

# django

## Введение

---

ЛЕКТОР: ИПАТОВА ЮЛИЯ НИКОЛАЕВНА

# Введение

---

Django — это высокоуровневый Python веб-фреймворк, который позволяет быстро создавать безопасные и поддерживаемые веб-сайты.

Django берёт на себя большую часть хлопот веб-разработки. Он бесплатный и с открытым исходным кодом, имеет растущее и активное сообщество, отличную документацию и множество вариантов как бесплатной, так и платной поддержки

Django предоставляет практически всё, что большинство разработчиков может пожелать, «из коробки». Он также быстр, безопасен и очень масштабируем. Код Django легко читать и поддерживать.

Популярные сайты, использующие Django (с домашней страницы Django), включают в себя: Disqus, Instagram, Knight Foundation, MacArthur Foundation, Mozilla, National Geographic, Open Knowledge Foundation, Pinterest, Open Stack.

# Полнота

---

Django следует философии «Всё включено» и предоставляет почти всё, что разработчики могут захотеть сделать «из коробки». Поскольку всё, что вам нужно, является частью единого «продукта», всё это безупречно работает вместе, соответствует последовательным принципам проектирования и имеет обширную и актуальную документацию.

# Многогранность

---

Django может быть (и был) использован для создания практически любого типа веб-сайтов — от систем управления контентом и wiki до социальных сетей и новостных сайтов. Он может работать с любой клиентской средой и может доставлять контент практически в любом формате (включая HTML, RSS-каналы, JSON, XML и т. д.).

Хотя Django предоставляет решения практически для любой функциональности, которая вам может понадобиться (например, для нескольких популярных баз данных, шаблонизаторов и т. д.), внутренне он также может быть расширен сторонними компонентами, если это необходимо.

# Безопасность

---

Django помогает разработчикам избежать многих распространённых ошибок безопасности, предоставляя фреймворк, разработанный чтобы «делать правильные вещи» для автоматической защиты сайта.

Например, Django предоставляет безопасный способ управления учётными записями пользователей и паролями, избегая распространённых ошибок, таких как размещение информации о сессии в файлы cookie, где она уязвима (вместо этого файлы cookie содержат только ключ, а фактические данные хранятся в базе данных) или непосредственное хранение паролей вместо хэша пароля.

Django, по умолчанию, обеспечивает защиту от многих уязвимостей, включая SQL-инъекцию, межсайтовый скриптинг, подделку межсайтовых запросов и клидджекинг (см. Website security для получения дополнительной информации об этих атаках).

# Масштабируемость

---

Код Django написан с использованием принципов и шаблонов проектирования, которые поощряют создание поддерживаемого и повторно используемого кода. В частности, в нём используется принцип «Don't Repeat Yourself» (DRY, «не повторяйся»), поэтому нет ненужного дублирования, что сокращает объём кода.

Django также способствует группированию связанных функциональных возможностей в повторно используемые «приложения» и, на более низком уровне, группирует связанный код в модули (в соответствии с шаблоном Model View Controller (MVC)).

# Переносимость

---

Django написан на Python, который работает на многих платформах. Это означает, что вы не привязаны к какой-либо конкретной серверной платформе и можете запускать приложения на многих версиях Linux, Windows и Mac OS X.

Кроме того, Django хорошо поддерживается многими веб-хостингами, которые часто предоставляют определённую инфраструктуру и документацию для размещения сайтов Django.

# Дополнительные возможности

---

- **Формы:** HTML-формы используются для сбора пользовательских данных для обработки на сервере. Django упрощает создание, проверку и обработку формы.
- **Аутентификация пользователя и разрешения:** Django включает надёжную систему аутентификации и авторизации пользователей, которая была построена с учётом безопасности.
- **Кеширование:** Django обеспечивает гибкое кеширование, чтобы вы могли хранить всю или часть отображаемой страницы, для того, чтобы она не вызывалась повторно, за исключением случаев, когда это необходимо.
- **Админ-панель:** Административная панель в Django включена по умолчанию при создании приложения с использованием основного каркаса.
- **Сериализация данных:** Django упрощает сериализацию и обслуживание ваших данных в таких форматах как XML или JSON.



# Особенности Django

---

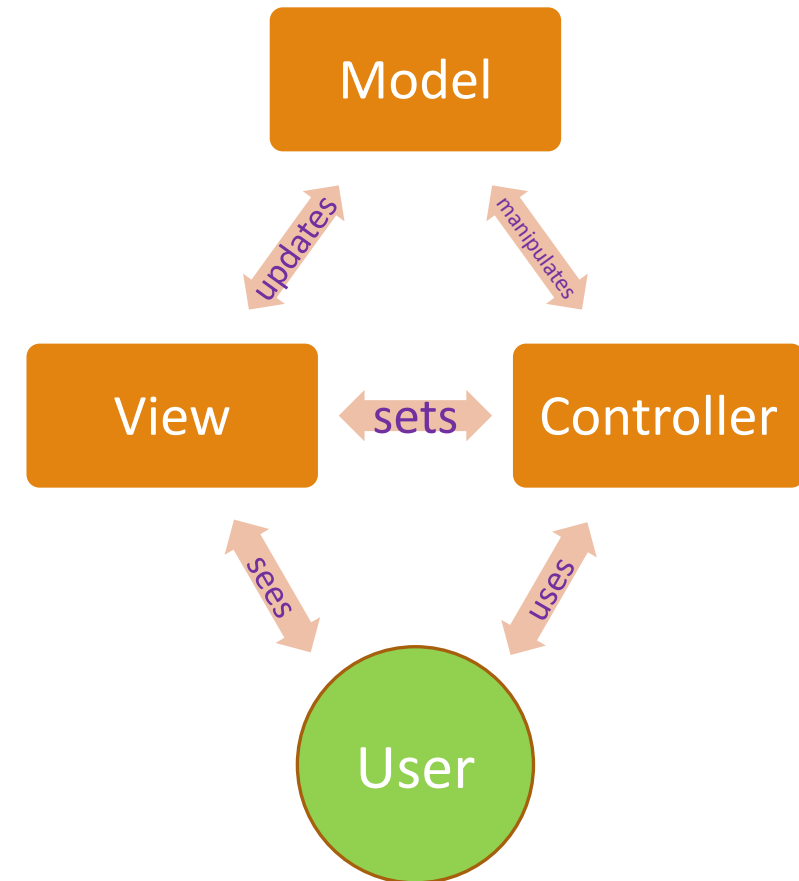
- ORM, API доступа к БД с поддержкой транзакций
- диспетчер URL на основе регулярных выражений
- расширяемая система шаблонов с тегами и наследованием
- подключаемая архитектура приложений, которые можно устанавливать на любые Django-сайты
- библиотека для работы с формами (наследование, построение форм по существующей модели БД)
- встроенная автоматическая документация по тегам шаблонов и моделям данных, доступная через административное приложение
- MVT архитектура.

# Model-View-Controller (MVC)

— архитектура программного обеспечения, в которой модель данных приложения, пользовательский интерфейс и управляющая логика разделены на три отдельных компонента, так, что модификация одного из компонентов оказывает минимальное воздействие на другие компоненты.

Шаблон MVC позволяет разделить данные, представление и обработку действий пользователя на три отдельных компонента:

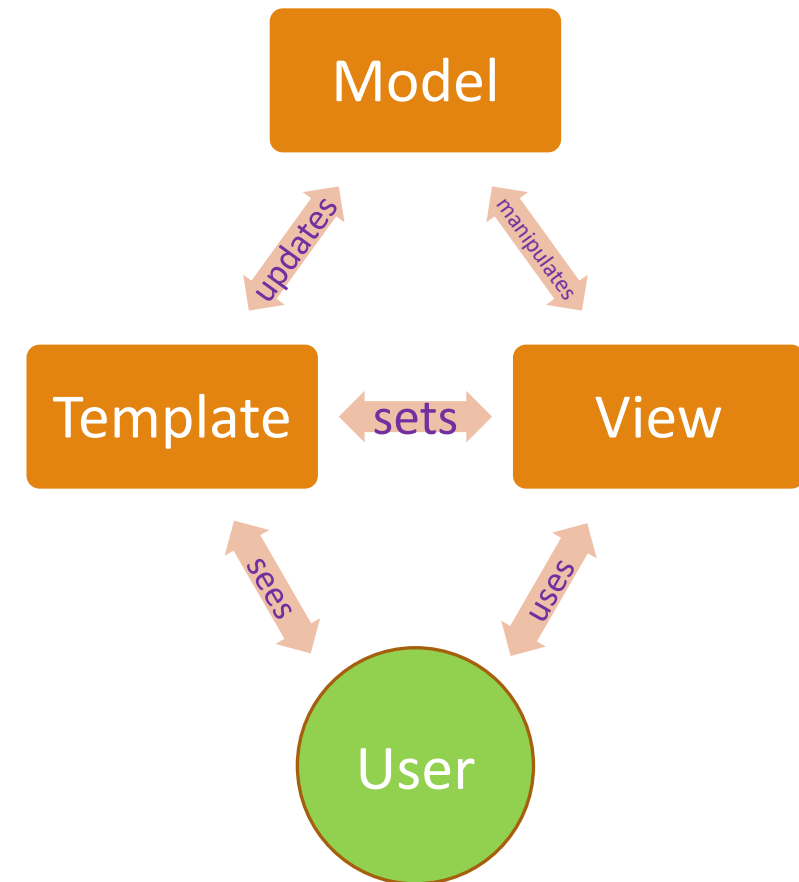
- **Модель** (Model)
- **Представление** (View)
- **Поведение** (Controller)



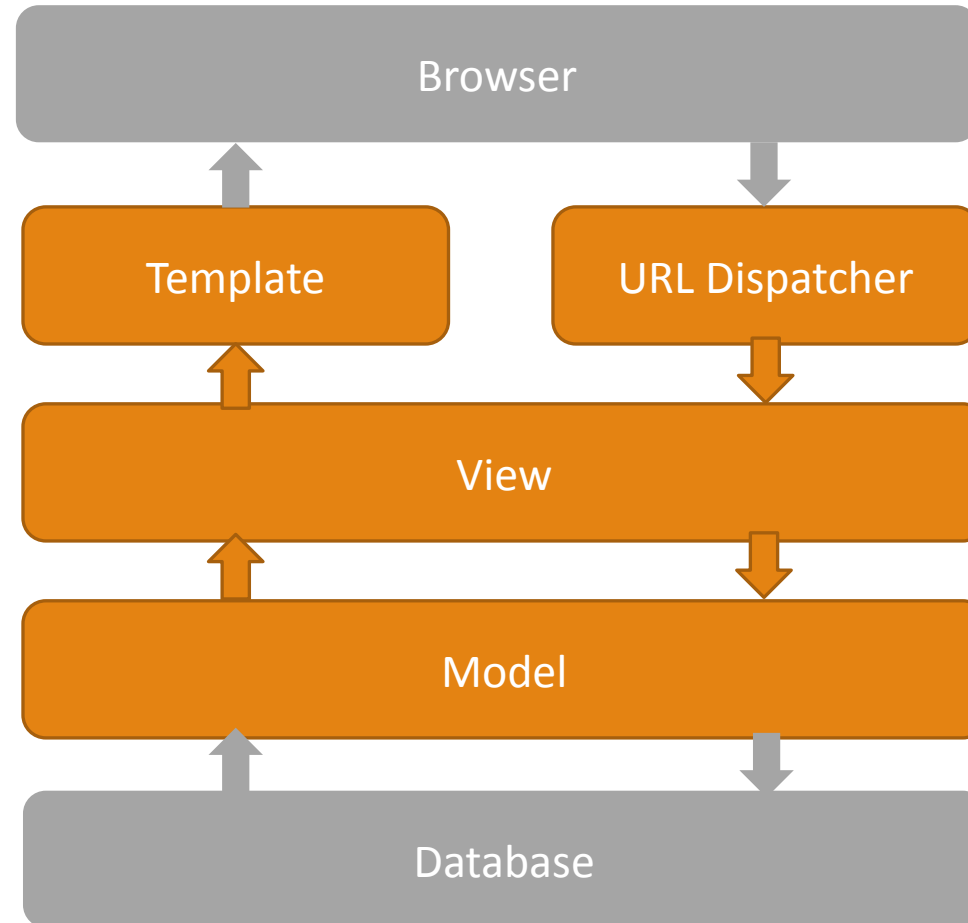
# Model-View-Template (MVT)

— модификация распространённого в веб-программировании паттерна MVC.

- Model, уровень доступа к данным. Здесь сосредоточена вся информация о данных: как получить к ним доступ, как осуществлять контроль, каково их поведение, каковы отношения между данными.
- Template (шаблон), уровень отображения. Здесь принимаются решения, относящиеся к представлению данных: как следует отображать данные на веб-странице или в ином документе.
- View, уровень логики. Здесь расположена логика доступа к модели и выбора подходящего шаблона (или шаблонов). Это мост между моделями и шаблонами.



# Django MVT



*представляет логику представления в виде сгенерированной разметки html.*

*получает запрос, обрабатывает его и отправляет в ответ пользователю некоторый ответ.*

*при получении запроса на основании запрошенного адреса URL определяет, какой ресурс должен обрабатывать данный запрос.*

*описывает данные, используемые в приложении. Отдельные классы, как правило, соответствуют таблицам в базе данных.*

# Django-проект

---

При установке Django устанавливается скрипт `django-admin.py`. А на Windows также исполняемый файл `django-admin.exe`.

`django-admin` предоставляет ряд команд для управления проектом Django. В частности, для создания проекта применяется команда `startproject`. Этой команде в качестве аргумента передается название проекта.

- **Проект** — набор настроек для экземпляра Django, включая конфигурацию базы данных, параметров для Django и настроек приложения.

*Проект создаётся при помощи соответствующей команды:*

```
... \> django-admin startproject mysite
```

# Структура проекта

---

- `__init__.py`: Файл необходим для того, чтобы Python рассматривал данный каталог как пакет, т.е., как группу модулей.
- `manage.py`: Это утилита командной строки, которая позволяет вам взаимодействовать с проектом различными методами.
- `settings.py`: Настройки для текущего проекта Django.
- `urls.py`: Описания URL для текущего проекта Django, так сказать «оглавление» для вашего сайта.
- `wsgi.py` и `asgi.py`: содержит свойства конфигурации WSGI (*Web Server Gateway Interface*) и ASGI (*Asynchronous Server Gateway Interface*). Они используются при развертывании проекта.

```
mysite/  
  manage.py  
  mysite/  
    __init__.py  
    settings.py  
    urls.py  
    asgi.py  
    wsgi.py
```

# Django-приложение

---

Веб-приложение или проект Django состоит из отдельных приложений. Вместе они образуют полноценное веб-приложение.

- Каждое приложение представляет какую-то определенную функциональность.
- Один проект может включать множество приложений. Это позволяет выделить группу задач в отдельный модуль и разрабатывать их относительно независимо от других.
- Мы можем переносить приложение из одного проекта в другой независимо от другой функциональности проекта.

```
... \> py manage.py startapp polls
```

# Django-приложение

---

После создания приложения, для того, чтобы оно задействовало в проекте, его необходимо зарегистрировать в файле *settings.py*.

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



# Структура приложения

---

- папка migrations: хранит информацию, которая позволяет сопоставить базу данных и определение моделей
- \_\_init\_\_.py: указывает интерпретатору python, что текущий каталог будет рассматриваться в качестве пакета
- admin.py: предназначен для административных функций, в частности, здесь производится регистрация моделей, которые используются в интерфейсе администратора

```
polls/  
    __init__.py  
    admin.py  
    apps.py  
    migrations/  
        __init__.py  
    models.py  
    tests.py  
    views.py
```

# Структура приложения

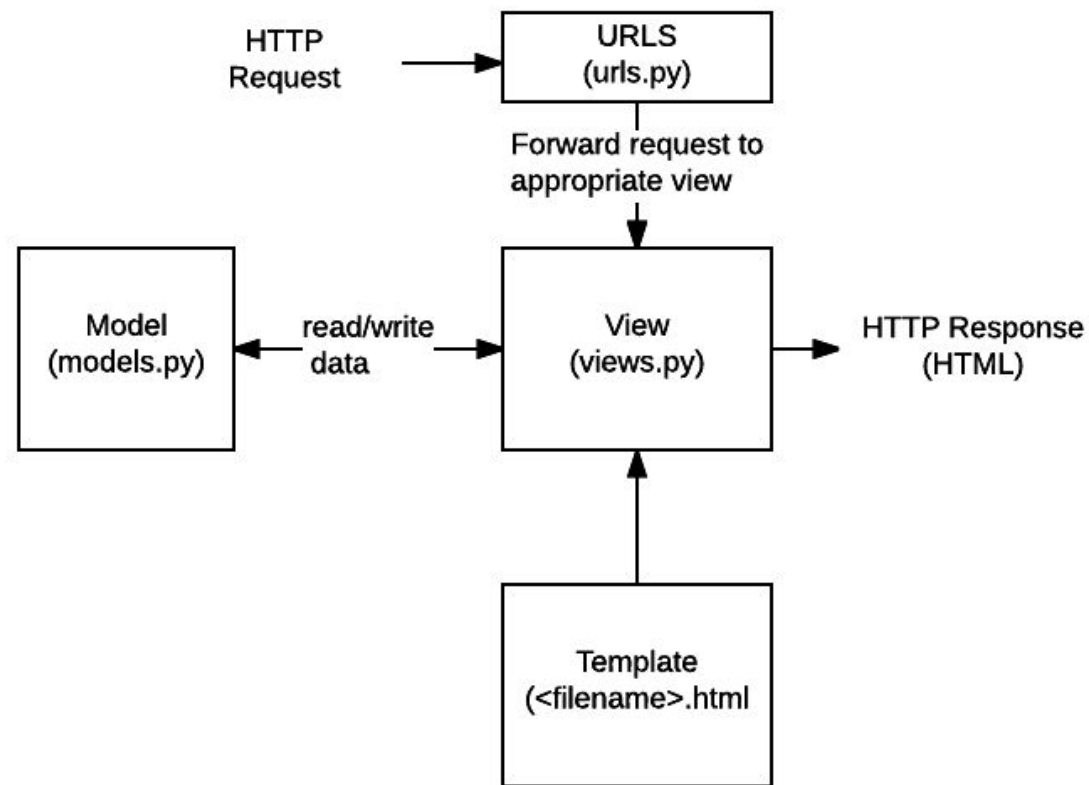
---

- `apps.py`: определяет конфигурацию приложения
- `models.py`: хранит определение моделей, которые описывают используемые в приложении данные
- `tests.py`: хранит тесты приложения
- `views.py`: определяет функции, которые получают запросы пользователей, обрабатывают их и возвращают ответ

```
polls/  
    __init__.py  
    admin.py  
    apps.py  
    migrations/  
        __init__.py  
    models.py  
    tests.py  
    views.py
```

# Основные строительные блоки

- **URLs:** Хотя можно обрабатывать запросы с каждого URL-адреса с помощью одной функции, гораздо удобнее писать отдельную функцию для обработки каждого ресурса.
- **View:** это функция обработчика запросов, которая получает HTTP-запросы и возвращает ответы.
- **Models:** Модели представляют собой объекты Python, которые определяют структуру данных приложения и предоставляют механизмы для управления (добавления, изменения, удаления) и выполнения запросов в базу данных.
- **Templates:** это текстовый файл, определяющий структуру или разметку страницы (например HTML-страницы), с полями для подстановки, которые используются для вывода актуального содержимого.



# Создание View

---

Определим какие-нибудь простейшие действия, которые будет выполнять данное приложение, например, отправлять в ответ пользователю строку "Hello, world. You're at the polls index."

Для этого перейдем в проекте приложения polls к файлу [views.py](#).

```
from django.http import HttpResponse

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

# Сопоставление маршрута

---

Теперь также в проекте *mysite* откроем файл [urls.py](#), который позволяет сопоставить маршруты с представлениями, которые будут обрабатывать запрос по этим маршрутам. Переменная `urlpatterns` определяет набор сопоставлений функций обработки с определ

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

urlpatterns = [
    path('polls/', views.index),
    path('admin/', admin.site.urls),
]
```

# Проверка работы

---

Для того, чтобы проверить работоспособность, необходимо сначала запустить сервер.

Django обладает встроенным сервером разработки.

Сервер разработки Django - это встроенный упрощенный веб-сервер, которым можно пользоваться в ходе разработки сайта. Он включен в состав Django для того, чтобы можно было быстро создать сайт, не отвлекаясь на настройку полноценного сервера.

```
... \> py manage.py runserver
```

Для запуска сервера используется команда.

По умолчанию команда *runserver* запускает сервер разработки на порту 8000 и принимает запросы на соединения только с локального компьютера.

# Проверка работы

---

