



Application server

Основные типы запросов



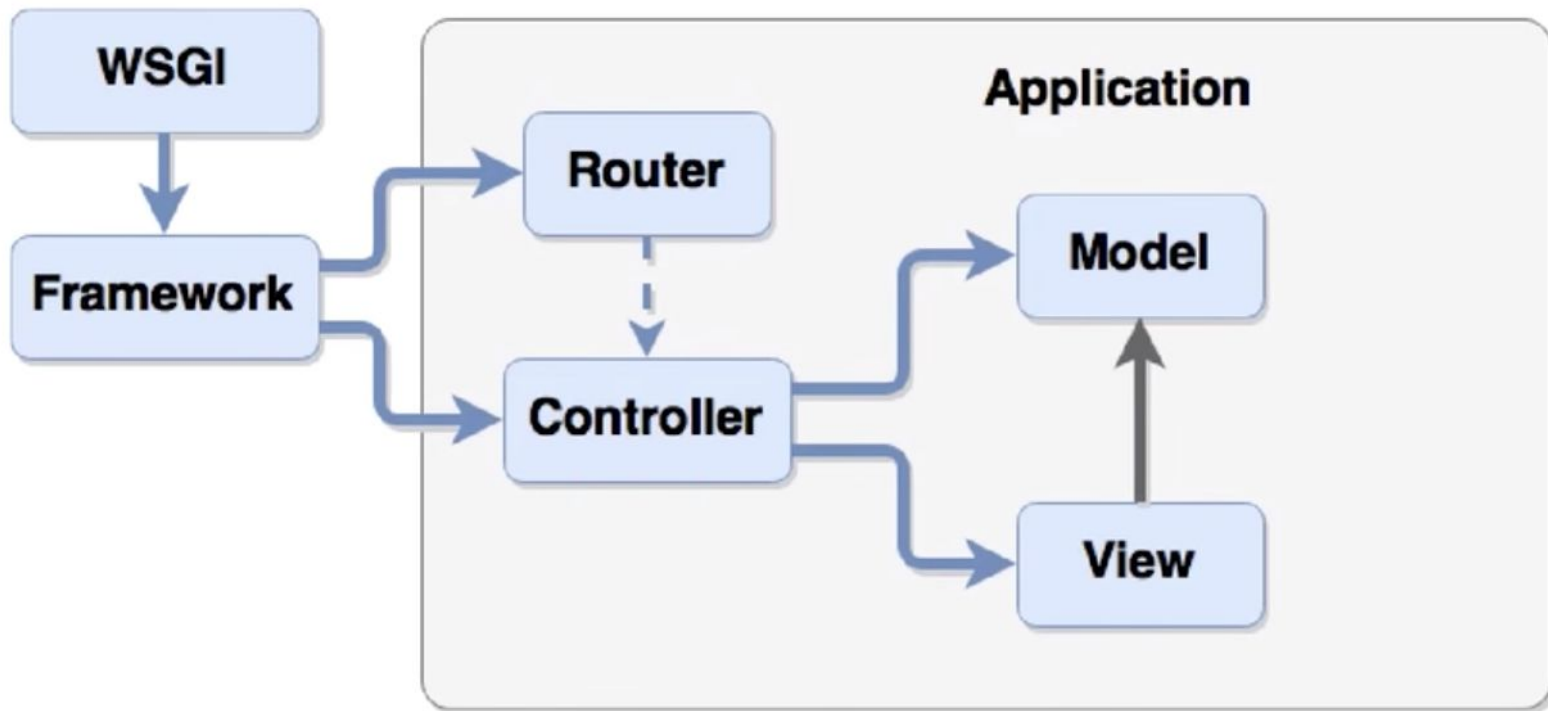
- Запросы статических файлов
- Запросы динамических документов
- Отправка данных форм
- AJAX - запросы
- Запросы к API сайтов
- Персистентные соединения

Основные задачи



- Маршрутизация URL
- Парсинг заголовков и параметров запроса
- Хранение состояния (сессии пользователя)
- Выполнение бизнес-логики
- Работа с базами данных
- Генерация HTML страниц или JSON ответа

MVC



Роли компонентов MVC



- Router - выбор контроллера по URL
- Model - реализация бизнес-логики приложения
- Controller - работа с http, связь с view
- View - генерация html или другого представления

Фреймворки MVC



Плюсы фреймворков



- Готовая архитектура
- Повторное использование кода
- Экономия ресурсов
- Участие в Open Source
- Проще найти программистов
- Проще обучать программистов

Соглашение об именовании



MVC --- MTV (Django)

Model --- Model

Router --- urls.py

Cotroller --- views.py

View --- templates

Создаем проект



```
django-admin startproject project_name
```

Структура проекта



```
messenger
|----- users
|         |----- models.py
|         |----- urls.py
|         |----- views.py
|
|----- manage.py
|----- messenger
|         |----- settings.py
|         |----- urls.py
|         |----- wsgi.py
```

Структура реального проекта (пример)



```
messenger
|----- users
|----- chats
|----- messages
...
|----- templates
|----- static
|----- manage.py
|----- application
|         |----- settings.py
|         |----- urls.py
|         |----- wsgi.py
```

Создаем приложение



```
python manage.py startapp chats
```

Конфигурация Django



```
# messenger/application/settings.py

ROOT_URLCONF = 'messenger.urls'
DATABASES = {
    'default': {
        'ENGINE':
'django.db.backends.sqlite3',
        'NAME': os.path.join(BASE_DIR,
'db.sqlite3')
    }
}

TEMPLATE_DIR = os.path.join(BASE_DIR, 'templates')
```

Пути в конфиге



Проблемы:

Проект может быть развернут в любой директории
Несколько копий проекта на одном сервере

Решения:

Абсолютные пути в каждом конфиге
Переменные окружения, \$PROJECT_PATH
Относительные пути

local_settings.py



```
# в конце messenger/application/settings.py

try:
    from .local_settings import *
except ImportError:
    pass
```

Маршрутизация в проекте



- Django начинает поиск с файла `ROOT_URLCONF`
- Загрузив файл, Django использует переменную `urlpatterns`
- Проходит по всем паттернам до первого совпадения
- Если не найдено - 404

Маршрутизация в проекте



```
# messenger/application/urls.py

from django.urls import path, re_path

urlpatterns = [
    path('', index, name='index'),
    re_path(r'^$', 'chats.views.index',
            name='index'),
    path('chats/', include('chats.urls')),
]
```

Маршрутизация в приложении



```
# messenger/chats/urls.py

from chats.views import chat_list

urlpatterns = [
    path('', chat_list, name='chat_list'),
    path('category/<int:pk>/',
         'chat_category',
         name='chat_category'),
    path('<chat_id>/', 'chat_detail',
         name='chat_detail'),
]
```

Особенности маршрутизации в Django



- Слеш (/) в начале роутов не указывается
- Можно указывать как имя, так и саму функцию
- Роуты описываются с помощью регулярных выражений
- Можно и нужно разносить роуты по приложениям
- Можно и нужно создавать именованные роуты
- Одно действие - один роут - один контроллер

Reverse routing



```
from django.core.urlresolvers import reverse
```

```
reverse('index')
```

```
reverse('chat_category', args=(10, ))
```

```
reverse('chat_detail', kwargs={'pk': 2})
```

в шаблонах

```
{% url 'chat_category' category_id %}
```

Django Views



Принимают первым параметром объект
`django.http.HttpRequest`

Возвращают объект `django.http.HttpResponse`

Django Views



```
# messenger.chats.views

# запрос вида /chat/?id=2

def chat_detail(request):
    try:
        chat_id = request.GET.get('chat_id')
        chat = Chat.objects.get(id=chat_id)
    except Chat.DoesNotExist:
        raise Http404
    return HttpResponse(chat.description,
                        content_type='text/plain')
```

Django Views



`request.method`

`request.GET`

`request.POST`

`request.COOKIES`

`request.FILES`

`request.META`

`request.session`

`request.user`

Квиз №2



Выберите вариант, где перечислены ВСЕ
валидные способы создать кортеж

1. `a = {}`; `a = tuple()`; `a = (2,)`
2. `a = ()`; `a = tuple()`; `a = (1, 2, 3,)`
3. `a = ()`; `a = tuple()`; `a = (2)`; `a = 2,`
4. `a = ()`; `a = tuple()`; `a = (2,)`; `a = 2,`

Квиз №2



Что умеет функция `type`?

1. Изменять тип объектов
2. Приводить объекты к другому типу
3. Изменять тип объектов и создавать новые типы
4. Проверять тип объектов и создавать новые типы

Квиз №2



В каком случае ваши тесты не запустятся в unittest?

1. Если в классе тесткейса не определен `__init__`
2. Если в классе тесткейса не определен `__setUp__`
3. Если тест-методы в тесткейсе не начинаются с `"test_"`
4. Если у вас установлена Django 2.X, но не установлена библиотека unittest

Квиз №2



Как итерироваться по ключу и значению в словаре?

1. `for key, value in my_dict:`
2. `for key, value in my_dict.keys(), my_dict.values():`
3. `for key, value in my_dict.items():`
4. `for key, value in my_dict.keys():`

Квиз №2



Какая ошибка будет при невалидном названии переменной?

1. `ValueError`
2. `NameError`
3. `SyntaxError`
4. `AttributeError`

Квиз №2



Выберите правильный вариант сортировки списка словарей по убыванию по ключу "age". `my_list = [{ 'name': 'Kate', 'age': 18}, { 'name': 'Mike', 'age': 17}]`

1. `sorted(my_list, key=lambda x: x[1], reverse=True)`
2. `sorted(my_list, reverse=True)`
3. `sorted(my_list, key=lambda x: x['age'], reverse=True)`
4. `sorted(my_list, key=lambda x: x['age'])`

Квиз №2



Что такое list comprehension?

1. Копирование списка
2. Создание списка на лету
3. Итератор

Квиз №2



Какой тип строк отключает экранирование?

1. f-строки
2. r-строки
3. unicode-строки
4. Экранирование всегда выключено

Квиз №2



Выберите НЕвалидный импорт

1. `from module import foo`
2. `import module`
3. `from module import *`
4. `import module as m`
5. `import foo from module`

Домашнее задание №4



- Создать и запустить Django проект - 3
- Реализовать "заглушки" для всех методов API, используя JsonResponse - 4
 - 1. Профиль
 - 2. Список чатов
 - 3. Страница чата
 - 4. Список контактов
 - (+ один метод, который должен возвращать html)
- Обрабатывать только нужные методы (GET/POST) - 1

Срок сдачи

желательно 24 октября



ТЕХНОТРЕК

**Спасибо за
внимание!**