Django Starter

Маршрутизация

Introduction

Лазорык Михаил

Software developer, 3 года опыта

mykhailo.lazoryk

mykhailo-lazoryk

Маршрутизация

План урока

1. Как **Django** обрабатывает запросы
2. Сопоставление маршрутов и представлений
3. Файл **urls.py**
4. Модуль **path**
5. Модуль **include**
6. Возможные значения атрибута **route**
7. Использование регулярных выражений

Как Django обрабатывает запросы

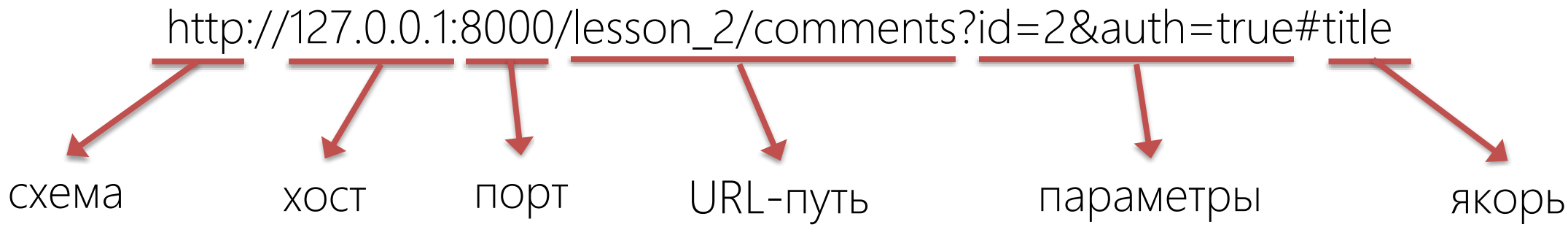
Все запросы, обрабатываемые в Django, обрабатываются в соответствии с файлами **urls.py** следующим образом:

1. Из **urls.py** берется объект **urlpatterns**
2. **urlpattern** последовательно перебирается (сверху вниз) на предмет соответствия запрашиваемого адреса какому-то из объявленных шаблонов
3. Если такое соответствие обнаружено, то происходит или передача в соответствующую функцию объекта запроса **request**, или передача адреса в другой файл **urls.py** приложения. Это зависит от того, какой функцией обрабатывается адрес по шаблону: **include** или **path**.
4. Результат функции, осуществляющей обработку переданного объекта запроса **request**, возвращается пользователю.
5. Если никакого соответствия **URL** шаблонам не найдено, то возвращается ошибка **404**.

URL это

URL (Uniform Resource Locator) – унифицированный указатель ресурса. Используется для записи ссылок на объекты в Интернете.

Структура **URL**:



URL структура

http://127.0.0.1:8000/lesson_2/comments?id=2&auth=true#title

http – схема обращения к ресурсу, указывает на сетевой протокол.

127.0.0.1 – доменное имя хоста, например **google.com**, **itvdn.com** и т.п.

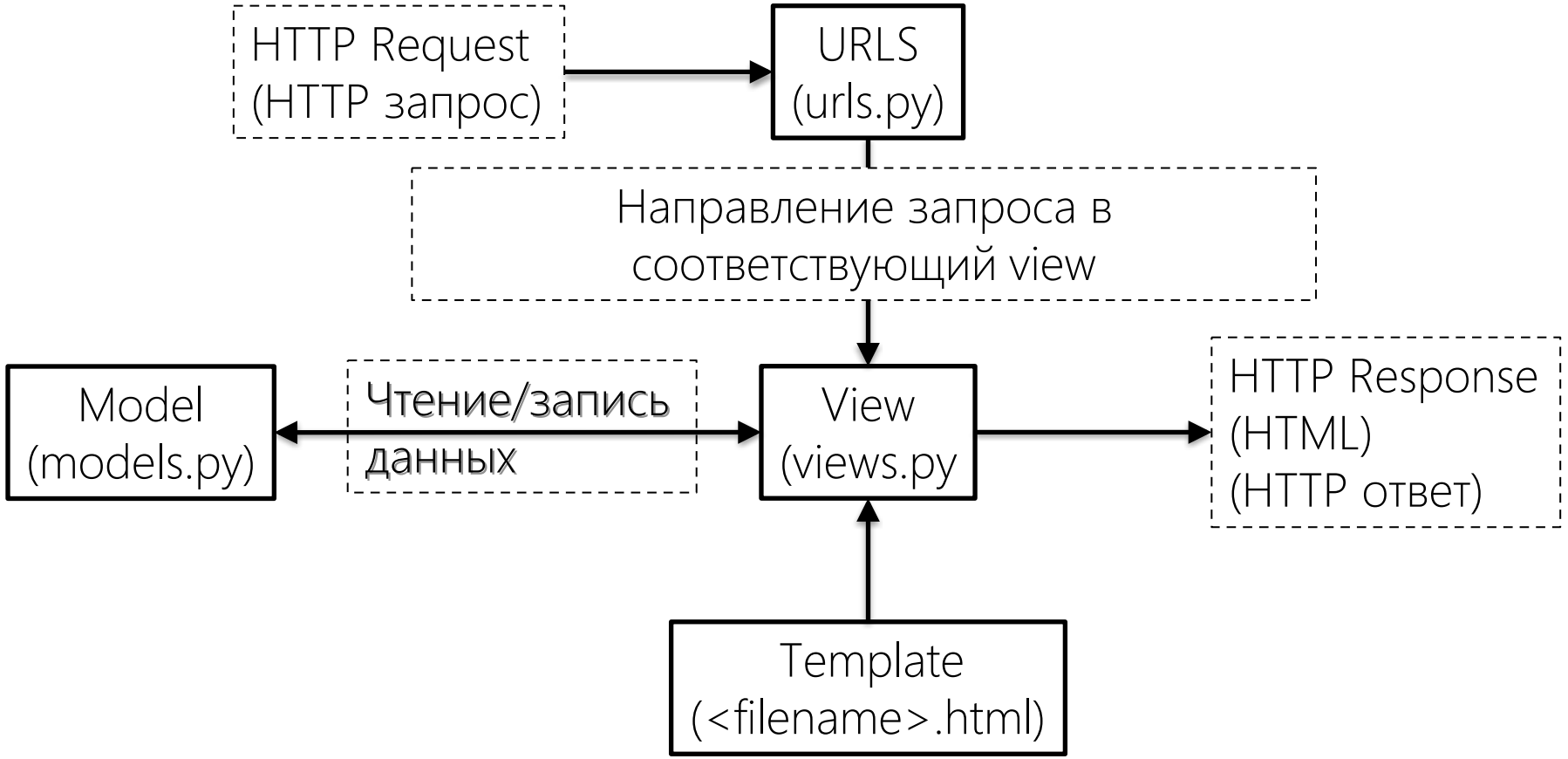
8000 – порт хоста для подключения. Если не указан, то для **http** порт=**80**.

lesson_2/comments – **URL**-путь к конкретной странице на хосте

id=2&auth=true – параметры, передаваемые хосту. Начинаются с символа **?**, разделитель параметров — знак **&** (амперсанд)

title – якорь, обычно это заголовок внутри документа, на который переходит браузер.

Структура файлов в реализации MVC в Django (MTV)



Сопоставление маршрутов и представлений

Сопоставление маршрутов и представлений в **django** осуществляется файлом, указанным в настройках **settings.py** проекта. По умолчанию это файл **urls.py**.

В **urls.py django** находит объект с именем urlpatterns, хранящий в себе список сопоставлений паттерн–представление (или паттерн–вложенный urlpatterns). При этом перебор всегда происходит в заданном порядке.

Файл urls.py

urls.py – это файл, описывающий то, как Django обрабатывает URL запросы.

В нём находится объект urlpatterns, который хранит в себе python список (list) обработчиков URL.

Данным обработчикам последовательно передается URL, и первому из них, кто сообщит о соответствии с адресом, будет передан URL вместе с запросом для обработки.

Обработчики, в общем случае, могут, или передавать URL с запросом в другой urls.py, или передавать объект запроса в функцию-обработчик.

Разбор файла django_courses/urls.py из первого урока

```
from django.contrib import admin  # стандартный интерфейс администратора
from django.urls import include, path  # функции для обработки URL


urlpatterns = [  # список шаблонов, определяющий маршрутизацию
    path('lesson_1/', include('lesson_1.urls')),
    path('admin/', admin.site.urls),
]
```

Функция path из пакета django.urls

path(route, view, kwargs=None, name=None)

route – (*обязательный атрибут*) это строка, содержащая URL шаблон, который django будет сравнивать с запрашиваемым URL-путем.

view – (*обязательный атрибут*) функция или класс, куда будет передаваться объект запроса для дальнейшей обработки, при совпадении URL шаблону из аргумента route.

kwargs – (*необязательный атрибут*) дополнительные аргументы для передачи в функцию/метод.

name - (*необязательный атрибут*) возможность задать имя для данного URL. Это имя в дальнейшем можно использовать во всём проекте для обращения к этому URL.

Новый lesson_2/urls.py

```
from django.urls import path  # функция для обработки пути

from . import views  # импорт всех view текущего проекта/приложения


urlpatterns = [
    path('index/', views.index, name='index-view') ,
    path('bio/<username>/', views.bio, name='bio') ,
]
```

Функция include из пакета django.urls

include(module, namespace=None)

Когда **django** встречает **include**, он обрезает совпадающую часть **URL**-пути, и передаёт остаток в модуль, указанный в **include**.

Атрибуты:

module – (обязательный) указывает модуль, в который нужно передать остаток **URL-**пути. Модуль может быть указан как строкой, так и **python-**объектом.

namespace – (необязательный) название пространства имен для модуля, в который передается остаток **URL-**пути.

Новый django_courses/urls.py

```
from django.urls import include, path # функции для обработки URL


urlpatterns = [
    path('lesson_1/', include('lesson_1.urls')),
    path('lesson_2/', include('lesson_2.urls')),
]
```

Возможные значения атрибута route

Напомним, что **route** – атрибут функции **path** (или **re_path**) из пакета **django.urls**, описывающий строку **URL** шаблона, который **django** сравнивает с запрошенным **URL** адресом.

Данная строка может быть 2 типов:

- Просто строка с шаблоном для функции **path**
- Выражение **Regex** (**Regular Expression**) для функции **re_path**

Строковый шаблон атрибута route для path

Правила составление шаблонов:

- Просто адрес, полное совпадение **'title/'**, при этом **'/'** должен быть только в конце, в начале **django** берет от предыдущего. Пример: **'index/'**
- Возможность передачи строки из **URL** как именованного параметра в вызываемую функцию: **'bio/<username>/'**
- Возможность передачи строки из **URL** как параметра определенного типа в вызываемую функцию **'articles/<str:title>/'**, при этом будет произведено соответствующее преобразование
- Возможность передачи строк из **URL** как параметров определенных типов в вызываемую функцию **path('articles/<str:title>/<int:section>/', views.section, name='article-section')**, при этом будет произведено соответствующее преобразование каждого параметра

Примеры строкового шаблона атрибута route

Используя данные правила, разберем варианты:

- path('index/', views.index, name='index-view') # совпадет с адресами, на конце которых есть «index/».

  # для возврата ответа вызовет функцию с аргументом по умолчанию request: views.index(request)

- path('bio/<username>/', views.bio, name='bio') # переход на /bio/user1/ вызовет views.bio(username='user1')
- path('articles/<str:title>/', views.article, name='article-detail')

  # articles/some_title/ даст в результате views.article(title='some_title', request)

- path('articles/<str:title>/<int:section>/', views.section, name='article-section') # articles/title1/5

  # вызов views.section(title='title1', section=5), где section – это обязательно int

- path('lesson_1/', include('lesson_1.urls')) # include добавит в наш UrlConf обработку URL из lesson_1.urls

Создание lesson_2/urls.py

```
from django.urls import path  # функции для обработки URL
from . import views  # импорт всех view текущего проекта/приложения


urlpatterns = [
    path('', views.index, name='index'),
    path('articles/2003/', views.special_case_2003),
    path('articles/<int:year>/', views.year_archive),
]
```

Использование регулярных выражений

Вместо `path`, можно использовать `re_path`, который позволяет использование регулярных выражений в `route` атрибуте для более точной, расширенной настройки критериев совпадения URL.

Синтаксис регулярных выражений в Python

В django для описания регулярных выражений используется тот же синтаксис, что и в модуле re в Python. Правила этого синтаксиса:

- Выражения regexp, в целом, описывают шаблон, с которым сравнивается текст
- Простые символы совпадают сами с собой, т.е. Text совпадет с Text
- Некоторые символы являются *метасимволами*, которые имеют особое значение

Regex. Метасимволы [ ] – «набор символов»

- [] – используются для определения набора символов, с которым будет проводится сравнение, например "c[abt]t" совпадет с "cat", "cbt", "ctt".
- Символы могут быть перечислены или последовательно, или в виде промежутка с использованием дефиса, например “a[b-d]c” означает тоже самое что и “a[bcd]c”. Если мы хотим чтобы нашелся только заглавный символ английского алфавита, то вы можете указать [A-Z].
- Внутри [ ] большинство метасимволов не работают, т.е. воспринимаются как обычные символы: [a$] будет совпадать с a или $.

Regex. Метасимволы ^, \

- [^] – используются для определения набора символов, с которым **не должно быть совпадения**, например "c[^abt]t" совпадет с "cct", не совпадет с "cat", "cbt", "ctt". Работает только вначале, т.е. [5^] совпадет с "5" или "^"
- \ - символ обратного слеша. Используется также как в Python – для экранирования любых метасимволов. Например "[\]]" будет совпадать с "]", а "\\" будет совпадать с "\".
- С помощью обратного слеша \ с использованием обычных символов, есть заданные наборы символов. Например, "\w" эквивалентно "[a-zA-Z0-9_]"

Regex. Наборы по умолчанию

- \d - [0-9] любая цифра.
- \D - [^0-9] любая не цифра.
- \s - [\t\n\r\f\v] любой пробельный символ.
- \S - [^ \t\n\r\f\v] любой не пробельный символ.
- \w - [a-zA-Z0-9_] любая цифра или буква в английском языке.
- \W - [^a-zA-Z0-9_] любая не цифра и не буква в английском языке.

Regex. Количественные метасимволы

- `()` – объединить символы в группу
- `?` - повторить 0 или 1 раз.
- `+` - повторить 1 или более раз.
- `*` - повторить 0 или более раз.
- `{n,m}` - повторить от `n` до `m` раз.
- `{n}` - повторить ровно `n` раз.
- `{n,}` - повторить `n` или более раз.

? - Выключение «жадности»

C(at)+ совпадет с Catat целиком

C(at)+? совпадет только с Cat

Regex. Примеры использования.

- `c(at)?` – объединить символы в группу
- `?` - повторить 0 или 1 раз.
- `+` - повторить 1 или более раз.
- `*` - повторить 0 или более раз.
- `{n,m}` - повторить от `n` до `m` раз.
- `{n}` - повторить ровно `n` раз.
- `{n,}` - повторить `n` или более раз.

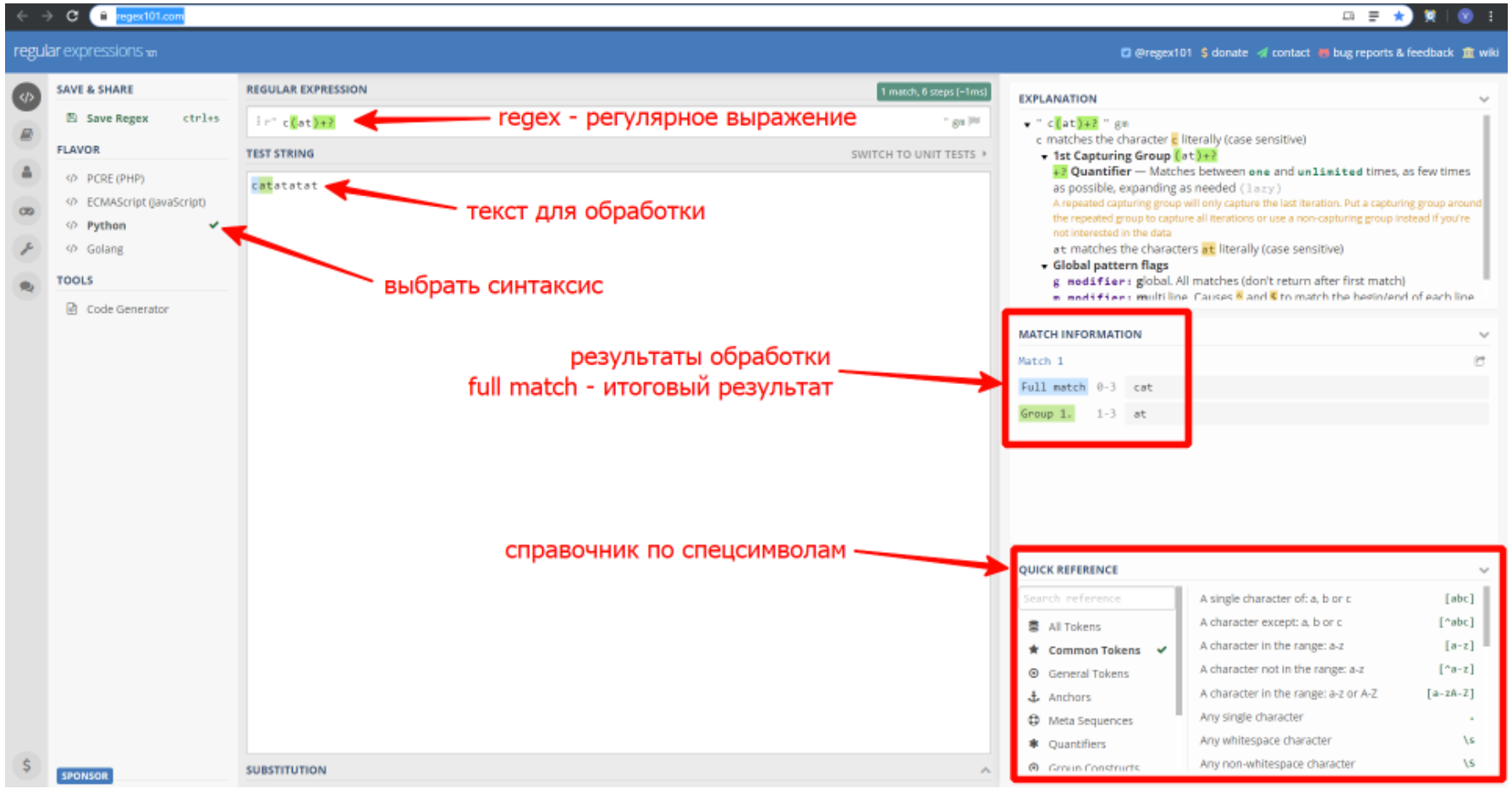
? - Выключение «жадности»

C(at)+ совпадет с Catat целиком

C(at)+? совпадет только с Cat

Regex. Проверка

https://regex101.com/



Закрепление урока

1. Как **Django** обрабатывает запросы
2. Сопоставление маршрутов и представлений
3. Файл **urls.py**
4. Модуль **path**
5. Модуль **include**
6. Возможные значения атрибута **route**
7. Использование регулярных выражений

Проверка знаний

TestProvider.com

TestProvider – это online сервис проверки знаний по информационным технологиям. С его помощью Вы можете оценить Ваш уровень и выявить слабые места. Он будет полезен как в процессе изучения технологии, так и для общей оценки знаний IT специалиста.

Проверьте как Вы усвоили данный материал на TestProvider.com

Успешное прохождение финального тестирования позволит Вам получить соответствующий Сертификат.

Спасибо за внимание! До новых встреч!

Лазорык Михаил
Software developer

Информационный видеосервис для разработчиков программного обеспечения