

Знакомство с Django. Подготовка и запуск проекта.



Максим
Филипенко



Максим Филипенко

Software Engineer at Marilyn Systems

 mfilipenko.yandex.ru

|  [prokactus](#)

|  [prokactus](#)

План занятия

1. [Что такое Django. Установка и запуск](#)
2. [Проект и приложение](#)
3. [Клиент и сервер](#)
4. [MVC и Django](#)
5. [Работа с урлами. Роутинг в Django](#)
6. [Как дебажить Django-проект](#)
7. [Что почитать](#)



Что такое DJANGO



ЧТО ТАКОЕ DJANGO

Django — фреймворк для быстрого создания веб-приложений, полностью написанный на Python.

Django является очень популярным проектом и используется многими крупными компаниями. Исходный код Django доступен на Github:

<https://github.com/django/django>



ПОЧЕМУ DJANGO ПРЕКРАСНО ПОДХОДИТ ДЛЯ ОБУЧЕНИЯ

- Грамотно спроектированная архитектура. Приложение легко организовать и даже новичкам будет понятно, где и что требуется искать.
- Прозрачная работа с базой данных. Вы всегда понимаете, какой именно запрос будет сгенерирован к базе данных.
- Серьезное отношение к безопасности. CSRF, XSS, sql-инъекция и прочим уязвимостям уделено много внимания. Вы сможете писать безопасные приложения легко.
- Огромное количество библиотек и уже написанного кода.
- Подробная документация (к сожалению, не переведена на русский).



УСТАНОВКА

```
$ pip install Django
```

Чтобы убедиться, что все установилось корректно, в консоли выполните команду:

```
$ python -m django --version
```

Готово!



СОГЛАШЕНИЯ ИЛИ КОНФИГУРАЦИЯ

Подход Ruby on Rails — «магия», но работает.

Подход Django — всё можно настроить и поменять под свои нужды (ну почти всё).



Проект и приложение



ЧТО ТАКОЕ ПРОЕКТ И ПРИЛОЖЕНИЕ

Под проектом можно понимать полноценный сайт. Это коллекция настроек, база данных и подключенные приложения.

Приложение — это изолированный кусок функциональности. Приложения могут переиспользоваться в различных проектах. Ближайшая аналогия — модули в Python.

ПРИМЕРЫ

- Приложение для работы с пользователями;
- Приложение для работы с email-рассылкой;
- Приложение для работы с заказами.

В идеале достичь полной переиспользуемости трудно и не всегда нужно.

Но в некоторых случаях может быть полезно.

СОЗДАНИЕ ПРОЕКТА

Запустите:

```
$ django-admin startproject django_netology
```

После этого у вас появится директория `django_netology`.



СТРУКТУРА ПРОЕКТА

Содержимое директории:

```
manage.py  
django_netology  
django_netology/__init__.py  
django_netology/settings.py  
django_netology/urls.py  
django_netology/wsgi.py
```

СОЗДАНИЕ ПРИЛОЖЕНИЯ

Приложение в Django — это своеобразный модуль с похожей функциональностью. Например, приложение для работы с email, с пользователями и т.д.

Создание приложения:

```
$ cd django_netology  
$ ./manage.py startapp app
```

`manage.py` — запускает команды в контексте Django-приложения.



Клиент и сервер



ЗАПРОС-ОТВЕТ

Клиент — инициирует запрос.

Сервер — отвечает клиенту на его запрос.

Что происходит, когда пользователь делает запрос в браузере?

<https://github.com/alex/what-happens-when>

КЛИЕНТ И СЕРВЕР

Веб-приложение реализует клиент-серверное взаимодействие. Пользователи шлют запросы к серверу, он выдает им результат в виде HTML или JSON данных.

Django-проект выступает в роли сервера. Для того, чтобы запустить проект выполните команды:

```
$ ./manage.py migrate # создает базу данных  
$ ./manage.py runserver # запускает проект
```



ЗАПУЩЕННЫЙ DJANGO-ПРОЕКТ



MVC и Django



ТЕПЕРЬ НАПИШЕМ ЧТО-НИБУДЬ СВОЁ

Django генерирует структуру проекта самостоятельно. Благодаря этому даже новые разработчики знают, где и что можно искать.

При разработке Django-приложений **очень важно** придерживаться соглашений.

Проекты на Django должны придерживаться паттерна MVC: model-viewcontroller (модель-представление-контроллер)

<https://habr.com/post/181772/>

DJANGO И РАЗДЕЛЕНИЕ ОТВЕТСТВЕННОСТИ

- Управление логикой при ответе -> view;
- Как будет выглядеть страница -> template;
- Состояние приложения -> model.

<https://docs.djangoproject.com/en/2.1/faq/general/#django-appears-to-be-amvc-framework-but-you-call-the-controller-the-view-and-the-view-the-template-how-come-you-don-t-use-the-standard-names>

Правило: **не мешать всё в одну кучу.**

base/views.py:

```
from django.http import HttpResponseRedirect
from django.shortcuts import render


def home_view(request):
    return HttpResponseRedirect('Здесь будет сайт!')
```

Необходимо добавить view-функцию в обработчик урлов
django_netology/urls.py:

```
...

urlpatterns = [
    path('', home_view, name='home'),
    ...
]
```

Рассмотрим структуру описания урла подробнее.



```
...  
urlpatterns = [  
    path('', home_view, name='home'),  
    ...  
]
```


- Первый параметр — фактический адрес, который будет указан в адресной строке браузера.
- Второй параметр — view-функция, которая будет вызвана при обработке запроса.
- `name` — позволяет получать конкретный урл по имени. Это позволяет приложению не ломаться, если урлы будут меняться и делает код более понятным.

Как получить урл по имени:

```
urlpatterns = [  
    path('', home_view, name='home'),  
    path('profile', profile_view, name='profile'),  
    path('long/address/orders', orders_view, name='orders')  
]
```

С помощью `manage.py shell` можно проверить работу:

```
> from django.urls import reverse  
  
> reverse('orders')  
'/long/address/orders'  
  
> reverse('profile')  
'/profile'
```

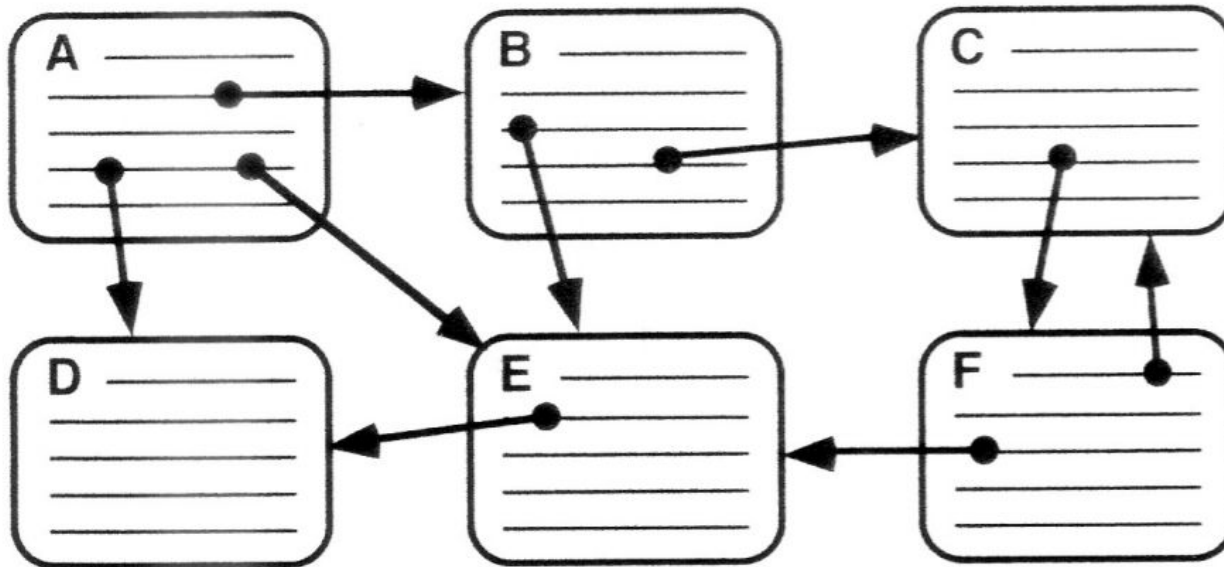



В функцию `reverse` можно также передавать параметры. Это нужно для формирования динамических урлов. Например:

```
reverse('accounts', kwargs={'account_id': 100})
```

Но об этом мы будем говорить подробнее на лекции про обработку запросов.

VIEW И ГИПЕРТЕКСТ



Каждый переход по ссылке обрабатывается view,
ответ возвращается пользователю.

КАК ДЕБАЖИТЬ DJANGO-ПРОЕКТ

- **print-функции;**

Django-проект — это Python приложение. Поэтому можно использовать возможности Python и использовать print'ы для дебага и отладки кода.

- **Точки останова (они же breakpoints);**

Удобнее всего использовать в IDE Pycharm.

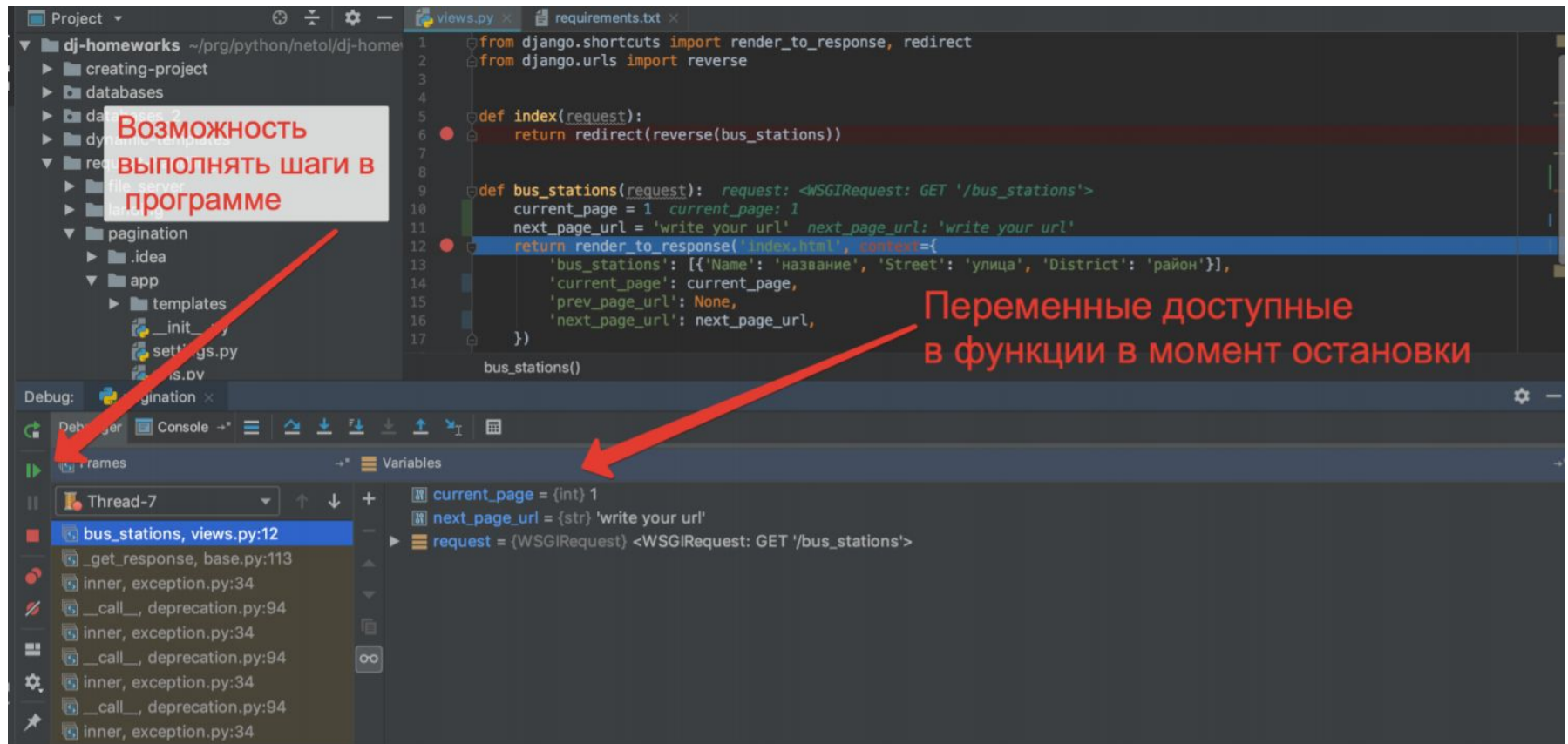
- **Сообщения об ошибках Django;**

Средство фреймворка. Если включен DEBUG-режим (по умолчанию во всех домашних работах именно так), то Django собирает и агрегирует информацию об ошибке.

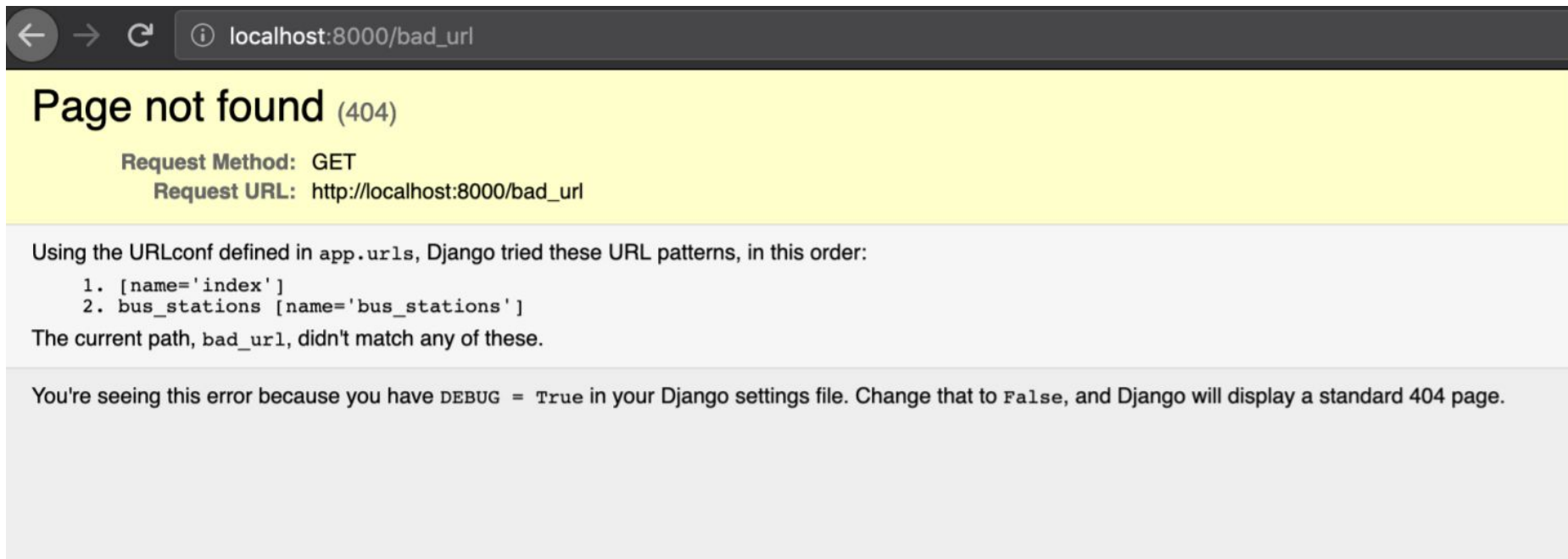
- **manage.py shell ;**


Запускает интерактивный интерпретатор в контексте Django-проекта.

ДЕМОНСТРАЦИЯ: ТОЧКИ ОСТАНОВА



ДЕМОНСТРАЦИЯ: ИНФОРМАЦИЯ ОБ ОШИБКЕ





Отладка — **очень важный процесс!** Сохраните себе эту информацию и используйте всегда при работе с домашними работами.

Помните, что проект на Django — это тот же Python-код, который выполняется интерпретатором.

ЧТО ПОЧИТАТЬ

- <https://docs.djangoproject.com/> — официальный сайт с документацией (английский);
- <https://developer.mozilla.org/ru/docs/Learn/Server-side/Django> — но не все статьи переведены;
- <https://tutorial.djangogirls.org/ru/> проект Django Girls, но полезно будет всем.



ИТОГИ

Сегодня на занятии мы рассмотрели:

1. как установить Django и создать проект и приложение;
2. как написать простые страницы;
3. как дебажить Django-проект.



Домашнее задание

Давайте посмотрим ваше [домашнее задание](#).

- Вопросы по домашней работе задаём в чате Slack!
- Задачи можно сдавать по частям.
- Зачёт по домашней работе проставляется после того, как приняты **все задачи**.

**Задавайте вопросы и
пишите отзыв о лекции!**

Максим Филипенко



mfilipenko.yandex.ru



[prokactus](https://github.com/prokactus)



[prokactus](https://facebook.com/prokactus)