Будничный Django

Владимир Филонов

Labbler.com

Программируем
8–14 часов в день

8–14 часов в день

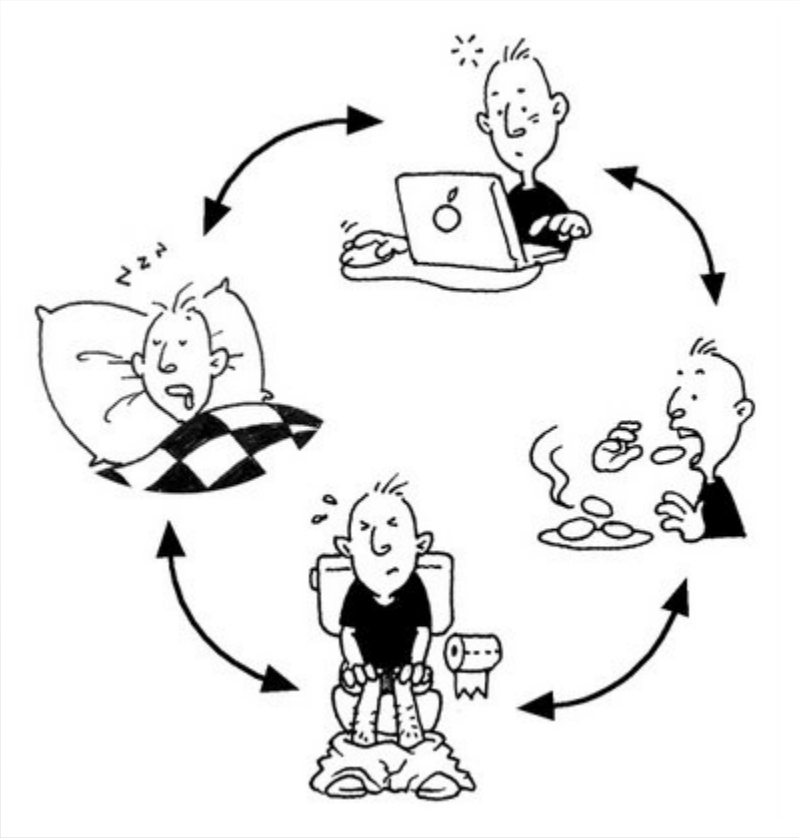
Почти каждый день

≈3000 часов в год

Как проходит это время?


≈30% мы творим
Ar
La
Cl
Me
Pr
Bo

50–70% – рутина



Как же так жить?

Задачи

- Упростить рутинную работу

Задачи

- Упростить рутинную работу
- Уменьшить влияние человеческого фактора

Задачи

- Упростить рутинную работу
- Уменьшить влияние человеческого фактора
- Сделать работу приятнее

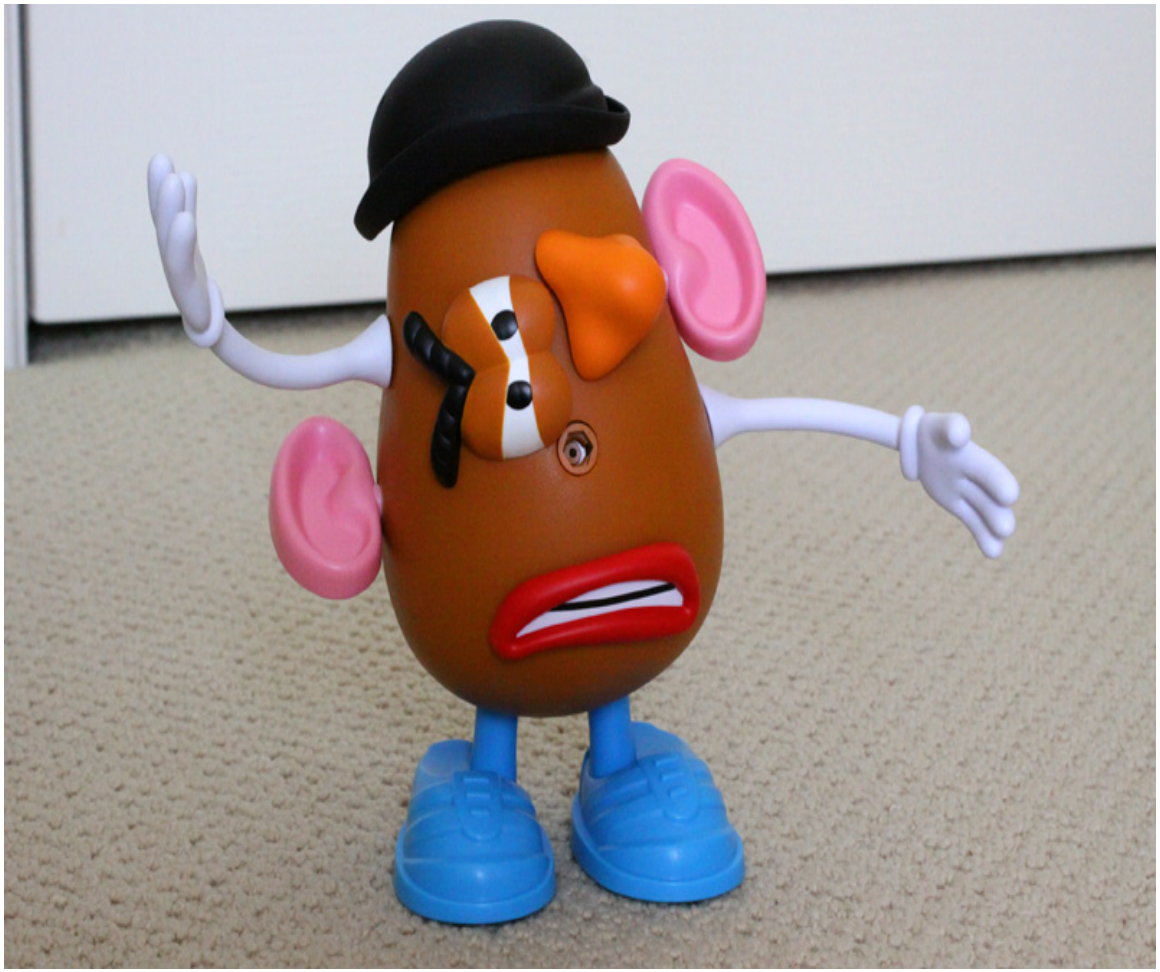
Что нам поможет

Кофе

Пиво

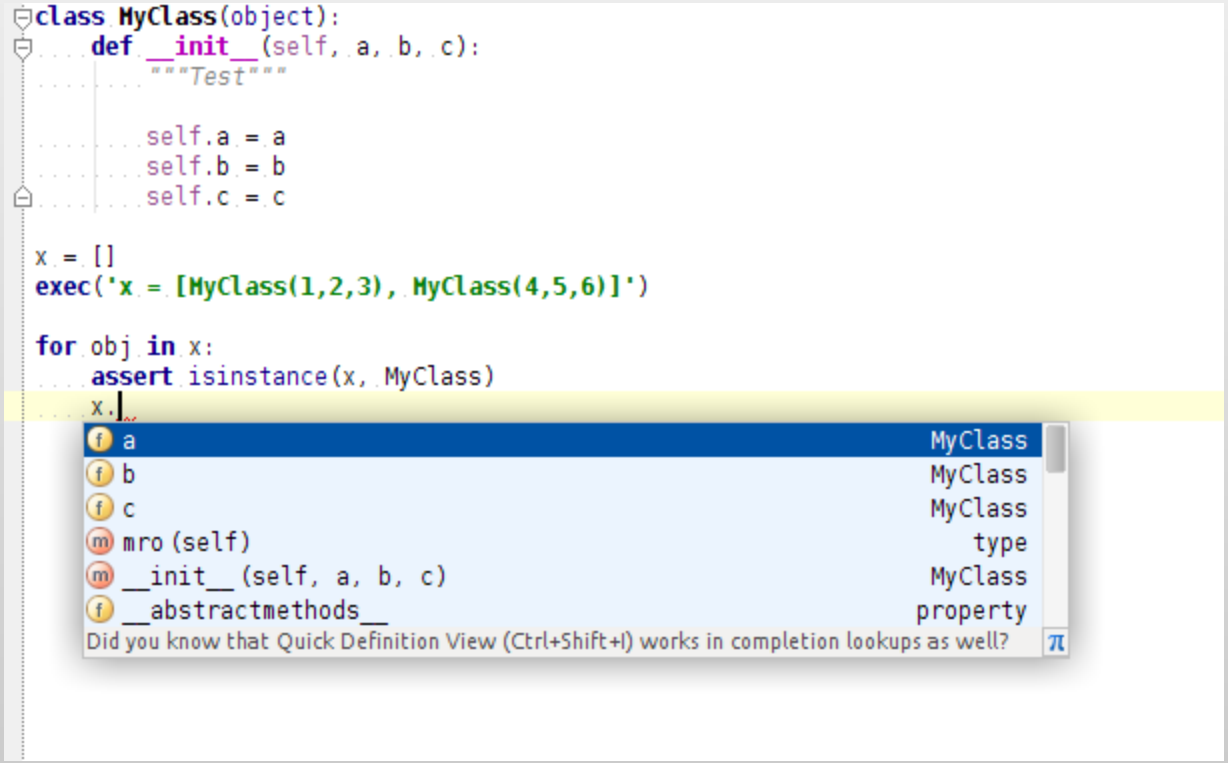
IDE

Голова и руки

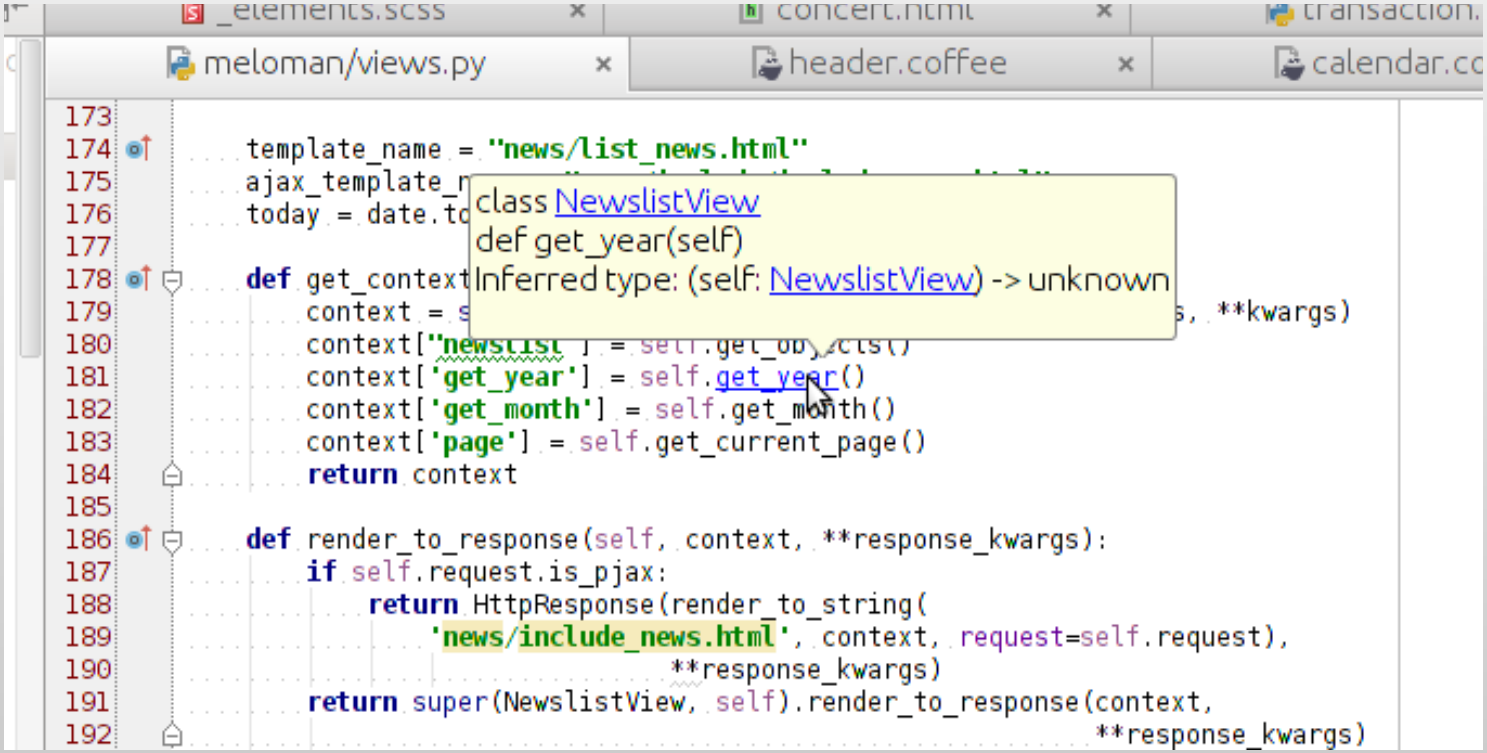


IDE

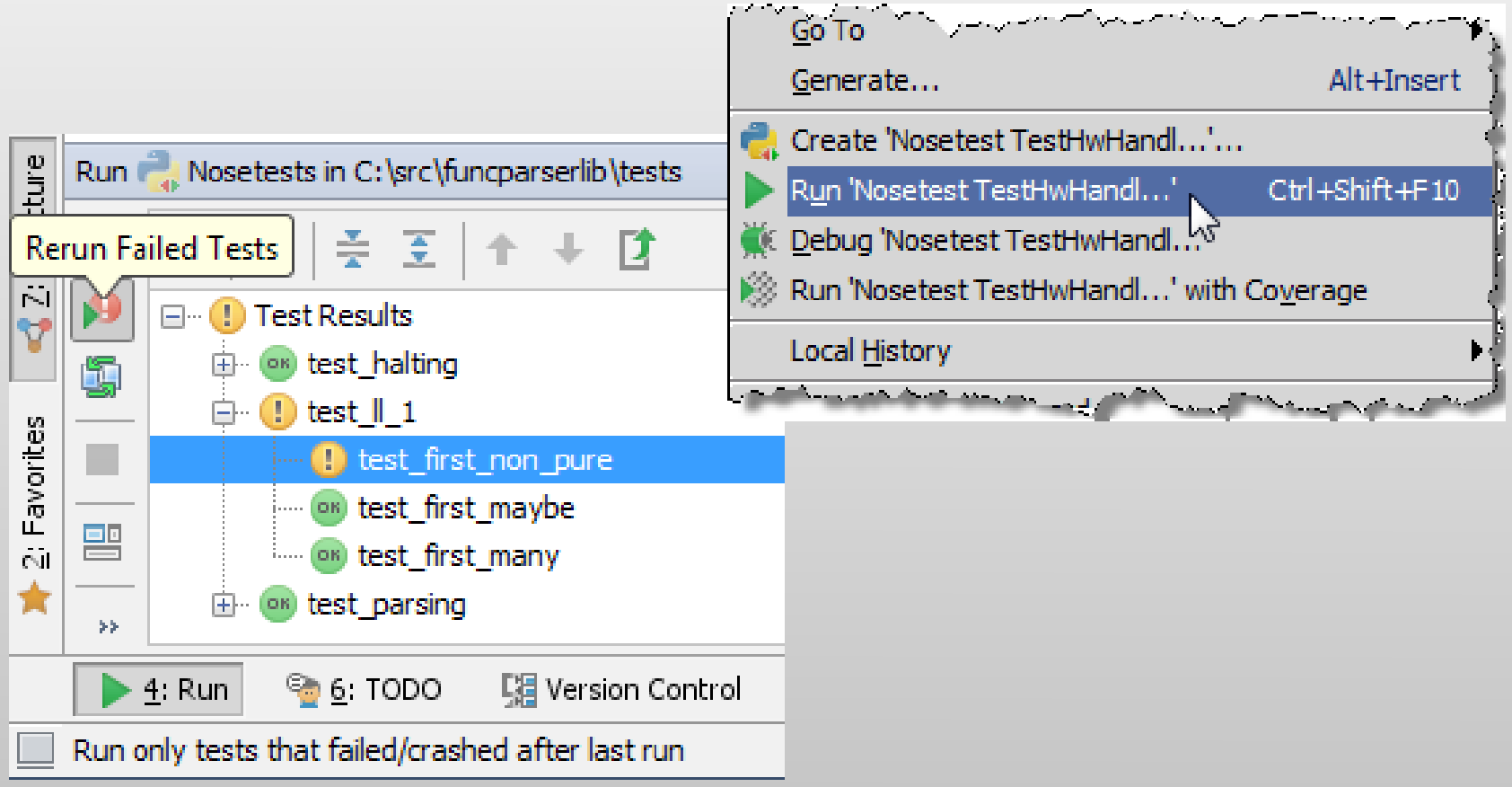
Автозаполнение



Переходы по ссылкам



Запуск тестов



Запуск тестов

```
./manage.py test labbler.audio.TestTranscoding.test_unsupported_format
```

PEP8, pylint и прочее

```
class StarkComment(Comment):
    """Render class for the comments in the top-comments
       display in the reddit toolbar"""
    _nodb = True

class MoreMessages(Printable):
    cachable = False

    new = False
    was_comment = False
    is_collapsed = True

    def __init__(self, parent, child):
        self.parent = parent
        self.child = child
```

PEP8: expected 2 blank lines, found 1

Пишем меньше кода

Пишем меньше **повторяющегося** кода

Выделение метода/функции

```
def users_list(request):

    ...

    if user.has_avatar():
        avatar = user.avatar
    else:
        avatar = DEFAULT_AVATAR

    ...

def user_profile(request):

    ...

    if user.has_avatar():
        avatar = user.avatar
    else:
        avatar = DEFAULT_AVATAR

    ...
```

```
def get_avatar(user):
    if user.has_avatar():
        avatar = user.avatar
    else:
        avatar = DEFAULT_AVATAR

def users_list(request):

    ...

    avatar = get_avatar(user)

    ...

def user_profile(request):

    ...

    avatar = get_avatar(user)

    ...
```

Mixin — расширяем классы

```
class SomeView(View):

    def get_queryset(self):
        try:
            page = int(self.request.GET.get("page", 1))
            if page < 0:
                page = 1
        except (TypeError, ValueError):
            page = 1


class SomeAnotherView(View):

    def get_queryset(self):
        try:
            page = int(self.request.GET.get("page", 1))
            if page < 0:
                page = 1
        except (TypeError, ValueError):
            page = 1
```

```
class PaginationMixin(object):

    def get_page(self):
        try:
            page = int(self.request.GET.get("page", 1))
            if page < 0:
                page = 1
        except (TypeError, ValueError):
            page = 1
        return page

class SomeView(View, PaginationMixin):

    def get_queryset(self):
        page = self.get_page()


class SomeAnotherView(View, PaginationMixin):

    def get_queryset(self):
        page = self.get_page()
```

Важно!

Нужно соблюдать меру

```
def do_something(obj):
    return do_something_else(obj)

def do_something_else(obj):
    return do_something_unexpected(obj)

def do_something_unexpected(obj):
    return do_something_unbelievable(obj)

def do_something_unbelievable(obj):
    return do_calculations(obj)

def do_calculations(obj):
    ...
    return result
```

```
class SomeMixin(SomeAnotherMixin):
    ...

class SomeAnotherMixin(OneMoreMixin, AndMoreAgainMinix):
    ...

class OneMoreMixin(AreYouKiddingMeMixin):
    ...

class AndMoreAgainMinix(object):
    ...

class AreYouKiddingMeMixin(object):
    ...
```

Декораторы

```
from django.db import transaction

def some_view(request):

    try:

        ...
        transaction.commit()

    except:
        ...
        transaction.rollback()
```

```
from django.db import transaction

@commit_on_success
def some_view(request):
    ...
```

Метапрограммирование

Метакласс

```
class Base(type):

    def __new__(cls, name, bases, attrs):
        new_cls = super(Base, cls).__new__(cls, name,
                                           bases, attrs)
        setattr(new_cls, 'HACKED', "!")
        return new_cls


class Parent(object):
    __metaclass__ = Base


class Main(Parent):
    data = "child"
```

Метакласс

```
class Base(type):

    def __new__(cls, name, bases, attrs):
        new_cls = super(Base, cls).__new__(cls, name,
                                           bases, attrs)
        setattr(new_cls, 'HACKED', "!")
        return new_cls


class Parent(object):
    __metaclass__ = Base


class Main(Parent):
    data = "child"
```

Сам метакласс

Метакласс

```
class Base(type):

    def __new__(cls, name, bases, attrs):
        new_cls = super(Base, cls).__new__(cls, name,
                                           bases, attrs)
        setattr(new_cls, 'HACKED', "!")
        re    cls

class Parent(object):
    __metaclass__ = Base

class Main(Parent):
    data = "child"
```

Имя класса, который будет сгенерирован

"Parent"

"Main"

Метакласс

```
class Base(type):

    def __new__(cls, name, bases, attrs):
        new_cls = super(Base, cls).__new__(cls, name,
                                           bases, attrs)
        setattr(new_cls, 'HACKED', "!")
        return n

class Parent(object):
    __metaclass__ = Base

class Main(Parent):
    data = "child"
```

Список классов-родителей

[object]

[Parent]

Метакласс

Словарь аттрибутов будущего класса

```
class Base(type):

    def __new__(cls, name, bases, attrs):
        new_cls = super(Base, cls).__new__(cls, name,
                                           bases, attrs)
        setattr(new_cls, 'HACKED', "!")
        return new_cls


class Parent(object):
    __metaclass__ = Base


class Main(Parent):
    data = "child"
```

Метакласс

```
class Base(type):

    def __new__(cls, name, bases, attrs):
        new_cls = super(Base, cls).__new__(cls, name,
```

```
Parent
{
    '__metaclass__': < class '__main__.Base' > ,
    '__module__': '__main__'
}
```

Метакласс

```
class Base(type):

    def __new__(cls, name, bases, attrs):
        new_cls = super(Base, cls).__new__(cls, name,
```

```
Main
{
    'data': 'child',
    '__module__': '__main__'
}
```

Метакласс

```
>>> print dir(Parent)
['HACKED', '__class__', '__delattr__',
'__dict__', '__doc__', ...]

>>> print dir(Main)
['HACKED', '__class__', '__delattr__',
'__dict__', '__doc__', ..., 'data']
```

Например

Метакласс

```
def calls_counter(name, func):

    counter_attr_name = "_{0}_calls_counter".format(name)


    def counter(instance, *args, **kwargs):
        setattr(instance, counter_attr_name,
                getattr(instance, counter_attr_name, 0) + 1)

        return func(instance, *args, **kwargs)

    return counter
```

Метакласс

```
class CounterBase(type):

    def __new__(cls, name, bases, attrs):
        new_cls = super(CounterBase, cls).__new__(cls, name,
                                                  bases, attrs)
        for obj_name, obj in attrs.items():
            if callable(obj) and not isinstance(obj, type):
                setattr(new_cls, obj_name,
                        calls_counter(obj_name, obj))
        return new_cls
```

Метакласс

```
class SomeClass(object):

    __metaclass__ = CounterBase

    def method1(self):
        pass

    def method2(self):
        pass
```

Метакласс

```
>>> p = SomeClass()
>>> p.method1()
>>> print p._method1_calls_counter
1
>>> p.method1()
>>> print p._method1_calls_counter
2
>>> p.method2()
>>> print p._method2_calls_counter
1
```

Python 2.X

```
__metaclass__ = Meta
```

Python 3.X

```
class X(metaclass=Meta)
```

Другие будничные проблемы

Повторные вычисления

Особенности ORM

```
class SomeClass():

    def get_queryset(self):
        return Entry.objects.all()

    def some_action(self):
        qs = self.get_queryset()
        return some_calculations(qs)

    def another_action(self):
        qs = self.get_queryset()
        return another_calculations(qs)
```

Особенности ORM

```
>>> some_obj = SomeClass()
>>> some_obj.some_action()
>>> len(connection.queries)
1
>>> some_obj.another_action()
>>> len(connection.queries)
2
```

Обычное решение

```
class SomeClass():

    def __init__():
        self.queryset = None

    def get_queryset(self):
        if not self.queryset:
            self.queryset = Entry.objects.all()
        return self.queryset

    def some_action(self):
        qs = self.get_queryset()
        return some_calculations(qs)

    def another_action(self):
        qs = self.get_queryset()
        return another_calculations(qs)
```

Обычное решение

```
>>> some_obj = SomeClass()
>>> some_obj.some_action()
>>> len(connection.queries)
1
>>> some_obj.another_action()
>>> len(connection.queries)
1
```

А если методов много?

```
class SomeClass():

    def get_queryset(self):
        return Entry.objects.all()

    def get_queryset_1(self):
        return Entry.objects.filter(something=1)

    ...

    def get_queryset_100(self):
        return Entry.objects.filter(something=100)
```

Или классов

```
class SomeClass():

    def get_queryset(self):
        return Entry.objects.all()

class SomeClass1():

    def get_queryset(self):
        return Entry.objects.all()

class SomeClass2():

    def get_queryset(self):
        return Entry.objects.all()
```

Так лень

```
class SomeClass():

    def __init__():
        self.queryset = None
        self.queryset_1 = None
        ...
        self.queryset_100 = None
```

Декоратор

```
def methodcache(name=None):

    def cache_decorator(func):
        if not name:
            cache_name = u"__{0}".format(func.__name__)
        else:
            cache_name = name

        def field_func(self, *args, **kwargs):
            if kwargs.pop("force") or not hasattr(self,
                                                  cache_name):
                setattr(self, cache_name, func(self, *args,
                                               **kwargs))
            return getattr(self, cache_name)
        return field_func
    return cache_decorator
```

Декоратор

```
class SomeClass():

    @methodcache(name="queryset")
    def get_queryset(self):
        return Entry.objects.all()

    @methodcache()
    def get_queryset_1(self):
        return Entry.objects.filter(something=1)

class SomeClass1():

    @methodcache(name="queryset")
    def get_queryset(self):
        return Entry.objects.all()
```

Обычное решение

```
>>> some_obj = SomeClass()
>>> some_obj.some_action()
>>> len(connection.queries)
1
>>> some_obj.another_action()
>>> len(connection.queries)
1
>>> some_obj.another_action(force=True)
>>> len(connection.queries)
2
```

Можно и Метакласс сделать

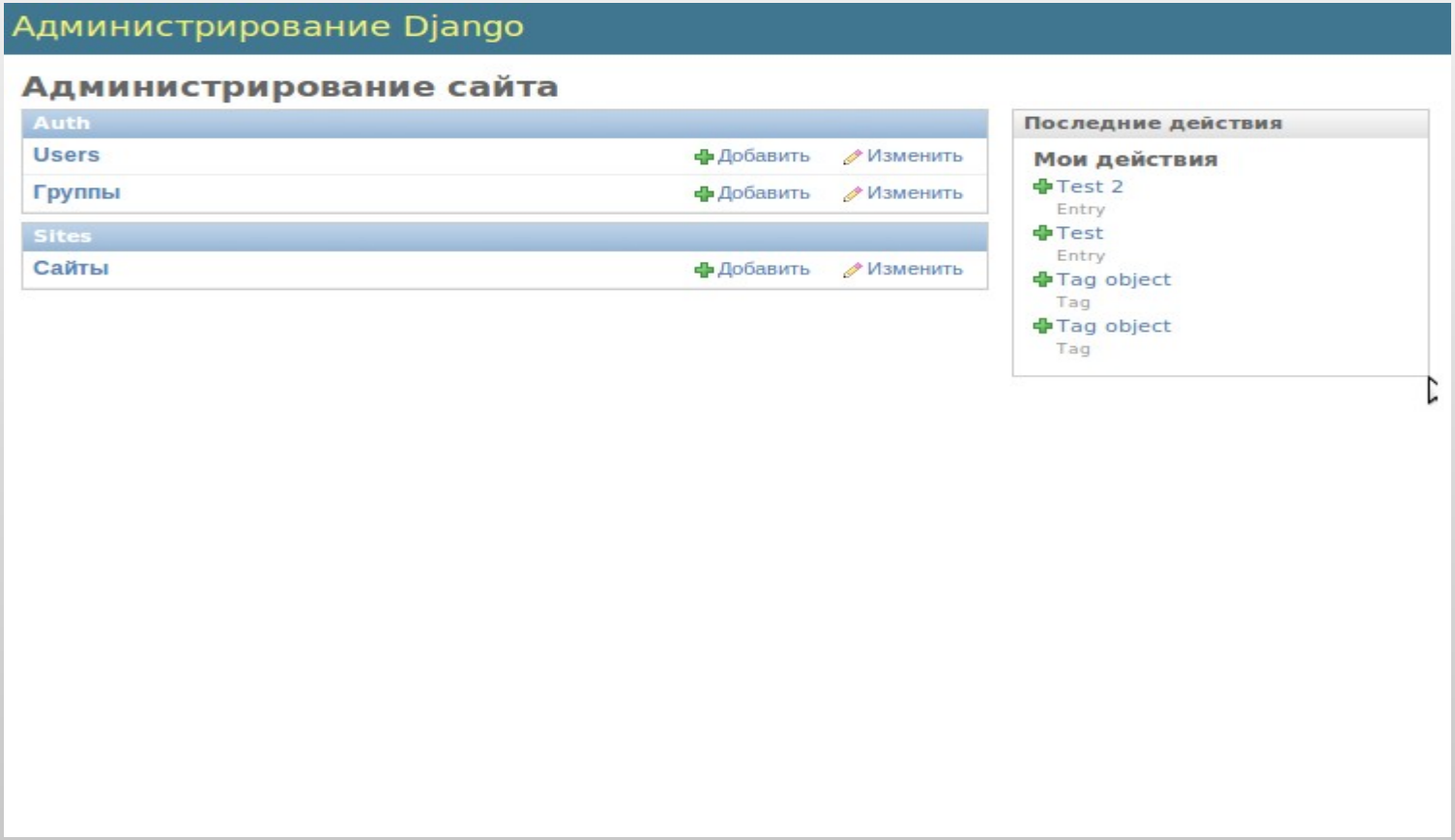
contrib.admin

Автоматическая регистрация

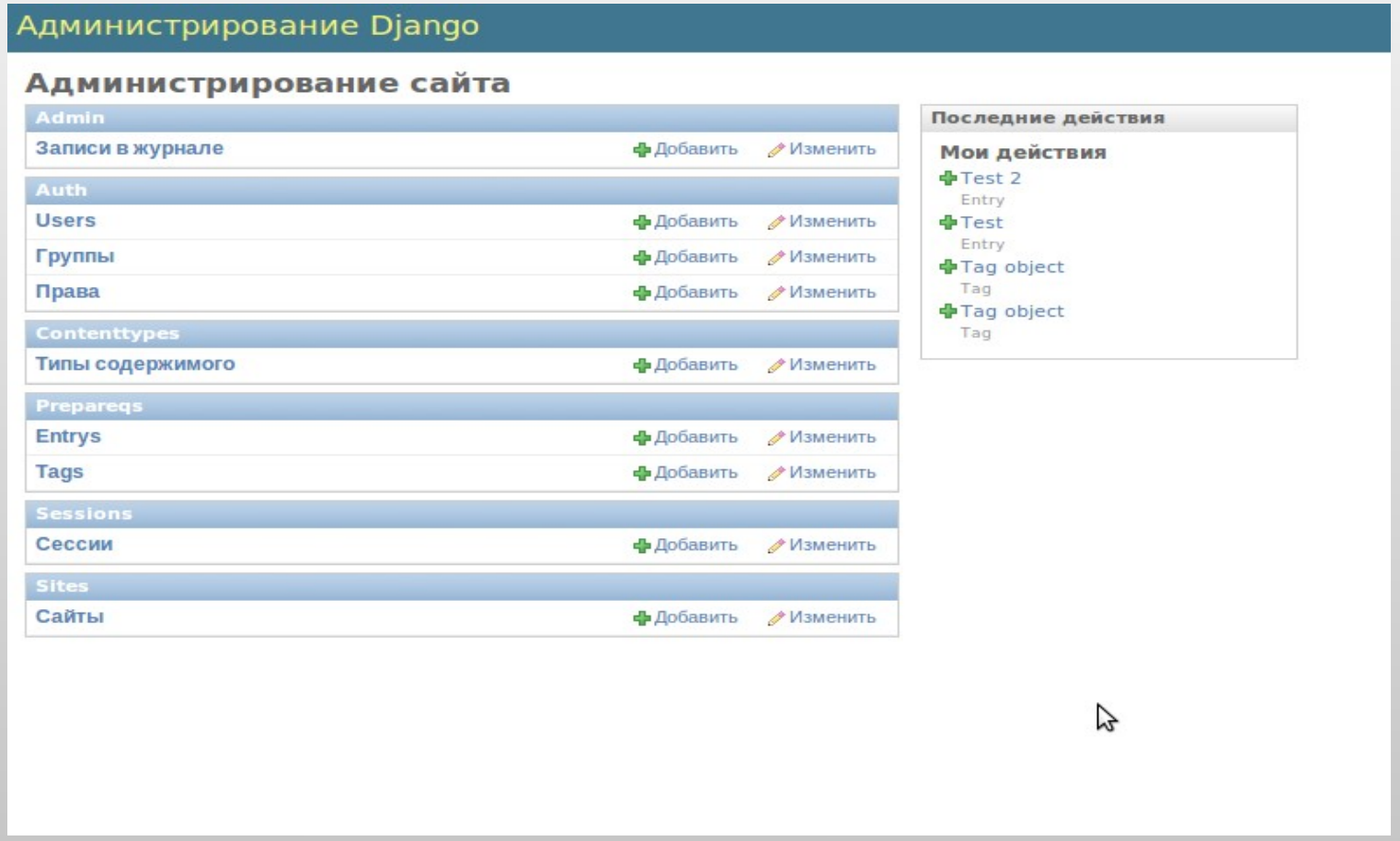
```
def autoregister():
    for model in get_models():
        try:
            admin.site.register(model)
        except AlreadyRegistered:
            Pass

# urls.py
admin.autodiscover()
autoregister()
```

Без авторегистрации



С авторегистрацией



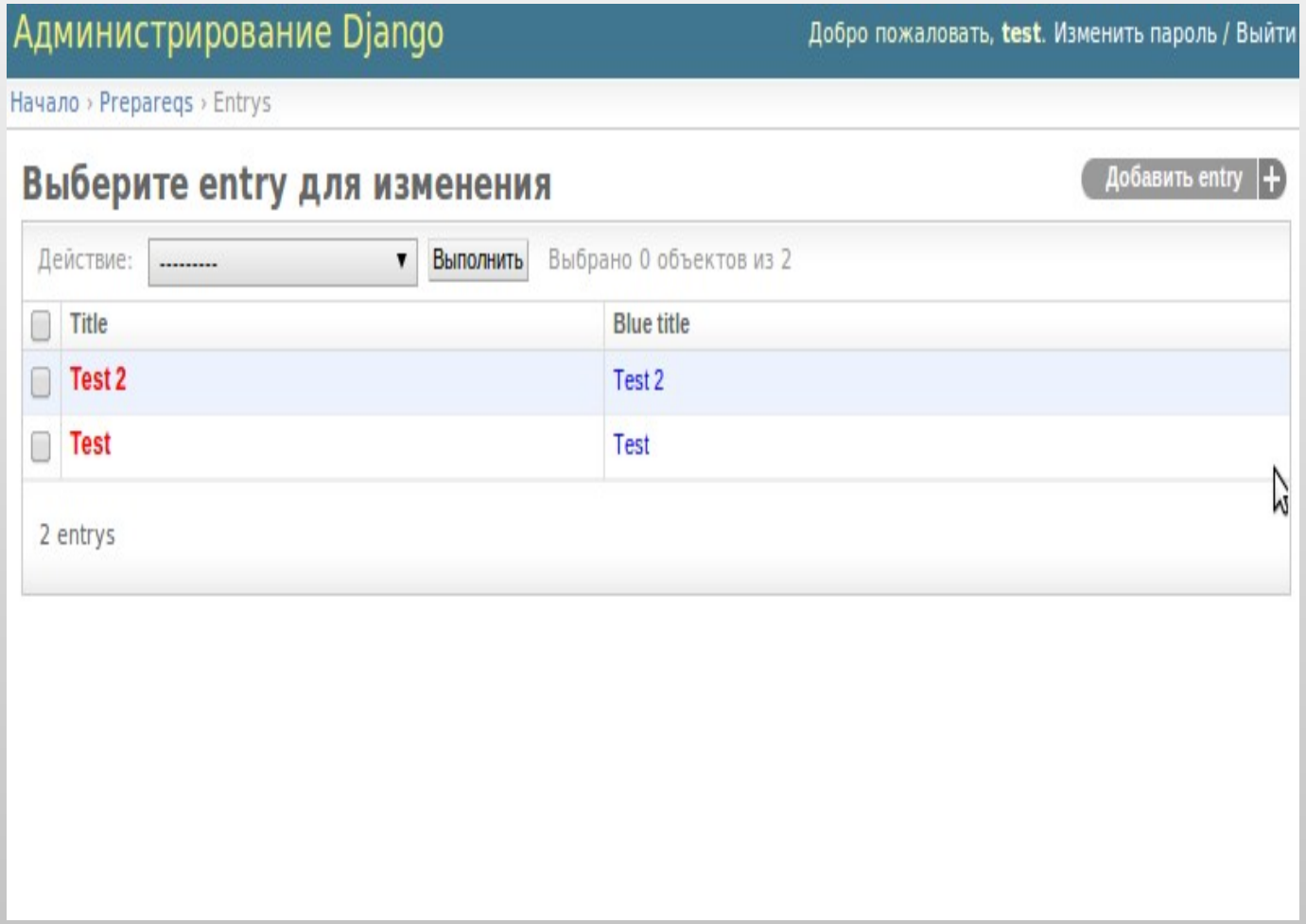
Методы-поля

```
class EntryAdmin(admin.ModelAdmin):
    list_display = ('red_title', 'blue_title')

    def red_title(self, obj):
      return u"""
       <font color='red'>{0}</font>
      """.format(obj.title)
    red_title.short_description = 'Title'
    red_title.allow_tags = True

    def blue_title(self, obj):
      return u"""
       <font color='blue'>{0}</font>
      """.format(obj.title)
    blue_title.allow_tags = True
```

Результат


Администрирование Django
Добро пожаловать, test. Изменить пароль / Выйти
Начало › Prepareqs › Entrys
Выберите entry для изменения
Добавить entry
Действие:

Выполнить
Выбрано 0 объектов из 2
Title
Blue title
Test 2
Test 2
Test
Test
2 entrys

И снова декоратор

```
def func_name(func):
    name = func.__name__.capitalize()
    return name.replace("_". " ")

def field(name=None, safe=False):

    def field_decorator(func):

        def field_func(self, *args, **kwargs):
            return func(self, *args, **kwargs)

        field_func.short_description = name or \
                                       func_name(func)
        field_func.allow_tags = safe

        return field_func
    return field_decorator
```

Методы-поля

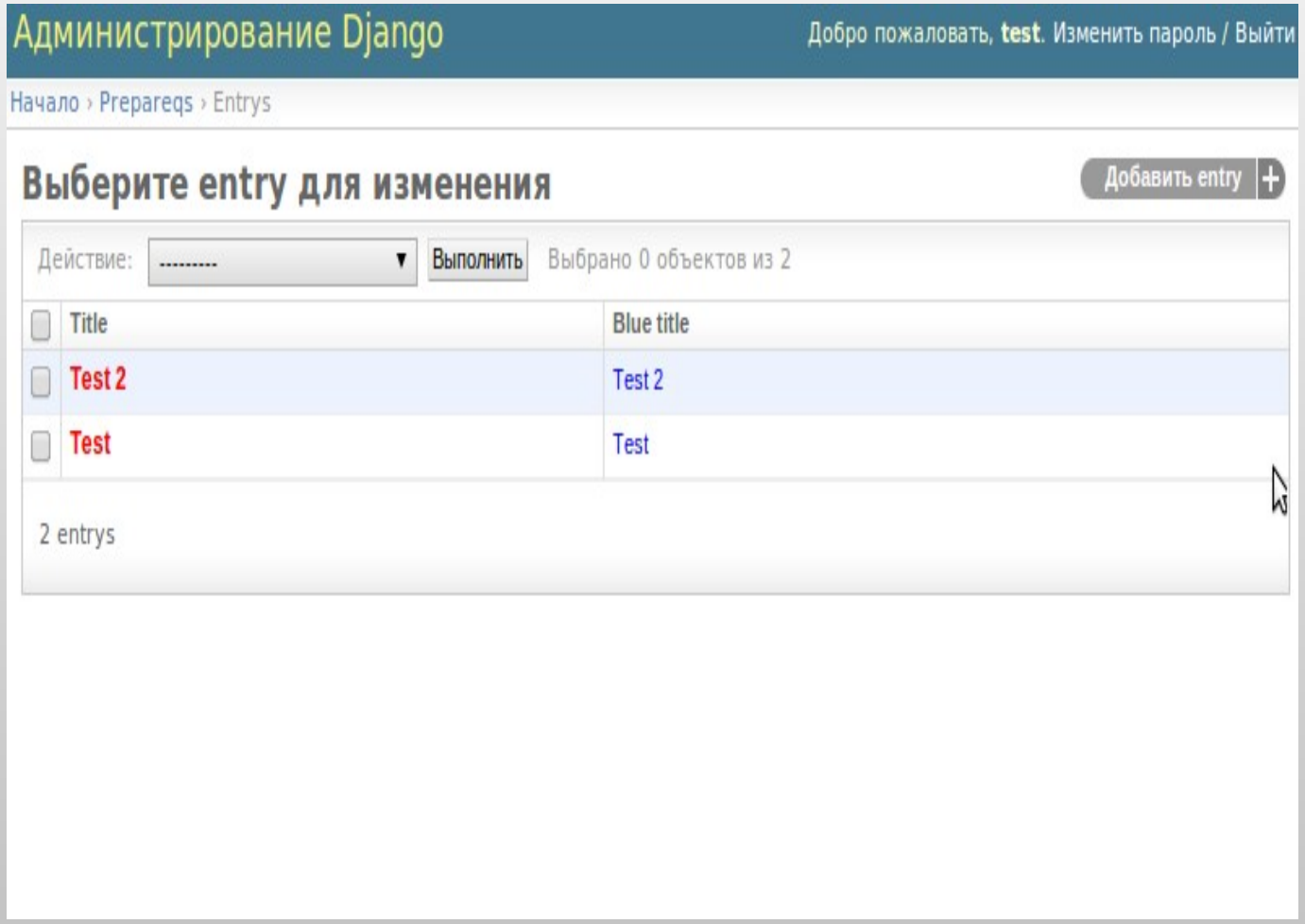
```
class EntryAdmin(admin.ModelAdmin):

    list_display = ('red_title', 'blue_title')

    @field("Title", True)
    def red_title(self, obj):
        return u"""
            <font color='red'>{0}</font>
        """.format(obj.title)

    @field(safe=True)
    def blue_title(self, obj):
        return u"""
            <font color='blue'>{0}</font>
        """.format(obj.title)
```

Результат


Администрирование Django
Добро пожаловать, test. Изменить пароль / Выйти
Начало › Prepareqs › Entrys
Выберите entry для изменения
Добавить entry
Действие:

Выполнить
Выбрано 0 объектов из 2
Title
Blue title
Test 2
Test 2
Test
Test
2 entrys

Если не видно разницы,
зачем писать больше? =)

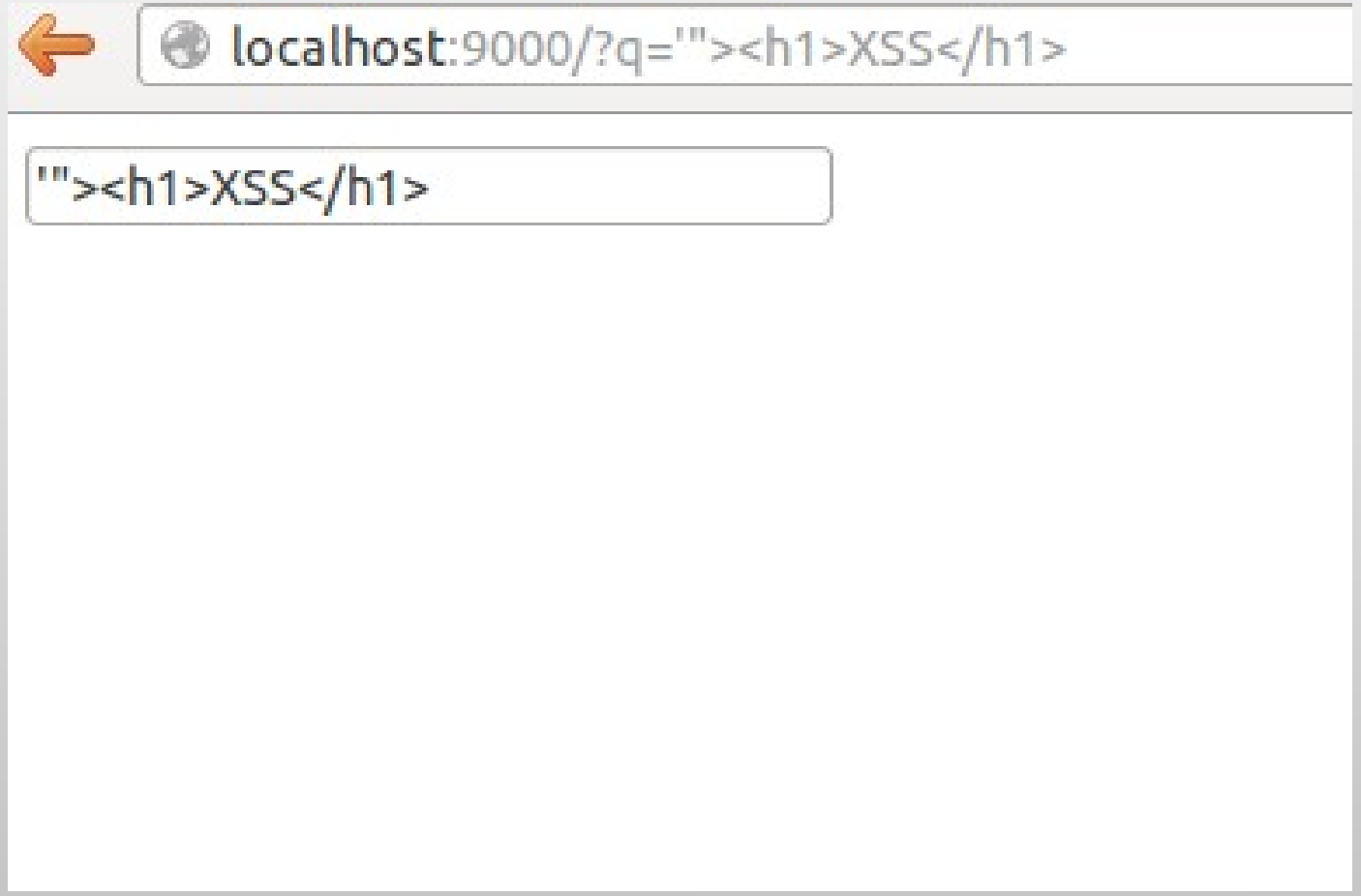
Будничные риски

XSS. Пассивные и хранимые

Все данные полученные от пользователя, потенциально опасны

Django вроде сама экранирует?

```
<input type="text" name="q"
value="{{ request.GET.q }}">
```



Почти...

```
{% cycle request.GET.q "" %}
```

и

```
{% firstof request.GET.q1 "" %}
```

cycle и firstof

Выход. Django <= 1.5

```
{% filter force_escape %}
    {% cycle request.GET.q "" %}

    {% firstof request.GET.q1 "" %}

{% endfilter %}
```

Выход. Django < 1.8

```
{% load cycle from future %}
{% load firstof from future %}

{% cycle request.GET.q "" %}

{% firstof request.GET.q1 "" %}
```

Django >= 1.8

```
{% cycle request.GET.q "" %}

{% firstof request.GET.q1 "" %}
```

Старайтесь не делать WYSIWYG и |safe

Составьте список разрешенных тегов и обрезайте лишние

На сервере

html5lib, BeautifulSoup

Entry.objects.raw()

Только так

```
Entry.objects.raw("""
    SELECT *
    FROM
    myapp_entry
    WHERE title = %s
""", [title])
```

Но не так

```
Entry.objects.raw("""
    SELECT *
    FROM
    myapp_entry
    WHERE title = %s
""",[title])
```

Ну, и чуть-чуть не Python

Не знаю как у вас...

```
user@host:~$ gut push

No command 'gut' found, did you mean:
 Command 'cut' from package 'coreutils' (main)
 Command 'git' from package 'git-core' (main)
gut: command not found
```

gut, got, gh

Нам поможет ~/.bashrc

~/.bashrc

```
...

alias gut=git
alias got=git
alias gh=hg

...
```

~/.bashrc + alias

```
user@host:~$ gut push
Everything up-to-date

user@host:~$ got push
Everything up-to-date
```

Еще чуть-чуть лени

```
ssh -i /path/to/project/key.pem user@111.222.111.222
```

Делаем скриптик project_ssh

~/bin/project_ssh

```
#!/bin/bash

ssh -i /path/to/project/key.pem
$1@111.222.111.222
```

~~ssh -i /path/to/project/key.pem user@111.222.111.222~~

./project_ssh

It’s all about music business

Artist
Label
Club
Media
Promoter
Booking

ВОПОСЫ? – vladimir@labbler.com
FACEBOOK - facebook.com/pyhoster

Спасибо!