coonf - Python Package for Conferences

Release 0.9.11 (beta release)

Krešimir Kumerički

27 Feb 2012

Physics Department, University of Zagreb
Email: kkumer@phy.hr

Abstract

This document describes how to use coonf - python package that can help in organization of a conference. It can handle on-line applications, submission of abstracts, creating WWW pages etc. Also, using only subset of coonf's features, one can set up WWW interface for easy insertion of any data into relational database and subsequent access to this data.

Contents

1 Introduction

Organizing a conference is usually full of various repetitive tasks which are, of course, best left to the computers. This collection of Python modules can be helpful if you want to automate some of the following tasks

- on-line application of participants and/or inserting their data into a database

- accessing and searching this data from command-line
- creating stylistically consistent WWW conference site with possibility of serving dynamical pages (e.g. listings of participants, various searches) using data from the above mentioned database (module `web`)
- on-line submission of abstracts (module `abstracts`)
- on-line uploads of files (e.g. PowerPoint presentations or proceedings contributions) (module `uploads`)

Some functionality of `coonf` is offered via WWW interface, some via command-line interface, and some by editing and executing python scripts. Some features are relegated to modules which operate independently (unless stated otherwise) and can be ignored if you don't need corresponding features. Note that these are *not* modules in the sense of Python modules/packages but just some convenient blocks into which various components of `coonf` are divided.

2 Requirements

To get everything to work you should have some basic knowledge about CGI scripting (not necessarily in Python, any CGI experience may be sufficient) and relational databases. Therefore, before even trying to put this package to work you should at least be able to:

- create and install simple CGI script ("Hello World" will do) on your machine. This can be quite involved when you do it for the first time. Consult relevant documentation and HOWTOs on the Web.
- create a simple table in your database and try inserting some data into it.

It goes without saying that you should also be able to create and install HTML pages on your server. Some basic knowledge of HTML forms is also needed, but there are example HTML forms given in package which can be used as templates.

For more advanced usage and for some of the modules, some knowledge of Python programming language is necessary. The more you know about Python and databases the more useful this package becomes.

2.1 Software Requirements

For minimal installation you need to have the following software installed (for modules 'web' and 'abstracts' some additional stuff is needed, as specified below):

- Python (http://www.python.org), at least version 2.2
- MySQL (http://www.mysql.com)
- Web server, e.g. Apache (http://www.apache.org)
- Additional Python packages:
  - MySQLdb Python database interface (http://sourceforge.net/projects/mysql-python)
  - SQLObject object-relational mapper (http://sqlobject.org)
  - eGenix.com mxDateTime extension (http://http://www.egenix.com/files/python/eGenix-mx-Extensions.html)
  - Python Cryptography Toolkit (http://www.dlitz.net/software/pycrypto/)

  E.g. on Debian Linux additional packages to be installed are python-mysqldb, python-sqlobject, python-egenix-mxdatetime and python-crypto
- If you want automatic emailing of confirmation messages to clients or administrators you also need email server (SMTP daemon) such as sendmail.

3 Installation and basic configuration of coonf

3.1 Installing coonf

You should obtain a recent source of coonf package, preferably at http://www.phy.hr/%7ekkumer/software/, unpack it, and install it by descending into directory where 'setup.py' is and giving command (probably as root)

```
python setup.py install
```

This installs coonf modules in '/usr/lib/python?.?/site-packages' or some other directory in your PYTHONPATH.

Move everything from 'HTML' and 'CGI' directories of coonf source distribution into directories where your web server can find them. In my case it goes like this:

```
shell> mv coonf-0.??/HTML /var/www/conference
shell> mv coonf-0.??/CGI /usr/lib/cgi-bin/conference
```

Python scripts in directory 'scripts' will be needed later for accomplishing various tasks. You can put them to some convenient place.

Edit 'config.py' from 'CGI' directory carefully. In principle, it is a Python source file, but the syntax is very simple and you should have no problems modifying it to suit your needs even if you don't know Python language. Comments there should lead you.

In particular, variables HTML_ROOT and CGI_ROOT should point to the directories where your have just put stuff from 'HTML' and 'DGI' directories.

This file contains database password, so it should be readable only to WWW server and people who will have administrative privileges to edit participants' data.

```
shell> chown -R www-data:confadmins /usr/lib/cgi-bin/conference
shell> chmod 660 /usr/lib/cgi-bin/conference/config.py
```

Also in the beginning of the files 'CGI_ROOT/cgi_init.py' and, if you need abstracts, 'CGI_ROOT/abstracts/pdfview.py', you have to specify absolute path to 'config.py'!

3.2 Creating database

Before proceeding, you should create a single database that will hold all the relevant data in various tables. See database documentation for how to do this. The name of this database should be specified as DB_NAME in 'config.py' as well as the username (DB_USER) and password (DB_PASSWD) needed for the connection.

For example, on MySQL, after installing MySQL software you should login as root, and then type:

```
shell> mysql --user=root mysql
mysql> create database <DB_NAME> DEFAULT CHARACTER SET utf8;
mysql> GRANT ALL PRIVILEGES ON <DB_NAME>.* TO <DB_USER>@localhost
           IDENTIFIED BY '<DB_PASSWD>';
mysql> GRANT ALL PRIVILEGES ON <DB_NAME>.* TO <DB_USER>@"%"
           IDENTIFIED BY '<DB_PASSWD>';
```

If needed, you can similarly grant access to other users.

Also, you can have more databases for e.g. more conferences. Than you just repeat the procedure and create separate databases and corresponding `HTML_ROOT` and `CGI_ROOT` direcotories, containing corresponding 'config.py'.

3.3 Configuring application form

The idea is to set up a HTML page with HTML application form so that entered data get inserted into a single row of table `People` in the database. This can be the usual data about conference participant (name, address, title of his talk etc.) but this could also be used in non-conference application to set up a simple WWW interface for inserting any kind of data into database. Configuration is done like this:

1. Create page with HTML form somewhere on your site. (You can probably just use 'publish_fancy.py' script from `web` module. In that case you should install module `web`, see Sect. 5) This HTML form should point to 'register.py' as its corresponding CGI script (`ACTION` attribute of HTML form).
2. Edit 'register.conf' (found in `CGI_ROOT` directory) and specify data that will be entered in form. Syntax is specified in file itself. Basically, to each item in HTML form there should be a matching `FIELD` item in 'register.conf'.
3. Edit 'register.py' (also found in `CGI_ROOT` directory) and edit some obvious strings there to specify content of the confirmation email and WWW page that will be returned to clients after they fill out the form.
4. Create database table `People` by single run of 'create_table_people.py' script with coonf configuration file as an argument.

```
shell> create_table_people.py /usr/lib/cgi-bin/conference/config.py
```

Now you are all set up and you should be able to make a test by pointing your WWW browser to application HTML page ('application.html' in `HTML_ROOT` by default) and submit some data to the database.

4 Accessing and modifying data from command line

Data entered via application form can be accessed via WWW with help of CGI scripts 'people.py', 'person.py' and 'search.py'. This is seen in default setup.

Access, and, additionally, editing of data is possible also via command line to users who have read permission on coonf configuration file ('config.py' in `CGI_ROOT` directory), which containts database password. You should preferably go to `CGI_ROOT` directory (or put it in your `PYTYONPATH` environment variable). Then start Python interpreter (or, better, IPython) and import coonf package, configuration data, and make this data available to coonf python module:

```
shell> python

>>> from config import *
>>> import __builtin__; __builtin__.cfg = cfg
>>> from coonf import *
```

Then connect to database and use `dbconnect()` to create a class of participants, which will be used for all access.

```
>>> People = dbconnect()
```

All data corresponding to a person with `id=5` can be printed out using command `print asciiRecord(People, 5)`, or shorter `print aR(People, 5)`

```
>>> print aR(People, 5)

================================[    5 ]================================
         name =>  albert
      surname =>  zweistein
        email =>  alby@princeton.edu
          tel =>  21342134
  institution =>  IAS
       street =>  N/A
         city =>  princeton
          zip =>  N/A
      country =>  US
         pres =>  O
    prestitle =>  Theory of everything
     banketYN =>  Y
         diet =>  N/A
================================================================================
```

Then you can change any of these attributes like this:

```
>>> People.get(5).surname = "dreistein"
```

Syntax is obvious but for people who know object-oriented programming it can be said here that each person is an instance of class `People`.

For searching you have two functions available: `asciiSearchAll()` or `aSA()` that searches all attributes, and `asciiSearchName()` or `aSN()` that searches only names and surnames. Arguments of these functions are strings (don't forget quotation marks!) and search is case-insensitive.

```
>>> asciiSearchName(People, 'ei')
============================
|  5 | albert | dreistein |
| 20 | steven |  weinberg |
============================
```

Deleting a person with `id=15` is simple:

```
>>> People.delete(15)
```

You can also do more complicated things. E.g. to find emails of all participants from Croatia:

```
>>> croats = People.select(People.q.country=='Croatia')
>>> print asciiTable(croats, 'id', 'name', 'surname', 'email')
================================================
| 1 |  prvi |      covjek |       njegov@email |
| 2 | drugi | participant |          his@email |
| 7 | again |    somebody | address@hidden.gov |
================================================
```

Of course, all this can also in principle be done by direct access to the database using SQL language.

5 Module `web` — automatic WWW site generation

This module requires python package HTMLgen. (On Debian: python-htmlgen.)

All pages are defined either in 'publish_simple.py' (simple pages that anyone with some knowledge of HTML can edit) or in 'publish_fancy.py' (sofisticated scripts that generate timetables, etc.)

First you should edit template file (`TEMPLATE_FILE` in 'config.py') and CSS file (`CSS_FILE` in 'config.py') to design general appearance of your WWW pages (and to correct URLs there!)

Then edit scripts 'publish_simple.py' and 'publish_fancy.py' to create actual pages with some content. Each page is defined by four strings as described in table

string	meaning
`path`	path to page relative to `HTML_ROOT`
`title`	title of the page
`update`	date of last update
`content`	HTML code of the main content of the page

Last three strings will be substituted in place of strings `@(TITLE)s`, `@(UPDATE)s` and `@(CONTENT)s` in `TEMPLATE_FILE`.

Running these two scripts (they take coonf configuration file as an obligatory argument) will publish pages to `HTML_ROOT`.

```
shell> publish_fancy.py /usr/lib/cgi-bin/conference/config.py
shell> publish_simple.py /usr/lib/cgi-bin/conference/config.py
```

This method of specifying all pages in a single file (or two) is convenient only for small sites with not much statical content. If you have pages with lot of content you will be better off by specifying `content` strings in external files and importing them.

6 Module `abstracts` — on-line submission of abstracts

This module requires pdflatex. (On Debian: tetex-base, tetex-bin and probably tetex-extra)

Installation:

1. Create database table for abstracts by single run of 'create_table_abstracts.py' script.

```
shell> create_table_abstracts.py /usr/lib/cgi-bin/conference/config.py
```

2. Create directory ‘abstracts/PDF’ (in `CGI_ROOT` dir) with permissions 777
3. Turn the module on by setting `MODULE_ABSTRACTS` and `EMAILING` in ‘config.py’ to 1.

Then (consult ‘register.py’) every client will get an email confirmation of application with URL of his private page where he can enter and modify his abstract. At the moment abstracts are set in LaTeXand resulting abstract is set as a nice PDF page.

7 Module `uploads` — on-line file uploads

Installation:

1. Create directory ‘uploads/FILES’ (in `CGI_ROOT` dir) with permissions 777
2. Turn the module on by setting `MODULE_UPLOADS` and `EMAILING` in ‘config.py’ to 1.

Then (consult ‘register.py’) every client will get an email confirmation of application with URL of his private page where he can upload file of his electronic presentation. Files will end up in ‘CGI_ROOT/uploads/FILES’ and they will have name of the form `<id>-<surname>-v<number>-<original filename>` where `<number>` is version number incremented by each submission of file with same name by the same person. Thus no files are overwritten.