Факультет компьютерный наук
Департаент программной инженерии

Высшая школа экономики

Москва 2023

индивидуальный программный курсовой проект

Модуль утилитарных абстракций моделей для фреймворка Django

Utility Model Abstractions Module for Django Framework

Руководитель:
внештатный преподаватель департамента программной инженерии
Веселко Никита Игоревич

Исполнитель:
студент группы БПИ 203
Черемицин Никита Сергеевич

Основные термины

- Фреймворк — программное обеспечение, облегчающее разработку и объединение разных компонентов большого программного проекта.
- Django — фреймворк для разработки веб-приложений на Python
- БД — база данных
- ООП — объектно-ориентированное программирование
- SDK (software development kit) — набор (пакет) средств (инструментов) разработки программного обеспечения
- ORM (object-relational mapping) — технология программирования, которая связывает базы данных с концепциями объектно-ориентированных языков программирования, создавая «виртуальную объектную базу данных»
- Лог — средство жруналирования событий
- Модель (здесь) — класс, содержащий набор полей, описывающих таблицу в БД, и наследующий django.db.models.base.Model (используется в составе Django-ORM)
- Кверисет (здесь) — класс, использующийся для получения записей моделей из таблиц БД и наследующий django.db.models.query.QuerySet (используется в составе Django-ORM)

Описание предметной области и проблематика

Django — один из наиболее популярных фреймворков для веб-разработки, а уровень моделей, реализованный как ORM и организующий работу с БД, — его ключевая составляющая.

При создании пользовательских моделей у разработчиков регулярно возникают следующие задачи:

- Сохранение логов в БД
- Безопасное обновление данных таблиц скриптами
- Организация работы с историческими строками таблиц

Однако Django "из коробки" не предоставляет варианты решений данных задач.

Актуальность работы

Модуль абстрактных моделей предоставит решение для упомянутых задач разработчиков.

Ключевые аспекты:

1. Реализация всей требуемой функциональности с использованием принципов ООП
2. Сохранение возможности гибкой настройки и переопределения поведения в пользовательских реализациях
3. Удовлетворение требований безопасности обновления данных

Цель и задачи проекта

Цель проекта — создания модуля абстрактных моделей (SDK) для решения частых задач разработки.

Задачи проекта:

- Создание абстрактной модели логов
- Создание абстрактной обновляемой модели
- Создание абстрактной исторической модели

Анализ существующих решений

Существует ряд решений, предлагающих разработчикам абстрактные модели:

- model-log
- django-model-history-log
- django-model-utils
- django-model-history

...и другие.

Однако ни одно из них не дает действительно удобных решений для работы с обновляемыми и историческими таблицами.

Функциональные требования

Требования к модели логов

- сохранение времени начала и окончания события, а также сообщений
- возможность настройки автоматического удаления старых логов
- для логов обновления: сохранение признаков (не)успешного завершения обновления

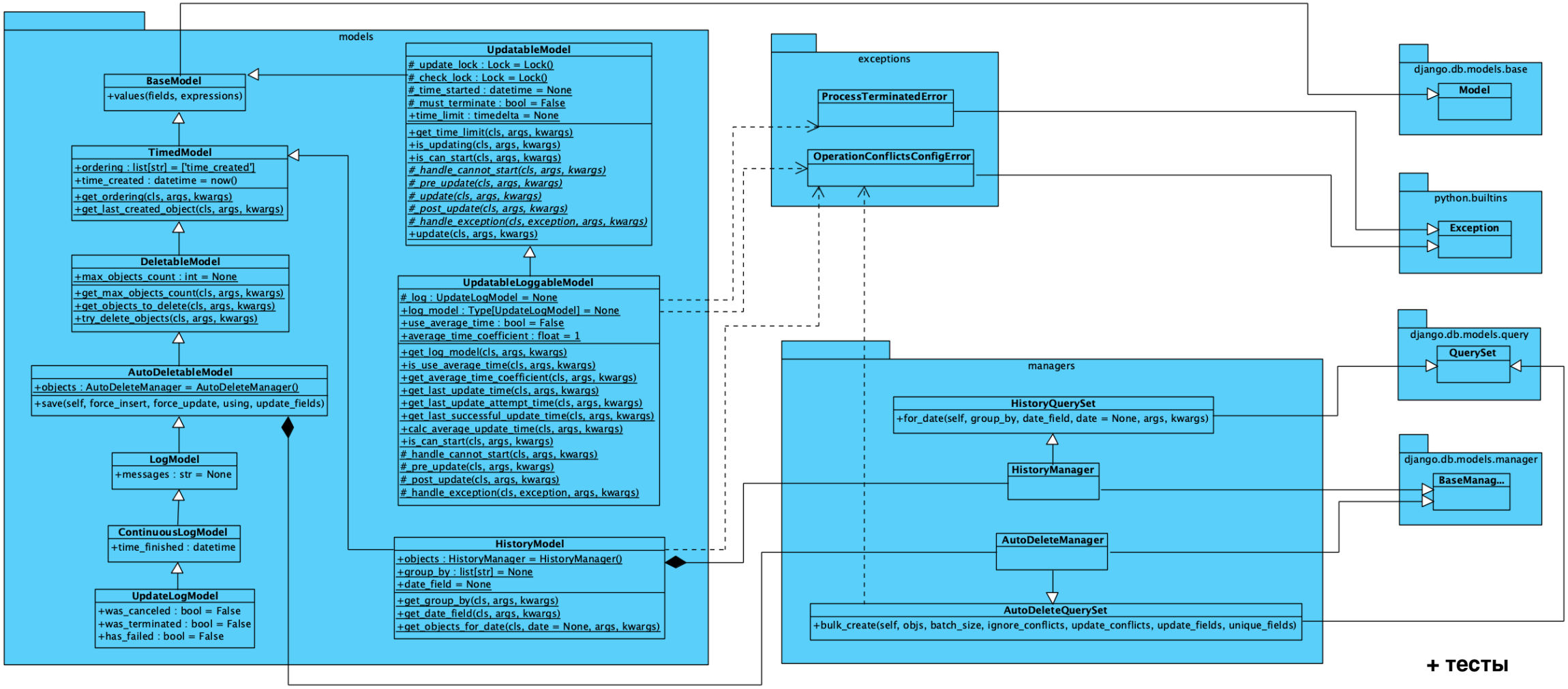
Требования к обновляемой модели

- гарантия отсутствия выбрасываемых исключений при вызове функции обновления
- соблюдение правила одного процесса обновления в любой момент времени
- логирование процесса обновления
- возможность настройки предельного времени обновления, после которого процесс может быть прерван новой попыткой
- возможность использования среднего времени обновления в качестве предельного времени обновления

Требования к исторической модели

- получение снимка данных на дату (путем выбора ближайшей к запрашиваемой дате и не превосходящей ее исторической записи по каждой из строк)

Диаграмма классов



+ тесты

Модель логов: автоматическое удаление

```
def test_auto_delete_after_create(self):
    mod = self.model
    mod.max_objects_count = 3
    o1 = mod.objects.create()
    self.assertQuerysetEqual(mod.objects.all(), [o1])
    o2 = mod.objects.create()
    self.assertQuerysetEqual(mod.objects.all(), [o1, o2])
    o3 = mod.objects.create()
    self.assertQuerysetEqual(mod.objects.all(), [o1, o2, o3])
    o4 = mod.objects.create()
    self.assertQuerysetEqual(mod.objects.all(), [o2, o3, o4])
    o5 = mod.objects.create()
    self.assertQuerysetEqual(mod.objects.all(), [o3, o4, o5])
```

Обновляемая модель: шаблонный метод и параметры

```python
@classmethod
def _handle_cannot_start(cls, *args, **kwargs):...

@classmethod
def _pre_update(cls, *args, **kwargs):...

@classmethod
def _update(cls, *args, **kwargs):...

@classmethod
def _post_update(cls, *args, **kwargs):...

@classmethod
def _handle_exception(cls, exception, *args, **kwargs):...

@classmethod
def update(cls, *args, **kwargs):...
```

```python
@classmethod
def _handle_cannot_start(cls, postpone_if_busy, send_email_when_finished):...

@classmethod
def _pre_update(cls, postpone_if_busy, send_email_when_finished):...

@classmethod
def _update(cls, postpone_if_busy, send_email_when_finished):...

@classmethod
def _post_update(cls, postpone_if_busy, send_email_when_finished):...

@classmethod
def _handle_exception(cls, exception, postpone_if_busy, send_email_when_finished):...

@classmethod
def update(cls, postpone_if_busy, send_email_when_finished):
    return super().update(postpone_if_busy, send_email_when_finished)
```

Обновляемая модель: шаблонный метод и безопасность

```
try:
    cls._pre_update(*args, **kwargs)
    cls._update(*args, **kwargs)
    cls._post_update(*args, **kwargs)

except Exception as exc:
    is_update_successful = False
    exception = exc
    try:
        cls._handle_exception(exc, *args, **kwargs)
    except Exception as exc2:
        exception_after_handling_attempt = exc2

return is_update_successful, exception, exception_after_handling_attempt
```

Обновляемая модель: шаблонный метод и логирование

```python
@classmethod
def _handle_cannot_start(cls, *args, **kwargs):
    log_model = cls.get_log_model(*args, **kwargs)
    log = log_model()
    log.time_finished = timezone.now()
    log.was_canceled = True
    log.save()

@classmethod
def _pre_update(cls, *args, **kwargs):
    log_model = cls.get_log_model(*args, **kwargs)
    cls._log = log_model.objects.create()

@classmethod
def _post_update(cls, *args, **kwargs):
    cls._log.time_finished = timezone.now()
    cls._log.save()
```

Обновляемая модель: единоличие процесса обновления

```
cls._check_lock.acquire()

if not cls.is_can_start(*args, **kwargs):
    cls._handle_cannot_start(*args, **kwargs)
    cls._check_lock.release()
    return False, None, None

cls._must_terminate = True

with cls._update_lock:
    cls._must_terminate = False
    cls._time_started = timezone.now()
    cls._check_lock.release()
```

Обновляемая модель: процесс обновления

```python
class UpdatableModelImpl(UpdatableModel):
    @classmethod
    def _update(cls):
        for i in range(5):
            if not cls._must_terminate:
                time.sleep(0.1)
            else:
                raise ProcessTerminatedError
```

```python
def update_and_check_result(self, is_ok_expected, exc_expected):
    result = self.model.update()
    self.assertEqual(result[0], is_ok_expected)
    self.assertEqual(type(result[1]), type(exc_expected))

def test_update_result_with_time_limit(self):
    self.model.time_limit = timedelta(seconds=0.3)
    t1 = Thread(target=self.update_and_check_result, args=[False, ProcessTerminatedError()])
    t1.start()
    time.sleep(0.1)
    t2 = Thread(target=self.update_and_check_result, args=[False, None])
    t2.start()
    time.sleep(0.3)
    t3 = Thread(target=self.update_and_check_result, args=[True, None])
    t3.start()
    time.sleep(0.2)
    t4 = Thread(target=self.update_and_check_result, args=[False, None])
    t4.start()
    for t in [t1, t2, t3, t4]:
        t.join()
```

Историческая модель: снимок данных на дату

```python
class HistoryModelImpl(HistoryModel):
    country = TextField()
    city = TextField()
    population = IntegerField()
    date_info = DateField()

    group_by = ['country', 'city']
    date_field = 'date_info'
```

```python
def assertPopulationValid(self, qs, values):
    self.assertQuerysetEqual(qs.values_list('population', flat=True), values)

def test_objects_for_date(self):
    mod = self.model
    rows = [
        ['A', 'x', 10, datetime(2010, 1, 1)],
        ['A', 'y', 20, datetime(2010, 1, 1)],
        ['B', 'x', 30, datetime(2010, 1, 1)],
        ['B', 'z', 40, datetime(2010, 1, 1)],
        ['A', 'x', 11, datetime(2010, 1, 3)],
        ['A', 'y', 22, datetime(2010, 1, 3)],
        ['B', 'x', 33, datetime(2010, 1, 4)],
        ['A', 'x', 12, datetime(2010, 1, 5)],
    ]
    for row in rows:
        mod.objects.create(country=row[0], city=row[1], population=row[2], date_info=row[3])
    self.assertPopulationValid(mod.get_objects_for_date(datetime(2009, 12, 31)), [])
    self.assertPopulationValid(mod.get_objects_for_date(datetime(2010, 1, 1)), [10, 20, 30, 40])
    self.assertPopulationValid(mod.get_objects_for_date(datetime(2010, 1, 2)), [10, 20, 30, 40])
    self.assertPopulationValid(mod.get_objects_for_date(datetime(2010, 1, 3)), [11, 22, 30, 40])
    self.assertPopulationValid(mod.get_objects_for_date(datetime(2010, 1, 4)), [11, 22, 33, 40])
    self.assertPopulationValid(mod.get_objects_for_date(datetime(2010, 1, 5)), [12, 22, 33, 40])
    self.assertPopulationValid(mod.get_objects_for_date(datetime(2010, 1, 6)), [12, 22, 33, 40])
```

Направления дальнейшего развития

1. Добавление аннотаций для параметров функций и возвращаемых значений
2. Документирование кода (классов, функций)
3. Создание вики-страницы модуля
4. Размещение модуля на портале PyPI

Список использованных источников

Python // [Электронный ресурс] — Режим доступа: свободный, URL: https://www.python.org/

Django // [Электронный ресурс] — Режим доступа: свободный, URL: https://www.djangoproject.com/

Wikipedia // [Электронный ресурс] — Режим доступа: свободный, URL: https://ru.wikipedia.org/wiki/SDK

Wikipedia // [Электронный ресурс] — Режим доступа: свободный, URL: https://ru.wikipedia.org/wiki/ORM

Wikipedia // [Электронный ресурс] — Режим доступа: свободный, URL: https://ru.wikipedia.org/wiki/База_данных

Wikipedia // [Электронный ресурс] — Режим доступа: свободный, URL: https://ru.wikipedia.org/wiki/Объектно-ориентированное_программирование

Wikipedia // [Электронный ресурс] — Режим доступа: свободный, URL: https://ru.wikipeda.org/wiki/Фреймворк

Wikipedia // [Электронный ресурс] — Режим доступа: свободный, URL: https://ru.wikipedia.org/wiki/Файл_журнала

Pypi // [Электронный ресурс] — Режим доступа: свободный, URL: https://pypi.org

Все ссылки актуальны по состоянию на 15.04.2023

Факультет компьютерный наук
Департаент программной инженерии

Высшая школа экономики

Москва 2023

Спасибо за внимание!

Черемицин Никита Сергеевич

nscheremitsin@edu.hse.ru