



# **Бэкенд-разработка на Python. Лекция N°3. Application Server**

Алена Елизарова





Не забудьте  
отметиться  
на портале



# План занятия

1. Паттерн MVC
2. Веб-фреймворки
3. Создание проекта на Django
4. Концепты внутри Django

---

## Квиз N°2

1. Встроенная функция id возвращает

- Случайное целое число
- Целое число, представляющее хэш объекта
- **Уникальный целочисленный идентификатор объекта**
- Строковое представление объекта

---

## Квиз N°2

2. Сравнение идентичности объектов (не значений) выполняется оператором

- **is**
- `==`
- with ...

---

## Квиз N°2

3. Какие типы данных являются неизменяемыми

- **int**
- **str**
- list
- set
- **tuple**
- dict

## Квиз N°2

4. Асимптотическая сложность добавления элемента в словарь в среднем  
(`d["key"] = "val"`)

- **$O(1)$**
- $O(n)$
- $O(n \log(n))$
- $O(n^2)$

## Квиз N°2

5. Асимптотическая сложность удаления первого элемента списка (`lst = [1,2,...]`;  
`lst.pop(0)`)

- $O(1)$
- **$O(n)$**
- $O(n \log(n))$
- $O(n^2)$



---

## Квиз N°2

6. В python допустимо множественное наследование?

- Да
- Нет

---

## Квиз N°2

7. `class CustomList(list): pass; c = CustomList();` Что вернет `isinstance(c, list)`?

- **True**
- False

## Квиз N°2

8. `d = collections.defaultdict(list)`; Как изменится `d` после применения `d["key"].append("val")`?

- Никак, будет `AttributeError`
- `{"key": ["val"]}`
- `{"key": "val"}`
- `[{"key": "val"}]`

## Квиз N°2

9. Наличие хотя бы одного из каких методов определяет дескрипторы?  
Выберите все

- `__add__`
- **`__get__`**
- **`__set__`**
- `__getattr__`
- **`__delete__`**
- `__del__`
- `__setattr__`

---

## Квиз N°2

10. Что такое метакласс?

- Базовый класс всех встроенных типов
- **Класс, экземпляры которого являются классами**
- Класс для наследования встроенных типов
- Класс для извлечения данных из дескрипторов

---

## Квиз N°2

11. Какой механизм управления существованием объектов является обязательным и неотключаемым?

- **Счетчик ссылок**
- Сборщик мусора



# MVC

паттерн



# Основные типы запросов

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

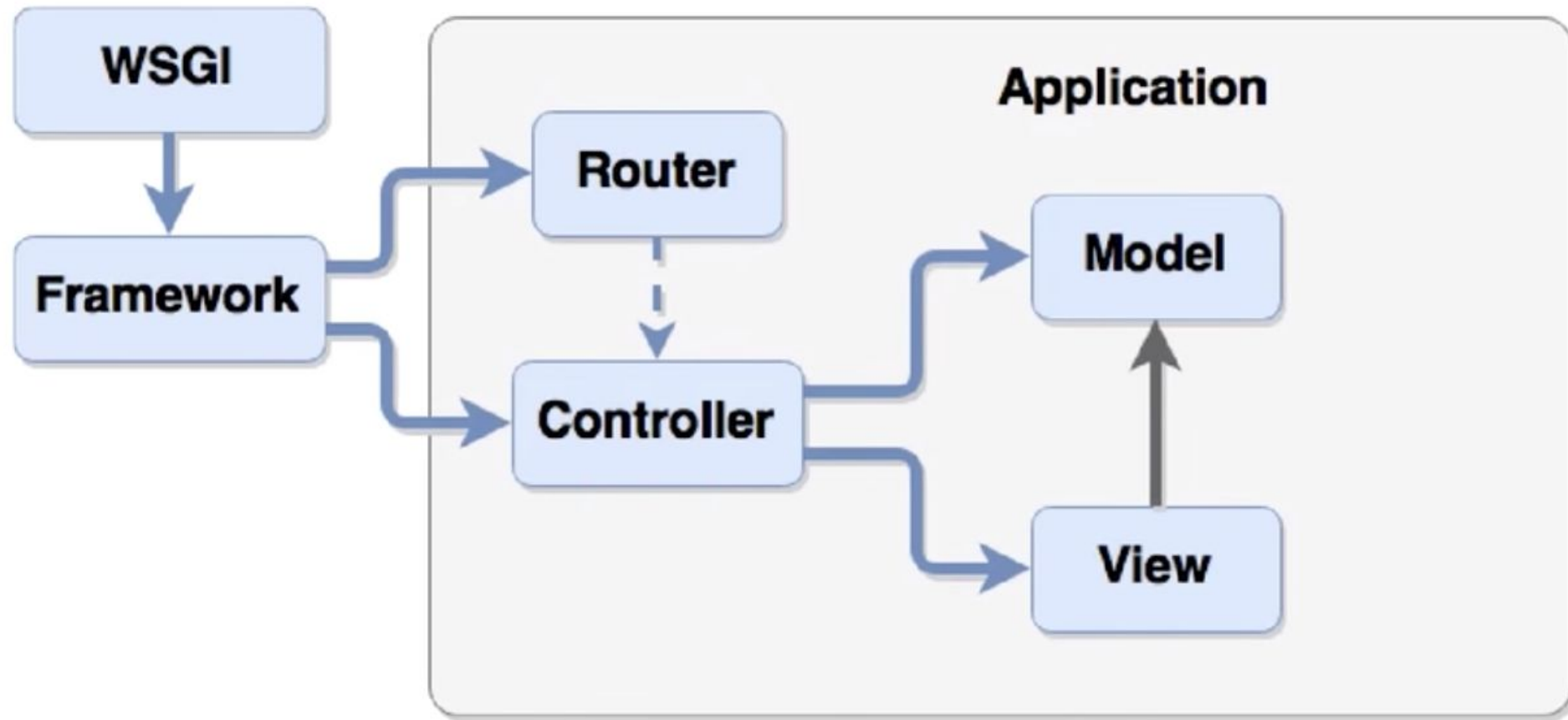




# Основные задачи

- 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

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



## Создаем приложение (внутри проекта)



```
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 путь(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`

# Django Templates

```
# in some_template.html
{% if latest_question_list %}
    <ul>
        {% for question in latest_question_list %}
            <li><a href="/polls/{{ question.id }}">{{ question.question_text
        }}</a></li>
        {% endfor %}
    </ul>
{% else %}
    <p>No polls are available.</p>
{% endif %}
```

```
# in views.py
def index(request):
    latest_question_list = Question.objects.order_by('-pub_date')[:5]
    context = {'latest_question_list': latest_question_list}
    return render(request, 'polls/index.html', context)
```

# Django Forms

```
# in some_form.html
<form action="/your-name/" method="post">
    <label for="your_name">Your name: </label>
    <input id="your_name" type="text" name="your_name" value="{{
current_name }}">
    <input type="submit" value="OK">
</form>
```

```
# in forms.py
class ContactForm(forms.Form):
    subject = forms.CharField(max_length=100)
    message = forms.CharField(widget=forms.Textarea)
    sender = forms.EmailField()
    cc_myself = forms.BooleanField(required=False)
```

---

# Django Forms

```
# in forms.py
class ArticleForm(ModelForm):
    class Meta:
        model = Article
        fields = ['pub_date', 'headline', 'content', 'reporter']
```

```
# in views.py
def some_view(request):
    form = ArticleForm(request.POST)
    if form.is_valid():
        form.save()
        return JsonResponse({})
    return JsonResponse({'errors': form.errors}, status=400)
```





# Документация

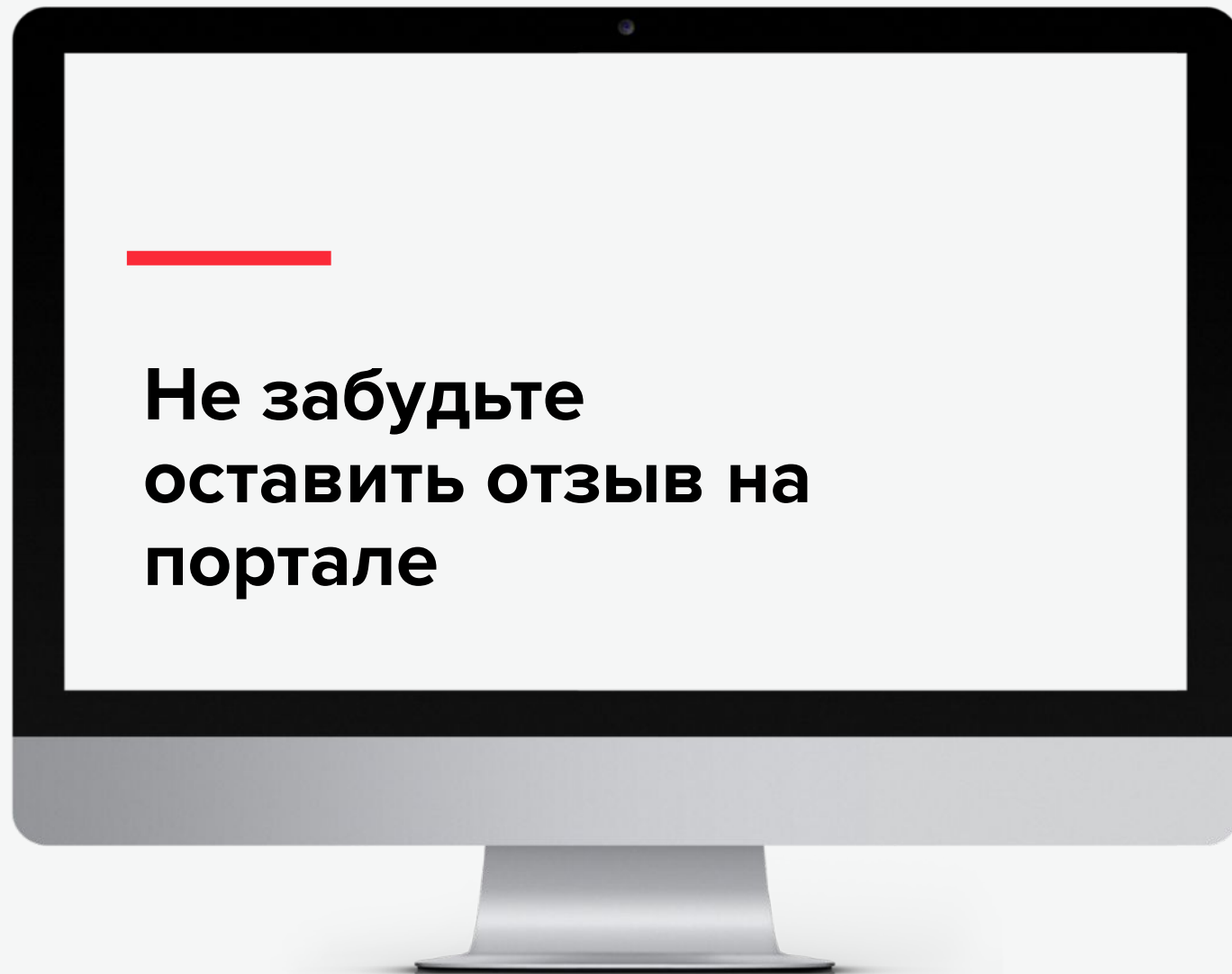
<https://docs.djangoproject.com/en/3.1/>

<https://www.feldroy.com/products/two-scoops-of-django-3-x>

<https://www.amazon.com/Django-APIs-Build-web-Python/dp/1093633948>

# Домашнее задание

- Создать и запустить Django проект
- Реализовать "заглушки" для методов API, используя JsonResponse (для проверки API можем использовать Postman и другие инструменты)
  1. Создание объекта
  2. Список объектов
  3. Детальная информация об объекте
  4. Функция, которая возвращает отрендеренный html (например, главная страница приложения)
- Внутри контроллера обрабатывать только нужные методы (GET/POST)



**Не забудьте  
оставить отзыв на  
портале**



**СПАСИБО  
ЗА ВНИМАНИЕ**

