Save The Change

Release 1.0.0

December 27, 2015

Contents

Save The Change takes this:

```
>>> lancelot = Knight.objects.get(name="Sir Lancelot")
>>> lancelot.favorite_color = "Blue"
>>> lancelot.save()
```

And does this:

```
UPDATE "roundtable_knight"
SET "favorite_color" = 'Blue'
```

Instead of this:

```
UPDATE "roundtable_knight"
SET "name" = 'Sir Lancelot',
    "from" = 'Camelot',
    "quest" = 'To seek the Holy Grail.',
    "favorite_color" = 'Blue',
    "epithet" = 'The brave',
    "actor" = 'John Cleese',
    "full_name" = 'John Marwood Cleese',
    "height" = '6''11"',
    "birth_date" = '1939-10-27',
    "birth_union" = 'UK',
    "birth_country" = 'England',
    "birth_county" = 'Somerset',
    "birth_town" = 'Weston-Super-Mare',
    "facial_hair" = 'mustache',
    "graduated" = true,
    "university" = 'Cambridge University',
    "degree" = 'LL.B.',
```

CHAPTER 1

Installation

Install Save The Change just like everything else:

```
$ pip install django-save-the-change
```

CHAPTER 2

Usage

Just add `SaveTheChange` to your model:

```
from django.db import models
from save_the_change.mixins import SaveTheChange

class Knight(SaveTheChange, models.model):
        ...
```

And that's it! Keep using Django like you always have, Save The Change will take care of you.

CHAPTER 3

How It Works

Save The Change overloads `__setattr__` and keeps track of what fields have changed from their stored value in your database. When you call `save()`, Save The Change passes those changed fields through Django's `update_fields` `kwarg`, and Django does the rest, sending only those fields back to the database.

CHAPTER 4

Caveats

Save The Change can't help you with `ManyToManyField`s nor reverse relations, as those aren't handled through `save()`. But everything else'll work.

CHAPTER 5

Goodies

Save The Change also comes with a second mixin, `TrackChanges`. Adding `TrackChanges` to your model will expose a few new properties and methods for tracking and manually reverting changes to your model before you save it.

You can also use `UpdateTogetherModel` in place of `Model` to add the new field `update_together` to your model's `Meta`, which allows you to specify that certain fields are dependent on the values of other fields in your model.

CHAPTER 6

Developer Interface

CHAPTER 7

History

7.1 1.1.0 (05/16/2014)

- Add proper support for ForeignKeys (thanks to Brandon Konkle and Brian Wilson).
- Add update_together field to model Meta, via UpdateTogetherModel.

7.2 1.0.0 (09/08/2013)

- Initial release.