Знакомство с фрэймворком

Зачем нужен Django. Сильные стороны фрэймворка. Установка и настройка. Общие сведения о структуре проекта. Знакомство с urlpatterns. Первая страничка и ее отображение.

[Рекомендуемый уровень знаний](#_c713xnuw3xem)

[Зачем нужен Django](#_ytqvmv91w6f)

[Сильные стороны фрэймворка](#_21kxz1xrglf6)

[Установка](#_q0n9z1ibz7o2)

[Определимся с версиями](#_1mbt9k9vnajt)

[Установим Django](#_1rmi05s9iphr)

[Установка под Ubuntu](#_vqcejib5g5lf)

[Установка Django под Windows](#_uz1dk1jzh4je)

[Несколько слов про IDE](#_65e95nezshfr)

[Создание нового проекта](#_uk6vo63t1qqn)

[Общие сведения о структуре проекта](#_2g077om7e18)

[Запуск сервера разработки](#_rqw9w0jsykea)

[Создание нового приложения](#_y5jowcbyzrcy)

[Несколько слов об URL](#_mz5528eajr29)

[Первая страничка](#_50qhvdoraudp)

[Подробнее рассмотрим обработку utl с помощью регулярки.](#_cv93i6srltg9)

[Домашнее задание](#_owpsqd9opgcf)

[Дополнительные материалы](#_y937sk8fclye)

[Используемая литература](#_uvp6qax5r1ok)

# Рекомендуемый уровень знаний

Для успешного прослушивания данного курса, у вас уже должны быть основные знания о языке программирования python.

Вот основные темы, которые вы должны знать на начало данного курса:

* Переменный ти типы данных;
* Логические выражения и операторы ветвления (if else);
* Циклы;
* Итераторы (строки, списки, кортежи, словари);
* Функции;
* Основы ООП (объектно ориентированое программирование);
* Регулярные выражения.

Также вы должны знать:

* Основы HTML;
* Основы CSS.

# Зачем нужен Django

Планирование и реализация Web-сайтов всегда сопровождается большими затратами усилий. Django представляет собой один из лучших на сегодняшний день фреймворков, который позволяет быстро вести разработку высокопроизводительных и полнофункциональных сайтов. С помощью django легко выстаиваются масштабируемые и легко расширяемые приложения для Web с дизайном любой степени сложности.

Абстрагируясь от низкоуровневого процесса Web-строительства, django предоставляет множество инструментов, реализующих стандартные шаблоны программирования (регистрация, авторизация, пагинация и прочие).

Для работы с базами данных (БД) Django использует [ORM](https://ru.wikipedia.org/wiki/ORM) (Object-Relational Mapping, рус. объектно-реляционное отображение), что позволяет абстрагироваться от конкретной БД и работать с базой на объектном уровне. Также, при необходимости, довольно просто сменить одну БД на другую.

На Django написано много больших и высоконагруженных веб-сайтов, вот лишь несколько примеров:

Новостной портал - <http://polit.ru/>

Сайт радиостанции Говорит Москва - <http://govoritmoskva.ru/>

Сайт по продаже недвижимости - <https://tranio.ru/>

# Сильные стороны фрэймворка

Прежде чем приступить к рассмотрению самого Django, кратко рассмотрим основные его преимущества:

* ORM

Django предоставляет простой механизм работы с базой без изучения синтаксиса SQL запросов, а также возможность абстрагироваться от конкретной БД

* Диспетчер URL, на основе регулярных выражений

Диспетчер URL - входная точка для любых запросов, связывающая адрес запроса с его обработчиком

* Шаблонизатор

Расширяема система шаблонов с тегами и наследование позволяет быстро и удобно создавать динамические станицы любой сложности

* Интернационализация

Встроенная система интернационализации помогает переводить сайты на различные языки

* Паттерн проектирования MVC (Мodel View Template)

Django поощряет свободное связывание и строгое разделение частей приложения. Если следовать этой философии, то легко вносить изменения в одну конкретную часть приложения без ущерба для остальных частей

* Готовые модули - готовые решения

Большое количество встроенных и подключаемых модулей имеют уже готовые шаблоны для решения многих рутинных задач, таких как: регистрация, авторизация, интеграция с социальными сетями и прочие.

* Промежуточные слои (Middleware)

Готовые обработки запросов и ответов позволяют упростить рутинные задачи построение правильных http-ответов и разборки http-запросов, что особенно полезно для начинающих программистов.

* Система кэширования

Кэширование часто используемых страниц позволяет существенно снизить нагрузку на сервер вашего сайта.

И это далеко не полный список всех возможностей фрэймворка Django.

Если вы поймете философию и научитесь правильно пользоваться данным фрэймворком, то написание ваших движков сайтов станет гораздо эффективнее и приятнее. Это именно то, чем мы будем заниматься на данном вебинаре.

Давайте приступим!

# Установка

Надеюсь, python у вас уже установлен, если нет: <https://www.python.org/downloads/>

## Определимся с версиями

Для работы нам будет достаточно любой версии питона, старше 3.4 (лучше устанавливать самую свежую)

Django будет достаточно версии 1.8+

## Установим Django

### Установка под Ubuntu

1. sudo apt-get install python3-pip - устанавливаем pip.
2. sudo pip3 install django - устанавливаем django, используя pip.
3. django-admin --version - проверяем установку, если увидели версию, то установка прошла успешно.

### Установка Django под Windows

1. Скачиваем [дистрибутив с оффициального сайта](https://www.djangoproject.com/download/).
2. Распаковываем.
3. Запускаем командную оболочку с правами администратора (WIN + R → cmd).
4. В командной оболочке переходим в папку с распакованным дистрибутивом.
5. python3 setup.py install - эта команда установит Django в системную директорию site-packages.

Замечание:

|  |
| --- |
| Если у вас интерпретатор не прописан в переменных среды, то вместо python3, необходимо указать полный путь к интерпретатору python, например: c:/python35/python.exe |

## Несколько слов про IDE

IDE - интегрированная среда разработки.

Вы можете использовать любую, привычную вам IDE, я пользуюсь [PyCharm](https://www.jetbrains.com/pycharm/download/#section=linux) (для полной поддержки django-проектов желательно иметь платную Professional версию).

На занятиях я буду показывать полезные инструменты PyCharm для работы с django-проектом, и всегда буду дублировать эти действия с помощью команд консоли, чтобы вы могли работать без привязки к какой-то конкретной IDE.

Работать вообще без IDE можно, но крайне не рекомендуется.

# Создание нового проекта

Django установлен, пришло время создать первый проект, выполнив консольную команду:

1. Запустите консоль
2. Перейдите в папку, в которой будет распологаться django-проект
3. Выполните команду:
   1. django-admin startproject <имя проекта> - Ubuntu
   2. django-admin.py startproject <имя проекта> - Windows

Замечание:

|  |
| --- |
| Скрипт django-admin.py должен быть в вашем системном пути, если вы устанавливали Django с помощью утилиты setup.py.  На Windows вам потребуется внести изменение в переменную среды PATH или указывать полный путь. |

После успешного выполнения команды, будет создана директория (с указанным вами именем) с основными файлами проекта.

# Общие сведения о структуре проекта

Договоримся, что первый проект назовем mysite (вы, конечно, можете его называть как хотите, но в тексте я буду подразумевать именно это имя проекта)

Давайте кратко рассмотрим назначение файлов проекта:

mysite/  
 \_\_init\_\_.py  
 settings.py  
 urls.py  
manage.py

\_\_init\_\_.py: Файл необходим для того, чтобы Python рассматривал данный каталог как пакет, т.е., как группу модулей. Это пустой файл и обычно вам не требуется добавлять что-либо в него.

manage.py: Это утилита командной строки, которая позволяет вам взаимодействовать с проектом различными методами. Наберите python manage.py help для получения информации о возможностях утилиты. Вы не должны изменять содержимое данного файла, он создан в данном каталоге в целях удобства.

settings.py: Настройки для текущего проекта Django. Посмотрите на содержимое файла, чтобы иметь представление о типах доступных параметров и их значениях по умолчанию.

urls.py: Описания URL для текущего проекта Django, «оглавление» для вашего сайта.

# Запуск сервера разработки

Для проверки правильности установки Django давайте запустим сервер разработки, чтобы посмотреть на наше приложение в действии.

Сервер разработки Django (также называемый «runserver» по имени команды, которая его запускает) — это встроенный лёгкий веб сервер, который вы можете использовать в процессе разработки вашего сайта. Он включен в Django для того, чтобы вы могли быстро приступить к разработке вашего сайта без траты времени на конфигурирование вашего боевого веб сервера (т.е., Apache) раньше времени. Этот сервер разработки отслеживает изменения в вашем коде и автоматически перезагружает его, помогая видеть вносимые вами изменения без перезагрузки веб сервера.

Для запуска сервера перейдите в каталог mysite (cd mysite), если вы ещё не сделали этого, и выполните команду:

|  |
| --- |
| python manage.py runserver |

Вы увидите нечто подобное этому:

|  |
| --- |
| Performing system checks...  System check identified no issues (0 silenced).  Django version 1.8.1, using settings mysite.settings'  Starting development server at http://127.0.0.1:8000/  Quit the server with CONTROL-C. |

Команда запускает сервер локально на порту 8000. Сервер принимает только локальные соединения с вашего компьютера. Теперь он запущен, посетите страницу http://127.0.0.1:8000/ с помощью браузера. Вы увидите страницу «Welcome to Django».

Несмотря на то, что сервер разработки очень удобен во время разработки, сопротивляйтесь искушению использовать этот сервер в боевом режиме. Этот сервер может обрабатывать только один запрос в единицу времени, и он не проходил никакого аудита на предмет безопасности. Когда придёт время поднять ваш сайт, обратитесь к главе «Развёртывание Django» для информации о том, как установить туда проект Django.

# 

# 

# Создание нового приложения

Проект принято разбивать на логические части - приложения.

Но в чём же разница между проектом и приложением? Разница в том, что первое является конфигурацией, а второе — кодом:

Проект — это экземпляр определённого набора кода Django-приложений и конфигурация для этих приложений.

С технической точки зрения существует одно требование к проекту — наличие файла конфигурации, который определяет способ соединения с базой данных, список установленных приложений, каталог с шаблонами и так далее.

Приложение — это переносимый набор некой функциональности, обычно включает в себя модели и представления, которые хранятся вместе в едином пакете языка Python.

Например, Django поставляется с рядом приложений, таких как система комментирования и автоматический интерфейс администратора. Важной особенностью этих приложений является то, что они переносимы и их можно использовать во множестве проектов.

Существует очень мало жёстких правил для соответствия вашего кода этой схеме. Если вы создаёте простой сайт, вы можете использовать единственное приложение. Если вы создаёте сложный сайт с несколькими независимыми частями, такими как интернет-магазин и форум, возможно, вы пожелаете разнести их в отдельные приложения, что позволит использовать их раздельно в других проектах.

Внутри каталога проекта mysite выполните эту команду для создания приложения books:

|  |
| --- |
| python manage.py startapp mainapp |

Эта команда ничего не выводит, она просто создаёт каталог books в каталоге mysite. Давайте посмотрим на содержимое этого каталога:

mainapp/  
 /migrations  
 \_\_init\_\_.py  
 \_\_init\_\_.py  
 admin.py  
 models.py  
 tests.py  
 views.py

В папке migrations - хранятся миграции (оставим это на потом)

admin.py - готовая админка (мы ей пользоваться не будем, почему - рассмотрим ниже)

models.py - описание моделей, для работы с БД

tests.py - тестирование (не рассматривается в рамках данного вебинара)

views.py - функции представления (именно здесь происходят основные действия)

Это и есть каркас вашего приложения, большинство файлов пустые. Наполняя данный каркас, мы разрабатываем приложение.

# Несколько слов об URL

Потерпите, еще совсем немного и мы приступим к созданию нашей первой страницы. Подробнее познакомимся url-обработчиком и общему подходу django к работе с красивыми url.

Красивая, элегантная схема URL является важной составляющей высококачественного веб-приложения. Django поощряет создание красивых схем URL и не захламляет их мусором, подобным .php или же .asp.

Проектируя модель URL для своего приложения, вы создаёте модуль Python, называемый менеджером URL-ов. Оглавление вашего приложения представляет собой простое соответствие между URL-адресом и вызываемой функцией на Python. URLconfs служит для выделения URL адресов из кода на Python.

Все url-обработчики находятся в файле url.py

Указанная схема URL содержит простые регулярные выражения для вызова соответствующей функции Python (соответствующего представления, “views”). Регулярные выражения используют скобки для “захвата” значения из URL-адреса. Когда пользователь запрашивает страницу, Django проходит по всем шаблонам по порядку и останавливается на первом подходящем (если ни один из шаблонов не подошёл, Django вызовет исключение 404.) Это происходит мгновенно, поскольку регулярные выражения компилируются ещё во время загрузки.

После того, как соответствующий шаблон найден, Django импортирует и вызывает представление, которое является по сути простой функцией на Python. Каждое представление передаёт объект запроса – он содержит возвращаемые метаданные – и значения, захваченные регулярным выражением.

Все гораздо проще, чем выглядит. Чтобы все стало понятнее, давайте рассмотрим на примере.

# Первая страничка

Достаточно теории, давайте, наконец, что-нибудь сделаем.

Постановка задачи: Добавить простенькую стартовую страницу для нашего сайта.

Уточнение: Когда мы перейдем по адресу: <http://127.0.0.1:8000/>, должна отобразиться наша страница

Приступим к реализации

1. Добавим url

Содержимое файла **urls.py** заменим на:

|  |
| --- |
| from django.conf.urls import url  from mainapp.views import \*  urlpatterns = [  url(r'^$', main),  ] |

Обратите внимание: url() - первым аргументом принимает шаблон в виде регулярного выражения, вторым ссылку на функцию-обработчик, т.е. функция main будет вызвана, когда uri запроса будет соответствовать заданной регулярке r’^$’ (подробнее про uri и регулярку ниже).  
Не забываем импортировать функцию main: from mainapp.views import \*

1. Добавим функцию отображения “views”

**views.py**

|  |
| --- |
| from django.shortcuts import render, render\_to\_response  def main(request):  return render\_to\_response("index.html") |

Обратите внимание: функция main() первым аргументом всегда получает объект ЗАПРОСА (подробнее в следующих уроках). И функция-обработчик всегда должна вернуть объект ОТВЕТА.

1. Создадим html-страницу

В корне проекта создадим папку templates, для хранения html-страниц (шаблонов). В этой папке создадим index.html

**index.html**

|  |
| --- |
| <!DOCTYPE html>  <html>  <head lang="en">  <meta charset="UTF-8">  <title></title>  </head>  <body>  <h1>Welcome</h1>  <p>  Hello! It's my site -)  </p>  </body>  </html> |

Надеюсь, что основы html вы знаете.

1. Укажем django, где искать шаблоны, для этого в файле settings.py добавим имя папки.

**settings.py** (добавить только ‘templates’)

|  |
| --- |
| TEMPLATES = [  {  'BACKEND': 'django.template.backends.django.DjangoTemplates',  'DIRS': ['**templates**'],  'APP\_DIRS': True,  'OPTIONS': {  'context\_processors': [  'django.template.context\_processors.debug',  'django.template.context\_processors.request',  'django.contrib.auth.context\_processors.auth',  'django.contrib.messages.context\_processors.messages',  ],  },  },  ] |

Перехдим по адресу <http://127.0.0.1:8000/> - видим нашу страницу!

### Подробнее рассмотрим обработку utl с помощью регулярки.

Мы использовали регулярку r’^$’ (^-начало строки, $-конец строки).

С регуляркой django будет сравнивать часть адреса, которую называют URN(подробно и понятно [здесь](http://handynotes.ru/2009/09/uri-url-urn.html)). Т.к. мы используем URL: http://127.0.0.1:8000/, то его URN - будет пустая строка, именно она подходит под r’^$’.

Например:

URL: http://127.0.0.1:8000/about - будет соответствовать регулярка r‘^about$’

# Домашнее задание

На первых 4-х занятиях мы будем делать небольшой сайт “About me”, в данном ДЗ вам предстоит самостоятельно начать работу над ним. Кое-что у вас может не получаться, это нормально, я преднамеренно выдал не всю информацию, необходимую для полного выполнения ДЗ. Я хочу посмотреть, с какими сложностями вы столкнетесь, и как будете искать пути их решения.

1. Установить django и создать новый проект (назвать произвольным именем), добавить в проект одно приложение.
2. Создать главную статическую страницу “About me”. На странице разместить немного информации о себе (ФИО, год рождения, фотографию, хобби). Оформить страницу насколько сможете, постарайтесь сделать красиво, обязательно используйте CSS.
3. На главной странице сделать меню с пунктами “Учеба”, “Работа”.
4. Создать отдельную html с информацией о месте (местах) своего обучения (школа, институт, доп.курсы). Сделать несложное оформление. Добавить возможность перехода на текущую страницу с главной и возможность вернуться на главную.
5. То же, что в предыдущем пункте, но для “Работы”. Если нигде не работали, то напишите несколько слов о месте желаемой работы или месте подработки.

Все проблемы и пути их решения подробно обсудим на следующем занятии.

# Дополнительные материалы

Все то, о чём сказано в методичке, но более подробно:

1. [Установка django](http://djbook.ru/rel1.7/topics/install.html#install-the-django-code)
2. [Создание нового проекта](http://djbook.ru/ch02s05.html)
3. [Книга Джанго](http://djbook.ru/rel1.9/)

# Используемая литература

Для подготовки данного методического пособия были использованы следующие ресурсы:

1. [Официальная документация](https://docs.djangoproject.com/en/1.9/).
2. [Перевод документации](http://djbook.ru/).