django-lb-workflow Documentation

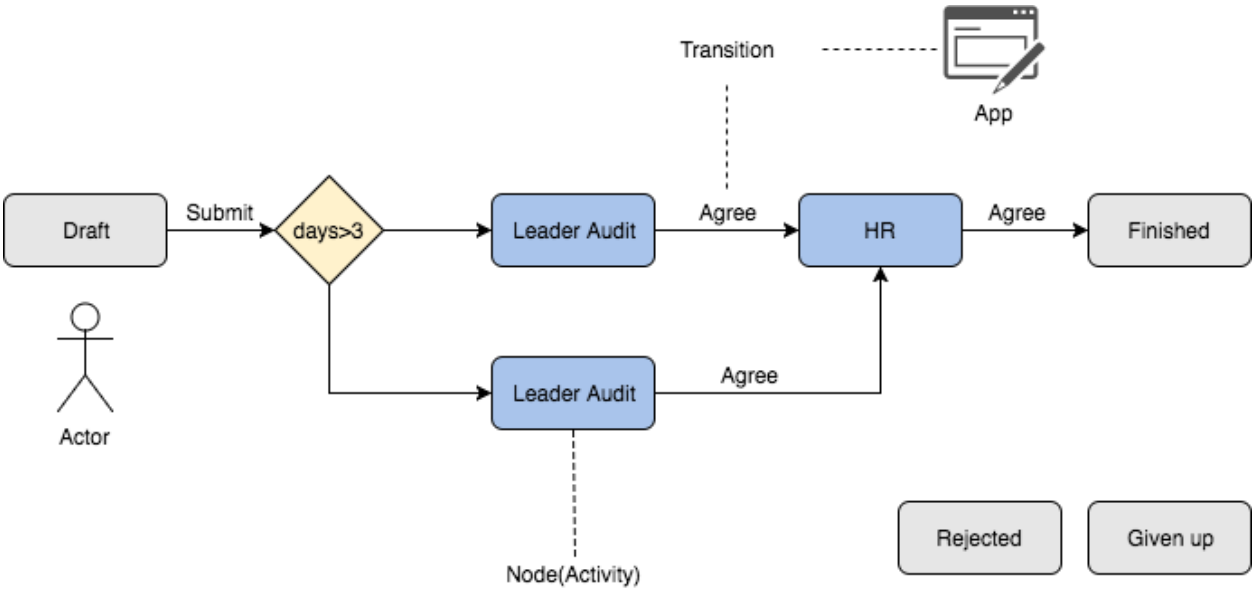
vicalloy

Aug 15, 2018

Contents

`django-lb-workflow` is a reusable workflow library for Django.

django-lb-workflow's source code hosted on GitHub.



CHAPTER 1

Demo site

Demo site: http://wf.haoluobo.com/

username: `admin` password: `password`

Switch to another user: http://wf.haoluobo.com/impersonate/search

Stop switch: http://wf.haoluobo.com/impersonate/stop

CHAPTER 2

Contents

2.1 Installation

2.1.1 Requirements

- python>=3.4
- django>=1.10
- jsonfield>=1.0.1
- pygraphviz>=1.3
- xlsxwriter>=0.9.6
- jinja2>=2.9.6
- django-lbutils>=1.0.3
- django-lbattachment>=1.0.2
- django-stronghold

The following packages are optional:

- django-compressor>=2.1.1
- django-bower>=5.2.0
- django-crispy-forms>=1.6
- django-lb-adminlte>=0.9.4
- django-el-pagination>=3.0.1
- django-impersonate

2.1.2 Installing django-lb-workflow

Install latest stable version into your python path using pip or easy_install:

```
pip install --upgrade django-lb-workflow
```

If you want to install `django-lb-workflow` with all option requires:

```
pip install --upgrade django-lb-workflow[options]
```

If you want to install development version (unstable), you can do so doing:

```
pip install git+git://github.com/vicalloy/django-lb-workflow.git#egg=django-lb-
↪workflow
```

Or, if you'd like to install the development version as a git repository (so you can `git pull` updates, use the `-e` flag with `pip install`, like so:

```
pip install -e git+git://github.com/vicalloy/django-lb-workflow.git#egg=django-lb-
↪workflow
```

Add `lbworkflow` to your `INSTALLED_APPS` in settings.py:

```
INSTALLED_APPS = (
    ...
    'lbworkflow',
)
```

Add `lbworkflow.urls` to you `url`:

```
urlpatterns = [
    ...
    url(r'^wf/', include('lbworkflow.urls')),  # url for lbworkflow
    url(r'^attachment/', include('lbattachment.urls')),  # url for lbattachment
]
```

Others: You should also config other required APPS, ex: `django-el-pagination`.

2.1.3 Sample code of using django-lb-workflow

You can find sample code of using django-lb-workflow in `testproject/` and `lbworkflow/tests/`.

2.2 Example

Throughout this tutorial, we'll walk you through the creation of a basic project and a issue process using default template.

2.2.1 Sample project

You can find sample code of using `django-lb-workflow` in `testproject/` and `lbworkflow/tests/`.

2.2.2 Start a new project and config it

Install `django-lb-workflow` with all option requires:

```
pip install --upgrade django-lb-workflow[options]
```

Creating a project:

```
$ django-admin.py startproject helloword
```

Add the following code in the file `settings.py`:

```
INSTALLED_APPS = [
    ...
    'helloword',

    'crispy_forms',
    'lbattachment',
    'lbadminlte',
    'lbutils',
    'compressor',
    'djangobower',
    'el_pagination',

    'stronghold',

    'lbworkflow',
]

MIDDLEWARE += [
    'stronghold.middleware.LoginRequiredMiddleware',
]

CRISPY_TEMPLATE_PACK = 'bootstrap3'

LBWF_APPS = {
}

STATICFILES_FINDERS = [
    'django.contrib.staticfiles.finders.FileSystemFinder',
    'django.contrib.staticfiles.finders.AppDirectoriesFinder',
]

# bower
STATICFILES_FINDERS += (('djangobower.finders.BowerFinder'),)
BOWER_COMPONENTS_ROOT = BASE_DIR

BOWER_INSTALLED_APPS = (
    'admin-lte#2.3.11',
    'font-awesome#4.7.0',
    'ionicons#2.0.1',

    'modernizr',
    # POLYFILLS: javascript fallback solutions for older browsers.
    # CSS3 selectors for IE 6-8.
    'selectivizr',
    # min/max width media queries for IE 6-8.
    'respond',
```

(continues on next page)

(continued from previous page)

```
    # CSS3 styles for IE 6-8.
    'pie',
    # HTML5 tag support for IE 6-8.
    'html5shiv',

    'masonry#4.1.1',
    'blueimp-file-upload#9.12.5',
    'flatpickr-calendar#2.5.6',
)

# django-compressor
STATICFILES_FINDERS += (('compressor.finders.CompressorFinder'),)
COMPRESS_PRECOMPILERS = (
    ('text/coffeescript', 'coffee --compile --stdio'),
    ('text/less', 'lessc {infile} {outfile}'),
    ('text/x-sass', 'sass {infile} {outfile}'),
    ('text/x-scss', 'sass --scss {infile} {outfile}'),
)

PROJECT_TITLE = 'LB-Workflow'

LOGIN_URL = '/admin/login/'
LOGOUT_URL = '/admin/logout/'

MEDIA_ROOT = os.path.join(BASE_DIR, 'media')
MEDIA_URL_ = '/media/'
MEDIA_URL = MEDIA_URL_

STATIC_ROOT = os.path.join(BASE_DIR, 'collectedstatic')

STRONGHOLD_PUBLIC_URLS = [
    r'^/admin/',
]
```

Edit the file urls.py:

```
from django.conf.urls import include
from django.conf.urls import url
from django.contrib import admin
from django.views.generic import RedirectView

urlpatterns = [
    url(r'^$', RedirectView.as_view(url='/wf/list/'), name='home'),
    url(r'^admin/', admin.site.urls),
    url(r'^wf/', include('lbworkflow.urls')),
    url(r'^attachment/', include('lbattachment.urls')),
]
```

Create base templates for project.

`helloword/templates/base.html`:

```
{% extends "lbadminlte/base.html" %}

{% load staticfiles %}

{% block head_ext %}
```

(continues on next page)

(continued from previous page)

```
  <link href="{% static '/css/lbworkflow.css' %}" rel="stylesheet" type="text/css" />
{% endblock %}

{% block footer_ext %}
  <script src="{% static 'js/lbworkflow.js' %}" type="text/javascript"></script>
  <script type="text/javascript">
    URL_UPLOAD_ATTACH = "{% url 'lbattachment_upload__' %}";
  </script>
{% endblock %}
```

helloword/templates/base_ext.html:

```
{% extends "lbadminlte/base_ext.html" %}

{% block left_side %}
  <section class="sidebar">
    <ul class="sidebar-menu">
      <li id="id-nav-todo">
        <a href="{% url 'wf_todo' %}">
          <i class="fa fa-th"></i> Todo
          <small class="badge pull-right bg-red todo-count hide"></small>
        </a>
      </li>
      <li id="id-nav-mywf">
        <a href="{% url 'wf_my_wf' %}">
          <i class="fa fa-th"></i> My
        </a>
      </li>
      <li id="id-nav-start-wf">
        <a href="{% url 'wf_start_wf' %}">
          <i class="fa fa-th"></i> Submit
        </a>
      </li>
      <li id="id-nav-list-wf">
        <a href="{% url 'wf_list_wf' %}">
          <i class="fa fa-th"></i> All
        </a>
      </li>
      <li id="id-nav-report-list">
        <a href="{% url 'wf_report_list' %}">
          <i class="fa fa-th"></i> Report list
        </a>
      </li>
    </ul>
  </section>
{% endblock %}
```

helloword/templates/base_form.html:

```
{% extends "lbadminlte/base_form.html" %}
```

Install required static package:

```
$ cd helloword
$ python manager bower install
```

run the following command to create database and create two superuser `admin` and `vicalloy`:

```
$ python manage.py migrate
$ python manage.py createsuperuser
$ python manage.py createsuperuser
```

Start the development server:

```
$ python manage.py runserver
```

Now, open a Web browser and go to "/" on your local domain – e.g., http://127.0.0.1:8000/ . You should see the admin's login screen. After login you can see the home page of this project.

2.2.3 Start a new flow

Create app and generate base code

Creating the issue app:

```
$ python manage.py startapp issue
```

Add `issue` to `INSTALLED_APPS` in `settings.py`:

```
INSTALLED_APPS = [
    ...
    'issue',
]
```

Creating models:

```
from django.db import models

from lbworkflow.models import BaseWFObj


class Issue(BaseWFObj):
    title = models.CharField('Title', max_length=255)
    summary = models.CharField('Summary', max_length=255)
    content = models.TextField('Content', blank=True)

    def __str__(self):
        return self.title
```

`python manager.py shell` to open django shell, and run the following code to generate skeleton code:

```
>>> from lbworkflow.flowgen import FlowAppGenerator
>>> from issue.models import Issue as wf_class
>>> FlowAppGenerator().gen(wf_class)
```

run the following command to update database:

```
$ python manage.py makemigrations issue
$ python manage.py migrate
```

Config flow

You can config flow in django admin or create a python file and execute it. Config the flow by code `issue/wfdata.py`:

```
from lbworkflow.core.datahelper import create_node
from lbworkflow.core.datahelper import create_category
from lbworkflow.core.datahelper import create_process
from lbworkflow.core.datahelper import create_transition


def load_data():
    load_issue()


def load_issue():
    """ load_[wf_code] """
    category = create_category('5f31d065-00cc-0020-beea-641f0a670010', 'HR')
    process = create_process('issue', 'Issue', category=category)
    create_node('5f31d065-00a0-0020-beea-641f0a670010', process, 'Draft', status=
↪'draft')
    create_node('5f31d065-00a0-0020-beea-641f0a670020', process, 'Given up', status=
↪'given up')
    create_node('5f31d065-00a0-0020-beea-641f0a670030', process, 'Rejected', status=
↪'rejected')
    create_node('5f31d065-00a0-0020-beea-641f0a670040', process, 'Completed', status=
↪'completed')
    create_node('5f31d065-00a0-0020-beea-641f0a670050', process, 'A1', operators=
↪'[vicalloy]')
    create_transition('5f31d065-00e0-0020-beea-641f0a670010', process, 'Draft,', 'A1')
    create_transition('5f31d065-00e0-0020-beea-641f0a670020', process, 'A1,',
↪'Completed')
```

Add the following code in the file settings.py:

```
LBWF_APPS = {
    'issue': 'issue',
}
```

run the following command to load flow config to database:

```
$ python manage.py callfunc lbworkflow.wfdata.load_data
$ python manage.py callfunc issue.wfdata.load_data
```

2.2.4 Submit and audit

Now we can start the server and submit a issue. We also can audit the issue.

Start the development server:

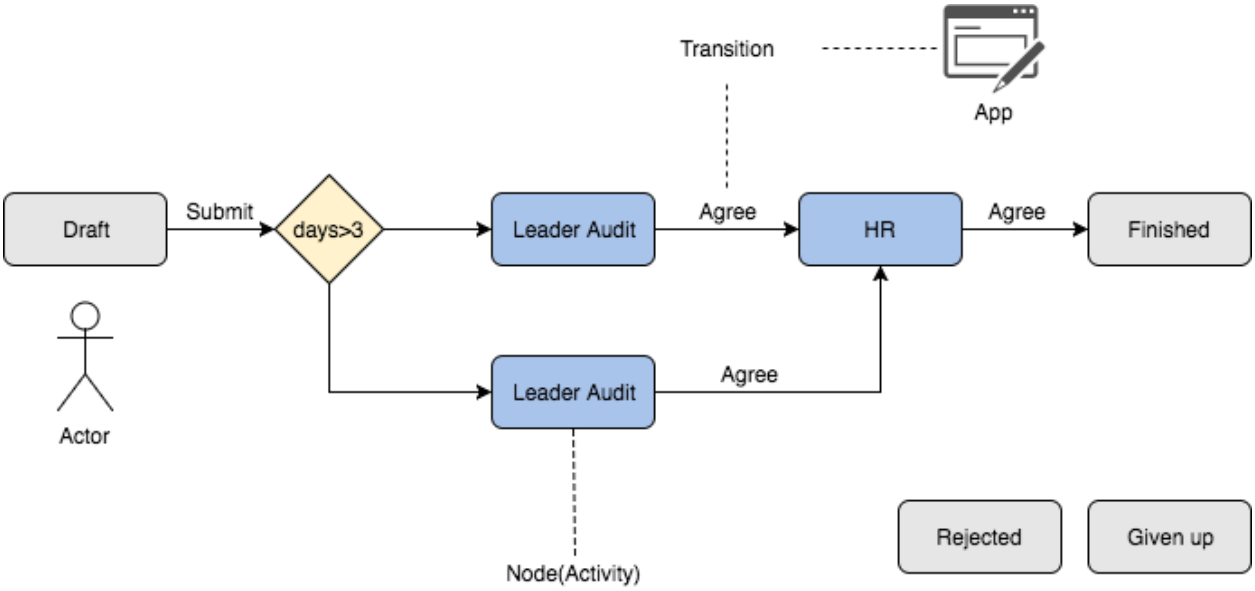
```
$ python manage.py runserver
```

- **Left menu**
  - `Todo` All task need you todo
  - `My` All process you submitted

- `Submit` Submit a new process
- `All` You process that you can see
- `Report list` Report list

2.3 Core concepts

`django-lb-workflow` is `Activity-Based Workflow`. Activity-based workflow systems have workflow processes comprised of activities to be completed in order to accomplish a goal.



2.3.1 Half Config

`django-lb-workflow` is `half config`.

- **`Data model/action/Layout of form` is written by code.**
  - They are too complex to config and the change is not too often.
- **The node(activity) and transition is configurable.**
  - The pattern is clear and the change is often.

2.3.2 Data model

Config

Process

A process holds the map that describes the flow of work.

The process map is made of nodes and transitions. The instances you create on the map will begin the flow in the draft node. Instances can be moved forward from node to node, going through transitions, until they reach the end node.

Node

Node is the states of an instance.

Transition

A Transition connects two node: a From and a To activity.

Since the transition is oriented you can think at it as being a link starting from the From and ending in the To node. Linking the nodes in your process you will be able to draw the map.

Each transition can have a condition that will be tested whether this transition is available.

Each transition is associated to a app that define an action to perform.

App

An application is a python view that can be called by URL.

Runtime

ProcessInstance

A process instance is created when someone decides to do something, and doing this thing means start using a process defined in `django-lb-workflow`. That's why it is called "process instance". The process is a class (=the definition of the process), and each time you want to "do what is defined in this process", that means you want to create an INSTANCE of this process.

So from this point of view, an instance represents your dynamic part of a process. While the process definition contains the map of the workflow, the instance stores your usage, your history, your state of this process.

Task

A task object represents a task you are performing.

Event

A task perform log.

BaseWFObj

A abstract class for flow model. Every flow model should inherit from it.

2.3.3 User Parser

`django-lb-workflow` use a text field to config users for `Node` and user a parser to cover it to Django model. The default parser is `lbworkflow.core.userparser.SimpleUserParser`. You can replace it with your implement.

2.3.4 Views and Forms

`django-lb-workflow` provide a set of views and forms to customized flow.

Classes for create/edit/list process instance is in `lbworkflow/views/generics.py`.

Classes for customize transition is in `lbworkflow/views/transition.py`.

Classes for customize form is in `lbworkflow/views/forms.py`.

url provide by `django-lb-workflow`

you can find all url in `lbworkflow/urls.py`

- **Main entrance.**
  - `wf_todo` List tasks that need current user to process.
  - `wf_my_wf` List processes that current user submitted.
  - `wf_start_wf` List the processes that current user can submit.
  - `wf_report_list` Each process have a default report. This url will list all report link.
- **Flow**
  - `wf_new [wf_code]` Submit a new process. `wf_code` used to specify which process to submit.
  - `wf_edit [pk]` Edit a process.
  - `wf_delete` Delete a process.
  - `wf_list [wf_code]` Default report for a process. `wf_code` used to specify the process.
  - `wf_detail [pk]` Display the detail information for a process.
  - `wf_print_detail [pk]` A page to display process information used for print.
- **Actions(App)**
  - `wf_agree` Agree a process.
  - `wf_back_to` Rollback process to previous node.
  - `wf_reject` Reject a process.
  - `wf_give_up` Give up a process.
  - `wf_batch_agree`
  - `wf_batch_reject`
  - `wf_batch_give_up`
  - `wf_execute_transition` Execute a transition for a process.
  - `wf_execute_transition [wf_code] [trans_func]` Execute a transition for a process with customize function.

2.4 Settings

The following settings are available for configuration through your project.

All available settings can find in `lbworkflow.settings`

2.4.1 List of available settings

LBWF_APPS

Default: `{}`

Specifies the APP of process.

```
>>> {'leave': 'lbworkflow.tests.leave'}.
```

`leave` is the wf_code of the process. `lbworkflow.tests.leave` is the app of the process.

LBWF_USER_PARSER

Default: `lbworkflow.core.userparser.SimpleUserParser`

`django-lb-workflow` use a text field to config users for `Node` and user a parser to cover it to Django model. You can replace it with your implement. The parse must a subclass of `lbworkflow.core.userparser.BaseUserParser`

LBWF_EVAL_FUNCS

Default: `{}`

A list of functions that can used in `Transition.condition`.

```
>>> {'get_dept': 'hr.models.get_dept'}.
```

`get_detp` can used in `Transition.condition`.

LBWF_WF_SEND_MSG_FUNCS

Default: `['lbworkflow.core.sendmsg.wf_print', ]`

A list of functions that used to send message when process node changed.

The function must define as `def wf_print(users, msg_type, event=None, ext_ctx=None)`
users: A list of user need send message to. msg_type: The type of message. Can be `notify/transfered/new_task`.

LBWF_GET_USER_DISPLAY_NAME_FUNC

Default: `lambda user:  "%s" % user`

A function used to get the display name of a user.