

Знакомство с Django. Подготовка и запуск проекта.



Александр
Иванов



Александр Иванов

Ведущий инженер-программист в
Лаборатории компьютерного моделирования

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?



Грамотно спроектированная архитектура



Прозрачная работа с базой данных



Серьезное отношение к безопасности



Огромное количество библиотек и написанного кода



Подробная документация *(на английском)*



Установка

Для установки **Django**, выполните команду в консоли:

```
$ pip install django
```

Чтобы убедиться, что все установилось **корректно**:

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



Проект и приложение

Что такое проект и приложение

Под **проектом** можно понимать **полноценный сайт**. Это:

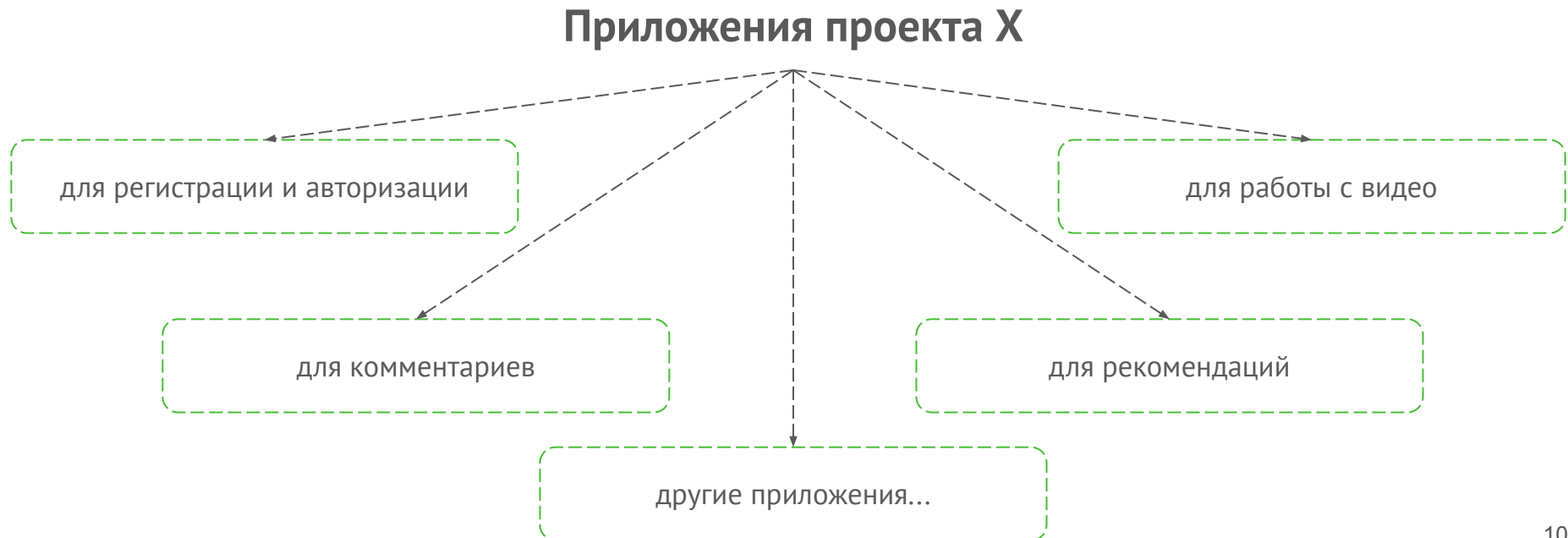
- коллекция настроек;
- база данных;
- подключенные приложения.

Например, YouTube - это **проект**.



Что такое проект и приложение

Приложение — это изолированная часть функциональности. Приложения **могут переиспользоваться** в различных проектах. Ближайшая аналогия — модули в Python.



Создание проекта

Запустите:

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

После этого у вас появится директория *django_netology*.





Структура проекта

Содержимое директории:

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

Создание приложения

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

Создание приложения:

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

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

Демонстрация создания проекта и приложения

Hey, ik ben Arnold!

Ik ben een front-end developer en student [application.nl](#)

</p>

Werkzaamheden

</article>

</section>

<section id="skills--section">

<h1 class="section--header">

My Skills.

</h1>

<section id="skills--section--wrap">

body main section.wrapper--page nav

VS Code Version Control Terminal



Клиент и сервер

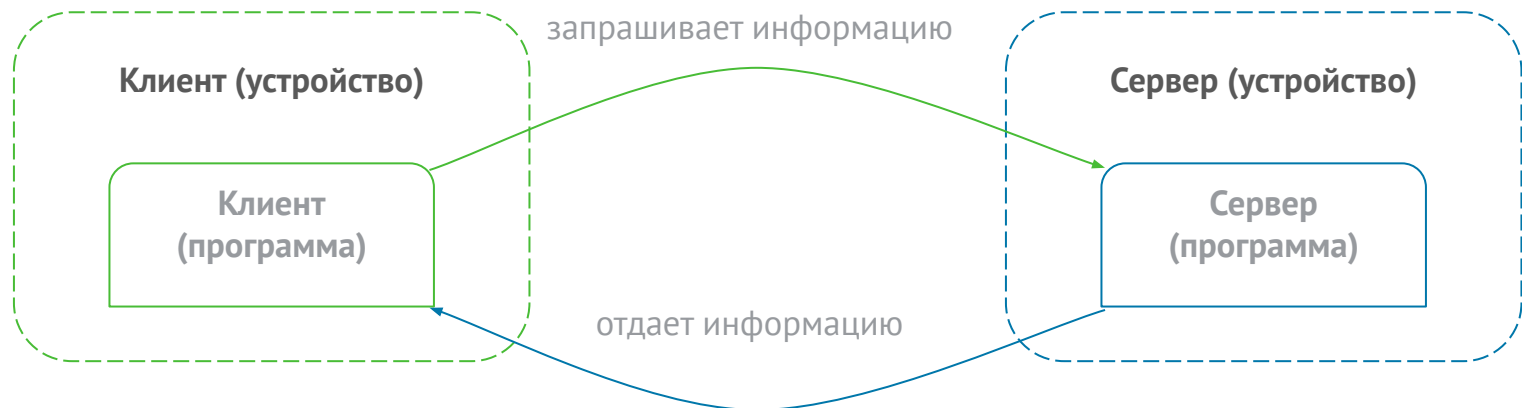
Клиент и сервер

Клиент:

1. программа, которая хочет получить информацию;
2. физическое устройство, на котором работает программа-клиент.

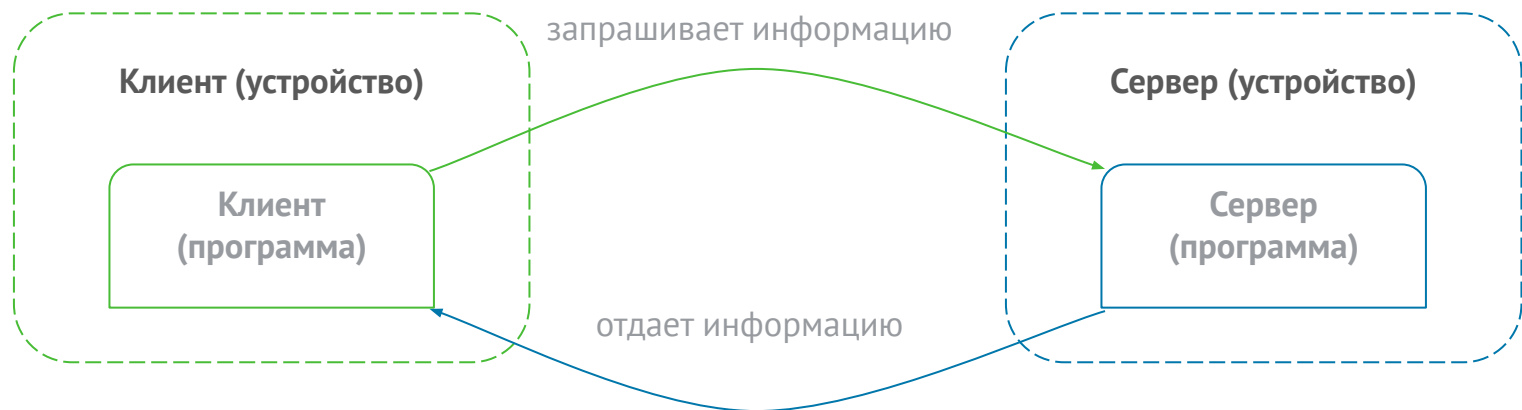
Сервер:

1. специальная программа, которая дает информацию;
2. физическое устройство, на котором запущена программа-сервер.



Клиент и сервер

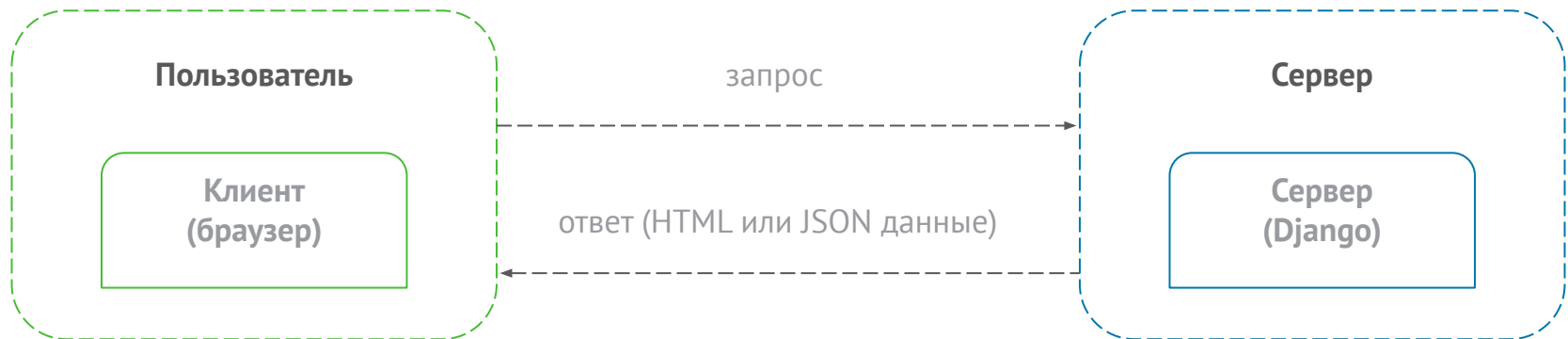
Обычно эти программы расположены на **разных вычислительных машинах** и взаимодействуют между собой по различным **протоколам**, но они могут быть расположены и на одной машине.



Клиент и сервер

Веб-приложение реализует **клиент-серверное взаимодействие**.

Пользователи шлют запросы к серверу, он выдает им результат в виде HTML или JSON данных.



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:

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

Запущенный Django-проект



MVC и Django

Теперь напишем что-нибудь свое

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-view-controller` (*модель-представление-контроллер*).

Дополнительная информация про MVC: <https://habr.com/post/181772/>

DJANGO и разделение ответственности

- управление логикой при ответе -> `view`;
- как будет выглядеть страница -> `template`;
- состояние приложения -> `model`.

[Дополнительный материал](#)

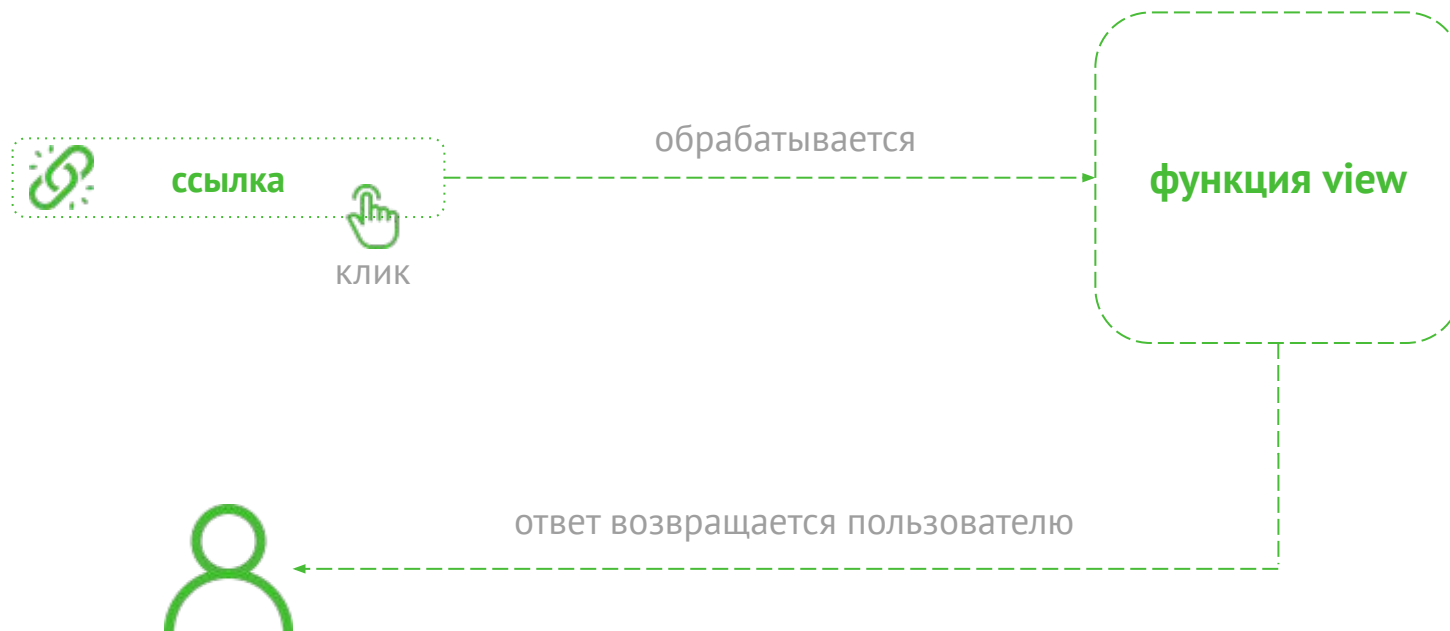
Правило: не мешать всё в одну кучу

Работа с урлами

Роутинг в Django

View и гипертекст

Пользователь взаимодействует с системой через гипертекст: клики, скролы, нажатия клавиш. Сервер реагирует на эти взаимодействия и с помощью view подготавливает новое состояние — ответ.



View

base/views.py:

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

def home_view(request):
    return HttpResponse('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

Маршрутизация

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

- `''` (первый параметр) — **фактический адрес**, который будет указан в адресной строке браузера;
- `home_view` (второй параметр) — **view-функция**, которая будет вызвана при обработке запроса;
- `name` (третий параметр) — позволяет получать конкретный **урл по имени**. Это позволяет приложению не ломаться, если урлы будут меняться и делает код более понятным.

Демонстрация view

Reverse

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'
```

Reverse

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

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

Демонстрация reverse



Как дебажить Django-проект

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.

manage.py shell

Запускает **интерактивный интерпретатор** в контексте Django-проекта.

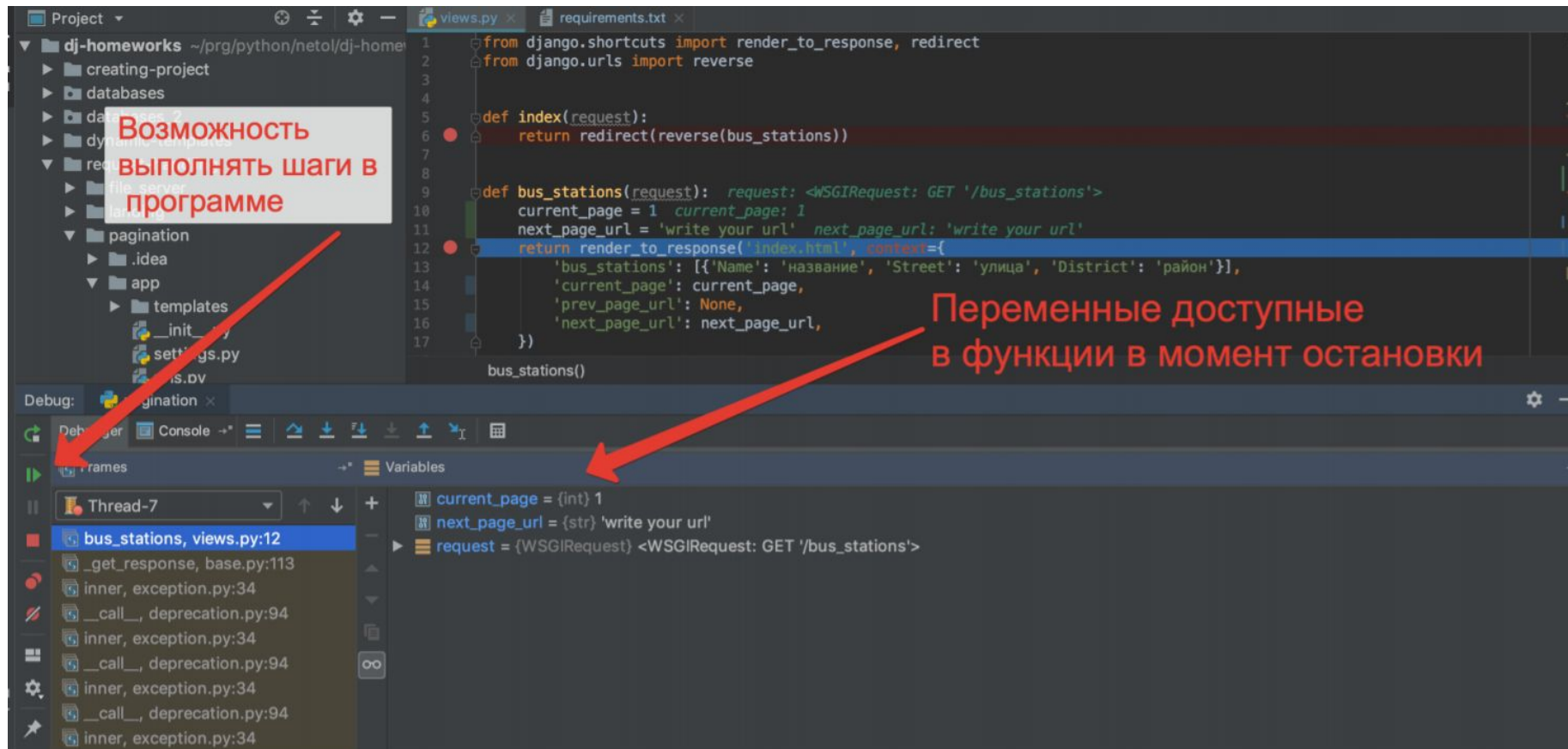
Точки останова (они же *breakpoints*)

Удобнее всего использовать в IDE Pycharm или VS Code.

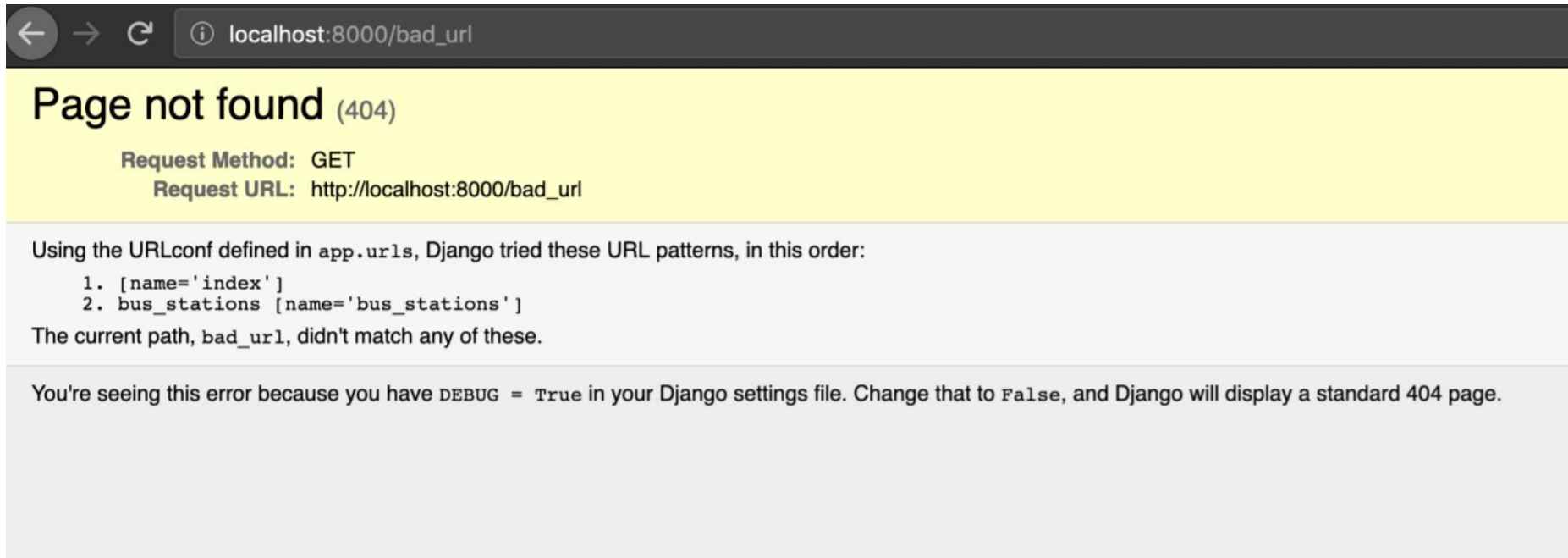
Сообщения об ошибках Django

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

Демонстрация: точки останова



Демонстрация: информация об ошибке



The screenshot shows a web browser window with the address bar displaying `localhost:8000/bad_url`. The page content is as follows:

Page not found (404)

Request Method: GET
Request URL: http://localhost:8000/bad_url

Using the URLconf defined in `app.urls`, Django tried these URL patterns, in this order:

1. `[name='index']`
2. `bus_stations [name='bus_stations']`

The current path, `bad_url`, didn't match any of these.

You're seeing this error because you have `DEBUG = True` in your Django settings file. Change that to `False`, and Django will display a standard 404 page.

Отладка

```

</script>
</head>
<body>
  <main>
    <section class="wrapper--page">
      <nav>
        <ul>
          <li id="left--item"></li>
          <li class="right--item"><a href="#contact--section" class="nav">
          <li class="right--item"><a href="#work--section" class="nav">
          <li class="right--item"><a href="#skills--section" class="nav">
        </ul>
      </nav>
      <section id="intro--section">
        <article id="intro--section--text">
          
          <h1 class="section--header">
            Hey, ik ben Arnold!
          </h1>
          <p class="section--text">
            Ik ben een front-end developer en student applicatieontwikkelaar
          </p>
          <a href="#work--section" class="pink-button scroll" id="intro--button">
          </article>
        </section>
        <section id="skills--section">
          <h1 class="section--header">
            Mijn Skills.
          </h1>
          <section id="skills--section--wrap">

```

Отладка

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

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



Чему мы научились?

- **Научились** устанавливать Django и создавать проект и приложение
- **Узнали** как писать простые страницы
- **Научились** дебажить Django-проект

Домашнее задание

Давайте посмотрим ваше [домашнее задание](#).

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

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

Александр Иванов