Django Starter

Модели (часть 1)

Introduction

Лазорык Михаил

Software developer, 3 года опыта

mykhailo.lazoryk

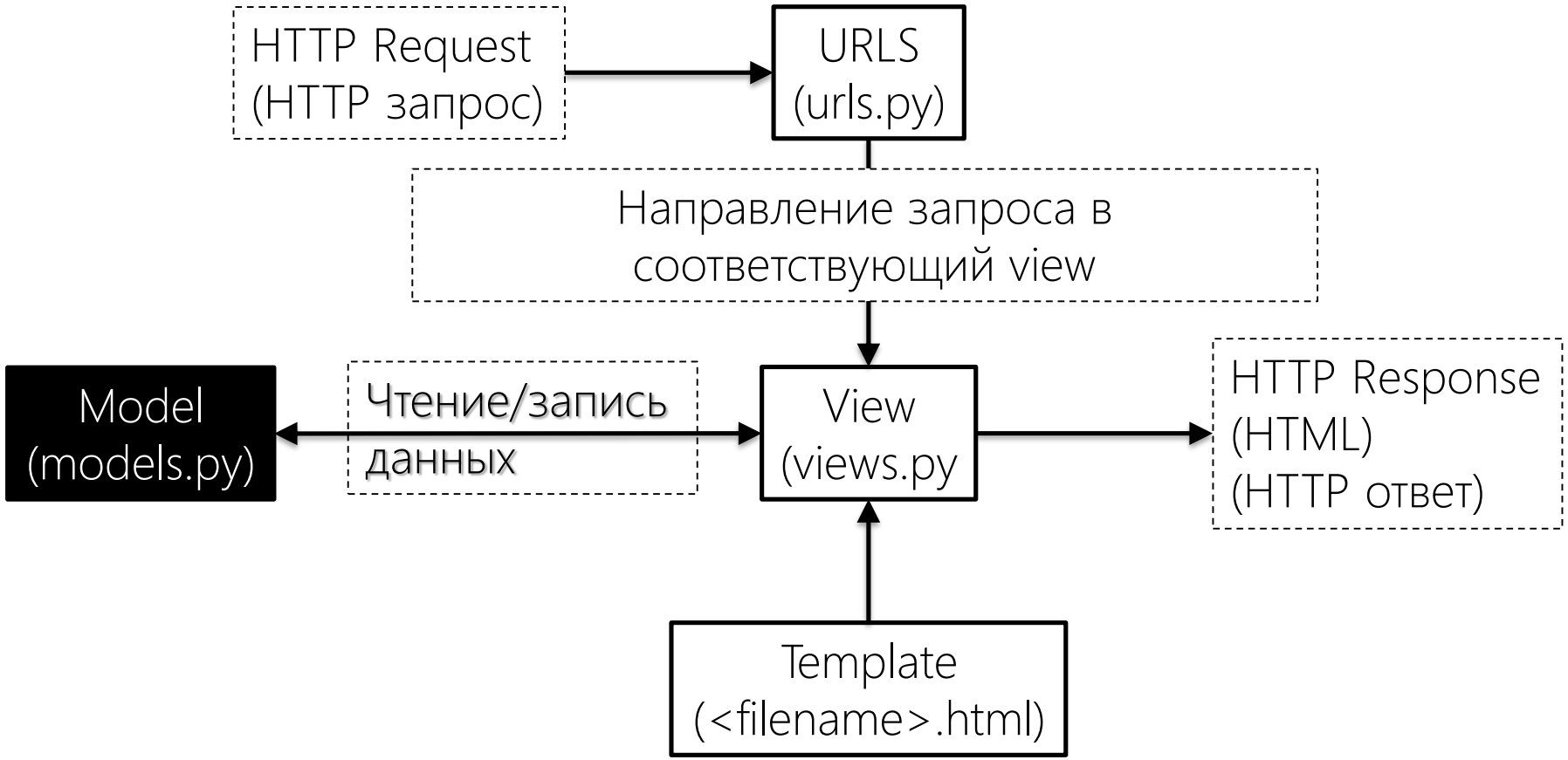
mykhailo-lazoryk

Модели (часть 1)

План урока

1. Модели в Django
2. Подключение PostgreSQL
3. Поля моделей
4. Отношение моделей

Структура файлов в реализации MVC в Django (MTV)



Модели в Django

Модели - это представление данных в качестве классов и их свойств.

Каждая модель представляет собой класс Python, который является подклассом

django.db.models.Model.

Например:

```
from django.db import models

class Person(models.Model):

    first_name = models.CharField(max_length=30)

    last_name = models.CharField(max_length=30)
```

```
CREATE TABLE myapp_person (

    "id" serial NOT NULL PRIMARY KEY,

    "first_name" varchar(30) NOT NULL,

    "last_name" varchar(30) NOT NULL);
```

Сравнение PostgreSQL vs SQLite

Преимущества

- Файловая: вся база данных хранится в одном файле, что облегчает перемещение.
- Стандартизированная: SQLite использует SQL; некоторые функции опущены (RIGHT OUTER JOIN или FOR EACH STATEMENT), однако, есть и некоторые новые.
- Отлично подходит для разработки и даже тестирования: во время этапа разработки большинству требуется масштабируемое решение.

Недостатки

- Отсутствие пользовательского управления: продвинутые БД предоставляют пользователям возможность управлять связями в таблицах в соответствии с привилегиями, но у SQLite такой функции нет.
- Невозможность дополнительной настройки.

Преимущества

- Полная SQL-совместимость.
- Сообщество: PostgreSQL поддерживается опытным сообществом 24/7.
- Поддержка сторонними организациями: несмотря на очень продвинутые функции, PostgreSQL используется в многих инструментах, связанных с РСУБД.
- Расширяемость: PostgreSQL можно программно расширить за счёт хранимых процедур.
- Объектно-ориентированность: PostgreSQL — не только реляционная, но и объектно-ориентированная СУБД.

Недостатки

- **Производительность:** В простых операциях чтения PostgreSQL может уступать своим соперникам.
- **Хостинг:** из-за вышеперечисленных факторов проблематично найти подходящего провайдера.

Установка PostgreSQL

Подключение PostgreSQL

- Для работы с PostgreSQL в Django будем использовать библиотеку psycopg2.

  Ссылка на PyPi: https://pypi.org/project/psycopg2/

- Последний наш шаг - настроить раздел DATABASES конфигурационного файла проекта settings.py.

```
'ENGINE': 'django.db.backends.postgresql_psycopg2',

'NAME': 'django_db',

'USER' : 'user_name',

'PASSWORD' : 'password',

'HOST' : '127.0.0.1',

'PORT' : '5432',
```

Поля моделей

Самая важная часть модели (и единственная необходимая часть модели) - это список полей базы данных, которые она определяет. Поля определяются атрибутами класса.

Каждое поле в модели должно быть экземпляром соответствующего класса Field. Django использует типы классов полей для определения нескольких вещей:

- Тип столбца, который сообщает базе данных, какой тип данных хранить (например, INTEGER, VARCHAR, TEXT).
- HTML-код по умолчанию widget, используемый при визуализации поля формы (например, <input type = "text">, <select>).
- Минимальные требования проверки, используемые в админке Django и в автоматически сгенерированных формах.

Основные типы полей

- IntegerField - целое число. Значения от -2147483648 до 2147483647 безопасны во всех базах данных, поддерживаемых Django.
- BooleanField - поле истина/ложь.
- CharField - строковое поле для строк малого и большого размера. Для больших объемов текста используйте TextField.
- TextField - большое текстовое поле.
- DateField - дата, представленная в Python экземпляром datetime.date. Имеет несколько дополнительных необязательных аргументов.
- DateTimeField - дата и время, представленные в Python экземпляром datetime.datetime. Принимает те же дополнительные аргументы, что и DateField.

Основные типы полей

- DurationField - поле для хранения периодов времени, смоделировано в Python с помощью timedelta. При использовании в PostgreSQL, используемый тип данных представляет собой interval, а в Oracle - тип данных представляет собой INTERVAL DAY (9) TO SECOND (6). В противном случае используется bigint микросекунд.

- EmailField - класс CharField, который проверяет, является ли значение действительным адресом электронной почты, используя EmailValidator.

- FileField - поле для загрузки файла.

- FloatField - число с плавающей точкой, представленное в Python экземпляром float.

- ImageField - наследует все атрибуты и методы из FileField, но также проверяет, что загруженный объект является допустимым изображением.

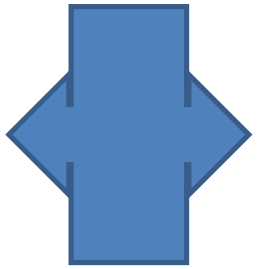
Основные типы полей

- DecimalField - десятичное число с фиксированной точностью, представленное в Python экземпляром Decimal. Он проверяет ввод с помощью DecimalValidator.
- URLField - CharField для URL, проверяется валидатором URLValidator.
- AutoField - хранит целочисленное значение, которое автоматически инкрементируется. Обычно применяется для первичных ключей.
- UUIDField - хранит строку, которая представляет UUID-идентификатор.
- GenericIPAddressField - хранит строку, которая представляет IP-адрес в формате IP4v или IP6v.

Поля отношений пример

User
Id – **number**
Name – **CharField**
Email – **EmailField**
LastBuy - **ForeignKey**

Product
Id – number
Label - **CharField**
Price – number
OnSale – m2m



Sale
Id – number
Label - **CharField**
DiscountValue - **number**
Products – m2m

Поля отношений

- ForeignKey - отношения многие-к-одному. Требуются два позиционных аргумента: класс, к которому относится модель, и опция on_delete.

- ManyToManyField - отношения многие ко многим. Требуется позиционный аргумент - класс, к которому относится модель, который работает точно так же, как и для ForeignKey, включая рекурсивные и ленивые отношения. Связанные объекты можно добавлять, удалять или создавать с помощью поля RelatedManager.

- OneToOneField - отношения один-к-одному. Концептуально это похоже на ForeignKey с unique=True, но «обратная» сторона отношения будет напрямую возвращать один объект.

Опции полей

- *null* - если True, Django будет хранить пустые значения как NULL в базе данных. По умолчанию установлено значение False.
- *blank* - если True, поле может быть пустым. По умолчанию установлено значение False.
- *choices* - первый элемент в каждом кортеже - это фактическое значение, которое должно быть установлено в модели, а второй элемент - удобочитаемое имя.
- *db_column* - имя столбца базы данных для использования в этом поле. Если это не указано, Django будет использовать имя поля.
- *db_tablespace* - имя табличного пространства базы данных, которое будет использоваться для индекса этого поля, если это поле проиндексировано.
- *default* - значение по умолчанию для поля. Это может быть значение или вызываемый объект. Если он вызывается, он будет вызываться каждый раз, когда создается новый объект.

Опции полей

- *editable* - если False, поле не будет отображаться ни админкой, ни какими-либо другими ModelForm. Они также пропускаются во время проверки модели. По умолчанию установлено значение True.

- *error_messages* - аргумент error_messages позволяет вам переопределить сообщения по умолчанию, которые вызовет поле. Передайте словарь с ключами, соответствующими сообщениям об ошибках, которые вы хотите переопределить.

- *help_text* - дополнительный «справочный» текст для отображения с виджетом формы. Это полезно для документации, даже если ваше поле не используется в форме.

- *primary_key* - если True, это поле является первичным ключом для модели.

Опции полей

- *unique* - если True, это поле должно быть уникальным во всей таблице.
- *unique_for_date* - задайте для него имя DateField или DateTimeField для требования, чтобы это поле было уникальным для значения поля даты.
- *unique_for_month* - как unique_for_date, но требует, чтобы поле было уникальным по отношению к месяцу.
- *unique_for_year* - как unique_for_date и unique_for_month.
- *verbose_name* - удобочитаемое имя для поля. Если подробное имя не указано, Django автоматически создаст его, используя имя атрибута поля, преобразовав подчеркивание в пробелы.
- *validators* - список валидаторов для этого поля.

План урока

1. Модели в Django
2. Подключение PostgreSQL
3. Поля моделей
4. Отношение моделей

Проверка знаний

TestProvider.com

TestProvider – это online сервис проверки знаний по информационным технологиям. С его помощью Вы можете оценить Ваш уровень и выявить слабые места. Он будет полезен как в процессе изучения технологии, так и для общей оценки знаний IT специалиста.

Проверьте как Вы усвоили данный материал на TestProvider.com

Успешное прохождение финального тестирования позволит Вам получить соответствующий Сертификат.

Спасибо за внимание! До новых встреч!

Лазорык Михаил
Software developer

tieto

django

Информационный видеосервис для разработчиков программного обеспечения

ITVDN

IT VIDEO DEVELOPERS NETWORK

TestProvider PROMETRIC TEST CENTER CyberBionic systematics Microsoft Partner Silver Learning Windows Azure Cloud Partner EBA European Business Association