django-basis Documentation

Release 1.0.0

Fredrik Carlsen

February 12, 2015

Contents

1 Installation 3

2 Usage of TimeStampModel 5

3 Usage of PersistentModel 7

4 Usage of BasisModel 9

5 Models 11

6 Usage of BasisSerializer 13

Simple reusable django app for basic model functionality.

CHAPTER 1

Installation

Install it with pip:

```
pip install django-basis
```

CHAPTER 2

Usage of TimeStampModel

Adds `created_at` and `updated_at` to models.

```
from basis.models import TimeStampModel

class Person(TimeStampModel):
    name = models.Charfield(max_length=50)

person = Person.objects.create(name="Fredrik"))
print person.created_at # datetime object1

person.name = "Rolf"
person.save()

print person.created_at # (datetime at the moment of the creation)
print person.updated_at # (datetime at the moment of the update)
```

CHAPTER 3

Usage of PersistentModel

Safe deletion of objects.

```
from basis.models import PersistentModel

class Person(PersistentModel):
    name = models.Charfield(max_length=50)

person = Person.objects.create(name="Fredrik"))

# SafeDelete person (safe delete)
person.delete()

print Person.objects.all().count() # 0 - excludes deleted users
print Person.all_objects.all().count() # 1 - includes deleted users

# Restore deleted person
person = Person.all_objects.get(id=person.id)
person.restore()

# If you really want to delete the object
person = Person.objects.create(name="Fredrik"))
person.delete(force=True)
```

CHAPTER 4

Usage of BasisModel

Includes the functionality of both PersistentModel and TimeStampModel, while adding the fields `created_by` and `updated_by`.

```
from basis.models import BasisModel

class Person(BasisModel):
    name = models.Charfield(max_length=50)

# Save changes on objects and register who did it
person = Person.objects.get(id=id)
person.name = "Fredrik"
person.save(current_user=request.user)

# Or create a new object and register who did it
person = Person.objects.create(name="Fredrik", current_user=request.user)

# See meta info about the object
print person.created_by # user object (creator)
print person.updated_by # user object (updater)
```

CHAPTER 5

Models

CHAPTER 6

Usage of BasisSerializer

Makes sure BasisModel objects created and updated have the `created_by` and `updated_by` fields set.

```
from basis.serializers imoprt BasisSerializer

class PersonSerializer(BasisSerializer):
    class Meta:
        model = Person
```