Framework – программная платформа, определяющая структуру проекта, облегчающая разработку.

Django – Web Framework на Python

Установка:

pip install django

Создание проекта:

django-admin startproject projectname

Структура Django проекта:

Django использует **MVC** (Model – View – Controller) шаблон проектирования.

- **Model** – описание таблицы в БД
- **View** – интерфейс с которым взаимодействует пользователь
- **Controller** – связующий объект между *model* и *view* (обрабатывает действия пользователя)

Создание приложения:

python manage.py startapp appname

После создания приложения, его необходимо добавить в список *installed_apps* в файле *settings.py* (зарегистрировать)

Django ORM (Object Relational Mapping) – объекто реляционное представление:

Описание моделей:

Все модели описываются в файлах *models.py*

Все модели должны наследоваться от базовой модели *Model*

Django автоматически добавляет *Primary Key* в каждую таблицу, в результате чего, нет необходимости в явном его указании (если его не надо переопределить)

Структура указания поля (атрибута) модели:

имя_атрибута = ТипАтрибута(опции)

Общие опции модели:

null: bool – возможность хранить пустые значения как *NULL*

blank: bool – может ли поле быть пустым

choices: tuple[tuple[any, any]] – варианты выбора значений (выпадающий список) для данного атрибута

default: any – значение по умолчанию, может быть значение или вызываемый объект (функция), если вызываемый объект, он будет вызываться каждый раз, когда создается новый объект

help_text: str – справочный текст для отображение с виджетом формы

primary_key: bool – первичный ключ (если стандартный необходимо переопределить)

verbose_name: str – удобочитаемой имя для поля, первая буква должна быть маленькой, *django* самостоятельно применит метод *title* там, где это необходимо

unique: bool – уникальность значений

В различных типах полей, присутствуют так же дополнительные атрибуты, такие как *max_length* к примеру

Класс **Meta –** используется для настройки модели в БД, а так же для отображения в админ панеле и тд.

Основные атрибуты класса **Meta:**

db_table: str – название таблицы в БД (по умолчанию используется название класса)

verbose_name: str – удобочитаемое имя таблицы в ед. ч.

verbose_name_plural: str – удобочитаемое имя таблицы в мн. ч.

ordering: tuple[str, str] – указание сортировки, если перед названием стоит “-”, это указывает на обратную сортировку

abstract: bool – указывает, что модель является базовым абстрактным классом (не создается в БД, используется в качестве родительского класса)

app_label: str – указание к какому приложение относится модель, если модель определена не в приложении, зарегистрированном в *installed_apps*

managed: bool – False – Django не управляет жизненным циклом таблицы (не создает и не удаляет), применяется, если таблица была уже представлена в БД (создана другим способом), вся остальная работа с данной таблицей остается неизменной

```python
from django.db import models


class Category(models.Model):
    name = models.CharField(
        max_length=24,
        unique=True,
        verbose_name='название',
        help_text='макс. 24'
    )
    parent = models.ForeignKey(
        'Category',
        on_delete=models.CASCADE,
        null=True,
        blank=True,
        verbose_name='родительская категория'
    )
    is_published = models.BooleanField(default=False)

    class Meta:
        db_table = 'categories'
        verbose_name = 'категория'
        verbose_name_plural = 'категории'
        ordering = ('-name', 'id')
```

Создание файлов миграции и мигрирование в БД:

python manage.py makemigrations – создает файлы миграции

python manage.py migrate – мигрирование в БД

Создание Супер Пользователя (Админ):

python manage.py createsuperuser

Добавление модели в Админ Панель:

Добавление моделей в админ панель, а так же кастомизация их отображения в Админ Панеле осуществляется в файле *admin.py*

Если необходимо изменять отображение модели в Админ Панеле, создается специализированный класс для кастомизации, данный класс наследуется от **ModelAdmin**

Основные опции ModelAdmin:

date_hierarchy: str – имя поля с типом *DateField* или *DateTimeField*, указание данной опции, добавляет навигацию по дате

empty_value_display: str – переопределяет отображение пустого (NULL) поля в БД

exclude: tuple[str] – список/кортеж имен полей, который не будут отображаться в форме

list_display: tuple[str] – список/кортеж имен полей, для отображения в списке

list_filter: tuple[str] – список/кортеж имен полей, для добавления фильтров, поле должно быть одного из типов: *BooleanField, CharField, DateField, DateTimeField, IntegerField, ForeignKey, ManyToManyField*

list_max_show_all: int – количество отображаемых записей при нажатии на “Показать все”

list_per_page: int – количество отображаемых записей на одной странице

search_fields: tuple[str] – список/кортеж имен полей, по которым можно производить поиск

search_help_text: str – текст подсказка для окна поиска (с Django 4.0)

```python
from django.contrib import admin

from .models import Category


@admin.register(Category)
class CategoryAdmin(admin.ModelAdmin):
    empty_value_display = 'H/Y'
    list_display = ('name', 'parent', 'is_published')
    list_filter = ('name', 'is_published', 'parent')
    search_fields = ('name', 'parent', 'id')
    search_help_text = 'введите имя категории/родительской категории или id'
```