Урок 16

Древовидные структуры в Django.

pip install django-mptt
from mptt.models import MPTTModel, TreeForeignKey

class Genre(MPTTModel):
        parent = TreeForeignKey('self', null=True, blank=True, related_name='children')

get_ancestors() - creates a QuerySet containing the ancestors of the model instance.

get_children()

get_descendants() - Creates a QuerySet containing descendants of the model instance, in tree order.

get_descendant_count()

get_next_sibling() get_previous_sibling() get_root()

insert_at(target, position='first-child', save=False) move_to(target, position='first-child')

Древовидные структуры в Django.

```
from mptt.admin import MPTTModelAdmin



class CustomMPTTModelAdmin(MPTTModelAdmin)
    # speficfy pixel amount for this ModelAdmin only:
    mptt_level_indent = 20

admin.site.register(Node, CustomMPTTModelAdmin)
```

Расширяем профайл пользователя.

```
from django.contrib.auth.models import User
class Profile(User,BodymetrikaMixin, AboutMixin):
   GENDER = (
            ('male', _('male')),
            ('female',_('female'))
          )

   birthday = models.DateField(verbose_name=_(u'Birthday'), blank=True)
   nickname = models.CharField(verbose_name=_(u'Nickname'),max_length=250, blank=True)
   about_me = models.TextField(verbose_name=_(u'About me'), blank=True)
   gender = models.CharField(verbose_name=_('Gender'),
                      choices=GENDER,
                      default='male',
                      max_length=6)
```

Миксины

```
from users.mixins import BodymetrikaMixin, AboutMixin
class BodymetrikaMixin(models.Model):
    HEIGHT = [(n,str(n)+' '+__('sm.')) for n in range(120,250)]
    height = models.IntegerField(
        verbose_name=_('Height'),
        choices=HEIGHT,
        default=0,
        null=True,
        blank=True)
    WEIGHT = [[n,str(n)+' '+__('kg.')] for n in range(20,200)]
    weight = models.IntegerField(
        verbose_name=_('Weight'),
        choices=WEIGHT,
        default=0,
        null=True,
        blank=True)
```

Кеширование

Оперативка, бд, ФС, псевдокеширование.

```
CACHES = {
    'default': {
        'BACKEND': 'django.core.cache.backends.filebased.FileBasedCache',
        'LOCATION': '/var/tmp/django_cache',
    }
}
```

```
from django.views.decorators.cache import cache_page

@cache_page(60 * 15)
def my_view(request):
```

Кеширование

```
urls.py
from django.views.decorators.cache import cache_page

urlpatterns = ('',
    (r'^foo/(\d{1,2})/$', cache_page(60 * 15)(my_view)),
)
```

```
{% load cache %}
{% cache 500 sidebar %}
    .. sidebar ..
{% endcache %}
```

```
>>> from django.core.cache import get_cache
>>> cache = get_cache('alternate')
>>> cache.set('my_key', 'hello, world!', 30)
>>> cache.get('my_key')
```

Прекрасный суп.

Beautiful Soup is a Python library for pulling data out of HTML and XML files.

```
pip install BeautifulSoup
```

```
from bs4 import BeautifulSoup
soup = BeautifulSoup(html_doc)
soup.title
soup.title.string
soup.title.parent.name
soup.find_all('a')
soup.find(id="link3")
for link in soup.find_all('a'):
    print(link.get('href'))
```