Django

debug and testing

План

logging
pdb
debug toolbar
extensions
testing
selenium

Дока наше все!

https://docs.djangoproject.com/en/1.7/topics/logging/

Logging

Не print'ом единым :)

```
import logging
logger = logging.getLogger(__name__)
```

```
logger.debug("Logging at DEBUG level")
logger.info("Logging at INFO level")
logger.warning("Logging at WARNING level")
logger.error("Logging at ERROR level")
```

Logging

Уровни логирования:

DEBUG: любая информация необходимая при разработке или отлавливании проблем
INFO: информирование о статусе чего-либо
WARNING: информирование о проблемах, которые не являются ошибками
ERROR: информация о ошибках
CRITICAL: критические проблемы

Logging

Имена логеров

```
import logging
logger = logging.getLogger(__name__)
# pybursa.views
```

Поддерживают наследование: pybursa.views будет перехвачен логером с именем pybursa

Имена это не обязательно имена модулей, а любые строки через точку

Logging

Конфигурация логирования

```
LOGGING = {
    'version': 1,
    'handlers': {
        'console': {
            'level': 'DEBUG',
            'class': 'logging.StreamHandler',
        },
    },
    'loggers': {
        'pybursa': {
            'handlers': ['console'],
            'level': 'DEBUG',
},},}
```

Logging

Конфигурация логирования

```
LOGGING = {
    'version': 1,
    'handlers': {
        'console': {
            'level': 'DEBUG',
            'class': 'logging.StreamHandler',
        },
    },
    'loggers': {
        'pybursa': {
            'handlers': ['console'],
            'level': 'DEBUG',
},},}
```

Logging

```
    'handlers': {
        'file': {
            'level': 'WARNING',
            'class': 'logging.FileHandler',
            'filename': os.path.join(BASE_DIR,
'debug.log'),
        },
        'console': {
            'level': 'DEBUG',
            'class': 'logging.StreamHandler',
        },
    },
```

Дока наше все!

https://docs.python.org/2/library/pdb.html

Дебагинг - pdb

Выполнение скрипта пошагово:

```
> python -m pdb script.py
```

Установка точки останова:

```
import pdb; pdb.set_trace()
```

pdb - команды

h(elp) - справка
w(here) - стектрейс текущей позиции
l(ist) - листинг текущей позиции
c(ontinue) - продолжение выполнения
a(rgs) - параметры текущей функции
q(uit) - прерывание и выход

r(eturn) - продолжить выполнение пока текущая функция не вернет значение
j(ump) - прыгнуть на номер строки
d(own) - прыгнуть внутрь функции
u(p) - выпрыгнуть из функции

Дока наше все!

https://github.com/django-debug-toolbar/django-debug-toolbar

http://django-debug-toolbar.readthedocs.org/en/1.2.2/

debug toolbar

```
pip install django-debug-toolbar
```

```
INSTALLED_APPS = (
    # ...
    'django.contrib.staticfiles',
    # ...
    'debug_toolbar',
)
```

Дальше магия!

debug toolbar

- packages versions
- time load
- settings (with defaults)
- http headers
- request
- SQL queries
- templates (with context processors)
- cache

debug toolbar

Команды:

python manage.py debugsqlshell

Выводит запросы к БД

Дока наше все!

https://github.com/django-extensions/django-extensions

http://django-extensions.readthedocs.org/en/latest/

extensions

```
pip install django-extensions

INSTALLED_APPS = (
    ...
    'django_extensions',
)
```

Дальше снова много классного!

extensions - manage commands

shell_plus - расширенный shell с предимпортами
admin_generator - создает admin файл
Продвинутые хелъперы создания структуры файлов: create_app, create_template_tags, create_command
graph_models - визуализация структуры БД
mail_debug - smtp сервер для дебаг
show_urls - текущая карта урлов
...

extensions - fields

- AutoSlugField
- CreationDateTimeField
- ModificationDateTimeField
- EncryptedCharField

Abstract TimeStampedModel with CreationDateTimeField and ModificationDateTimeField

Дока наше все!

https://docs.djangoproject.com/en/1.7/intro/tutorial05/

https://docs.djangoproject.com/en/1.7/topics/testing/

https://docs.python.org/2/library/unittest.html

testing

Все просто!

python manage.py test

Ran 0 tests in 0.000s
OK

Нужно написать первый :)

testing

```
courses
    __init__.py
    admin.py
    models.py
    tests.py
    urls.py
    views.py
```

```
from django.test import TestCase
# Create your tests here.
```

testing

```
class CourseTests(TestCase):

    def test_course_create(self):
        course1 = Course.objects.create(
                        name='PyBursa02',
                        start_date=date(2015, 2, 16))
        self.assertEqual(Course.objects.all().count(), 1)
```

testing - views

```
def test_pages(self):
    from django.test import Client
    client = Client()

    response = client.get('/courses/')
    self.assertEqual(response.status_code, 200)
```

```
course1 = Course.objects.create(name='PyBursa02',
                                start_date=date(2015, 2, 16))

response = client.get('/courses/1/')
self.assertEqual(response.status_code, 200)
self.assertContains(response, "PyBursa02")
```

Дока наше все!

https://docs.djangoproject.com/en/1.7/topics/testing/tools/#liveservertestcase

http://docs.seleniumhq.org/docs/

testing - selenium

pip install selenium

```
from django.test import LiveServerTestCase
from selenium.webdriver.firefox.webdriver import WebDriver

class MySeleniumTests(LiveServerTestCase):
    @classmethod
    def setUpClass(cls):
        cls.selenium = WebDriver()
        super(MySeleniumTests, cls).setUpClass()

    @classmethod
    def tearDownClass(cls):
        cls.selenium.quit()
        super(MySeleniumTests, cls).tearDownClass()
```

testing - selenium

```
def test_login(self):
    self.selenium.get('%s%s' %
                      (self.live_server_url, '/login/'))
    username_input = self.selenium.find_element_by_name(
                                                "username")
    username_input.send_keys('myuser')
    password_input = self.selenium.find_element_by_name(
                                                "password")
    password_input.send_keys('secret')
    self.selenium.find_element_by_xpath(
                        '//input[@value="Log in”]').click()
```