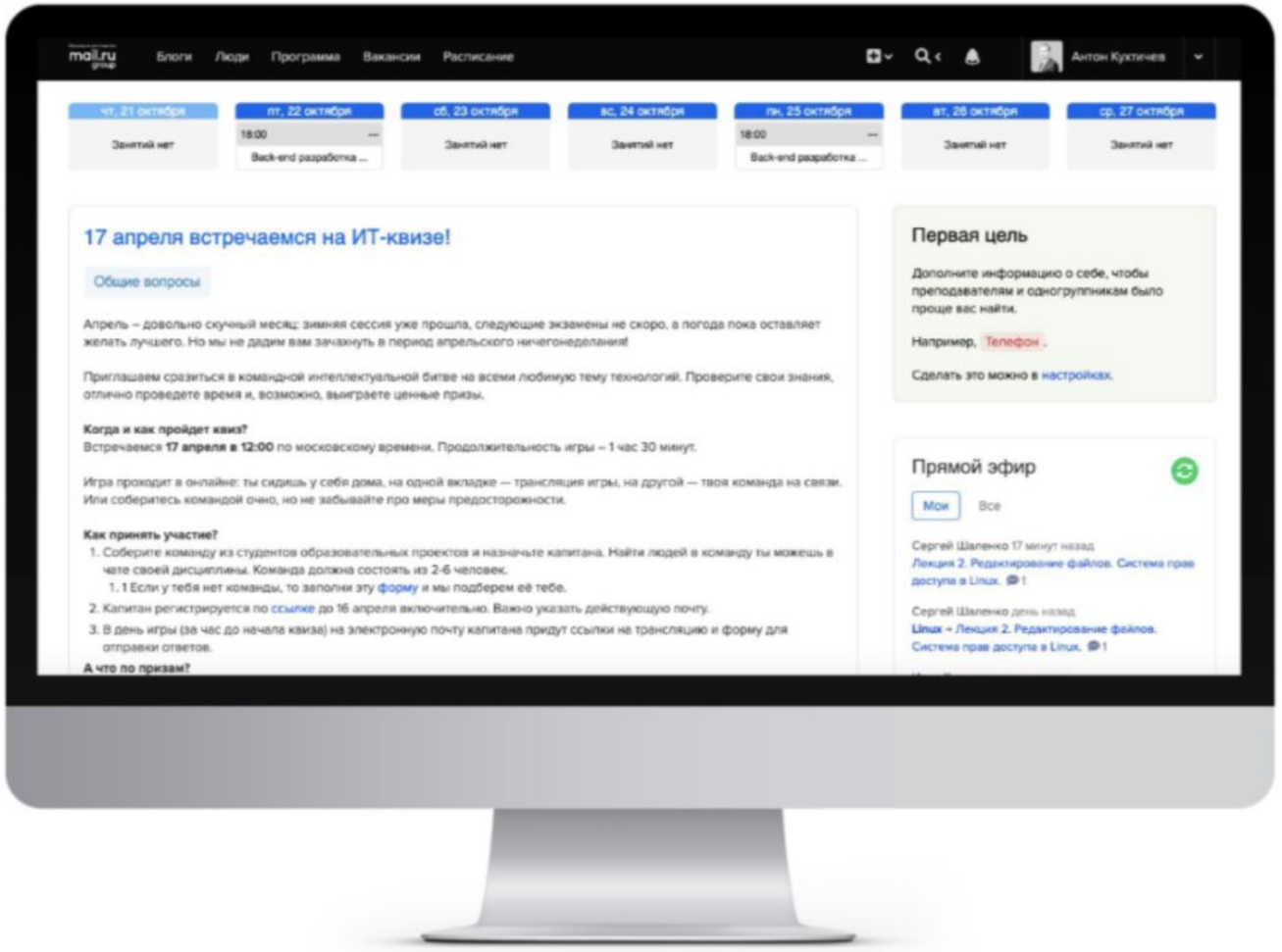
Бэкенд-разработка на Python. Лекция №4. Application Server

Алена Елизарова

образование

Напоминание отметиться на портале



План занятия

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

MVC

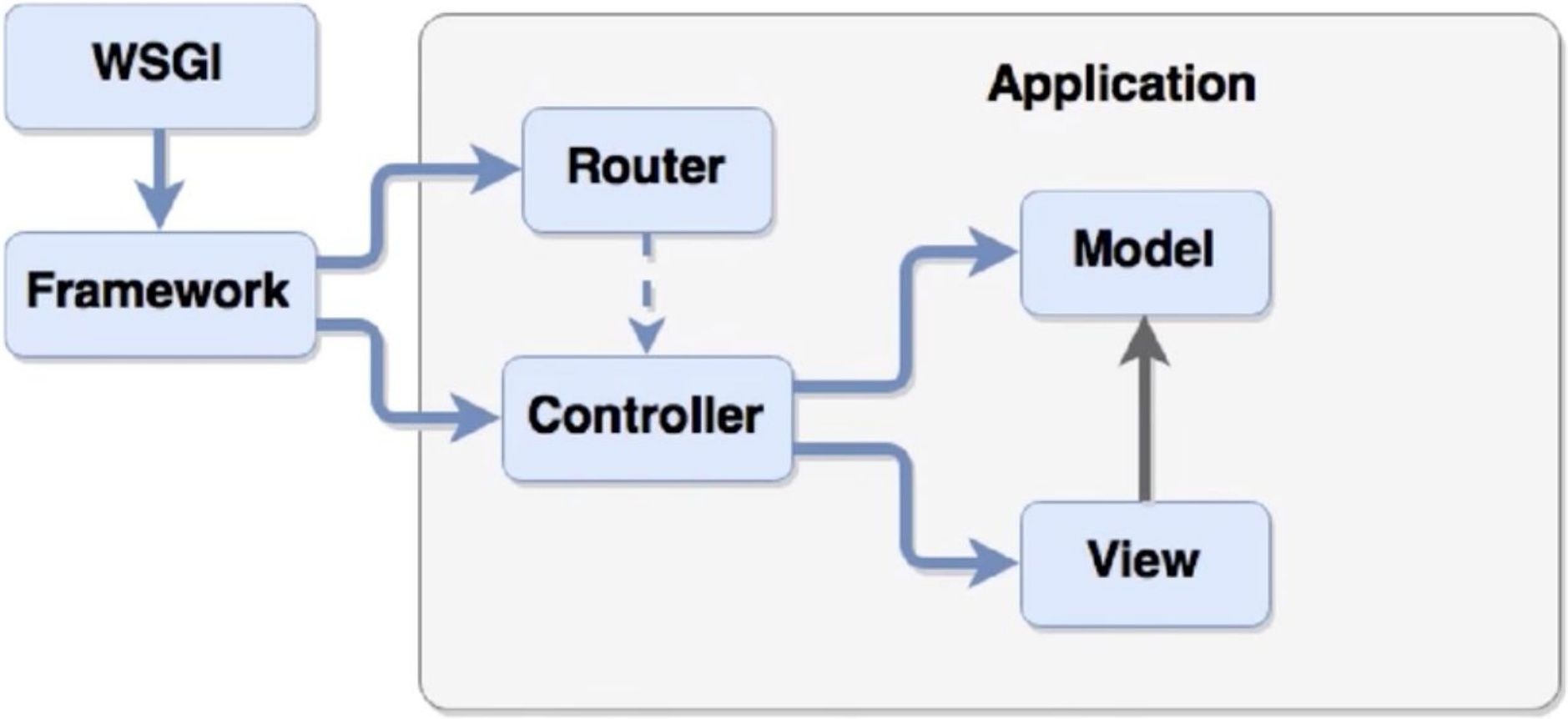
Основные типы запросов

- 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 состоянии (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

Django

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

Django Views

Принимают первым параметром объект django.http.HttpRequest

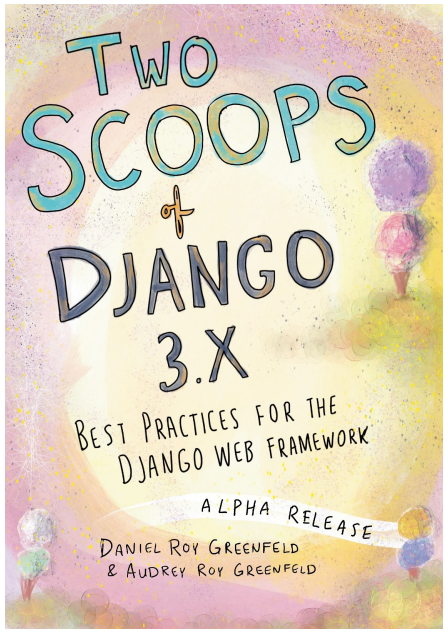
Возвращают объект django.http.HttpResponse

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 (ClassBased)

```
# views.py
from django.shortcuts import get_object_or_404
from django.views.generic import ListView
from books.models import Book, Publisher

class PublisherBookListView(ListView):
    template_name = 'books/books_by_publisher.html'
    def get_queryset(self):
        self.publisher = get_object_or_404(Publisher, name=self.kwargs['publisher'])
        return Book.objects.filter(publisher=self.publisher)

# urls.py
    path('books/<publisher>/', PublisherBookListView.as_view()),
```



What Type of View Should You Use?

A handy flowchart for those moments when you can't decide whether to implement a particular view as function-based or class-based.

Start Here

Do one of the generic class-based views closely match what you have in mind?

no

Do you need to subclass your view to create other views?

no

yes

yes

Can you use a class-based view just by overriding attributes?

no

Do you have to dive into Django's source code just to get it working as a class-based view?

yes

Will your view be painfully complex to implement with a class-based view?

e.g. Does your view process more than one form?

yes

no

yes

Use a class-based view, then!

Use a function-based view, then!

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)
```

Middlewares

```
class UserIdHeaderMiddleware(object):

    def __init__(self, get_response):
        self.get_response = get_response
        self.logger = logging.getLogger(intranet)

    def __call__(self, request):
        response = self.get_response(request)
        try:
            response['X-User-Id'] = request.user.id or 'anonymous'
        except Exception as e:
            self.logger.exception(e)
        return response
```

Middlewares

```
class DeletedMiddleware(object):
    def __init__(self, get_response):
        self.get_response = get_response

    def __call__(self, request):
        return self.get_response(request)

    def process_exception(self, request, exception):
        # ObjectWasDeleted - custom error
        if isinstance(exception, ObjectWasDeleted):
            return render(request, 'deleted.html')
```

Документация

https://docs.djangoproject.com/en/3.2/

https://docs.djangoproject.com/en/3.2/faq/

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

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

Домашнее задание

1. Создать и запустить Django проект

2. Реализовать "заглушки" для методов API, используя JsonResponse

   (для проверки API можем использовать Postman и другие инструменты)

   - Создание объекта
   - Список объектов
   - Детальная информация об объекте

3. Функция, которая возвращает отрендеренный html (например, главная страница приложения)
4. Внутри контроллера обрабатывать только нужные методы (GET/POST)

Не забудьте оставить отзыв на портале

Спасибо
за внимание

образование