://github.com/Project-SILPA/Silpa-Flask.git
$ cd Silpa-Flask
$ pip install -r requirements.txt
$ python silpa.py
```

If you want to Install all modules:

```
$ pip install -r requirement-modules.txt
```

Note: Previously we were suggesting use of `modules.txt` but now we are unable properly update the pypi modules in time so we suggest use of `head` of git repo. But note that this might lead to some breakage.

To enable module a line should be added to *silpa.conf*. By default all modules will be enabled if you don't want this behavior mark `no` infront of module name under `modules` section. `modules_display` section is used to display a text in the side bar of SILPA main page. Tweak it if required.

Warning: *normalizer* and *silpa_common* modules are helper modules which is required by the current modules. Do not add a line to silpa.conf for this module. Its not a web module pure python module

1.1 VirtialEnv Instructions

If you are on Mac OS X or Linux, chances are that one of the following two commands will work for you:

```
$ sudo easy_install virtualenv
```

or even better:

```
$ sudo pip install virtualenv
```

One of these will probably install virtualenv on your system. Maybe it's even in your package manager. If you use Ubuntu, try:

```
$ sudo apt-get install python-virtualenv
```

Once you have virtualenv installed, just fire up a shell-session and create your own environment.

```
$ git clone git://github.com/copyninja/Silpa-Flask.git
$ cd Silpa-Flask
$ virtualenv silpa

New python executable in silpa/bin/python
Installing distribute............done.
```

Now, whenever you want to work on a project, you only have to activate the corresponding environment. On OS X and Linux, do the following:

```
$ . silpa/bin/activate
```

If you are a Windows user, the following command is for you:

```
$ silpa\scripts\activate.bat
```

Either way, you should now be using your virtualenv (notice how the prompt of your shell-session has changed to show the active environment).

Now you can just enter the following command to get Flask activated in your virtualenv:

```
$ pip install Flask
```

A few seconds later and you are good to go.

You can start the silpa application by .. code-block:: shell-session

    python silpa.py Running on http://127.0.0.1:5000/

Well not exactly. You will see error messages saying Failed to import module xyz. That means you need to install the modules. Here is an example module installation for Soundex. Repeat this for other modules.

```
mkdir modules
cd modules
git clone git://github.com/copyninja/Soundex.git
cd Soundex
python setup.py install
```

And restart the server by just killing and running python silpa.py again.

CHAPTER 2

Silpa-Flask

Silpa-Flask is a webapp built using flask to host the silpa modules and provide a web interface for them as well as make them accessible through the JSONRPC interface.

2.1 Writing a module for Silpa-Flask

All silpa modules are implemented as classes with member functions which will be exposed through the JSONRPC interface. All the modules are required to implement a `getInstance()` method that will return an instance of the corresponding class. If you are also adding a web interface then you are required to create a templates folder in the root of your python package and create a jinja template with the name `<modulename>.html`. The template should also extend the template `silpa.html`. If this is not clear then feel free to look at the source of any of the modules.

CHAPTER 3

Modules

this is the list of the currently available modules. As of now some of them are fully functional and ready for use while others are purely experimental with limited functionality.

3.1 the modules

- fortune
- katapayadi numbers
- text similarity
- stemmer
- shingling
- sort
- guesslang
- payyans
- hyphenate
- soundex
- dictionary
- chardetails
- syllabalizer
- complex script render
- spellcheck
- indicngram
- approxsearch
- transliterate
- normalizer
- silpa_common
- indiccallendar

3.2 flask-webfonts

we have also developed a flask extension for serving and showcasing webfonts. flask-webfonts

CHAPTER 4

SILPA Webservice APIs

Warning: Althoght SILPA JSON RPC Api is fully functional and silpa-flask itself uses it, the the api for listing the available methods is not functional now. We have plans for a http based api in the near future. As of now all the python functions available in the silpa modules can be accessed through the JSON RPC interface.

4.1 Introduction

Silpa provides a set of webservice APIs over json-rpc protocol. Silpa services can be used from any programming language which has an RPC implementation. The request and response formats of APIs are in json. It is recommended to read the json-rpc documentation if you are not familiar with that.

This page explains the available usage and sample usage from python application. Implementing this in other programming languages should not be difficult and should be a matter of changing the programming language syntax.

For using the APIs you need the jsonrpc library of python. You can get this from here.

4.2 Concepts

JSON-RPC wraps an object, allowing you to call methods on that object and get the return values. It also provides a way to get error responses. The specification goes into the details (though in a vague sort of way). Here's the basics:

All access goes through a `POST` to a single URL.

The `POST` contains a `JSON` body that looks like:

```
{"method": "methodName",

"id": "arbitrary-something",
"params": [arg1, arg2, ...]}
```

The id parameter is just a convenience for the client to keep track of which response goes with which request. This makes asynchronous calls (like an XMLHttpRequest) easier. We just send the exact same id back as we get, we never look at it.

The response is `JSON`. A successful response looks like:

```
{"result": the_result,

"error": null,
"id": "arbitrary-something"}
```

The error response looks like:

```
{"result": null,
"error": {"name": "JSONRPCError",
"code": (number 100-999),
"message": "Some Error Occurred",
"error": "whatever you want\n(a traceback?)"},
"id": "arbitrary-something"}
```

It doesn't seem to indicate if an error response should have a 200 response or a 500 response. So as not to be completely stupid about HTTP, we choose a 500 resonse, as giving an error with a 200 response is irresponsible.

4.3 Usage

just send a http `POST` request to the `JSONRPC` url `dev.silpa.org.in/JSONRPC`:

```
{"method": "moduelname.methodName",

 "id": "arbitrary-something",
 "params": [arg1, arg2, ...]}
```

you will recieve a JSON reply with required result.

CHAPTER 5

Frequently Asked Questions

5.1 I am interested in contributing to this application. To whom should I contact?

Thanks. Just drop a mail to santhosh dot thottingal at gmail dot com

5.2 Can I use this application in my machine without having internet access?

Yes you can. See the installation instructions.

5.3 Can I use this application in windows?

Yes you can. See the installation instructions.

5.4 Can I host this application in my domain?

Yes you can and you are encouraged to do so. Note that you should follow the AGPL licensing terms and conditions.

5.5 Who is sponsoring the development of this project?

Nobody.

5.6 When did this project development start?

March 2009.

5.7 Is this application available in any GNU/Linux distributions?

No. But we would be happy if somebody help to get it packaged.

5.8 I found a bug in one module. How can I report?

Oops!. Just drop a mail to silpa-discuss@nongnu.org

CHAPTER 6

Credits

Many people contributed in direct and indirect way in the development of silpa. This page attempts to list their names.

- Baiju. M, Swathanthra Malayalam Computing for his mlsplit program for using it as a base for syllabalizer for many languages.
- Laxminarayan Kamath for testing and feature suggestions.
- Rajeesh Nambiar and Nishan Naseer of SMC for their contributions for Font converter.
- Guess Language module is based on the python implementation by Kent S Johnson of guesslanguage.cpp by Jacob R Rideout for KDE which itself is based on Language::Guess by Maciej Ceglowski.
- Spellcheck dictionaries for many languages are contributed by hunspell dictionaries maintained by language communities.
- The python based spellchecker is a highly customized version of basic python spellchecker by Peter Norvig.
- The python implementation of tex hyphenation algorithm is by Wilbert Berendsen, and it is based on Text::Hyphen of Ruby.
- The sort module uses Python UCA implementation The thirukkural quotes collection for Random Quote module is by Manvendra Bhangui and the original text is taken from Shakthi Kannan's collection.
- The chanakya quotes is from Chanakya Niti website and prepared by Girish Venkatachalam.
- The fontmap of ASCII Tamil font, Valluvar for encoding converter was contributed by Kevin.
- Shantanu Oak for Rs 10K donation to meet expenses of project development (March 1, 2011)
- Netdotnet.com sponsors our server! Silpa Project thanks netdotnet.com for sponsoring server for silpa.smc.org.in instance.
- SILPA Mirror instances Silpa is mirrored in the following addresses.
- Santhosh Thottingal runs a mirror instance at http://thottingal.in/silpa
- Ashik Salahudeen runs a mirror instance at http://aashiks.in/silpa

CHAPTER 7

Indices and tables

- *genindex*
- *modindex*
- *search*