Разработка программного обеспечения на языке Python

Лекция 5. Модель пользователя

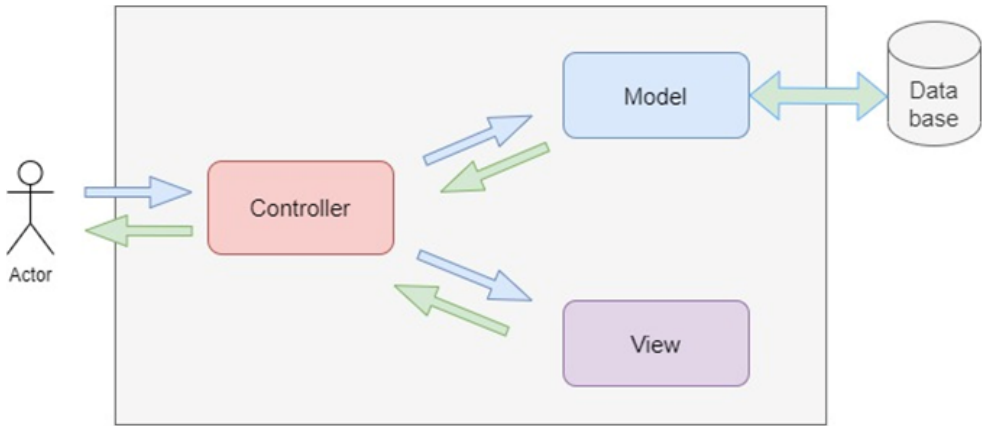
Посмотрите видеоуроки и ответьте на контрольные вопросы после лекции

Архитектура проекта Django

00:00 / 03:34

В данной теме обобщим материал, изученный ранее, и рассмотрим архитектуру проекта джанго.

При создании веб-приложений наиболее распространен архитектурный паттерн MVC – Model View Controller.

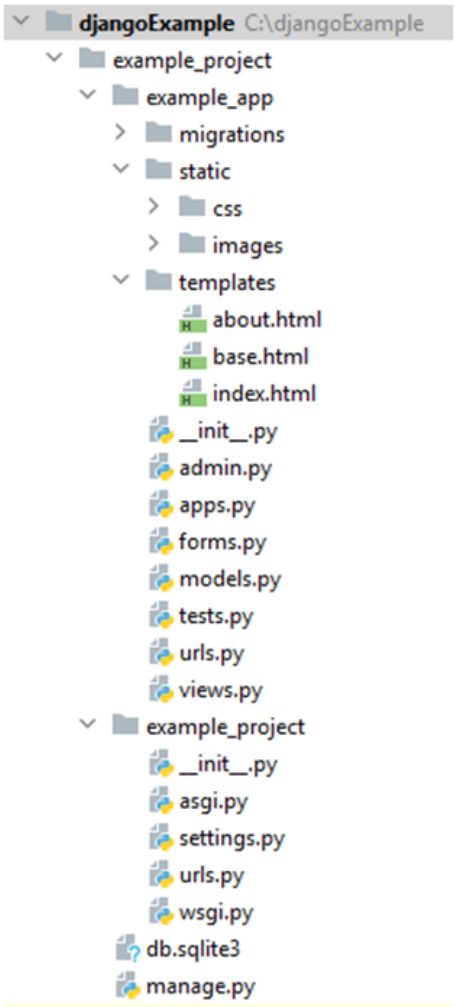


Способ организации кода MVC («Модель-представление-контроллер») включает следующие компоненты:

- Модель – логическое представление данных, совокупность методов, правил и ограничений работы с данными.
- Представление (view) – компонент, отображающий пользователю данные в зависимости от изменения модели.
- Контроллер – программный посредник, обрабатывающий действия пользователя и сообщающий модели, как она должна измениться.

В Django используется аналогичный по смыслу паттерн, но имеет он немного другое название – MTV – Model Template View.

Рассмотрим на примере, как может выглядеть приложение на Django. Так, общая структура проекта джанго включает каталог с названием проекта, каталоги приложений (в нашем примере это одно приложение) и файла manage.py с помощью которого происходит управление проектом – например, запуск сервера. В папке проекта находятся его настройки и маршрутизатор. В папке приложения – файлы, необходимые для его работы, в том числе модели и представления.



Слой Model является слоем доступа к данным. Здесь хранятся модели ORM. То есть описание классов, на основе которых в дальнейшем будут созданы таблицы базы данных. Каждый класс соответствует одной таблице и содержит описание ее полей.

models.py

```python
from django.db import models


class Company(models.Model):
    name = models.CharField(max_length=50)

    def __str__(self):
        return str(self.name)


class Product(models.Model):
    name = models.CharField(max_length=50)
    price = models.IntegerField()
    company = models.ForeignKey(Company, on_delete=models.CASCADE)

    def __str__(self):
        return str(self.name)+" "+str(self.company.name)
```

Слой Template содержит представления, в данном случае это страницы html. На слайде вы видите пример кода шаблонизатора вместе с html разметкой. Например, страница index.html содержит форму для ввода пользовательских данных и таблицу, в которой будут выведены записи из базы данных. Здесь также подключаются статические файлы.

index.html

```html
<!DOCTYPE html>
{% load static %}
<html>
<head>
    <meta charset="utf-8" />
    <title>Store</title>
    <link rel="stylesheet" href="{% static 'css/style.css' %}" />
</head>
<body class="container">
    <form method="POST" action="create/">
        {% csrf_token %}
        <table>{{form}}</table>
        <input type="submit" value="Добавить" >
    </form>

    {% if products.count > 0 %}
    <h2>Список товаров</h2>
    <table class="t">
        <tr><th>Номер</th><th>Название</th><th>Цена</th><th>Поставщик</th></tr>
        {% for product in products %}
        <tr><td>{{ product.id }}</td><td>{{ product.name }}</td>
            <td>{{ product.price }}</td><td>{{ product.company.name }}</td></tr>
        {% endfor %}
    </table>
    {% endif %}
</body>
</html>
```

Слой Views схож со слоем Controller в паттерне MVC.

Но, в отличии от стандартной архитектуры MVC, в джанго во views отсутствует маршрутизация url, так как они обрабатываются Django в отдельном файле. На скриншоте показана функция, соответствующая главной странице. В ней в зависимости от метода вызова происходит переход на нужную страницу, либо выводится сообщение об ошибке.

views.py

```python
from django.shortcuts import render
from django.http import HttpResponse, HttpResponseRedirect
from .models import *
from .forms import *


def index(request):
    if request.method == "POST":
        form = ProductForm(request.POST)
        # проверка валидности данных
        if form.is_valid():
            # получение данных поля name формы
            name = form.cleaned_data["name"]
            price = form.cleaned_data["price"]
            form.save()
            return HttpResponseRedirect("/")
        else:
            return HttpResponse("Некорректные данные!")
    else:
        form = ProductForm()
        products = Product.objects.all()
        return render(request, "index.html", {"form": form, "products": products})
```

Все URL и связанные с ними функции прописываются в отдельном файле. Для этого используются регулярные выражения.

urls.py

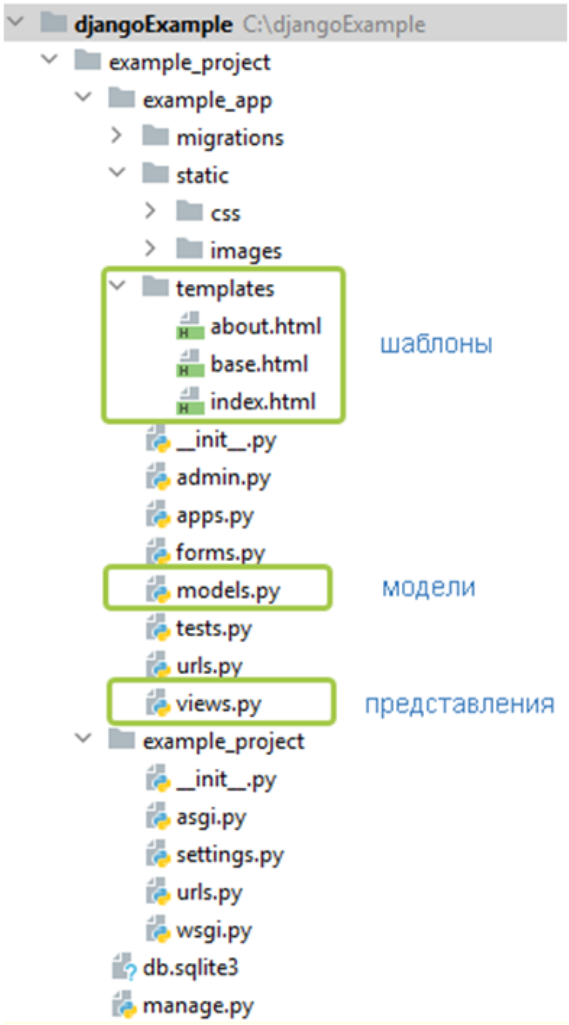
```python
from django.urls import path
from .views import *

urlpatterns = [
    path('', index),
    path('create/', create),
    path('about/', about),
    path('info/', info),
    path('contact/', contact),
]

```

В файле settings прописываются все настройки проекта, такие как хост и порт, пути к директориям и т.д. На скриншоте показан пример приложений, т.е. пакетов Python с определенным функционалом, используемых в проекте.

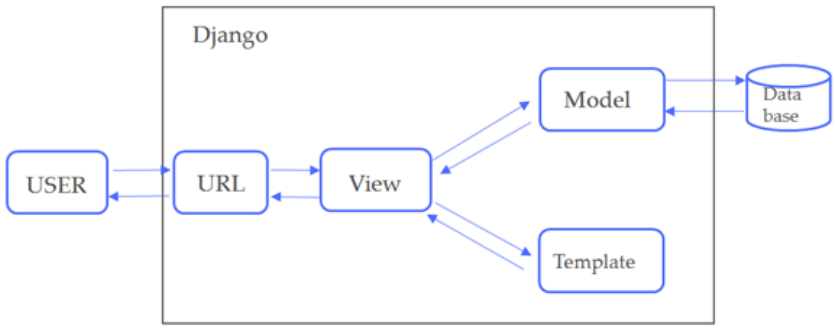
settings.py

```python
# Application definition

INSTALLED_APPS = [
    'example_app',
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
]

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]
```

Так, общая структура проекта со всеми описанными компонентами может выглядеть следующим образом.



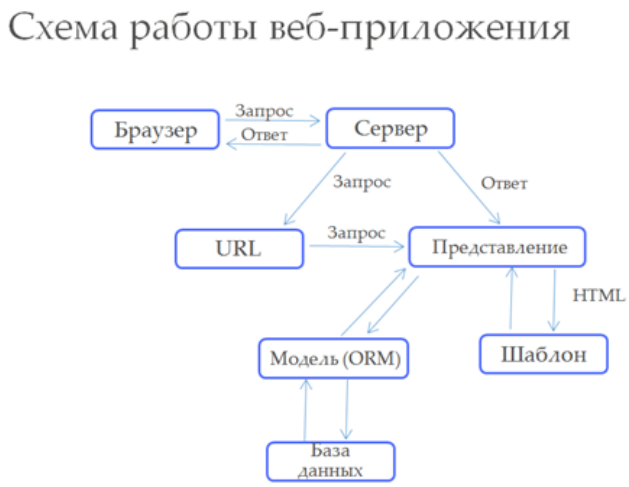
Django реализует архитектурный паттерн MVT (Модель Представление Шаблон).



Таким образом, структура Django включает следующие компоненты:

- URL-маршрутизаторы, перенаправляющие HTTP-запрос от браузера или любого иного веб-клиента в представления;
- Представление View: получает запрос, обрабатывает его и отправляет пользователю некоторый ответ. Для создания ответа может применять шаблоны Template. Представление описывает основную логику и алгоритмы приложения, обращение к модели и базе данных. В архитектуре MVC этому компоненту соответствуют контроллеры.
- Model: описывает схему данных. Отдельные классы соответствуют таблицам в базе данных.
- Template: Это часть приложения, которую пользователь видит в браузере. Часто это HTML-шаблоны, которые используются представлением для демонстрации пользователю полученных от модели данных. В MVC этому компоненту соответствует View, то есть представления.

Примерная схема работы приложения на Django: Когда к приложению приходит запрос, то URL определяет, с каким ресурсом сопоставляется данный запрос. Ресурс – функция представления View – получает запрос и определенным образом обрабатывает его. При этом может обращаться к моделям и базе данных, получать из нее данные, или, наоборот, сохранять их. Результат обработки запроса отправляется обратно, и этот результат пользователь видит в своем браузере. Как правило, результат обработки запроса представляет сгенерированный html-код на основе шаблонов (Template).



Итак, мы рассмотрели фреймворк Django, и увидели, что при его настройке следует соблюдать определенные правила и придерживаться определенной структуры организации кода.

Модель пользователя

ПРЕДЫДУЩИЙ ЭЛЕМЕНТ КУРСА

◄ Задание 8. Вывод данных

Перейти на...

СЛЕДУЮЩИЙ ЭЛЕМЕНТ КУРСА

Задание 9. Личный кабинет администратора ►

Контакты +7(391) 206-27-05
info-ms@sfu-kras.ru