Debian Member Portfolio Service Documentation

Release 0.3.1

Jan Dittberner

February 08, 2014

Contents

The Debian Member Portfolio Service is a web application that provides links to information regarding the activities of a person related to the Debian Project.

The service was originally implemented and is hosted by Jan Dittberner at http://portfolio.debian.net/.

CHAPTER 1

Development of Debian Member Portfolio Service

The Debian Member Portfolio Service is implemented in Python using the Pylons web application framework.

The following sections describe how to setup a local development environment for the Debian Member Portfolio Service.

All instructions assume that you work on a Debian system. You should use Python 2.7 for development.

1.1 Setup of a local development

To start working on the source code you need to have git installed:

```
sudo aptitude install git
```

The canonical git repository for the Debian Member Portfolio Service is available at http://debianstuff.dittberner.info/git/debianmemberportfolio.git. To get a clone of the source code you change to a directory of your choice and invoke git clone:

```
cd ~/src
git clone http://debianstuff.dittberner.info/git/debianmemberportfolio.git
```

You should use virtualenv to separate the development environment from your system wide Python installation. You can install virtualenv using:

```
sudo aptitude install python-virtualenv
```

When you have **virtualenv** installed you should create a virtual environment for Debian Member Portfolio Service development and install the requirements using pip:

```
mkdir ~/.virtualenvs
virtualenv --distribute ~/.virtualenvs/dmportfolio
. ~/.virtualenvs/dmportfolio/bin/activate
cd ~/src/debianmemberportfolio
pip install -r wheezyreq.pip
```

Note: The Debian Member Portfolio Service instance at http://portfolio.debian.net/ is running on a Debian Wheezy server, therefore `wheezyreq.pip` contains dependency versions matching that Debian release.

The dependency download and installation into the virtual environment takes some time.

After you have your virtual environment ready you need to setup the project for development:

```
python setup.py develop
```

Debian Member Portfolio Service needs the JQuery JavaScript library to function properly. The JQuery library is not included in the git clone and must be copied into the subdirectory `debianmemberportfolio/public/javascript/jquery`. On Debian systems you can install the package libjs-jquery and place a symlink to the directory `/usr/share/javascript` into `debianmemberportfolio/public`:

```
sudo aptitude install libjs-jquery
ln -s /usr/share/javascript debianmemberportfolio/public
```

1.1.1 Prepare for first startup

The Debian Member Portfolio Service uses data from the Debian keyring to get information regarding PGP keys and names related to email addresses. Before you can run the service you need to fetch a copy of the keyring and prepare it for use by the code.

Note: You need rsync and gnupg for these tasks:

```
sudo aptitude install rsync gnupg
```

When you have both installed you can run:

```
. ~/.virtualenvs/dmportfolio/bin/activate
./synckeyrings.sh
python debianmemberportfolio/model/keyringanalyzer.py
```

The first synchronizes the keyrings in `$HOME/debian/keyring.debian.org` with files on the keyring.debian.org host. And the second generates a key/value database in `debianmemberportfolio/model/keyringcache` that is used by the code.

1.1.2 Run a development server

Pylons uses PasteScript to run a development server. You can run a development server using:

```
paster serve --reload development.ini
```

The output of this command should look like the following:

```
Starting subprocess with file monitor
Starting server in PID 31377.
serving on http://127.0.0.1:5000
```

You can now access your development server at the URL that is printed by the command.

If you want to stop the development server press `Ctrl + C`.

1.2 Common development tasks

1.2.1 Add new URL

Debian Member Portfolio Service uses a ini style configuration file `debianmemberportfolio/model/portfolio.ini` to configure the generated URL patterns. The actual URL generation is done in `DdportfolioController` in the

`urllist()` method.

If you want to add a new URL type you have to add a line in `portfolio.ini` and an entry in `DdportfolioController`'s `_LABELS` dictionary. The top level dictionary keys correspond to sections in the ini file. The dictionary values are dictionaries themselves that contain a special key `label` that defines the label of the section in the output and keys for each entry to be rendered in that section. The values in these sub-dictionaries are strings marked for translation using the `_()` function from `pylons.i18n`.

The patterns in `portfolio.ini` can contain the following placeholders that are filled at runtime:

Placeholder	Replacement
%(aliothusername)s	user name on alioth.debian.org
%(email)s	email address (URL encoded)
%(emailnoq)s	email address
%(firstchar)s	first character of the email address
%(forumsid)d	forum user id
%(gpgfp)s	GNUPG/PGP key fingerprint
%(name)s	full name (i.e. John Smith)
%(username)s	Debian user name
%(wikihomepage)s	full name in camel case (i.e. JohnSmith)

The replacement of placeholders is performed in the `urllist()` method. And uses data from the Debian keyring. Access to the pre-parsed keyring data is performed using the `build_data()` function of the module `debianmemberportfolio.model.dddatabuilder`, which uses several helper functions from `debianmemberportfolio.model.keyfinder` to access the key information.

CHAPTER 2

Source documentation

The sections below contain mostly autogenerated documentation of the source code of the Debian Member Portfolio Service.

2.1 Controllers

2.1.1 portfolio controller

2.1.2 error controller

class debianmemberportfolio.controllers.error.**ErrorController**
Generates error documents as and when they are required.

The ErrorDocuments middleware forwards to ErrorController when error related status codes are returned from the application.

This behaviour can be altered by changing the parameters to the ErrorDocuments middleware in your config/middleware.py file.

document()
Render the error document

img(*id*)
Serve Pylons' stock images

style(*id*)
Serve Pylons' stock stylesheets

2.1.3 showformscripts controller

This file defines the ShowformscriptsController used to generate the JavaScript code in forms.

class debianmemberportfolio.controllers.showformscripts.**ShowformscriptsController**
This controller is used to support data entry in showform.

It provides code for generating JavaScript as well as JSON responses for autocompletion of fields.

fetchdddata()
This action fetches the data for a given mail address and returns them as JSON.

index()
This action generates the helper script for the showform page.

2.1.4 template controller

This file contains the TemplateController used to render templates.

2.2 Library code

2.2.1 app_globals

The application's Globals object

class debianmemberportfolio.lib.app_globals.**Globals**(*config*)
Globals acts as a container for objects available throughout the life of the application

2.2.2 base

The base Controller API

Provides the BaseController class for subclassing.

debianmemberportfolio.lib.base.**render**(*template_name*, *extra_vars=None*, *cache_key=None*, *cache_type=None*, *cache_expire=None*)
Render a template with Mako

Accepts the cache options cache_key, cache_type, and cache_expire.

2.2.3 helpers

Helper functions

Consists of functions to typically be used within templates, but also available to Controllers. This module is available to templates as 'h'.

2.3 Model

Model classes and model related utilities for the Debian Member Portfolio service.

2.3.1 dddatabuilder

This file contains code to build a representation of a person based on keyring data associated to a given email address.

debianmemberportfolio.model.dddatabuilder.**build_data**(*email_address*)
Build a DD data structure from a given email address.

2.3.2 form

This file contains the form definitions used in the controllers.

class debianmemberportfolio.model.form.**DDDataRequest**(**args*, ***kw*)
Validation schema for DDData request.

Messages

badDictType: The input must be dict-like (not a `%(type)s`: `%(value)r`)

badType: The input must be a string (not a `%(type)s`: `%(value)r`)

empty: Please enter a value

missingValue: Missing value

noneType: The input must be a string (not None)

notExpected: The input field `%(name)s` was not expected.

class `debianmemberportfolio.model.form.`**DeveloperData**(**args*, ***kw*)
Validation schema for DeveloperData.

Messages

badDictType: The input must be dict-like (not a `%(type)s`: `%(value)r`)

badType: The input must be a string (not a `%(type)s`: `%(value)r`)

empty: Please enter a value

missingValue: Missing value

noneType: The input must be a string (not None)

notExpected: The input field `%(name)s` was not expected.

2.3.3 keyfinder

This module provides tools for finding PGP key information from a given keyring.

`debianmemberportfolio.model.keyfinder.`**getFingerprintByEmail**(*email*)
Gets the fingerprints associated with the given email address if available.

`debianmemberportfolio.model.keyfinder.`**getLoginByEmail**(*email*)
Gets the logins associated with the given email address if available.

`debianmemberportfolio.model.keyfinder.`**getLoginByFingerprint**(*fpr*)
Gets the login associated with the given fingerprint if available.

`debianmemberportfolio.model.keyfinder.`**getRealnameByEmail**(*email*)
Gets the real names associated with the given email address if available.

2.3.4 keyringanalyzer

This is a tool that analyzes GPG and PGP keyrings and stores the retrieved data in a file database. The tool was inspired by Debian qa's carnivore.

`debianmemberportfolio.model.keyringanalyzer.`**process_gpg_list_keys_line**(*line*, *fpr*)
Process a line of gpg –list-keys –with-colon output.

`debianmemberportfolio.model.keyringanalyzer.`**process_keyrings**()
Process the keyrings and store the extracted data in an anydbm file.

2.3.5 urlbuilder

CHAPTER 3

Credits

The Debian Member Portfolio Service contains contributions from several people.

3.1 Code

- Jan Dittberner <jandd at debian dot org>
- Paul Wise <pabs at debian dot org>
- Olivier Berger <olivier.berger at telecom-sudparis dot eu>

3.2 Translations

- Jan Dittberner
- Daniel Manzano (Brazilian Portuguese)
- Izharul Haq (Indonesian)
- Stéphane Aulery (French)

If you think your name is missing please tell me (Jan Dittberner) about your contribution and I'll add you.

CHAPTER 4

Indices and tables

- *genindex*
- *search*

d