Simple Lottery 0.4.0

Simple Lottery Handbook

User Handbook for Simple Lottery

Publican

BOOK PUBLISHING TOOL

Sascha Peter

Simple Lottery 0.4.0 Simple Lottery Handbook
User Handbook for Simple Lottery
Edition 0

Author Sascha Peter sascha.o.peter@gmail.com

This documentation is written to give an overview over the Simple Lottery system, it's functionalities and guidance through the installation and setup process.

Preface

1. Document Conventions

This manual uses several conventions to highlight certain words and phrases and draw attention to specific pieces of information.

In PDF and paper editions, this manual uses typefaces drawn from the *Liberation Fonts*[1] set. The Liberation Fonts set is also used in HTML editions if the set is installed on your system. If not, alternative but equivalent typefaces are displayed. Note: Red Hat Enterprise Linux 5 and later include the Liberation Fonts set by default.

1.1. Typographic Conventions

Four typographic conventions are used to call attention to specific words and phrases. These conventions, and the circumstances they apply to, are as follows.

`Mono-spaced Bold`

Used to highlight system input, including shell commands, file names and paths. Also used to highlight keys and key combinations. For example:

> To see the contents of the file **`my_next_bestselling_novel`** in your current working directory, enter the **`cat my_next_bestselling_novel`** command at the shell prompt and press **`Enter`** to execute the command.

The above includes a file name, a shell command and a key, all presented in mono-spaced bold and all distinguishable thanks to context.

Key combinations can be distinguished from an individual key by the plus sign that connects each part of a key combination. For example:

> Press **`Enter`** to execute the command.
>
> Press **`Ctrl`**+**`Alt`**+**`F2`** to switch to a virtual terminal.

The first example highlights a particular key to press. The second example highlights a key combination: a set of three keys pressed simultaneously.

If source code is discussed, class names, methods, functions, variable names and returned values mentioned within a paragraph will be presented as above, in **`mono-spaced bold`**. For example:

> File-related classes include **`filesystem`** for file systems, **`file`** for files, and **`dir`** for directories. Each class has its own associated set of permissions.

Proportional Bold

This denotes words or phrases encountered on a system, including application names; dialog box text; labeled buttons; check-box and radio button labels; menu titles and sub-menu titles. For example:

> Choose **System** → **Preferences** → **Mouse** from the main menu bar to launch **Mouse Preferences**. In the **Buttons** tab, select the **Left-handed mouse** check box and click

[1] https://fedorahosted.org/liberation-fonts/

> **Close** to switch the primary mouse button from the left to the right (making the mouse suitable for use in the left hand).
>
> To insert a special character into a **gedit** file, choose **Applications** → **Accessories** → **Character Map** from the main menu bar. Next, choose **Search** → **Find…** from the **Character Map** menu bar, type the name of the character in the **Search** field and click **Next**. The character you sought will be highlighted in the **Character Table**. Double-click this highlighted character to place it in the **Text to copy** field and then click the **Copy** button. Now switch back to your document and choose **Edit** → **Paste** from the **gedit** menu bar.

The above text includes application names; system-wide menu names and items; application-specific menu names; and buttons and text found within a GUI interface, all presented in proportional bold and all distinguishable by context.

Mono-spaced Bold Italic or ***Proportional Bold Italic***

Whether mono-spaced bold or proportional bold, the addition of italics indicates replaceable or variable text. Italics denotes text you do not input literally or displayed text that changes depending on circumstance. For example:

> To connect to a remote machine using ssh, type **ssh *username*@*domain.name*** at a shell prompt. If the remote machine is **example.com** and your username on that machine is john, type **ssh john@example.com**.
>
> The **mount -o remount *file-system*** command remounts the named file system. For example, to remount the **/home** file system, the command is **mount -o remount /home**.
>
> To see the version of a currently installed package, use the **rpm -q *package*** command. It will return a result as follows: ***package-version-release***.

Note the words in bold italics above — username, domain.name, file-system, package, version and release. Each word is a placeholder, either for text you enter when issuing a command or for text displayed by the system.

Aside from standard usage for presenting the title of a work, italics denotes the first use of a new and important term. For example:

> Publican is a *DocBook* publishing system.

1.2. Pull-quote Conventions

Terminal output and source code listings are set off visually from the surrounding text.

Output sent to a terminal is set in **mono-spaced roman** and presented thus:

```
books        Desktop   documentation  drafts  mss    photos   stuff  svn
books_tests  Desktop1  downloads      images  notes  scripts  svgs
```

Source-code listings are also set in **mono-spaced roman** but add syntax highlighting as follows:

```
package org.jboss.book.jca.ex1;

import javax.naming.InitialContext;
```

```
public class ExClient
{
   public static void main(String args[])
       throws Exception
   {
      InitialContext iniCtx = new InitialContext();
      Object         ref    = iniCtx.lookup("EchoBean");
      EchoHome       home   = (EchoHome) ref;
      Echo           echo   = home.create();

      System.out.println("Created Echo");

      System.out.println("Echo.echo('Hello') = " + echo.echo("Hello"));
   }
}
```

1.3. Notes and Warnings

Finally, we use three visual styles to draw attention to information that might otherwise be overlooked.

Note

Notes are tips, shortcuts or alternative approaches to the task at hand. Ignoring a note should have no negative consequences, but you might miss out on a trick that makes your life easier.

Important

Important boxes detail things that are easily missed: configuration changes that only apply to the current session, or services that need restarting before an update will apply. Ignoring a box labeled 'Important' will not cause data loss but may cause irritation and frustration.

Warning

Warnings should not be ignored. Ignoring warnings will most likely cause data loss.

2. We Need Feedback!

Whilst this document has been created with most care, potential errors and typos may happen. We appreciate your feedback and are happy to receive it via email to sascha.o.peter@gmail.com

Requirements

Simple Lottery is a web application written in Python, using the Django web framework.

1.1. Software Requirements

In order to run this project, you will need the following software installed:

1. Python 3.3.2
2. PIP
3. Virtualenv

The web application can be run on any Operating System supporting those requirements

1.2. PIP package requirements

Additional requirements which need to be installed through pip:

1. Django 1.8.5
2. django-localflavor 1.1
3. psycopg2 2.6.1

This additional requirements can be found in the pip-requirements.txt file in the main project folder

Installation

The following sections will guide you through the setup and installation of the software on your device.

2.1. Setup Python 3 on MAC

Simple-Shop has been built and developed in Python 3.3.2 and Django 1.8. One of the easiest ways to set up python3, or generally multiple versions of python on your environment is through *https://github.com/yyuu/pyenv*.

The following commands are a quick cheat sheet for you if you installed everything correctly with pyenv.

To install a version **`pyenv install version.number`**

To set it/switch to it **`pyenv global version.number`**

Then to check the set version **`pyenv version`**

And the overall system version **`python --version`**

Python 3.3.2 required!

The software has been tested with Python 3.3.2 and Django 1.8. Please make sure you have the right version installed.

2.2. Setup Python 3 on CentOS

Simple Lottery has been built and tested in Python 3.3.2 on CentOS 7. Since CentOS 7 only ships with Python 2.7, an alternative Python installation is needed. The easiest way to do so on a RHEL based system are Software collections *https://www.softwarecollections.org/en/docs/*. You are free to use any other way to setup Python 3.3.2 on your device.

Please follow the instructions of the software collections documentation closely. It only takes a few steps to set it up.

After successfull installation, please test your python version with: **`python --version`**

Python 3.3.2 required!

The software has been tested with Python 3.3.2 and Django 1.8. Please make sure you have the right version installed.

2.3. Setup Virtual Environment

When working with different Python versions or different projects, it's suggested to use separate environments for each project to handle their requirements.

For MAC users: One way to achieve this is to use virtualenv. Pyenv has a plugin for virtualenv *https://github.com/yyuu/pyenv-virtualenv*. But you are free to use any virtualenv solution you prefer.

Make sure you activate your virtual environment!

Depending on your operating system and virtualenv solution you went for, the command to activate the virtualenv may vary. With pyenv-virtualenv the command would be **`pyenv activate your-virtual-env`**

2.4. Install project dependencies

When working with different Python versions or different projects, it's suggested to use separate environments for each project to handle their requirements.

After activating the virtual environment for your simple_lottery project, you can install all project requirements through PIP. All project dependencies are listed in the pip-requirements.txt file in the root of the project.

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

Configuration

The following sections will guide you through the configuration and setup of the website project itself and how to locally launch it in your pre-prepared environment of the previous chapter.

3.1. Configure the project

The root folder of the project will present you with the following folder hiearchy:

- Simple_Lottery_Handbook/
- lottery/
- media/
- simple_lottery/
- static/
- usermanagement/
- **`.coverage`**
- **`.gitignore`**
- **`README.md`**
- **`circle.yml`**
- **`manage.py`**
- **`pip-requirements.txt`**

Please change into simple_lottery and create a copy of **`settings_local.template`** with the name **`settings_local.py`**.

```
cd simple_lottery/
cp settings_local.template settings_local.py
vi settings_local.py
```

Editor of choice

The last line with **`vi settings_local.py`** is just an example. Please use your editor of choice for editing the python file.

Most variables in the settings_local.py will already be as they are needed.

SECRET_KEY is required for the project to run!

SECRET_KEY is used in many security related places within the django framework and should never be shared nor exposed. Here is an example of a secret key (don't use for productive use): 6*kpa+0m+%k(d-$_xbh#b7zch1_9ox0!35a^@q4fv=i*p!4*to

3.2. Create the database

Change into the root folder of simple_lottery if you have not already and execute **`python manage.py migrate`**. This will setup the database hierarchy and tables.

After creating the database, you'll need to execute **`python manage.py createsuperuser`** to create an admin user. The information asked will consist out of *`username`*, *`email address`*, *`password`*.

Make sure your virtual environment is active!

In order to execute the manage.py script you'll have to have your environment active with the right python version, otherwise you'll encounter potential issues / errors.

3.3. Start the webapplication

When you finished the configuration, please execute **`python manage.py runserver`** and head over to *http://localhost:8000* in your browser of choice.

The admin panel is accessible under *http://localhost:8000/admin/*.

Features

Simple Lottery is a web application written in Python, using the Django web framework.

4.1. Lotteries

Simple Lottery provides an easy to use lottery management system which is accessible through the admin panel under /admin/.

Every Lottery can have a Title, StartDate, EndDate, StartTime, EndTime and a maximum amount of entries. By providing the maximum amount of entries, the system will automatically create all entries and define one entry as a winning entry ticket.

Users joining a lottery will randomly be assigned to one entry, which either will fail or win.

A list of all lotteries and winners can be found under /admin/lottery_winner/ - a corresponding link is available in the right-top corner of the admin interface near your username and other navigational links.

4.2. User Management

Users have to be registered in order to join a lottery. There are a hand full required fields, including their address, in order to sign up as a new user on the web application.

Users can select which lottery they would like to enter and will be informed about which ones they already entered.

The admin user can manage all users and their profile information through the admin interface in the user management module.

4.3. Responsive Webdesign

Whilst simplistic, the user interface has been written in Twitter's Bootstraps. Twitter Bootstrap is a framework which enables responsive websites. The website uses a few basic elements including the grid layout in order to enable a responsive website which works across all screen sizes.

Future

Simple Lottery, whilst functional, is not a finished product or web application. Many functions are rough around their edges, need optimization or improvements.

5.1. Potential Plans

The following points represent a small list of ideas and aspects which could be done when more time and resources get invested into this web application to make it a production ready product.

To make this lottery system ready, I would:

1. Add more unit tests to all modules
2. Turn email into a required field
3. Add more inline documentation
4. Expand the frontend and make it more pretty with custom CSS (LessCSS/SaSS)
5. Provide the user with an option to have a list of entered lotteries in the past
6. Provide either a management command to send out an email to winners after the lottery finishes or give the user an on-page solution where he can see which lotteries he has won after they finished
7. Provide filters or an overall better admin view for the winner list

Appendix A. Revision History

Revision 0.0-0 Tue Oct 6 2015 **Sascha Peter** *sascha.o.peter@gmail.com*

Initial creation by publican

Index