Обработка запросов: *views.py*, *urls.py*.

Шередеко Василий, piphon@gmail.com

Диспетчеризация URL

urls.py

https://docs.djangoproject.com/en/2.0/topics/http/urls/

```
from django.urls import path
from articles import views

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

Создание шаблона URL

`django.urls.path`

```
path(pattern, view, *, name)
```

- `pattern` - Шаблон для разбора URL
- `view` - обработчик данного шаблона или другии шаблоны
- `name` - Имя шаблона URL

Создание шаблона URL

Шаблонные переменные,

- Примеры:
  - Пустой URL (``)
  - Строка (`<str:name>/`)
  - Число (`<int:id>/`)
  - Slug (`<int:id>/`)

Формирование URL

`django.urls.reverse`

```
reverse(name, *, args, kwargs)
```

- `name` - Имя шаблона URL
- `args` - Коллекция аргументов
- `kwargs` - Словарь аргументов

Подключение других шаблонов URL

`django.urls.path`

```
path(pattern, includes)
```

- `pattern` - Шаблон для разбора URL
- `includes` - коллекция URL'ов или вызов `include`

Подключение других шаблонов URL

`django.urls.include`

```
include(module)
```

- `module` - Модуль содержащий коллекцию URL'ов в переменной `urlpatterns`

Обработка ошибок

- `handler404` - ресурс не найден
- `handler500` - необработанная ошибка
- `handler403` - доступ запрещен
- `handler400` - запрос содержит ошибки

Простейшая функция вида

```
from django.http import HttpRequest, HttpResponse

def home(request: HttpRequest) -> HttpResponse:
    return HttpResponse(
        '<b>Hello world!</b>',
        content_type='text/html'
        )
```

`HttpRequest`

Основные свойства объекта:

- `request.method` - HTTP метод
- `request.path` - относительный путь до ресурса (URL)
- `request.scheme` - протокол запроса: `HTTP`
- `request.GET` - параметры запроса
- `request.POST` - значения из формы

HTTP методы:

- `GET` - получение ресурса с сервера
- `POST` - отправка ресурса на сервер
- `DELETE` - удаление ресурса на сервер
- `PUT`, `PATCH`, `OPTIONS` и другие

`HttpRequest`

Менее используемые свойства объекта:

- `request.FILES` - файлы
- `request.COOKIES` - Ку́ки браузера
- `request.session` - Сессия браузера.
- `request.scheme` - Схема запроса (`http` или `https`)
- `request.body` - Необработанные данные тела запроса

HttpResponse

```
HttpResponse(
    content,
    content_type,
    status,
    reason,
    charset
)
```

HttpResponse

Аргументы конструктора HttpResponse:

- `content`: `str | bytearray` - тело ответа
- `content_type`: `str` - MIME-тип ответа:
  - `plain/text` - обычный текст.
  - `plain/html` - HTML текст.

HttpResponse

Аргументы конструктора HttpResponse:

- `status`: `int` - код HTTP ответа.
- `reason`: `str` - Текст к коду HTTP ответа:
  - `Ok`
  - `Not found`

HttpResponse

Самые используемые HTTP статусы:

- `200` - все ок
- `301` - постоянное перенаправление
- `302` - временное перенаправление
- `400` - плохой запрос
- `401` - неавторизованный пользователь
- `403` - запрещенный ресурс
- `404` - ресурс не найден
- `405` - метод запрещен
- `418` - Я чайник (I’m a teapot)
- `500` - Ошибка приложения

Перенаправление

`django.shortcuts.redirect`

```
def view(request):
    return redirect('https://google.com/", permanent=False
```

Перенаправление

`django.shortcuts.redirect`

`redirect(to, permanent)`

- `to` - куда перенаправлять:
  - URL
  - Имя шаблона URL
- `permanent`: `bool` - постоянное или временное перенаправление.
- `*args`, `**kwargs` - передаются в формирование из шаблона URL

Декораторы видов

`django.views.decorators.http.require_http_methods`

Указание разрешенных HTTP-методов для обработки запроса:

```
@require_http_methods(["GET", "POST"])
def view(request):
    if request.method == 'POST':
        return ...
    return ...
```