Диспетчеризация URL запросов Django

Процесс поиска

Запрос пользователя проходит несколько этапов, прежде чем попасть в нужную ему функцию или класс, которые обработают данный запрос и ответят пользователю:

1. Запрос попадает в корневой файл диспетчеризации (прописаный в settings → ROOT_URLCONF)

2. В данной файле DJANGO ищет список urlpatters

3. Запрос проходит по каждому шаблону, останавливается на первом соответствующем

4. Django вызывает заданое представление или другой файл диспетчеризации, если он указан, если указан файл диспетчеризации, повторяются пункты 2 и 3

5. Если ни один шаблон не удовлетворяет, Django генерирует исключение

Для включения в список urlpatterns других модулей URLConf, используется функция include

...

path(‘some_url’, include(‘app.urls’)),

…

Аргументы функции **path**:

route – строка содержащая URL шаблон

view – ссылка на представление или другой URLConf

kwargs – словарь дополнительных аргументов, которые будут переданы в представление

name для именования URL адреса, что позволяет ссылаться на него из других мест, хорошим опытом будет добавление названия приложения к имени URL адреса для избежания коллизий

Шаблон URL запроса может содержать как конвертеры пути, так и регулярные выражения

Конвертеры пути

Синтаксис: <**type:argname**>

Функция представления, отвечающая, за обработку данного шаблона, обязана принимать аргумент/ы с указанными именани в шаблоне

Допустимо указание значения по умолчанию для аргументов, на случай, если данные не были переданы в URL

Конвертеры пути по умолчанию:

str – любая не пустая строка, за исключением “/”

int – не отрицательное число

slug – любая строка, состоящая из цифр, букв ASCII, а так же “_” и “-”

uuid – соответствует форматированному UUID

path – любая не пустая строка, включая “/”, позволяет сопостовлять полный путь, а не его часть, как в *str*

Регистрация пользовательских конвертеров пути:

Для создания пользовательского конвертора, необходимо реализовать атрибут класса **regex** содержащий регулярное выражение

Метод to_python, для преобразования данных из строки в тип данных

Метод to_url, для обратного процесса, преобразования типа данных в строку, которая будет использоваться в URL. Если метод не может преобразовать, он должен вызывать исключение ValueError

Регистрация пользовательского конвертера происходит в файле диспетчеризации

from django.url import register_converter

register_converter(CustomConverter, ‘представление’)

Регулярные выражения в шаблоне

Для указания регулярных выражений в шаблоне URL, используется функция re*_path*, а не *path*

Обработка HTTP ошибок

Для обработки исключений, в Django присутствуют соответствующий handler для каждой ошибки. Посмотреть результат работы данных handler, можно только с DEBUG=False

from django.conf.urls import handler404

urlpatters = […]

handler404 = ‘app.views.page_not_found_view’

Функции представления

Простейшая функция представления, должна обязательно принимать аргумент request, а так же аргументы, переданные из URL, если таковые имеются, и возвращать объект ответа: HttpResponse, JsonResponse, StreamingHttpResponse, FileResponse в зависимости от типа возвращаемых данных, либо вызывать какую-либо ошибку в соответствии с логикой

def simple_view(request, argname):

<processing>

return HttpResponse('Hello')

Основные атрибуты HttpRequest

scheme – указание схемы запроса (http, https)

body – тело запроса в виде байт строки

path – полный путь запроса

method – метод запроса (GET, POST, …)

encoding – кодировка данных

content_type – тип контекта, извлеченный из заголовка CONTENT_TYPE

COOKIES – словарь содержащий cookies запроса

headers – заголовки запроса

Основные методы HttpRequest

get_host() - возвращает исходный хост запроса

get_port() - возвращает исходный порт запроса

is_secure() - возвращает True, если схема запроса HTTPS

Ограничение методов запроса

Для того, чтобы указать, какой именно тип запроса должен поступать на вход функции представления (по умолчанию функции представления обрабатывают любой метод запроса) Django предоставляет набор декораторов

@require_http_methods(["GET", "POST"]) – для указания нескольких методов запроса

@require_GET() – для указания, что может обрабатываться только **GET** запрос

@require_POST() - для указания, что может обрабатываться только **POST** запрос

@require_save() - для указания, что может обрабатываться безопасный метод запроса (**GET, HEAD**)

Если метод запроса не соответствует, то декораторы возвращают ошибку **HttpResponseNotAllowed**

Дополнительные функции

render

Функция, для объединения HTML шаблона с контекстом в виде словаря

Аргументы:

request – объект запроса, является обязательным аргументом функции **render**

template_name – имя используемого шаблона, является обязательным аргументом функции **render**

context – словарь с контекстом

content_type – тип возвращаемых данных, по умолчанию **text/html**

statuse – статус ответа, по умолчанию **200**

redirect

Функция перенаправления запроса на другой URL адрес, аргументами функции могут выступать модели, имя представления, абсолютный или относительный URL

get_object_or_404

Совершает SELECT запрос к указанной модели с фильтром, для получения одной записи, если записи не будет, будет сгенерировано исключение **Http404**

get_list_or_404

Совершает SELECT запрос у казанной модели с фильтром, для получения списка записей, если записей не будет, будет сгенерировано исключение **Http404**