Django Phantom Theme Documentation

Release 1.1

Przemyslaw 'bespider' Pajak for EggForSale

Oct 30, 2017

Contents

Phantom is an admin theme for Django Framework. It allows for registering custom database options and provides a clean and modern web interface. The application is designed to be responsive and adopt to mobile and tablet devices. To achieve this we have used the twitter bootstrap 3.x framework grid. The application is constantly evolving so make sure you checkout github for the latest updates and fixes.

CHAPTER 1

Features

- A clean and modern user interface
- Hand-written pure HTML5/CSS3 code with indented HTML output
- Responsive interface, optimized for mobile phones and tablets
- Register custom database settings (options) editable from the UI. You can use all standard Django form fields for these settings
- Register your applications to the top-bar navigation
- Refurbished original django admin widgets
- **Added new django admin widgets:**
  - AutoCompleteTextInput
  - BootstrapRadioRenderer
  - BootstrapCurrencyDecimalWidget
  - BootstrapPercentageDecimalWidget
  - URLThumbWidget
  - Select2MultipleWidget
  - Select2Widget
- Basic profile model and views ready to extend
- Mechanism for opening the original Django admin popup windows with fancybox
- Zinnia Blog admin index

CHAPTER 2

Authors and Contributors

bespider (@bespider) for EggForSale (@eggforsale) created Django Phantom Theme.

CHAPTER 3

Licence

Django Phantom Theme is licensed under Creative Commons Attribution-NonCommercial 3.0 license.

Licence and pricing: http://www.eggforsale.com/catalogue/django-phantom-theme_15/

CHAPTER 4

Support or Contact

Having trouble with Django Phantom Theme? Check out the detail page at http://www.eggforsale.com/catalogue/django-phantom-theme_15/ or contact support@eggforsale.com and we'll help you sort it out.

CHAPTER 5

Instalation

Install Django Phantom Theme from the github repository.:

```
$ git clone https://github.com/eggforsale/django-phantom-theme
$ cd django-phantom-theme
$ python setup.py install
```

or just unzip the github package and copy phantom dir to your project.

Requirements

BabelDjango - http://babel.edgewall.org/wiki/BabelDjango

Set Up

To use the Django Phantom Theme website you first need to include phantom in your INSTALLED_APPS setting. You also not need to include django.contrib.admin in INSTALLED_APPS and place it after phantom for Django Phantom Theme to work properly. In settings.py.:

```
INSTALLED_APPS = (
    'phantom',
    'django.contrib.admin',
    ...
)
```

Django Phantom Theme uses the phantom.middleware.PopupMiddleware middleware to replace the standard django admin popups with fancybox. Make sure the middleware is enabled in your MIDDLEWARE_CLASSES setting. In settings.py.:

```
MIDDLEWARE_CLASSES = (
    ...
```

```
    'phantom.middleware.PopupMiddleware',
    ...
)
```

You also need to add 'django.core.context_processors.request' to TEMPLATE_CONTEXT_PROCESSORS setting in your Django project settings.py file.:

```
TEMPLATE_CONTEXT_PROCESSORS = (
    ...
    'django.core.context_processors.request',
    ...
)
```

Register the Django Phantom Theme urls

To register the admin site views, use the following (inside your urls.py).:

```
from phantom import admin_site

admin.autodiscover()
admin_site._registry.update(admin.site._registry)

patterns = (''
    url(r'^admin/', include(admin_site.urls)),
    ...
)
```

You do not need to register the django admin urls as well, the Django Python Theme admin_site extends the original admin class. Beside that we need to tell django to use our customized user model instead of the default one as the authentication model.:

```
# use our own user model

AUTH_USER_MODEL = "phantom.User"
```

Settings

With Django Phantom Theme you can optionaly disable the app index view (the one that lists an application's models). Doing so will raise "Page Not Found" (404) errors when accessing the application urls and will also hide all corresponding links from breadcrumbs.:

```
ADMIN_DISABLE_APP_INDEX = True
```

ModelAdmin registration and auto-discovery

Normally, to register your normal ModelAdmin class with Django Phantom Theme you should use phantom.admin_site instead of the original django.contrib.admin.site instance (in admin.py).:

```
from django.contrib import admin
from models import DemoModel
from phantom import admin_site

class DemoAdmin(admin.ModelAdmin):
    pass

admin_site.register(DemoModel, DemoAdmin)
```

Top-bar navigation

Django Phantom Theme provides a top navigation bar. If you wish, you can register an application's admin models along with an accompanying image to the top-bar as follows.:

```
from phantom import admin_site
admin_site.register_top_menu_item('sites', icon_class="glyphicon-th")
```

The icon_class argument can be any icon from the ones that ship with bootstrap 3.0.

The above snippet will register the django.contrib.admin.sites application to the top bar. Note however that if the application you try to register is not yet registered with the admin website, an Exception will be raised. Therefore, a safe place to put this code is in your urls.py module, right after the auto-discovery code. If you want to register the current application, you could use the admin.py module and place the code right after the ModelAdmin registrations.

Django Phantom Theme provides two custom ModelAdmin attributes to achieve this behavior: order and separator. You can use them like this.:

```
class DemoOneAdmin(admin.ModelAdmin)
    ...
    order = 1

class DemoTwoAdmin(admin.ModelAdmin)
    ...
    order = 2
    separator = True
```

The above will place DemoOneAdmin before DemoTwoAdmin. A separator line will also be drawed before the DemoTwoAdmin item.

If you do not set a custom ModelAdmin order, Django Phantom Theme will use the standard alphabetical order for your models.

You can exclude a certain model from the top-bar navigation. To do so set the exclude_from_top_menu attribute to True.

Admin db options

You can register sets of custom options that editable from the admin interface.

Each set of options is defined by extending the phantom.admin_options.OptionSetAdmin class.:

```
class DemoOptions(OptionSetAdmin):
    optionset_label = 'demo-options'
    verbose_name = 'Demo Options'
```

```
    option_1 = SiteOption(field=forms.CharField(
            widget=forms.Textarea(
                    attrs = {'class' : 'form-control'}
            ),
            required=False
    ))

    option_2 = SiteOption(field=forms.CharField(
            widget=forms.TextInput(
                    attrs = {'class' : 'form-control'}
            ),
    ))

admin_site.register_options(DemoOptions)
```

The optionset_label attribute is the equivalent of the app_label for models. By defining a verbose_name you can explicitly set how you want this option-set label to be displayed.

Each option is implemented as a member of the OptionSetAdmin sub-class, exactly like you would do in a database model. The options must be of the phantom.admin_options.SiteOption type. The field argument of the SiteOption constructor can refer to any standard django form field class instance.

Retrieving option values

To retrieve a single option you can use the get_option() method.:

```
from phantom.utils import get_option

option = get_option('demo-options', 'option_1')
```

If you want to retrieve all options of a single option-set at once use the get_options() method.:

```
from phantom.utils import get_options

options = get_options('demo-options')
```

CHAPTER 6

Admin inline customizations

Collapsing inlines

With Django Phantom Theme you can collapse your inlines, like you do with your fieldsets. Collapsing an admin inline is easy and works for both stacked and tabular inlines.:

```
class DemoInline(admin.StackedInline):
    ...
    collapse = True
```

Modal inlines

Another nice option is the inline modal functionality. It can be really useful when you have a lot of fields in your inline model. Add modal=True to the StackedInline class and your inline form will open in a popup-style modal window.:

```
class DemoInline(admin.StackedInline):
    ...
    modal = True
```

Side navigation for change forms

You can optionally enable a left menu navigation for your change form pages on any model. This will automatically list and track all fieldsets and inlines set in the ModelAdmin.:

```
class DemoModelAdmin(admin.ModelAdmin):
    ...
    fieldsets = (...)
    inlines = (...)
    affix=True
```

Sortable changelists

You can enable a "sorting mode" in the changelist view for orderable objects by subclassing phantom.admin.SortableModelAdmin instead of admin.ModelAdmin.

By default Django Phantom Theme expects the ordering model field to be named "order" (it must be an IntegerField). If the name is different you need to set the "sorting_order_field" attribute.

If you use django-mptt for nested categories, you can enable nested ordering like so.:

```
class CategoryAdmin(SortableModelAdmin):
    sortable_mptt = True
```

The sorting mechanism assumes items are orderd by the ordering field in the default queryset. If that's not true, you should override the "sortables_ordered" method to provide a proper default ordering.:

```
class CategoryAdmin(SortableModelAdmin):
    def sortables_ordered(self, queryset):
            return queryset.order_by("order")
```

Model icons

You can set an accompanying icon class for each of your models in the ModelAdmin class.:

```
class DemoModelAdmin(admin.ModelAdmin):
    ....
    title_icon = 'glyphicon-cog'
```

Base user profile

Django Phantom Theme comes with a base profile fo user model.:

```
AUTH_PROFILE_MODULE = "phantom.Profile"
```

Settings view in model admin

You can add settings view to model admin similar to this one and new button will appear.:

```
def settings_view(self, request, form_url='', extra_context=None):
    info = self.model._meta.app_label, self.model._meta.module_name
    if request.method == 'POST':
        form = YoutubeSettingsForm(request.POST)
        if form.is_valid():
            form.save()
    else:
        try:
            obj = Settings.objects.get(site__exact = current_site_id())
        except Settings.DoesNotExist:
            form = YoutubeSettingsForm()
        else:
```

```
            form = YoutubeSettingsForm(instance=obj)

    context = {
        'title': _('Settings'),
        'media': self.media,
        'form' : form,
        'app_label': info[0],
    }
    return render_to_response(
        "admin/youutube/settings_form.html",
        context,
        context_instance=RequestContext(request)
    )

def get_urls(self):
    from django.conf.urls import patterns, url

    def wrap(view):
        def wrapper(*args, **kwargs):
            return self.admin_site.admin_view(view)(*args, **kwargs)
        return update_wrapper(wrapper, view)

    info = self.model._meta.app_label, self.model._meta.module_name

    urlpatterns = patterns('',
        url(r'^settings/$',
            wrap(self.settings_view),
            name='%s_%s_settings' % info),
    )
    return urlpatterns + super(VideoAdmin, self).get_urls()
```

CHAPTER 7

Sample settings

Django settings.py for demo project.:

```
"""
Django settings for demo_phantom project.

For more information on this file, see
https://docs.djangoproject.com/en/1.6/topics/settings/

For the full list of settings and their values, see
https://docs.djangoproject.com/en/1.6/ref/settings/
"""

# Build paths inside the project like this: os.path.join(BASE_DIR, ...)
import os
BASE_DIR = os.path.dirname(os.path.dirname(__file__))


# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/1.6/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = '...'

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

TEMPLATE_DEBUG = True

ALLOWED_HOSTS = []


# Application definition

INSTALLED_APPS = (
    'phantom',
```

```
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
)

MIDDLEWARE_CLASSES = (
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
    'phantom.middleware.PopupMiddleware',
)

TEMPLATE_CONTEXT_PROCESSORS = (
    "django.contrib.auth.context_processors.auth",
    "django.core.context_processors.request",
    "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",
)

ROOT_URLCONF = 'demo_phantom.urls'

WSGI_APPLICATION = 'demo_phantom.wsgi.application'


# Database
# https://docs.djangoproject.com/en/1.6/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
    }
}

# Internationalization
# https://docs.djangoproject.com/en/1.6/topics/i18n/

LANGUAGE_CODE = 'en-us'

TIME_ZONE = 'UTC'

USE_I18N = True

USE_L10N = True

USE_TZ = True
```

```
# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/1.6/howto/static-files/

STATIC_URL = '/static/'

TEMPLATE_DIRS = (
    os.path.join(BASE_DIR,  'templates'),
)

AUTH_USER_MODEL = 'phantom.User'
```

Django urls.py for Demo project.:

```
from django.conf.urls import patterns, include, url

# Uncomment the next two lines to enable the admin:
from django.contrib import admin
from phantom import admin_site

admin.autodiscover()
admin_site._registry.update(admin.site._registry)

urlpatterns = patterns('',
    # Examples:
    # url(r'^$', 'demo.views.home', name='home'),
    # url(r'^demo/', include('demo.foo.urls')),
    url(r'^blog/', include('zinnia.urls')),
    url(r'^comments/', include('django.contrib.comments.urls')),

    # Uncomment the admin/doc line below to enable admin documentation:
    url(r'^admin/doc/', include('django.contrib.admindocs.urls')),

    # Uncomment the next line to enable the admin:
    url(r'^admin/', include(admin_site.urls)),
)
```

CHAPTER 8

Templates for popular django applications

Django Phantom Theme comes with templates for the following popular django applications:

- Django Zinnia Blog http://django-blog-zinnia.com
- Django MPTT https://github.com/django-mptt/django-mptt