prospect Documentation

Release 1.0

Victor Ripplinger

January 30, 2015

Contents

Managing your prospects easily

CHAPTER 1

Abstract

`django-prospect` is an application that allows you to save information about your prospects and then easily find them in your list.

With `django-prospect` you will be able to save reports of your e-mails, phone calls and meetings.

`django-prospect` will help you to make appointments and will remind you of prospects you must phone.

CHAPTER 2

Table of contents

2.1 Setup a new project

2.1.1 Install django-prospect

Create a project folder and enter it. Now you must install django-intranet dependencies and create a new project:

```
$ virtualenv apps
$ source apps/bin/activate
(apps)$ pip install Django PIL MySQL-python South
(apps)$ pip install django-prospect

(apps)$ django-admin.py startproject myintranet
(apps)$ cd myintranet
```

Now it's time to setup your database and to configure the settings module accordingly.

Configuration of the settings module

Here is a possible configuration of the file `settings.py`. Note that the variables `ABSOLUTE_PATH`, `AUTHENTICATION_BACKENDS`, `ANONYMOUS_USER_ID`, `LOGIN_URL`, `LOGOUT_URL`, `LOGIN_REDIRECT_URL`, `PASSWORD_CHANGE_REDIRECT_URL`, `INTRANET_TITLE`, `AUTH_PROFILE_MODULE`, `PASSWORD_CHANGE_REDIRECT_URL` are specific to django-intranet and must be specified. (Don't forget the module imports at the beginning of the file.) Also check in the `INSTALLED_APPS` variable that you have not only django.contrib apps but also all the other apps like in the following example:

```
# -*- coding: utf-8 -*-
# Django settings for django-prospect project.
import os
from django.core.urlresolvers import reverse_lazy

ABSOLUTE_PATH = os.path.dirname(__file__)

DEBUG = True
TEMPLATE_DEBUG = DEBUG

ADMINS = (
    ('admin', 'admin@ionyse.com'),
)

MANAGERS = ADMINS
```

```
INTERNAL_IPS = ('127.0.0.1', )

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql', # Add 'postgresql_psycopg2', 'mysql', 'sqlite3' or 'ora
        'NAME': 'django_prospect_bdd',                      # Or path to database file if using sqli
        'USER': 'admin',                      # Not used with sqlite3.
        'PASSWORD': 'the_password',                  # Not used with sqlite3.
        'HOST': '',                      # Set to empty string for localhost. Not used with sqlite3.
        'PORT': '',                      # Set to empty string for default. Not used with sqlite3.
    }
}

# Local time zone for this installation. Choices can be found here:
# http://en.wikipedia.org/wiki/List_of_tz_zones_by_name
# although not all choices may be available on all operating systems.
# On Unix systems, a value of None will cause Django to use the same
# timezone as the operating system.
# If running in a Windows environment this must be set to the same as your
# system time zone.
TIME_ZONE = 'Europe/Paris'

# Language code for this installation. All choices can be found here:
# http://www.i18nguy.com/unicode/language-identifiers.html
LANGUAGE_CODE = 'fr-fr'

SITE_ID = 1
SITE_NAME = "Django Prospect"
GRAPPELLI_ADMIN_TITLE = SITE_NAME

# If you set this to False, Django will make some optimizations so as not
# to load the internationalization machinery.
USE_I18N = True

# If you set this to False, Django will not format dates, numbers and
# calendars according to the current locale.
USE_L10N = True

# If you set this to False, Django will not use timezone-aware datetimes.
USE_TZ = True

# Absolute filesystem path to the directory that will hold user-uploaded files.
# Example: "/home/media/media.lawrence.com/media/"
MEDIA_ROOT = os.path.join(ABSOLUTE_PATH, 'medias')

# URL that handles the media served from MEDIA_ROOT. Make sure to use a
# trailing slash.
# Examples: "http://media.lawrence.com/media/", "http://example.com/media/"
MEDIA_URL = '/_medias/'

# Absolute path to the directory static files should be collected to.
# Don't put anything in this directory yourself; store your static files
# in apps' "static/" subdirectories and in STATICFILES_DIRS.
# Example: "/home/media/media.lawrence.com/static/"
STATIC_ROOT = os.path.join(ABSOLUTE_PATH, 'collected_static')

# URL prefix for static files.
# Example: "http://media.lawrence.com/static/"
```

```
STATIC_URL = '/_static/'

# Additional locations of static files
STATICFILES_DIRS = (
    # Put strings here, like "/home/html/static" or "C:/www/django/static".
    # Always use forward slashes, even on Windows.
    # Don't forget to use absolute paths, not relative paths.
    # os.path.join(ABSOLUTE_PATH, 'static'),
)

# List of finder classes that know how to find static files in
# various locations.
STATICFILES_FINDERS = (
    'django.contrib.staticfiles.finders.FileSystemFinder',
    'django.contrib.staticfiles.finders.AppDirectoriesFinder',
#    'django.contrib.staticfiles.finders.DefaultStorageFinder',
)

# Make this unique, and don't share it with anybody.
SECRET_KEY = '_$6a=hh50yz!o@(oks0+#6hx+8tmm3^ga#5_9%)xw0hrda%l^b'

# List of callables that know how to import templates from various sources.
TEMPLATE_LOADERS = (
    'django.template.loaders.filesystem.Loader',
    'django.template.loaders.app_directories.Loader',
#     'django.template.loaders.eggs.Loader',
)

TEMPLATE_CONTEXT_PROCESSORS = (
    'django.contrib.auth.context_processors.auth',
    'django.core.context_processors.debug',
    'django.core.context_processors.i18n',
    'django.core.context_processors.media',
    'django.core.context_processors.static',
    'django.core.context_processors.tz',
    'django.contrib.messages.context_processors.messages',
    'django.core.context_processors.request',
)

MIDDLEWARE_CLASSES = (
    'django.middleware.common.CommonMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    # Uncomment the next line for simple clickjacking protection:
    # 'django.middleware.clickjacking.XFrameOptionsMiddleware',
)

ROOT_URLCONF = 'myintranet.urls'

# Python dotted path to the WSGI application used by Django's runserver.
WSGI_APPLICATION = 'myintranet.wsgi.application'

TEMPLATE_DIRS = (
    # Put strings here, like "/home/html/django_templates" or "C:/www/django/templates".
    # Always use forward slashes, even on Windows.
    # Don't forget to use absolute paths, not relative paths.
```

```
    os.path.join(ABSOLUTE_PATH, 'templates'),
)

INSTALLED_APPS = (
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.sites',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'django.contrib.admin',
    'django.contrib.admindocs',
    'admin_tools',
    'admin_tools.dashboard',
    'intranet',
    'floppyforms',
    'bootstrap',
    'south',
    'guardian',
    'prospect',
    'xlrd',
)

# A sample logging configuration. The only tangible logging
# performed by this configuration is to send an email to
# the site admins on every HTTP 500 error when DEBUG=False.
# See http://docs.djangoproject.com/en/dev/topics/logging for
# more details on how to customize your logging configuration.
LOGGING = {
    'version': 1,
    'disable_existing_loggers': False,
    'filters': {
        'require_debug_false': {
            '()': 'django.utils.log.RequireDebugFalse'
        }
    },
    'handlers': {
        'mail_admins': {
            'level': 'ERROR',
            'filters': ['require_debug_false'],
            'class': 'django.utils.log.AdminEmailHandler'
        }
    },
    'loggers': {
        'django.request': {
            'handlers': ['mail_admins'],
            'level': 'ERROR',
            'propagate': True,
        },
    }
}


AUTHENTICATION_BACKENDS = (
    'intranet.backends.ObjectPermissionBackend',
    'django.contrib.auth.backends.ModelBackend',
    'guardian.backends.ObjectPermissionBackend',
)
```

```
# Configuration of Django Guardian
ANONYMOUS_USER_ID = -1

LOGIN_URL = reverse_lazy('login')
LOGOUT_URL = reverse_lazy('logout')
LOGIN_REDIRECT_URL = reverse_lazy('dashboard')
PASSWORD_CHANGE_REDIRECT_URL = reverse_lazy('profile')

INTRANET_TITLE = "Intranet"
AUTH_PROFILE_MODULE = 'intranet.UserProfile'
PASSWORD_CHANGE_REDIRECT_URL = reverse_lazy('profile')
```

When the database and the settings are configured, type these commands:

```
(apps)$ python manage.py syncdb --noinput --migrate
(apps)$ python manage.py createsuperuser
```

2.1.2 Configure the urls

Open the file `urls.py` in the `myintranet/myintranet` folder and there you can setup the URLs of your project. For example:

```
# -*- coding: utf-8 -*-
from django.conf.urls import patterns, include, url
from django.core.urlresolvers import reverse_lazy
from django.conf import settings
from django.views.generic import RedirectView
from django.contrib import admin

from intranet.urls import urlpatterns as intranet_urlpatterns

admin.autodiscover()

# Default URLs patterns from intranet
urlpatterns = intranet_urlpatterns

# Add specific patterns
urlpatterns += patterns('',

                        url(r'^$', RedirectView.as_view(url=reverse_lazy('dashboard'))),

                        # Apps
                        url(r'^prospect/', include('prospect.urls')),

                        # Admin URLs
                        url(r'^_admin/', include(admin.site.urls)),
                        url(r'^_admin_tools/', include('admin_tools.urls')),

                        )

if settings.DEBUG:
    from django.conf.urls.static import static
    from django.contrib.staticfiles.urls import staticfiles_urlpatterns

    urlpatterns += static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
    urlpatterns += staticfiles_urlpatterns()
```

2.1.3 Configure the navbar

In the `myintranet/myintranet` folder, create a folder named `templates`. In that folder create a folder named `intranet`. In that last folder create a text file named `navbar.html` where you will define your navigation bar, like this:

```
{% load intranet_extra i18n %}
{% load url from future %}

<div class="navbar navbar-fixed-top">
  <div class="navbar-inner">
    <div class="container-fluid">
      <a class="btn btn-navbar" data-toggle="collapse" data-target=".nav-collapse">
        <span class="icon-bar"></span>
        <span class="icon-bar"></span>
        <span class="icon-bar"></span>
      </a>
      <a class="brand" href="{% url 'dashboard' %}">{% intranet_title %}</a>
      <div class="nav-collapse">
        <ul class="nav">
          <li class="divider-vertical"></li>
          <li><a href="{% url 'dashboard' %}">
            <i class="icon-user icon-white"></i> {% trans "Dashboard" %}
          </a></li>
          <li class="divider-vertical"></li>
          <li><a href="{% url 'prospect-list' %}">
            <i class="icon-briefcase icon-white"></i> {% trans "Prospects" %}
          </a></li>
          <li class="divider-vertical"></li>
          <li><a href="{% url 'field-list' %}">
            <i class="icon-th-large icon-white"></i> {% trans "Fields" %}
          </a></li>
          <li class="divider-vertical"></li>
          <li><a href="{% url 'activity-list' %}">
            <i class="icon-th-list icon-white"></i> {% trans "Activities" %}
          </a></li>
          <li class="divider-vertical"></li>
        </ul>
        {% if request.user.is_authenticated %}
          <p class="navbar-text pull-right"><a href="{% url 'logout' %}">{% trans "Logout" %}</a></p>
          <ul class="nav pull-right"><li class="divider-vertical"></li></ul>
          <p class="navbar-text pull-right">{% trans "Logged in as" %} <a href="#">{{ request.us
        {% endif %}
      </div><!--/.nav-collapse -->
    </div>
  </div>
</div>
```

2.1.4 Temporary file folder

Just one more thing, for the feature of file import to work correctly, you need to use a temporary file folder. If it doesn't work with the default value, please define `FILE_UPLOAD_TEMP_DIR` in your settings.

That's it. You are now ready to run the server and use the application.

2.2 Features

Add your prospects to the list, enter information about them. You can later modify this information, delete them from your list.

2.2.1 Navigation bar

The application handles several types of data, which you can access to from the navigation bar at the top of the page:

- **Prospects:** Here will be saved the information about your prospects, their companies and contact informations.
- **Fields:** Each prospect can be stored into an activity field. These fields will then help you find prospects in a particular field more easily.
- **Activities:** Here will be saved the summary of all your activities (phone calls, e-mails, meetings...) with your prospects.

2.2.2 Importing a prospect batch

On the page of the prospect list, you will find an 'Import list' button.

This feature allows you to import a list of prospect from an XLS file (Excel document) or a CSV file.

The first row of this file must contain the field names (like "company name", "first name", "field"...) and the others have to contain your data.

If you have several sheets in your XLS file, the application will try to import data of all sheets and will warn you if a sheet seems to be empty.

Import options

A select list allows you to precise the language in which the file is written (for now only English or French).

`django-prospect` can deal with duplicates.

Every time you try to insert a new duplicate, the application is able to check if it is a duplicate, that is, an entry that has already been saved in the database. A new entry will be detected as a duplicate if it has the same company name or both the same first name and last name as an existing entry.

You can choose between three options to tell `django-prospect` how to deal with duplicates:

- Ignore duplicates. (Don't insert entries detected as duplicates.)
- Prompt me. (You will be able to skip or modify every entry detected as duplicate.)
- Allow duplicates. (The application will not try to detect duplicates at all. So if ther is a duplicate, it will be inserted in the database.)

Field name associations

The application will try to associate each field name of the first row of your file with a field name of the database.

For example, if you have a column named "town", its data will automatically be saved in the "city" field of the database using a list of defined word. You will be warned for each field name that failed to be associated with a field name in the database.

Feedbacks and error handling

During the import of each entry, you will be warned of every error or ambiguous case, and you will asked what to do.

For each entry, you will then be able to edit it and then save it, or to skip it. At any time you can decide to give up and cancel all the rest of entry insertions.

2.2.3 Activity summary

Every time you give or receive a phone call, an e-mail, a fax, or you take part in a meeting, you can go to the detail page of the affected prospect and add a new activity report.

The detail page of each prospect also displays a summary of the last activities you had with that prospect. You can then display all the activities clicking on the button “Show all”.

CHAPTER 3

Source code

Don’t hesitate to fork our project and improve it : https://bitbucket.org/ionyse/django-prospect

CHAPTER 4

Screencast

Have a look at the screencast : http://www.youtube.com/watch?v=SKMMKFfzpkg