ТЕХНОТРЕК

Лекция 7

Реализация API

Елизарова Алена

Application programming interface (API)

Описание способов (набор классов, процедур, функций, структур или констант), которыми одна компьютерная программа может взаимодействовать с другой программой.

Форматы передачи данных

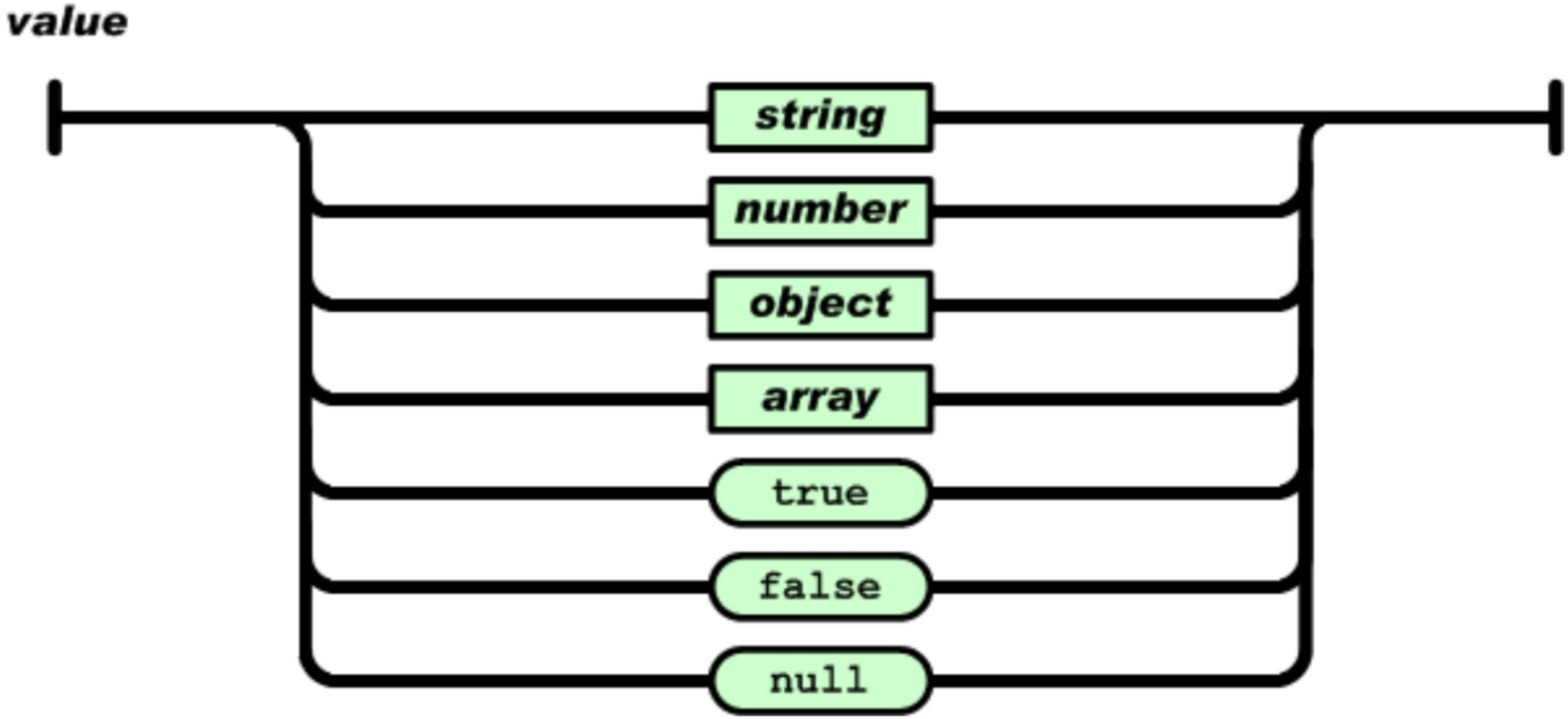
- Текстовые форматы (JSON, XML, CSV);
- Бинарный формат (Apache Thrift, Protocol Buffers);

Формат JSON

JSON (JavaScript Object Notation) — текстовый формат обмена данными, основанный на JavaScript.

```
{
    "first_name": "Иван",
    "last_name": "Иванов",
    "phone_numbers": [
        "812 123-1234",
        "916 123-4567"
    ]
}
```

Формат JSON



Формат JSON

Преимущества:

- Легко читается человеком;
- Компактный;
- Для работы с JSON есть множество библиотек;
- Больше структурной информации в документе.

Формат JSON

- JSON - это формат данных - он содержит только свойства, а не методы;
- JSON требует двойных кавычек, которые будут использоваться вокруг строк и имен свойств;
- Вы можете проверить JSON с помощью приложения, такого как jsonlint;
- JSON может фактически принимать форму любого типа данных, который действителен для включения внутри JSON, а не только массивов или объектов.

Декораторы в Python

Это функция, которая принимает функцию в качестве единственного аргумента и возвращает новую функцию, с дополнительными функциональными возможностями.

```
def my_decorator(function):
    def wrapper(*args, **kwargs):
        print('It is decorator logic')
        return function(*args, **kwargs)
    return wrapper

@my_decorator
def foo():
    print('It is main function')
```

Сериализация данных из БД

```
from django.http import JsonResponse
from chat.models import Chat

def chat_detail(request, chat_id):
    chat = get_object_or_404(Chat, id=chat_id)
    return JsonResponse({
        'data': {'id': chat_id, 'title': chat.title}
    })


def chat_list(request):
    chats = Chat.objects.filter(is_active=True).values(
        'id', 'title', 'description'
    )
    return JsonResponse({
        'data': list(chats)
    })
```

Class-based views

```
from django.http import HttpResponseRedirect
from django.shortcuts import render
from django.views import View

from .forms import MyForm

class MyFormView(View):
    form_class = MyForm
    initial = {'key': 'value'}
    template_name = 'form_template.html'

    def get(self, request, *args, **kwargs):
        form = self.form_class(initial=self.initial)
        return render(request, self.template_name, {'form': form})

    def post(self, request, *args, **kwargs):
        form = self.form_class(request.POST)
        if form.is_valid():
            # <process form cleaned data>
            return HttpResponseRedirect('/success/')

        return render(request, self.template_name, {'form': form})
```

Functional views

```
from django.http import HttpResponseRedirect
from django.shortcuts import render

from .forms import MyForm

def myview(request):
    if request.method == "POST":
        form = MyForm(request.POST)
        if form.is_valid():
            # <process form cleaned data>
            return HttpResponseRedirect('/success/')
    else:
        form = MyForm(initial={'key': 'value'})

    return render(request, 'form_template.html', {'form': form})
```

Django Form

```
# forms.py
from django import forms


class FeedbackForm(forms.Form):
    email = forms.EmailField(max_length=100)
    message = forms.CharField()

    def clean(self):
        if is_spam(self.cleaned_data):
            self.add_error('message', 'Это спам')
```

Django Form

```
# forms.py
from django import forms

class PostForm(forms.Form):
    title = forms.CharField(max_length=100)
    text = forms.CharField()
    days_active = forms.IntegerField(required=False)

    def clean_text(self):
        if is_correct(self.cleaned_data['message']):
            return self.cleaned_data['message']
        return 'Текст содержал нецензурную лексику и был удален'

    def save(self):
        return Post.objects.create(**self.cleaned_data)
```

Django Form

```
# forms.py
from django import forms

class PostForm(forms.Form):
    title = forms.CharField(max_length=100)
    text = forms.CharField()
    days_active = forms.IntegerField(required=False)

    def clean_text(self):
        if is_correct(self.cleaned_data['message']):
            return self.cleaned_data['message']
        return 'Текст содержал нецензурную лексику и был удален'

    def save(self):
        return Post.objects.create(**self.cleaned_data)
```

Django Form

`BooleanField` - флаг

`CharField` - текстовое поле

`EmailField` - почтовый адрес

`PasswordField` - пароль

`DateField` - дата

`DateTimeField` - время и дата

`FileField` - загрузка файла

Django ModelForm

```
# forms.py
from django import forms

class PostForm(forms.ModelForm):
    class Meta:
        model = Post
        fields = [‘title’, ‘text’]


# метод save уже определен
# сохраняем в модель, указанную в Meta
# валидация полей проходит через типы, объявленные в модели
```

Django ModelForm

```
# forms.py
from django import forms

class PostForm(forms.ModelForm):
    class Meta:
        model = Post
        fields = [‘title’, ‘text’]


# метод save уже определен
# сохраняем в модель, указанную в Meta
# валидация полей проходит через типы, объявленные в модели
```

Валидация формы в views

```
def add_post(request):
    form = PostForm(request.POST)
    if form.is_valid():
        post = form.save()
        return JsonResponse({
            'msg': 'Пост сохранен',
            'id': post.id
        })
    return JsonResponse({'errors': form.errors},
status=400)
```

Django Rest Framework

https://www.django-rest-framework.org/

Домашнее задание №7

Реализовать методы для:

- отправки сообщения
- получения списка сообщений чата
- прочтения сообщения

Валидировать входные параметры API с помощью форм

Переписать заглушки всех предыдущих методов

Срок сдачи

желательно 14 ноября

ТЕХНОТРЕК

Спасибо за внимание!