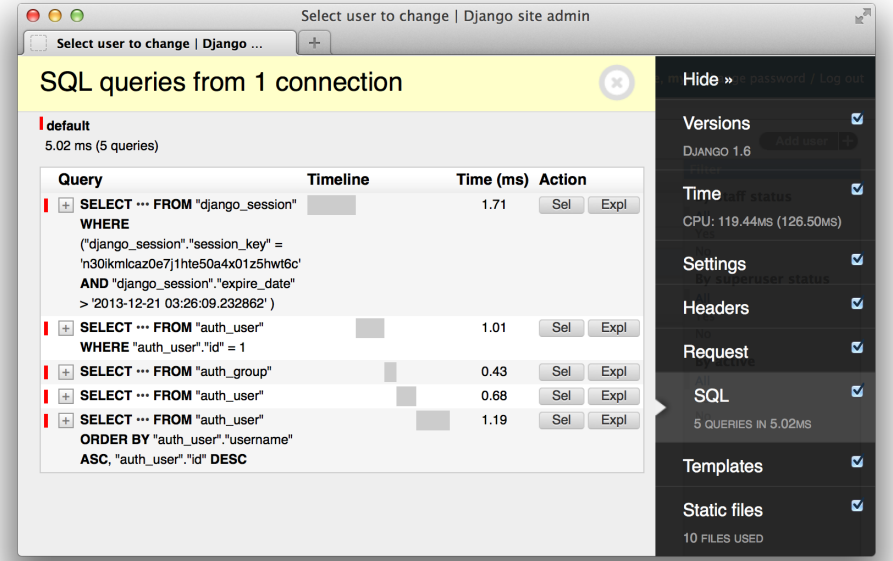
Урок 15

Дебагер django



Устанавливаем дебагер Django.

```
pip install django-debug-toolbar
INSTALLED_APPS = (
    # ...
    'django.contrib.staticfiles',
    'debug_toolbar',
)
MIDDLEWARE_CLASSES = (
    'debug_toolbar.middleware.DebugToolbarMiddleware',
)
from django.conf import settings
from django.conf.urls import include, patterns, url

if settings.DEBUG:
    import debug_toolbar
    urlpatterns += patterns('',
        url(r'^__debug__/', include(debug_toolbar.urls)),
    )
```

REST API на основе tastypie.

```
sudo pip install django-tastypie
```

```
from tastypie.utils.timezone import now
from django.contrib.auth.models import User
from django.db import models
from django.utils.text import slugify


class Entry(models.Model):
    user = models.ForeignKey(User)
    pub_date = models.DateTimeField(default=now)
    title = models.CharField(max_length=200)
    slug = models.SlugField()
    body = models.TextField()

    def __unicode__(self):
        return self.title
    def save(self, *args, **kwargs):
        # For automatic slug generation.
        if not self.slug:
            self.slug = slugify(self.title)[:50]
        return super(Entry, self).save(*args, **kwargs)
```

Creating Resources

```
# myapp/api.py
from tastypie.resources import ModelResource
from myapp.models import Entry


class EntryResource(ModelResource):
    class Meta:
        queryset = Entry.objects.all()
        resource_name = 'entry'
```

Hooking Up The Resource

```
# urls.py
from django.conf.urls.defaults import *
from myapp.api import EntryResource

entry_resource = EntryResource()

urlpatterns = patterns('',
    # The normal jazz here...
    (r'^blog/', include('myapp.urls')),
    (r'^api/', include(entry_resource.urls)),
)
```

- http://127.0.0.1:8000/api/entry/?format=json
- http://127.0.0.1:8000/api/entry/1/?format=json
- http://127.0.0.1:8000/api/entry/schema/?format=json
- http://127.0.0.1:8000/api/entry/set/1;3/?format=json

Creating More Resources

```
# myapp/api.py
from django.contrib.auth.models import User
from tastypie import fields
from tastypie.resources import ModelResource
from myapp.models import Entry


class UserResource(ModelResource):
    class Meta:
        queryset = User.objects.all()
        resource_name = 'user'


class EntryResource(ModelResource):
    user = fields.ForeignKey(UserResource, 'user')

    class Meta:
        queryset = Entry.objects.all()
        resource_name = 'entry'
```

Adding To The Api

```
# urls.py
from django.conf.urls.defaults import *
from tastypie.api import Api
from myapp.api import EntryResource, UserResource

v1_api = Api(api_name='v1')
v1_api.register(UserResource())
v1_api.register(EntryResource())

urlpatterns = patterns('',
    # The normal jazz here...
    (r'^blog/', include('myapp.urls')),
    (r'^api/', include(v1_api.urls)),
)
```

Настройки

```
from tastypie.authentication import BasicAuthentication

class SillyAuthentication(Authentication):
    def is_authenticated(self, request, **kwargs):
        if 'daniel' in request.user.username:
          return True

        return False

class UserResource(ModelResource):
    class Meta:
        queryset = User.objects.all()
        resource_name = 'user'
        excludes = ['email', 'password', 'is_active', 'is_staff', 'is_superuser']
        fields = ['username', 'first_name', 'last_name', 'last_login']
        allowed_methods = ['get']
        authentication = BasicAuthentication()
```

Настройки

```
from tastypie.authorization import DjangoAuthorization
from tastypie.serializers import Serializer
from tastypie.throttle import BaseThrottle
from tastypie.paginator import Paginator

class UserResource(ModelResource):
    class Meta:
.....
        # Add it here.
       authorization = DjangoAuthorization()
        serializer = Serializer(formats=['json', 'jsonp', 'xml', 'yaml', 'html', 'plist'])
        throttle = BaseThrottle(throttle_at=100, timeframe=2)
        paginator_class = Paginator
```

Настройки

API_LIMIT_PER_PAGE

TASTYPIE_ALLOW_MISSING_SLASH

```
TASTYPIE_DATETIME_FORMATTING = 'rfc-2822'
TASTYPIE_DEFAULT_FORMATS = ['json', 'xml']
```

Возможности

Tastypie features support for GeoDjango! Resources return and accept GeoJSON.

It is common to use django to provision OAuth 2.0 tokens for users and then have Tasty Pie use these tokens to authenticate users to the API.

```
import requests
import json

response = requests.get('http://localhost:8000/api/v1/entry/1/')
event = json.loads(response.content)
response = requests.post('http://localhost:8000/api/v1/entry/',
                         data=json.dumps(event),
                         headers={'content-type': 'application/json'})
```

Приемы резнесения настроек на серверах.

_local.py -> local.py

```
from local import *
```

```
.gitignore
.idea
*.pyc
/media
local.py
db.sqlite3
*~
```

Интеграция аутентификации через соц. сети.

Django Social Auth

```
pip install django-social-auth
INSTALLED_APPS = (
    ...
    'social_auth'
)
./manage.py syncdb
```

Интеграция аутентификации через соц. сети.

```
AUTHENTICATION_BACKENDS = (
    'social_auth.backends.twitter.TwitterBackend',
    'social_auth.backends.facebook.FacebookBackend',
    'social_auth.backends.google.GoogleOAuthBackend',
    'social_auth.backends.google.GoogleOAuth2Backend',
    'social_auth.backends.google.GoogleBackend',
    'social_auth.backends.yahoo.YahooBackend',
    'social_auth.backends.browserid.BrowserIDBackend',
    'social_auth.backends.contrib.linkedin.LinkedinBackend',
    'social_auth.backends.contrib.disqus.DisqusBackend',
    'social_auth.backends.contrib.livejournal.LiveJournalBackend',
    'social_auth.backends.contrib.orkut.OrkutBackend',
    'social_auth.backends.contrib.foursquare.FoursquareBackend',
    'social_auth.backends.contrib.github.GithubBackend',
    'social_auth.backends.contrib.vk.VKOAuth2Backend',
    'social_auth.backends.contrib.live.LiveBackend',
    'social_auth.backends.contrib.skyrock.SkyrockBackend',
    'social_auth.backends.contrib.yahoo.YahooOAuthBackend',
    'social_auth.backends.contrib.readability.ReadabilityBackend',
    'social_auth.backends.contrib.fedora.FedoraBackend',
    'social_auth.backends.OpenIDBackend',
    'django.contrib.auth.backends.ModelBackend',
)
```

```
urlpatterns = patterns('',
    ...
    url(r'', include('social_auth.urls')),
    ...
)
TWITTER_CONSUMER_KEY         = ''
TWITTER_CONSUMER_SECRET      = ''
FACEBOOK_APP_ID              = ''
……..
SKYROCK_CONSUMER_SECRET      = ''
YAHOO_CONSUMER_KEY           = ''
YAHOO_CONSUMER_SECRET        = ''
READABILITY_CONSUMER_SECRET  = ''
READABILITY_CONSUMER_SECRET  = ''
```

Интеграция аутентификации через соц. сети.

```
LOGIN_URL          = '/login-form/'
LOGIN_REDIRECT_URL = '/logged-in/'
LOGIN_ERROR_URL    = '/login-error/'
SOCIAL_AUTH_LOGIN_REDIRECT_URL = '/another-login-url/'
SOCIAL_AUTH_NEW_USER_REDIRECT_URL = '/new-users-redirect-url/
SOCIAL_AUTH_NEW_ASSOCIATION_REDIRECT_URL = '/new-association-redirect-url/'
SOCIAL_AUTH_DISCONNECT_REDIRECT_URL = '/account-disconnected-redirect-url/'
SOCIAL_AUTH_BACKEND_ERROR_URL = '/new-error-url/'

{% url "socialauth_begin" "backend-name" %}
{% url "socialauth_disconnect" "backend-name" %}
{% url "socialauth_disconnect_individual" "backend-name" association_id %}
```