Django APIs

[Django][Python]

・http://docs.djangoproject.com/en/dev/

View レイヤー

shortcut

■ render_to_response

・テンプレートとともにコンテキスト辞書を与えると、テキストが描画された HttpResponse オブジェクトを返す

```
render_to_response(template[, dictionary][, context_instance][, mimetype])
```

Model レイヤー

Field オプション

■ Field.null

・True に設定すると、エンプティ値を NULL としてデータベースに格納する。
・デフォルトは False

■ Field.blank

・True に設定すると、ブランク値が許容される。デフォルトは False。
・純粋にデータベースの問題である、null オプションとは異なり、値の妥当性検査に関係する。False の場合、入力必須となる。

■ Field.choices

・選択フィールドとして、2 要素を持つタプルの繰り返しを利用する。
・Django admin は、このオプションが与えら得れた場合、テキストフィールドの変わりにセレクトボックスを利用する。

```
from django.db import models
class MyModel1(models.Model):
      WEEK_CHOICES  =  (('0','Sun'),('1','Mon'),('2','Tue'),('3','Wed'),('4','Thu'),('5','Fri'),('6
','Sat'))
    week = models.CharField(max_length=1, choices=WEEK_CHOICES)
```

■ Field.db_column

・データベースのカラム名を指定する。指定しない場合、フィールド名を使用する。

■ Field.db_index

・True に設定した場合、以下のように CREATE INDEX 文を出力できる。

```
C:\work\py\django\mysite>python manage.py sqlindexes myapp1
```

```
BEGIN;
CREATE INDEX "myapp1_mymodel1_field1" ON "myapp1_mymodel1" ("field1");
COMMIT;
```

■ Field.db_tablespace

・フィールドにインデックスを張る場合に使用するテーブルスペースを指定。
・デフォルトは、DEFAULT_INDEX_TABLESPACE セッティング
・バックエンドがサポートしない場合は無視される。

■ Field.default

・フィールドのデフォルト値
・呼び出し可能オブジェクトを指定することもできる。その場合、呼ばれるたびに新規オブジェクトが作成される。

■ Field.editable

・False に設定すると、モデルクラスから自動的に生成される、admin もしくは forms においてフィールドは編集不可になる。

■ Field.help_text

■ Field.unique

■ Field.unique_for_date

■ Field.unique_for_month

■ Field.verbose_name

Field タイプ

■ AutoField

・自動的に有効な ID をインクリメントして取得する IntegerField 。
・主キーフィールドはモデルに暗黙的に追加され、AutoField となるため、通常は直接使用する必要はない。
・save メソッドを呼び出すまでは None

■ BooleanField

項目	値
デフォルト Widget	CheckboxInput
未設定時	False
値の正規化	True or False
バリデーション	チェックが ON の場合 True
エラーメッセージの Key	required

例

```
from django.http import HttpResponse
from django.template import Context, loader
from django.forms import *

def index(request):
    t = loader.get_template('foobar/index.html')
    form = None
    result = ''
    if request.method == 'POST':
        form = HogeForm(request.POST)
        if form:
            if form.is_valid():
                pass
           (略)

class HogeForm(Form):
    is_foo = BooleanField(required=False)
```

【注意】required=False

Field の派生クラスは、バリデーションの設定が、required=True だが、required=False としないと、チェック ON でない場合、form.is_valid() が失敗してしまう。

■ CharField

■ CommaSeparatedIntegerField

■ DateField

■ DateTimeField

■ DecimalField

■ EmailField

■ FileField

・Django File アップロード例

■ FilePathField

■ FloatField

■ ImageField

■ IntegerField

■ IPAddressField

■ NullBooleanField

■ PositiveIntegerField

■ PositiveSmallIntegerField

■ SlugField

■ SmallIntegerField

■ TextField

■ TimeField

■ URLField

■ XMLField

Forms

・http://docs.djangoproject.com/en/dev/topics/forms/#topics-forms-index

Built-in Field

■ BooleanField

■ CharField

■ ChoiceField

項目	値
デフォルト Widget	Select
未設定時	'' (空文字)
値の正規化	ユニコードオブジェクト
バリデーション	リストの値に存在
エラーメッセージの Key	required、invalid_choice

使用例

・Form

```
from django.forms import *

class FooForm(Form):
    CHOICE_LIST = (('01','ham'),('02','spam'))
    bar = CharField(widget=Select(choices=CHOICE_LIST))

    def conv(self, data):
        choiced = self.cleaned_data['bar']
                :
```

・テンプレート

```
<form action='/hogehoge/' method='POST'>
  {% if form %}
  BAR:{{form.bar}}
  {% endif %}
```

■ TypedChoiceField

■ DateField

■ DateTimeField

■ DecimalField

■ EmailField

■ FileField

■ FilePathField

■ FloatField

■ ImageField

■ IntegerField

■ IPAddressField

■ MultipleChoiceField

■ NullBooleanField

■ RegexField

■ TimeField

■ URLField

Built-in widgets

■ TextInput

■ PasswordInput

■ HiddenInput

■ MultipleHiddenInput

■ FileInput

■ DateInput

■ DateTimeInput

■ TimeInput

■ Textarea

行列を指定する例

```
from django.forms import *
class HogeForm(Form):
    data = CharField(widget=Textarea(attrs=dict(rows='40', cols='50')))
```

■ CheckboxInput

■ Select

■ NullBooleanSelect

■ SelectMultiple

■ RadioSelect

■ CheckboxSelectMultiple

■ MultiWidget

■ SplitDateTimeWidget

■ SelectDateWidget

Template レイヤー

・http://docs.djangoproject.com/en/dev/topics/templates/#topics-templates

タグ

・組み込みタグリファレンス

■変数

・変数は、{{ variable }} のように記述する。
・ピリオド "." で、変数の属性を参照できる。

■ for

for ループの中で利用できる変数

変数	内容
forloop.counter	1 から始まるカウンタの現在位置
forloop.counter0	0 から始まるカウンタの現在位置
forloop.revcounter	1 から始まり最後から現在していくカウンタの現在位置
forloop.revcounter0	0 から始まり最後から現在していくカウンタの現在位置
forloop.first	繰り返し最初の要素の場合 True

forloop.last	繰り返し最後の要素の場合 True
forloop.parentloop	ネストしたループの場合、親ループの現在の値を参照

配列の各要素をループする。

・views.py

```
def index(request):
    ctx = {}
    ctx['rows'] = range(1,4,1)
    return render_to_response('test/index.html', ctx)
```

・index.html

```
<table>
  {% for i in rows %}
  <tr><td>{{ i }}</td><td><input type="text"></td></tr>
  {% endfor %}
</table>
```

・結果

アイテムを繰り返し処理する

```
{% for itm in item_list %}
  <a href="{{itm.detailPageURL}}"><img src="{{itm.smallImageURL}}"/></a>
 {% endfor %}
```

■ if と else

・変数を評価し、変数が true なら ブロックを表示する

例

```
{% if rows %}
  rows size : {{ rows|length }}.
{% else %}
  no rows.
{% endif %}
```

結果

■ ifequal

・2 つの引数が等しい場合にブロックを出力する。

```
{% ifequal style 'text' %}
  <div>
    スタイル変数が 'text' の場合このブロックを出力
  </div>
{% endifequal %}
```

テンプレートの継承

基底

・派生でオーバーライドする箇所を、block [名前] 〜 endblock で作成

```
 <!DOCTYPE  html  PUBLIC  "-//W3C//DTD  HTML  4.01  Transitional//EN"  "http://www.w3.org/TR/html4
/loose.dtd">
 <html>
 <head>
   <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
   {% block html_header %}
   {% endblock %}
 </head>
 <body>
   {% block content %}
   {% endblock %}
 </body>
 </html>
```

派生

・基底ファイルを extends で指定
・オーバーライドする箇所を block [名前] 〜 endblock で記述

```
{% extends "base.html" %}
{% block html_header %}
  <script type="text/javascript"><!--
  //--></script>
  <title>Title</title>
{% endblock %}
{% block content %}
  Content
{% endblock %}
```

HTML をそのまま出力する

・|safe とする

```
{{digest.summary|safe}}
```

フィルター

・組み込みテンプレートフィルタリファレンス

・フィルターを利用して、変数の表示を変更できる
・次のように記述する << プラグインは存在しません。>>
・フィルターは連鎖できる。<< プラグインは存在しません。>>
・引数をとるフィルターもある。<< プラグインは存在しません。>>
・スペースを含むフィルターの引数は引用符で囲む << プラグインは存在しません。>>
・Django は約 30 種類の組み込みテンプレートフィルタを用意している。

■ default

・変数が、false か 空の場合、デフォルトを表示する。そうでない場合は、変数の値を表示する。

```
{{ value|default:"nothing" }}
```

■ length

・長さを返す。

```
{{ value|length }}
```

■ striptags

・タグを除いた値を返す。

```
{{ value|striptags }}
```

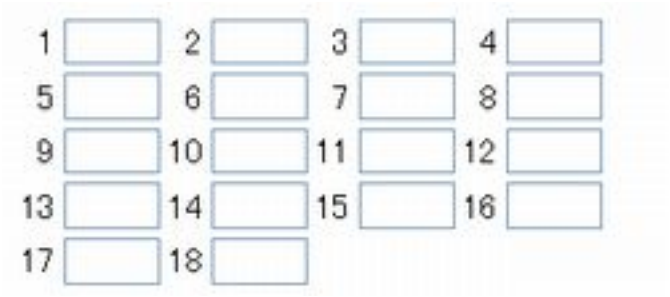
■ divisibleby

・値が引数で割り切れるかどうか判定。割り切れれば True
・mod 演算の代替として使える

例

```
<table>
  <tr>
  {% for num in numlist %}
    <td align='right'>{{ num }}</td>
    <td><input type='text' size='6'></td>
    {% if num|divisibleby:"4" %}
    </tr><tr>
    {% endif %}
  {% endfor %}
  </tr>
</table>
```

結果



Forms