Django 管理画面逆引きメモ

Python　Django

この記事は最終更新日から1年以上が経過しています。

リスト画面 (Change List View)

表示させる項目を指定

```
ModelAdmin.list_display = ('hoge',)
```

Boolを返すModelのカスタムメソッドは直接指定可能。

ついでに　__unicode__　or　__str__　も指定できるよ。

```
ModelAdmin.list_display = ('__str__', 'model_custom_method')
```

詳細画面へのリンクを貼る項目を指定

デフォルトでは一番左側に表示される奴にリンクが付く

```
ModelAdmin.list_display = ('hoge', 'fuga')
ModelAdmin.list_display_links = ('fuga',)
```

カスタムした値を表示

```
ModelAdmin.list_display = ('custom_hoge',)

def custom_hoge(self, obj):
    return u'Hoge'
custom_hoge.short_description = u'表示名'
custom_hoge.allow_tags = True  # htmlタグ許可
```

ユーザー入力の場合は `format_html('hoge')` 関数をつかってエスケープしておくとよい。

デフォルトのソート順

```
ModelAdmin.ordering = ('created_at',)  # 作成時間でソート
```

カスタムアクション (リストビュー上部のセレクト部分のやつ)

```
ModelAdmin.actions = ['hoge']  # 定義した関数などをリストアップ
ModelAdmin.actions_on_top = True  # ページ上部に表示
ModelAdmin.actions_on_bottom = True  # ページ下部に表示
```

カスタムフィールドのソート

Django のソートは全て DB のクエリレベルで行うので、実際にDBにカラムが存在しないフィールドはソートできない。
ただ、カスタムフィールドが、あるフィールドの代理になっているときは、そのフィールド名を指定するとソートできる。

```
ModelAdmin.list_display = ('number_str',)
```

```
# number が数字(int)を返すとする
def number_str(self, obj):
    return str(obj.number)
number_str.admin_order_field = 'number'
```

`list_display` でDBにカラムのないカスタムフィールドをソート可能にする方法

StackOverflow Django admin: how to sort by one of the custom list_display fields that has no database field

追加・変更画面

特定のフィールドを更新の対象にしたい

２つの指定方法がある

fields

1. fields は `ModelAdmin.readonly_fields` の値も指定できるが、編集はできない
2. `list_display` などと違い、Model の field か `ModelAdmin.form` しか指定できない

```
ModelAdmin.fields = ('hoge', 'fuga')  # hoge, fuga フィールドのみ
ModelAdmin.fields = (('hoge', 'fuga'), 'piyo')  # hoge, fuga か
```

exclude

```
ModelAdmin.exclude = ('hoge',)  # hoge は編集対象とならない
```

外部キーの値も、同じ画面で追加/編集できるようにする

```
# Inline モデルを定義
class SomeForeignKeyModelInline(admin.TabularInline):
    model = SomeForeignKeyModel
```

```
# Inline モデルを表示させたいモデルの管理画面に追加
class HogeAdmin(admin.ModelAdmin):
    list_display = ('hoge', 'fuga')
    inlines = [SomeForeignKeyModelInline]
```

fieldsets オプションでリッチな画面

```
ModelAdmin.fieldsets = (
    ( '名前', {'オプション名': ('オプション値',)} ),
    (None, {
        'fields': ('hoge', 'fuga')
    }),
)
```

カスタムフォーム

デフォルトでは ModelForm がよしなに管理画面用のフォームを生成するが、自分でカスタムしたフォームも指定可能

```
ModelAdmin.form = HogeForm
```

モデルに定義されていないカスタムのフォームフィールドを管理画面に表示させたい

stackOverflow: django admin - add custom form fields that are not part of the model

既存の自動生成されるフォームをカスタマイズしたい

`get_form` メソッドでフックする

```
ModelAdmin.get_form(request, obj=None, **kwargs)
```

管理画面テンプレートの上書き

設定

管理画面の元テンプレート

は `contrib/admin/template/admin` に格納されている。

上書きする場合は、 `TEMPLATE_DIR` で設定されているディレクトリに `admin/` を追加する。

上書きしたい管理対象を、上記ディレクトリ配下
に、 `app_name/modelname/` で追加する。
アプリ名以下のテンプレートはアプリ配下の全てのモデルに適応される。
モデル名以下のテンプレートは、そのモデルの管理画面のみ適応される。
ディレクトリ名は全て小文字。

テンプレート作成方針

基本的に必要な部分だけを上書きするほうが効率的。

例:
hoge アプリの fuga モデルの一覧画面を上書きしたいときは `admin/hoge/fuga/change_list.html` を作成。

あとは以下のように変更したい `{% block %}` 上書きしていく。

```
{% extend 'admin/change_list.html' %}
{% block 'content' %}
<!--
なにか追加したいフォームや変数など
-->
{{ block.super }}
{% endblock 'content' %}
```

独自機能追加

管理画面に独自機能を追加したい場合、必要になりそうな作業のメモ

URL の追加

独自機能を提供するURLを追加する。
Django管理画面がデフォルトで生成するURLをカスタマイズする必要がある。

`get_urls` を使う.

```
from django.conf.urls import url
from django.http import HttpResponse
class HogeAdmin(admin.ModelAdmin):
    def get_urls(self):
        urls = super(HogeAdmin, self).get_urls()
        my_urls = [
            url(r'^hoge/$', self.admin_site.admin_view(self.ho
        ]
        return my_urls + urls

    def hoge(self, request):
        return HttpResponse('OK')
```

`self.admin_site.admin_view` は追加したいビューに、パーミッションのチェックと `never_cache` を付与するもの。基本的には、これでラップしておくほうが良い。

`return my_url + urls` の順番に注意。 `my_urls` を先にしないと、管理画面標準の URL が優先されて
独自定義した URL のビューまで辿りつけない。

既存管理画面の動作を上書きする

各 View関数の動作をオーバーライドすればよい

```
class HogeAdmin(admin.ModelAdmin):
    # 一覧画面の動作を上書きする例
    def changelist_view(self, request, extra_context=None):
        extra_context = extra_context or {}
        extra_context['hoge'] = 'hoge'
        return super(HogeAdmin, self).changelist_view(request,
```

その他

同一の Model を複数の ModelAdmin で管理する

StackOverflow Multiple ModelAdmins/views for same model in Django admin