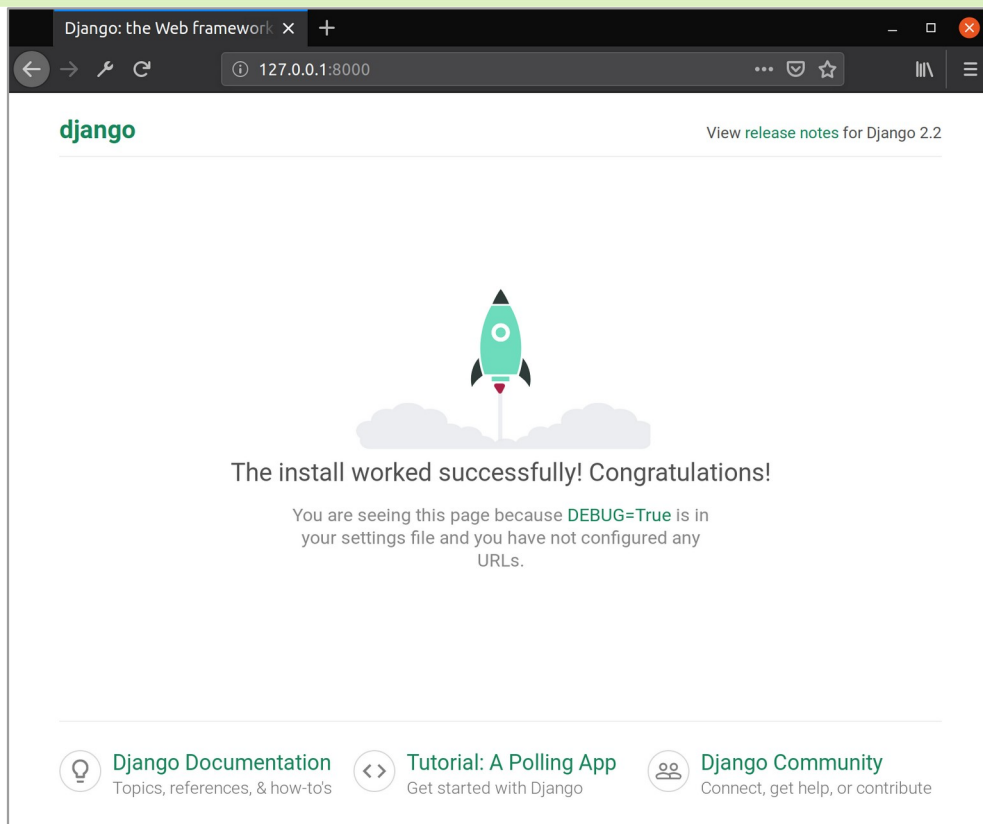


Django Tutorial (April 23, 2022)



23.04.2022

Ильина А. В. (ЛИТ ОИЯИ, ГУ "Дубна")

Документация

Django-документация:
<https://docs.djangoproject.com/>

Немного истории и основных понятий

- **Авторы:** Adrian Holovaty (Адриан Головатый), Simon Willison (Саймон Виллисон)
- **Первый выпуск:** 21 июля 2005 (16 лет назад)
- *Джанго Рейнхардт* — французский джазовый гитарист — музыкант, в честь которого получил название фреймворк
- Один из основных принципов фреймворка — **DRY** (англ. Don't repeat yourself)
- Для работы с базой данных Django использует собственный **ORM**, в котором модель данных описывается классами Python, и по ней генерируется схема базы данных



Возможности Django

- **ORM**, API доступа к БД с поддержкой **транзакций**
- **встроенный интерфейс администратора** с уже имеющимися переводами на многие языки
- **диспетчер URL** на основе *регулярных выражений*
- расширяемая **система шаблонов** с тегами и наследованием
- система **кеширования**
- интернационализация
- подключаемая архитектура приложений, которые можно устанавливать на любые Django-сайты
- «generic views» — шаблоны функций контроллеров
- **авторизация и аутентификация**, подключение внешних модулей аутентификации: LDAP, OpenID и проч.
- система фильтров («middleware») для построения дополнительных обработчиков запросов, как например включённые в дистрибутив фильтры для кеширования, сжатия, нормализации URL и поддержки анонимных сессий
- библиотека для работы с **формами** (наследование, **построение форм по существующей модели БД**)
- встроенная автоматическая документация по тегам шаблонов и моделям данных, доступная через административное приложение

Помимо возможностей, встроенных в ядро фреймворка, существуют пакеты, расширяющие его возможности. Возможности, предоставляемые пакетами, а также полный перечень пакетов удобно отслеживать через специальный ресурс — **www.djangopackages.com**.

Где используется Django?

- Instagram, 
- Disqus, 
- Mozilla, 
- The Washington Times, 
- Pinterest,  **Pinterest**
- YouTube,  **YouTube**
- Google 
- ...

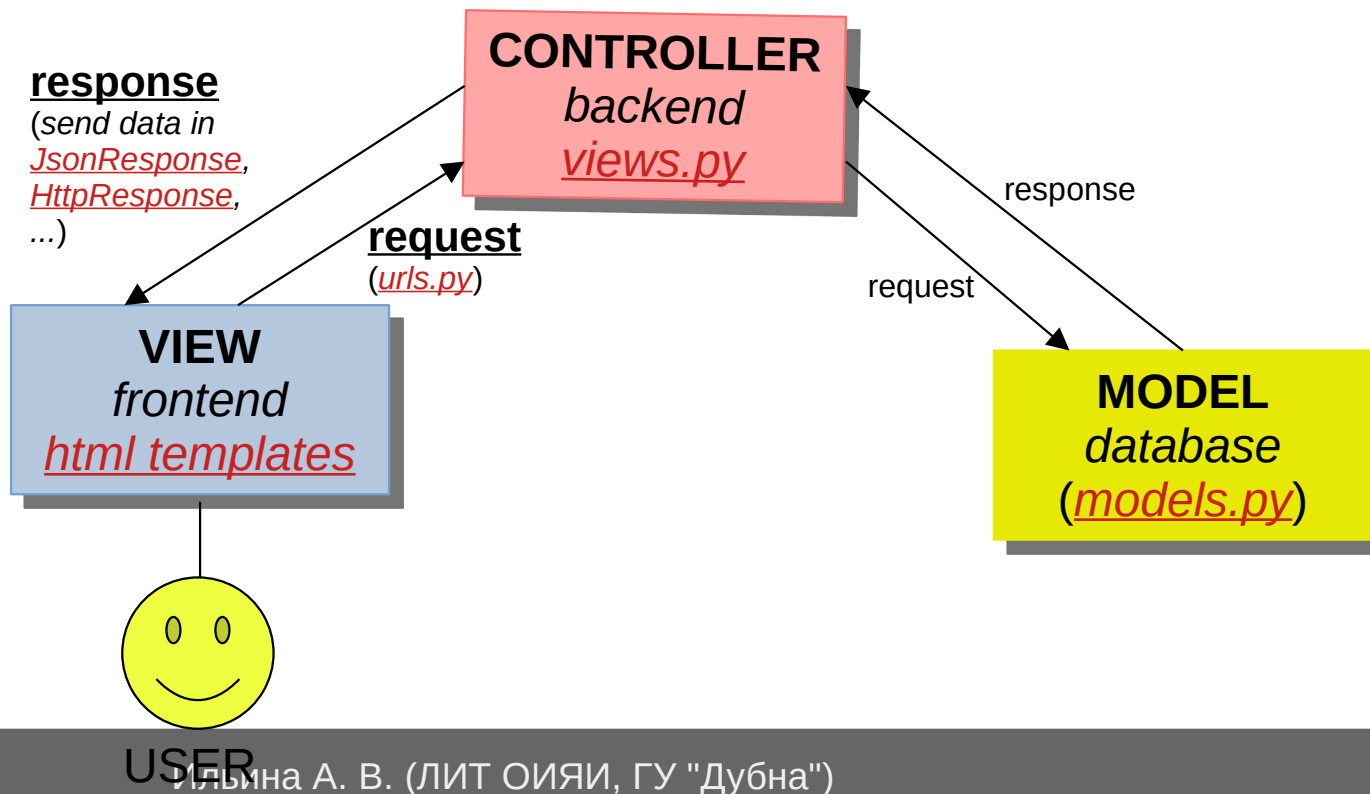
Также Django используется в качестве веб-компонента в проектах:

- Graphite — система построения графиков и наблюдения
- FreeNAS — свободная реализация системы хранения и обмена файлами и др.

Django — это backend-фреймворк, использующий шаблон проектирования Model-View-Controller (MVC)

Видео, как
работает
Django:

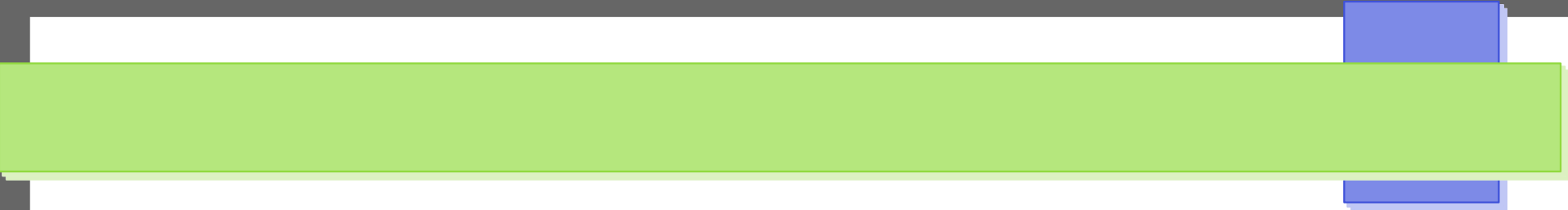
<https://www.youtube.com/watch?v=RL09RJhbOrQ>



Deploy — как развернуть веб-приложение?

Django можно развернуть на PaaS-сервисах RedHat:

- OpenShift, в том числе и бесплатно
- Heroku
- На хостинге PythonAnywhere
- Google App Engine



Создаём веб-приложение

Какие технологии будем использовать?

- На стороне сервера:
 - Python
 - Django
- На стороне клиента:
 - HTML/JavaScript
 - Bootstrap5 (документация: <https://getbootstrap.com/docs/5.0/getting-started/introduction/>)
 - AJAX
- База данных:
 - sqlite (по умолчанию в Django)

Создание виртуальной среды, загрузка Django

Используемые версии на момент создания tutorials:
python==3.8.10 (с 3.10 тоже работает), Django==4.0.4

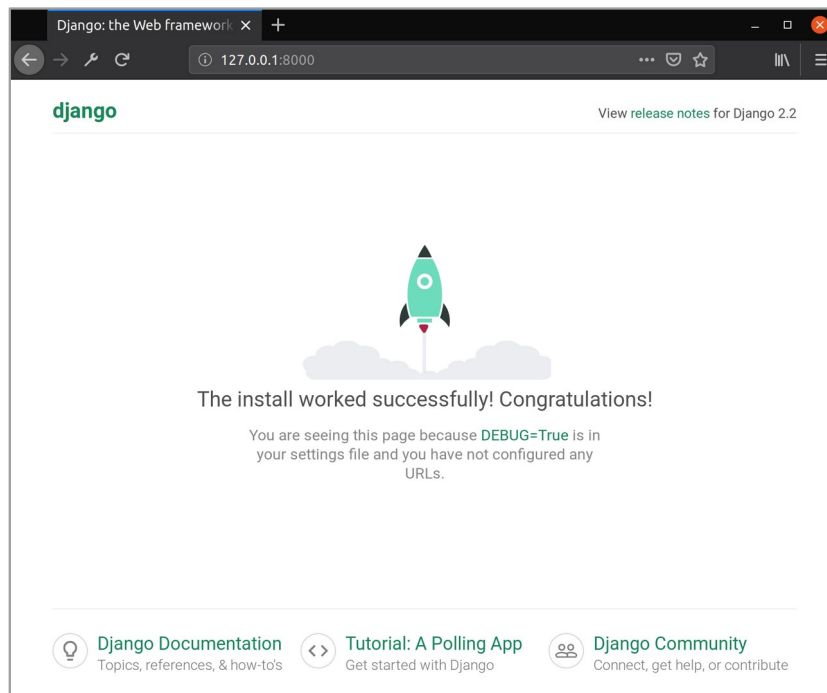
- `python3 -m venv venv`
- `source venv/bin/activate`
- `pip install Django`
- `pip list`
- `deactivate && pip list && source venv/bin/activate`
- `python -m django --version`

Создание проекта

- `django-admin startproject our_tutorial_project`
появится папка, заходим в неё, рассматриваем файлы
- **manage.py**: утилита командной строки, которая позволяет вам взаимодействовать с этим проектом Django различными способами. Вы можете прочитать все подробности об `manage.py` в `django-admin` и `manage.py`.
- **Внутренний каталог `our_tutorial_project/`** — это фактический пакет Python для вашего проекта. Его имя — это имя пакета Python, которое вам нужно будет использовать для импорта чего-либо внутри него (например, `mysite.urls`).
- **`mysite/__init__.py`**: пустой файл, который сообщает Python, что этот каталог следует рассматривать как пакет Python.
- **`mysite/settings.py`**: Настройки/конфигурация для этого проекта Django.
- **`mysite/urls.py`**: объявления URL для этого проекта Django; «оглавление» вашего сайта на Django.
- **`mysite/asgi.py`**: точка входа для ASGI-совместимых веб-серверов для обслуживания вашего проекта.
- **`mysite/wsgi.py`**: точка входа для WSGI-совместимых веб-серверов для обслуживания вашего проекта.

Запуск сайта

- `python our_tutorial_project/manage.py runserver`



Создание нашего приложения

- **cd our_tutorial_project**
- **python manage.py startapp app**
появилась ещё папка, рассматриваем файлики
- подключаем созданное приложение в проект Django (our_tutorial_project/settings.py):

```
INSTALLED_APPS = [  
'django.contrib.admin',  
'django.contrib.auth',  
'django.contrib.contenttypes',  
'django.contrib.sessions',  
'django.contrib.messages',  
'django.contrib.staticfiles',  
'app.apps.AppConfig'  
]
```

Создаем первое представление (app/views.py)

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

```
def index(request):  
    return HttpResponse("Hello, world. You're at the app index.")
```

Создаем первый маршрут (app/urls.py)

```
from django.urls import path
```

```
from . import views
```

```
urlpatterns = [  
    path("", views.index, name='index'),  
]
```

Добавляем наш созданный маршрут в головные (our_tutorial_project/urls.py)

```
from django.contrib import admin  
from django.urls import include, path
```

```
urlpatterns = [  
    path("", include(('app.urls', 'app'), namespace='app')),  
    path('admin/', admin.site.urls),  
]
```


Проверяем, что работает

- **python manage.py runserver**
- переходим на <http://127.0.0.1:8000/>

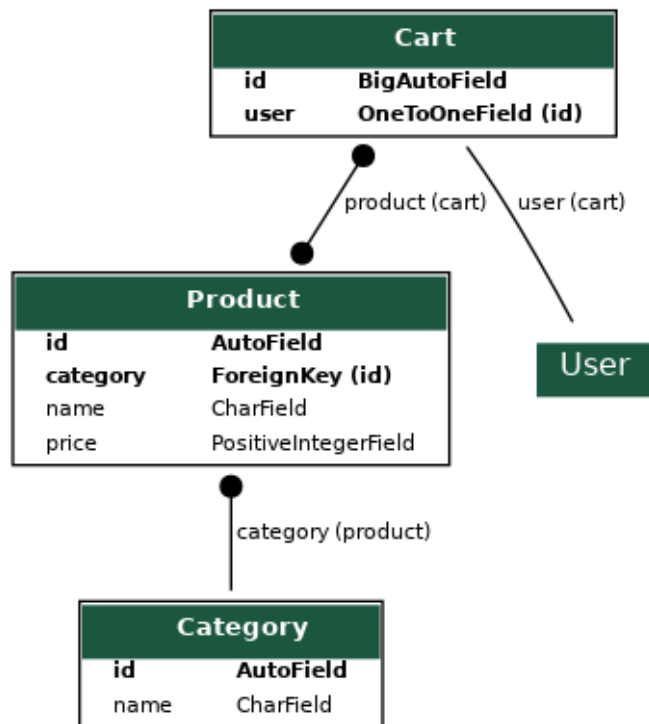
Описание задачи

Будем писать интернет-магазин

Для этого нам понадобится БД и сущности:

- Пользователь нашего магазина
- Товар
 - Название
 - Цена
 - Категория
- Категория
 - название
- Корзина пользователя
 - товары

ERD



Какие БД можно подключать в Django помимо дефолтной sqlite?

- PostgreSQL
- MariaDB
- MySQL
- Oracle
- SQLite
- CockroachDB
- Firebird
- Google Cloud Spanner
- Microsoft SQL Server

<https://docs.djangoproject.com/en/4.0/ref/databases/>

Создаём нужные модели (таблицы) (app/models.py)

```
from django.db import models
from django.contrib.auth.models import User

class Category(models.Model): # категория товара
    id = models.AutoField(primary_key=True, help_text="Unique ID for this Category")
    name = models.CharField(max_length=200)

    def __str__(self):
        """
        String for representing the Model object.
        """
        return self.name
```

Создаём нужные модели (таблицы) (app/models.py)

```
class Product(models.Model): # товар
    id = models.AutoField(primary_key=True, help_text="Unique ID for this Product")
    name = models.CharField(max_length=200)
    price = models.PositiveIntegerField(default=100)
    category = models.ForeignKey('Category', on_delete=models.CASCADE)

    def __str__(self):
        """
        String for representing the Model object.
        """
        return f'{self.name}, {self.price}, {self.category}'
```

Создаём нужные модели (таблицы) (app/models.py)

```
class Cart(models.Model): # корзина пользователя
    user = models.OneToOneField(User, on_delete=models.CASCADE)
    product = models.ManyToManyField(Product, blank=True)

    def __str__(self):
        return f'{self.user.get_full_name()} (@{self.user.get_username()}), {'
        '.join(self.product.all().values_list('name', flat=True))}'
```

Регистрируем эти модели (app/admin.py)

```
from django.contrib import admin  
from .models import *
```

```
admin.site.register(Category)
```

```
admin.site.register(Product)
```

```
admin.site.register(Cart)
```

делаем миграцию (синхронизацию с БД):

```
python manage.py makemigrations
```

```
python manage.py migrate
```


Заходим в админку, смотрим на получившиеся таблички

- Делаем миграцию:
python manage.py makemigrations
python manage.py migrate
- Создаём суперпользователя-админа:
python manage.py createsuperuser
- Заходим на <http://127.0.0.1:8000/admin/> под админской учёткой и видим, что наши таблицы появились в БД
- Даём нашему админу First Name и Last Name
- Добавляем несколько категорий и несколько товаров (2-3)

Add product

Name:

Price:

Category:



Создаём для нашего пользователя корзину

Add cart

User:

anna



Product:

CP-1 Hair silk essence, 100, Cosmetics

здесь ничего не выбираем,
т. к. будем добавлять товары
прямо с нашего интернет-магазина
(из html-странички)



Hold down "Control", or "Command" on a Mac, to select more than one.

Save and add another

Save and continue editing

SAVE

Пишем страничку с каталогом товаров

- Создаём папку **app/templates**
- Создаём в ней **base.html** (это «корневой» html, от него будут как бы наследоваться все наши странички)
- Скопировать base.html

Пишем страничку с каталогом товаров (ajax)

- Создаём в templates **index.html** (это главная страница сайта, которая сразу будет содержать каталог доступных товаров)
- Скопировать index.html

CP-1 Hair silk essence

Price: 100

Category: Cosmetics

Add to My Cart

Show My Cart

Potato

Price: 2

Category: Food

Add to My Cart

views.py

```
def index(request):  
    return render(request, 'index.html', context={})
```

```
def get_all_products(request):  
    all_products = Product.objects.all()  
    result = []  
    for product in all_products:  
        result.append(  
            {  
                'id': product.id,  
                'name': product.name,  
                'price': product.price,  
                'category': product.category.name  
            }  
        )  
    return JsonResponse(result, safe=False)
```

```
def add_product_to_cart(request):  
    product_id = request.GET.get('product_id')  
  
    this_user = request.user  
    this_user_cart = Cart.objects.get(user=this_user)  
    this_user_cart.product.add(product_id)  
  
    return HttpResponse(status=200)
```

urls.py

```
from django.urls import path
```

```
from . import views
```

```
urlpatterns = [  
    path("", views.index, name='index'),  
    path('get_all_products', views.get_all_products,  
        name='get_all_products'),  
    path('add_product_to_cart',  
        views.add_product_to_cart,  
        name='add_product_to_cart'),  
]
```

Пишем страничку с корзиной пользователя (Django template tags)

- Создаём в templates **cart.html** (это страница корзины нашего пользователя)
- Скопировать cart.html

My Cart

Hello, Anna Ilina!

Category	Name	Price	Action
Cosmetics	CP-1 Hair silk essence	100	DELETE
Food	Potato	2	DELETE

[Go to Store](#)

views.py

```
def show_cart(request):
    this_user = request.user
    user_cart = Cart.objects.get(user=this_user)
    user_products = user_cart.product.all()
    result = {}
    for product in user_products:
        result[product.category.name] = {
            'id': product.id,
            'name': product.name,
            'price': product.price
        }
    return render(request, 'cart.html', context={'result': result})
```

```
def delete_a_product(request):
    product_id = request.GET.get('product_id')
    this_user = request.user
    this_user_cart = Cart.objects.get(user=this_user)
    this_user_cart.product.remove(product_id)
    return HttpResponse(status=200)
```

urls.py

```
from django.urls import path

from . import views

urlpatterns = [
    path("", views.index, name='index'),
    path('get_all_products', views.get_all_products,
         name='get_all_products'),
    path('add_product_to_cart', views.add_product_to_cart,
         name='add_product_to_cart'),
    path('show_cart', views.show_cart, name='show_cart'),
    path('delete_a_product', views.delete_a_product,
         name='delete_a_product'),
]
```

Результаты

- Мы создали веб-приложение, у которого backend реализован на **Django**, а frontend — на **HTML/JavaScript (+ Bootstrap5)**.
- База данных — **sqlite**, имеет 3 сущности (+ User).
- На страничке [index.html](#) расположен каталог товаров, содержащихся в БД, из которого можно добавлять товары в свою корзину.
 - реализован с использованием AJAX-запросов (REST-архитектура)
- На страничке [cart.html](#) можно просматривать и удалять добавленные товары.
 - реализована с помощью встроенных Django-тегов (Django template tags)