Django-money Documentation

Release 1.3

Jacob Hansson

Jan 10, 2021

Contents

CHAPTER 1

django-money

A little Django app that uses `py-moneyed` to add support for Money fields in your models and forms.

- Django versions supported: 1.11, 2.1, 2.2, 3.0, 3.1
- Python versions supported: 3.5, 3.6, 3.7, 3.8, 3.9
- PyPy versions supported: PyPy3

If you need support for older versions of Django and Python, please refer to older releases mentioned in the release notes.

Through the dependency `py-moneyed`, `django-money` gets:

- Support for proper Money value handling (using the standard Money design pattern)
- A currency class and definitions for all currencies in circulation
- Formatting of most currencies with correct currency sign

1.1 Installation

Using *pip*:

```
$ pip install django-money
```

This automatically installs `py-moneyed` v0.8 (or later).

Add `djmoney` to your `INSTALLED_APPS`. This is required so that money field are displayed correctly in the admin.

```
INSTALLED_APPS = [
    ...,
    'djmoney',
    ...
]
```

1.2 Model usage

Use as normal model fields:

```
from djmoney.models.fields import MoneyField
from django.db import models


class BankAccount(models.Model):
    balance = MoneyField(max_digits=14, decimal_places=2, default_currency='USD')
```

To comply with certain strict accounting or financial regulations, you may consider using `max_digits=19` and `decimal_places=4`, see more in this StackOverflow answer

It is also possible to have a nullable `MoneyField`:

```
class BankAccount(models.Model):
    money = MoneyField(max_digits=10, decimal_places=2, null=True, default_
↪currency=None)

account = BankAccount.objects.create()
assert account.money is None
assert account.money_currency is None
```

Searching for models with money fields:

```
from djmoney.money import Money


account = BankAccount.objects.create(balance=Money(10, 'USD'))
swissAccount = BankAccount.objects.create(balance=Money(10, 'CHF'))

BankAccount.objects.filter(balance__gt=Money(1, 'USD'))
# Returns the "account" object
```

1.3 Field validation

There are 3 different possibilities for field validation:

- by numeric part of money despite on currency;
- by single money amount;
- by multiple money amounts.

All of them could be used in a combination as is shown below:

```python
from django.db import models
from djmoney.models.fields import MoneyField
from djmoney.money import Money
from djmoney.models.validators import MaxMoneyValidator, MinMoneyValidator


class BankAccount(models.Model):
    balance = MoneyField(
        max_digits=10,
        decimal_places=2,
        validators=[
            MinMoneyValidator(10),
            MaxMoneyValidator(1500),
            MinMoneyValidator(Money(500, 'NOK')),
            MaxMoneyValidator(Money(900, 'NOK')),
            MinMoneyValidator({'EUR': 100, 'USD': 50}),
            MaxMoneyValidator({'EUR': 1000, 'USD': 500}),
        ]
    )
```

The `balance` field from the model above has the following validation:

- All input values should be between 10 and 1500 despite on currency;
- Norwegian Crowns amount (NOK) should be between 500 and 900;
- Euros should be between 100 and 1000;
- US Dollars should be between 50 and 500;

1.4 Adding a new Currency

Currencies are listed on moneyed, and this modules use this to provide a choice list on the admin, also for validation.

To add a new currency available on all the project, you can simple add this two lines on your `settings.py` file

```python
import moneyed
from moneyed.localization import _FORMATTER
from decimal import ROUND_HALF_EVEN


BOB = moneyed.add_currency(
    code='BOB',
    numeric='068',
    name='Peso boliviano',
    countries=('BOLIVIA', )
)

# Currency Formatter will output 2.000,00 Bs.
_FORMATTER.add_sign_definition(
    'default',
    BOB,
    prefix=u'Bs. '
)

_FORMATTER.add_formatting_definition(
    'es_BO',
```

(continues on next page)

(continued from previous page)

```
    group_size=3, group_separator=".", decimal_point=",",
    positive_sign="",  trailing_positive_sign="",
    negative_sign="-", trailing_negative_sign="",
    rounding_method=ROUND_HALF_EVEN
)
```

To restrict the currencies listed on the project set a `CURRENCIES` variable with a list of Currency codes on `settings.py`

```
CURRENCIES = ('USD', 'BOB')
```

The list has to contain valid Currency codes

Additionally there is an ability to specify currency choices directly:

```
CURRENCIES = ('USD', 'EUR')
CURRENCY_CHOICES = [('USD', 'USD $'), ('EUR', 'EUR €')]
```

1.5 Important note on model managers

Django-money leaves you to use any custom model managers you like for your models, but it needs to wrap some of the methods to allow searching for models with money values.

This is done automatically for the "objects" attribute in any model that uses MoneyField. However, if you assign managers to some other attribute, you have to wrap your manager manually, like so:

```
from djmoney.models.managers import money_manager


class BankAccount(models.Model):
    balance = MoneyField(max_digits=10, decimal_places=2, default_currency='USD')
    accounts = money_manager(MyCustomManager())
```

Also, the money_manager wrapper only wraps the standard QuerySet methods. If you define custom QuerySet methods, that do not end up using any of the standard ones (like "get", "filter" and so on), then you also need to manually decorate those custom methods, like so:

```
from djmoney.models.managers import understands_money


class MyCustomQuerySet(QuerySet):

   @understands_money
   def my_custom_method(*args, **kwargs):
       # Awesome stuff
```

1.6 Format localization

The formatting is turned on if you have set `USE_L10N = True` in the your settings file.

If formatting is disabled in the configuration, then in the templates will be used default formatting.

In the templates you can use a special tag to format the money.

In the file `settings.py` add to `INSTALLED_APPS` entry from the library `djmoney`:

```
INSTALLED_APPS += ('djmoney', )
```

In the template, add:

```
{% load djmoney %}
...
{% money_localize money %}
```

and that is all.

Instructions to the tag `money_localize`:

```
{% money_localize <money_object> [ on(default) | off ] [as var_name] %}
{% money_localize <amount> <currency> [ on(default) | off ] [as var_name] %}
```

Examples:

The same effect:

```
{% money_localize money_object %}
{% money_localize money_object on %}
```

Assignment to a variable:

```
{% money_localize money_object on as NEW_MONEY_OBJECT %}
```

Formatting the number with currency:

```
{% money_localize '4.5' 'USD' %}
```

```
Return::

    Money object
```

1.7 Testing

Install the required packages:

```
git clone https://github.com/django-money/django-money

cd ./django-money/

pip install -e ".[test]" # installation with required packages for testing
```

Recommended way to run the tests:

```
tox
```

Testing the application in the current environment python:

```
make test
```

1.8 Working with Exchange Rates

To work with exchange rates, add the following to your `INSTALLED_APPS`.

```
INSTALLED_APPS = [
    ...,
    'djmoney.contrib.exchange',
]
```

Also, it is required to have `certifi` installed. It could be done via installing `djmoney` with `exchange` extra:

```
pip install "django-money[exchange]"
```

To create required relations run `python manage.py migrate`. To fill these relations with data you need to choose a data source. Currently, 2 data sources are supported - https://openexchangerates.org/ (default) and https://fixer.io/. To choose another data source set `EXCHANGE_BACKEND` settings with importable string to the backend you need:

```
EXCHANGE_BACKEND = 'djmoney.contrib.exchange.backends.FixerBackend'
```

If you want to implement your own backend, you need to extend `djmoney.contrib.exchange.backends.base.BaseExchangeBackend`. Two data sources mentioned above are not open, so you have to specify access keys in order to use them:

`OPEN_EXCHANGE_RATES_APP_ID` - '<your actual key from openexchangerates.org>'

`FIXER_ACCESS_KEY` - '<your actual key from fixer.io>'

Backends return rates for a base currency, by default it is USD, but could be changed via `BASE_CURRENCY` setting. Open Exchanger Rates & Fixer supports some extra stuff, like historical data or restricting currencies in responses to the certain list. In order to use these features you could change default URLs for these backends:

```
OPEN_EXCHANGE_RATES_URL = 'https://openexchangerates.org/api/historical/2017-01-01.
↪json?symbols=EUR,NOK,SEK,CZK'
FIXER_URL = 'http://data.fixer.io/api/2013-12-24?symbols=EUR,NOK,SEK,CZK'
```

Or, you could pass it directly to `update_rates` method:

```
>>> from djmoney.contrib.exchange.backends import OpenExchangeRatesBackend
>>> backend = OpenExchangeRatesBackend(url='https://openexchangerates.org/api/
↪historical/2017-01-01.json')
>>> backend.update_rates(symbols='EUR,NOK,SEK,CZK')
```

There is a possibility to use multiple backends in the same time:

```
>>> from djmoney.contrib.exchange.backends import FixerBackend,␣
↪OpenExchangeRatesBackend
>>> from djmoney.contrib.exchange.models import get_rate
>>> OpenExchangeRatesBackend().update_rates()
>>> FixerBackend().update_rates()
>>> get_rate('USD', 'EUR', backend=OpenExchangeRatesBackend.name)
>>> get_rate('USD', 'EUR', backend=FixerBackend.name)
```

Regular operations with `Money` will use `EXCHANGE_BACKEND` backend to get the rates. Also, there are two management commands for updating rates and removing them:

```
$ python manage.py update_rates
Successfully updated rates from openexchangerates.org
$ python manage.py clear_rates
Successfully cleared rates for openexchangerates.org
```

Both of them accept `-b/--backend` option, that will update/clear data only for this backend. And `clear_rates` accepts `-a/--all` option, that will clear data for all backends.

To set up a periodic rates update you could use Celery task:

```
CELERYBEAT_SCHEDULE = {
    'update_rates': {
        'task': 'path.to.your.task',
        'schedule': crontab(minute=0, hour=0),
        'kwargs': {}  # For custom arguments
    }
}
```

Example task implementation:

```
from django.utils.module_loading import import_string

from celery import Celery
from djmoney import settings


app = Celery('tasks', broker='pyamqp://guest@localhost//')


@app.task
def update_rates(backend=settings.EXCHANGE_BACKEND, **kwargs):
    backend = import_string(backend)()
    backend.update_rates(**kwargs)
```

To convert one currency to another:

```
>>> from djmoney.money import Money
>>> from djmoney.contrib.exchange.models import convert_money
>>> convert_money(Money(100, 'EUR'), 'USD')
<Money: 122.8184375038380800 USD>
```

Exchange rates are integrated with Django Admin.

django-money can be configured to automatically use this app for currency conversions by settings `AUTO_CONVERT_MONEY = True` in your Django settings. Note that currency conversion is a lossy process, so automatic conversion is usually a good strategy only for very simple use cases. For most use cases you will need to be clear about exactly when currency conversion occurs, and automatic conversion can hide bugs. Also, with automatic conversion you lose some properties like commutativity (`A + B == B + A`) due to conversions happening in different directions.

1.9 Usage with Django REST Framework

Make sure that `djmoney` and is in the `INSTALLED_APPS` of your `settings.py` and that `rest_framework` has been installed. MoneyField will automatically register a serializer for Django REST Framework through `djmoney.apps.MoneyConfig.ready()`.

You can add a serializable field the following way:

```
from djmoney.contrib.django_rest_framework import MoneyField

class Serializers(serializers.Serializer):
    my_computed_prop = MoneyField(max_digits=10, decimal_places=2)
```

Built-in serializer works in the following way:

```
class Expenses(models.Model):
    amount = MoneyField(max_digits=10, decimal_places=2)


class Serializer(serializers.ModelSerializer):
    class Meta:
        model = Expenses
        fields = '__all__'

>>> instance = Expenses.objects.create(amount=Money(10, 'EUR'))
>>> serializer = Serializer(instance=instance)
>>> serializer.data
ReturnDict([
    ('id', 1),
    ('amount_currency', 'EUR'),
    ('amount', '10.000'),
])
```

Note that when specifying individual fields on your serializer, the amount and currency fields are treated separately. To achieve the same behaviour as above you would include both field names:

```
class Serializer(serializers.ModelSerializer):
    class Meta:
        model = Expenses
        fields = ('id', 'amount', 'amount_currency')
```

1.10 Customization

If there is a need to customize the process deconstructing `Money` instances onto Django Fields and the other way around, then it is possible to use a custom descriptor like this one:

```
class MyMoneyDescriptor:

    def __get__(self, obj, type=None):
        amount = obj.__dict__[self.field.name]
        return Money(amount, "EUR")
```

It will always use `EUR` for all `Money` instances when `obj.money` is called. Then it should be passed to `MoneyField`:

```
class Expenses(models.Model):
    amount = MoneyField(max_digits=10, decimal_places=2, money_descriptor_
↪class=MyMoneyDescriptor)
```

1.11 Background

This project is a fork of the Django support that was in http://code.google.com/p/python-money/

This version adds tests, and comes with several critical bugfixes.

CHAPTER 2

Contents

2.1 Changelog

2.1.1 Unreleased - TBD

2.1.2 1.3 - 2021-01-10

Added

- Improved localization: New setting `CURRENCY_DECIMAL_PLACES_DISPLAY` configures decimal places to display for each configured currency. #521 (wearebasti)

Changed

- Set the default value for `models.fields.MoneyField` to `NOT_PROVIDED`. (tned73)

Fixed

- Pin `pymoneyed<1.0` as it changed the `repr` output of the `Money` class. (Stranger6667)
- Subtracting `Money` from `moneyed.Money`. Regression, introduced in `1.2`. #593 (Stranger6667)
- Missing the right `Money.decimal_places` and `Money.decimal_places_display` values after some arithmetic operations. #595 (Stranger6667)

2.1.3 1.2.2 - 2020-12-29

Fixed

- Confusing "number-over-money" division behavior by backporting changes from `py-moneyed`. #586 (wearebasti)
- `AttributeError` when a `Money` instance is divided by `Money`. #585 (niklasb)

2.1.4 1.2.1 - 2020-11-29

Fixed

- Aggregation through a proxy model. #583 (tned73)

2.1.5 1.2 - 2020-11-26

Fixed

- Resulting Money object from arithmetics (add / sub / ...) inherits maximum decimal_places from arguments #522 (wearebasti)
- `DeprecationWarning` related to the usage of `cafile` in `urlopen`. #553 (Stranger6667)

Added

- Django 3.1 support

2.1.6 1.1 - 2020-04-06

Fixed

- Optimize money operations on MoneyField instances with the same currencies. #541 (horpto)

Added

- Support for `Money` type in `QuerySet.bulk_update()` #534 (satels)

2.1.7 1.0 - 2019-11-08

Added

- Support for money descriptor customization. (Stranger6667)
- Fix `order_by()` not returning money-compatible queryset #519 (lieryan)
- Django 3.0 support

Removed

- Support for Django 1.8 & 2.0. (Stranger6667)
- Support for Python 2.7. #515 (benjaoming)
- Support for Python 3.4. (Stranger6667)
- `MoneyPatched`, use `djmoney.money.Money` instead. (Stranger6667)

Fixed

- Support instances with `decimal_places=0` #509 (fara)

2.1.8 0.15.1 - 2019-06-22

Fixed

- Respect field `decimal_places` when instantiating `Money` object from field db values. #501 (astutejoe)
- Restored linting in CI tests (benjaoming)

2.1.9 0.15 - 2019-05-30

Warning: This release contains backwards incompatibility, please read the release notes below.

Backwards incompatible changes

- Remove implicit default value on non-nullable MoneyFields. Backwards incompatible change: set explicit `default=0.0` to keep previous behavior. #411 (washeck)
- Remove support for calling `float` on `Money` instances. Use the `amount` attribute instead. (Stranger6667)
- `MinMoneyValidator` and `MaxMoneyValidator` are not inherited from Django's `MinValueValidator` and `MaxValueValidator` anymore. #376
- In model and non-model forms `forms.MoneyField` uses `CURRENCY_DECIMAL_PLACES` as the default value for `decimal_places`. #434 (Stranger6667, andytwoods)

Added

- Add `Money.decimal_places` for per-instance configuration of decimal places in the string representation.
- Support for customization of `CurrencyField` length. Some cryptocurrencies could have codes longer than three characters. #480 (Stranger6667, MrFus10n)
- Add `default_currency` option for REST Framework field. #475 (butorov)

Fixed

- Failing certificates checks when accessing 3rd party exchange rates backends. Fixed by adding *certifi* to the dependencies list. #403 (Stranger6667)
- Fixed model-level `validators` behavior in REST Framework. #376 (rapIsKal, Stranger6667)
- Setting keyword argument `default_currency=None` for `MoneyField` did not revert to `settings.DEFAULT_CURRENCY` and set `str(None)` as database value for currency. #490 (benjaoming)

Changed

- Allow using patched `django.core.serializers.python._get_model` in serializers, which could be helpful for migrations. (Formulka, Stranger6667)

2.1.10 0.14.4 - 2019-01-07

Changed

- Re-raise arbitrary exceptions in JSON deserializer as *DeserializationError*. (Stranger6667)

Fixed

- Invalid Django 1.8 version check in `djmoney.models.fields.MoneyField.value_to_string`. (Stranger6667)
- InvalidOperation in `djmoney.contrib.django_rest_framework.fields.MoneyField.get_value` when amount is None and currency is not None. #458 (carvincarl)

2.1.11 0.14.3 - 2018-08-14

Fixed

- `djmoney.forms.widgets.MoneyWidget` decompression on Django 2.1+. #443 (Stranger6667)

2.1.12 0.14.2 - 2018-07-23

Fixed

- Validation of `djmoney.forms.fields.MoneyField` when `disabled=True` is passed to it. #439 (stinovlas, Stranger6667)

2.1.13 0.14.1 - 2018-07-17

Added

- Support for indirect rates conversion through maximum 1 extra step (when there is no direct conversion rate: converting by means of a third currency for which both source and target currency have conversion rates). #425 (Stranger6667, 77cc33)

Fixed

- Error was raised when trying to do a query with a *ModelWithNullableCurrency*. #427 (Woile)

2.1.14 0.14 - 2018-06-09

Added

- Caching of exchange rates. #398 (Stranger6667)
- Added support for nullable `CurrencyField`. #260 (Stranger6667)

Fixed

- Same currency conversion getting MissingRate exception #418 (humrochagf)
- *TypeError* during templatetag usage inside a for loop on Django 2.0. #402 (f213)

Removed

- Support for Python 3.3 #410 (benjaoming)
- Deprecated `choices` argument from `djmoney.forms.fields.MoneyField`. Use `currency_choices` instead. (Stranger6667)

2.1.15 0.13.5 - 2018-05-19

Fixed

- Missing in dist, `djmoney/__init__.py`. #417 (benjaoming)

2.1.16 0.13.4 - 2018-05-19

Fixed

- Packaging of `djmoney.contrib.exchange.management.commands`. #412 (77cc33, Stranger6667)

2.1.17 0.13.3 - 2018-05-12

Added

- Rounding support via `round` built-in function on Python 3. (Stranger6667)

2.1.18 0.13.2 - 2018-04-16

Added

- Django Admin integration for exchange rates. #392 (Stranger6667)

Fixed

- Exchange rates. TypeError when decoding JSON on Python 3.3-3.5. #399 (kcyeu)
- Managers patching for models with custom `Meta.default_manager_name`. #400 (Stranger6667)

2.1.19 0.13.1 - 2018-04-07

Fixed

- Regression: Could not run w/o `django.contrib.exchange` #388 (Stranger6667)

2.1.20 0.13 - 2018-04-07

Added

- Currency exchange #385 (Stranger6667)

Removed

- Support for `django-money-rates` #385 (Stranger6667)
- Deprecated `Money.__float__` which is implicitly called on some `sum()` operations #347. (jonashaag)

Migration from django-money-rates

The new application is a drop-in replacement for `django-money-rates`. To migrate from `django-money-rates`:

- In `INSTALLED_APPS` replace `djmoney_rates` with `djmoney.contrib.exchange`
- Set `OPEN_EXCHANGE_RATES_APP_ID` setting with your app id
- Run `python manage.py migrate`
- Run `python manage.py update_rates`

For more information, look at `Working with Exchange Rates` section in README.

2.1.21 0.12.3 - 2017-12-13

Fixed

- Fixed `BaseMoneyValidator` with falsy limit values. #371 (1337)

2.1.22 0.12.2 - 2017-12-12

Fixed

- Django master branch compatibility. #361 (Stranger6667)
- Fixed `get_or_create` for models with shared currency. #364 (Stranger6667)

Changed

- Removed confusing rounding to integral value in `Money.__repr__`. #366 (Stranger6667, evenicoulddoit)

2.1.23 0.12.1 - 2017-11-20

Fixed

- Fixed migrations on SQLite. #139, #338 (Stranger6667)
- Fixed `Field.rel.to` usage for Django 2.0. #349 (richardowen)
- Fixed Django REST Framework behaviour for serializers without `*_currency` field in serializer's `Meta.fields`. #351 (elcolie, Stranger6667)

2.1.24 0.12 - 2017-10-22

Added

- Ability to specify name for currency field. #195 (Stranger6667)
- Validators for `MoneyField`. #308 (Stranger6667)

Changed

- Improved `Money` support. Now `django-money` fully relies on `pymoneyed` localization everywhere, including Django admin. #276 (Stranger6667)
- Implement `__html__` method. If used in Django templates, an `Money` object's amount and currency are now separated with non-breaking space (`&nbsp;`) #337 (jonashaag)

Deprecated

- `djmoney.models.fields.MoneyPatched` and `moneyed.Money` are deprecated. Use `djmoney.money.Money` instead.

Fixed

- Fixed model field validation. #308 (Stranger6667).
- Fixed managers caching for Django >= 1.10. #318 (Stranger6667).
- Fixed `F` expressions support for `in` lookups. #321 (Stranger6667).
- Fixed money comprehension on querysets. #331 (Stranger6667, jaavii1988).
- Fixed errors in Django Admin integration. #334 (Stranger6667, adi-).

Removed

- Dropped support for Python 2.6 and 3.2. (Stranger6667)
- Dropped support for Django 1.4, 1.5, 1.6, 1.7 and 1.9. (Stranger6667)

2.1.25 0.11.4 - 2017-06-26

Fixed

- Fixed money parameters processing in update queries. #309 (Stranger6667)

2.1.26 0.11.3 - 2017-06-19

Fixed

- Restored support for Django 1.4, 1.5, 1.6, and 1.7 & Python 2.6 #304 (Stranger6667)

2.1.27 0.11.2 - 2017-05-31

Fixed

- Fixed field lookup regression. #300 (lmdsp, Stranger6667)

2.1.28 0.11.1 - 2017-05-26

Fixed

- Fixed access to models properties. #297 (mithrilstar, Stranger6667)

Removed

- Dropped support for Python 2.6. (Stranger6667)
- Dropped support for Django < 1.8. (Stranger6667)

2.1.29 0.11 - 2017-05-19

Added

- An ability to set custom currency choices via `CURRENCY_CHOICES` settings option. #211 (Stranger6667, ChessSpider)

Fixed

- Fixed `AttributeError` in `get_or_create` when the model have no default. #268 (Stranger6667, lobziik)
- Fixed `UnicodeEncodeError` in string representation of `MoneyPatched` on Python 2. #272 (Stranger6667)
- Fixed various displaying errors in Django Admin . #232, #220, #196, #102, #90 (Stranger6667, arthurk, mstarostik, eriktelepovsky, jplehmann, graik, benjaoming, k8n, yellow-sky)
- Fixed non-Money values support for `in` lookup. #278 (Stranger6667)
- Fixed available lookups with removing of needless lookup check. #277 (Stranger6667)
- Fixed compatibility with `py-moneyed`. (Stranger6667)
- Fixed ignored currency value in Django REST Framework integration. #292 (gonzalobf)

2.1.30 0.10.2 - 2017-02-18

Added

- Added ability to configure decimal places output. #154, #251 (ivanchenkodmitry)

Fixed

- Fixed handling of `defaults` keyword argument in `get_or_create` method. #257 (kjagiello)
- Fixed handling of currency fields lookups in `get_or_create` method. #258 (Stranger6667)
- Fixed `PendingDeprecationWarning` during form initialization. #262 (Stranger6667, spookylukey)
- Fixed handling of `F` expressions which involve non-Money fields. #265 (Stranger6667)

2.1.31 0.10.1 - 2016-12-26

Fixed

- Fixed default value for `djmoney.forms.fields.MoneyField`. #249 (tsouvarev)

2.1.32 0.10 - 2016-12-19

Changed

- Do not fail comparisons because of different currency. Just return `False` #225 (benjaoming and ivirabyan)

Fixed

- Fixed `understands_money` behaviour. Now it can be used as a decorator #215 (Stranger6667)
- Fixed: Not possible to revert MoneyField currency back to default #221 (benjaoming)
- Fixed invalid `creation_counter` handling. #235 (msgre and Stranger6667)
- Fixed broken field resolving. #241 (Stranger6667)

2.1.33 0.9.1 - 2016-08-01

Fixed

- Fixed packaging.

2.1.34 0.9.0 - 2016-07-31

NB! If you are using custom model managers **not** named `objects` and you expect them to still work, please read below.

Added

- Support for `Value` and `Func` expressions in queries. (Stranger6667)
- Support for `in` lookup. (Stranger6667)
- Django REST Framework support. #179 (Stranger6667)
- Django 1.10 support. #198 (Stranger6667)
- Improved South support. (Stranger6667)

Changed

- Changed auto conversion of currencies using djmoney_rates (added in 0.7.3) to be off by default. You must now add `AUTO_CONVERT_MONEY = True` in your `settings.py` if you want this feature. #199 (spookylukey)
- Only make `objects` a MoneyManager instance automatically. #194 and #201 (inureyes)

Fixed

- Fixed default currency value for nullable fields in forms. #138 (Stranger6667)
- Fixed `_has_changed` deprecation warnings. #206 (Stranger6667)
- Fixed `get_or_create` crash, when `defaults` is passed. #213 (Stranger6667, spookylukey)

Note about automatic model manager patches

In 0.8, Django-money automatically patches every model managers with `MoneyManager`. This causes migration problems if two or more managers are used in the same model.

As a side effect, other managers are also finally wrapped with `MoneyManager`. This effect leads Django migration to point to fields with other managers to `MoneyManager`, and raises `ValueError` (`MoneyManager` only exists as a return of `money_manager`, not a class-form. However migration procedure tries to find `MoneyManager` to patch other managers.)

From 0.9, Django-money only patches `objects` with `MoneyManager` by default (as documented). To patch other managers (e.g. custom managers), patch them by wrapping with `money_manager`.

```
from djmoney.models.managers import money_manager


class BankAccount(models.Model):
    balance = MoneyField(max_digits=10, decimal_places=2, default_currency='USD')
    accounts = money_manager(MyCustomManager())
```

2.1.35 0.8 - 2016-04-23

Added

- Support for serialization of `MoneyPatched` instances in migrations. (AlexRiina)
- Improved django-money-rates support. #173 (Stranger6667)
- Extended `F` expressions support. (Stranger6667)
- Pre-commit hooks support. (benjaoming)
- Isort integration. (Stranger6667)
- Makefile for common commands. (Stranger6667)
- Codecov.io integration. (Stranger6667)
- Python 3.5 builds to tox.ini and travis.yml. (Stranger6667)
- Django master support. (Stranger6667)
- Python 3.2 compatibility. (Stranger6667)

Changed

- Refactored test suite (Stranger6667)

Fixed

- Fixed fields caching. #186 (Stranger6667)
- Fixed m2m fields data loss on Django < 1.8. #184 (Stranger6667)
- Fixed managers access via instances. #86 (Stranger6667)
- Fixed currency handling behaviour. #172 (Stranger6667)
- Many PEP8 & flake8 fixes. (benjaoming)
- Fixed filtration with `F` expressions. #174 (Stranger6667)
- Fixed querying on Django 1.8+. #166 (Stranger6667)

2.1.36 0.7.6 - 2016-01-08

Added

- Added correct paths for py.test discovery. (benjaoming)
- Mention Django 1.9 in tox.ini. (benjaoming)

Fixed

- Fix for `get_or_create` / `create` manager methods not respecting currency code. (toudi)
- Fix unit tests. (toudi)
- Fix for using `MoneyField` with `F` expressions when using Django >= 1.8. (toudi)

2.1.37 0.7.5 - 2015-12-22

Fixed

- Fallback to `_meta.fields` if `_meta.get_fields` raises `AttributeError` #149 (browniebroke)
- pip instructions updated. (GheloAce)

2.1.38 0.7.4 - 2015-11-02

Added

- Support for Django 1.9 (kjagiello)

Fixed

- Fixed loaddata. (jack-cvr)
- Python 2.6 fixes. (jack-cvr)
- Fixed currency choices ordering. (synotna)

2.1.39 0.7.3 - 2015-10-16

Added

- Sum different currencies. (dnmellen)
- `__eq__` method. (benjaoming)

- Comparison of different currencies. (benjaoming)
- Default currency. (benjaoming)

Fixed

- Fix using Choices for setting currency choices. (benjaoming)
- Fix tests for Python 2.6. (plumdog)

2.1.40 0.7.2 - 2015-09-01

Fixed

- Better checks on `None` values. (tsouvarev, sjdines)
- Consistency with South declarations and calling `str` function. (sjdines)

2.1.41 0.7.1 - 2015-08-11

Fixed

- Fix bug in printing `MoneyField`. (YAmikep)
- Added fallback value for current locale getter. (sjdines)

2.1.42 0.7.0 - 2015-06-14

Added

- Django 1.8 compatibility. (willhcr)

2.1.43 0.6.0 - 2015-05-23

Added

- Python 3 trove classifier. (dekkers)

Changed

- Tox cleanup. (edwinlunando)
- Improved `README`. (glarrain)
- Added/Cleaned up tests. (spookylukey, AlexRiina)

Fixed

- Append `_currency` to non-money ExpressionFields. #101 (alexhayes, AlexRiina, briankung)
- Data truncated for column. #103 (alexhayes)
- Fixed `has_changed` not working. #95 (spookylukey)
- Fixed proxy model with `MoneyField` returns wrong class. #80 (spookylukey)

2.1.44 0.5.0 - 2014-12-15

Added

- Django 1.7 compatibility. (w00kie)

Fixed

- Added `choices=` to instantiation of currency widget. (davidstockwell)
- Nullable `MoneyField` should act as `default=None`. (jakewins)
- Fixed bug where a non-required `MoneyField` threw an exception. (spookylukey)

2.1.45 0.4.2 - 2014-07-31

2.1.46 0.4.1 - 2013-11-28

2.1.47 0.4.0.0 - 2013-11-26

Added

- Python 3 compatibility.
- tox tests.
- Format localization.
- Template tag `money_localize`.

2.1.48 0.3.4 - 2013-11-25

2.1.49 0.3.3.2 - 2013-10-31

2.1.50 0.3.3.1 - 2013-10-01

2.1.51 0.3.3 - 2013-02-17

Added

- South support via implementing the `south_triple_field` method. (mattions)

Fixed

- Fixed issues with money widget not passing attrs up to django's render method, caused id attribute to not be set in html for widgets. (adambregenzer)
- Fixed issue of default currency not being passed on to widget. (snbuchholz)
- Return the right default for South. (mattions)
- Django 1.5 compatibility. (devlocal)

2.1.52 0.3.2 - 2012-11-30

Fixed

- Fixed issues with `display_for_field` not detecting fields correctly. (adambregenzer)
- Added South ignore rule to avoid duplicate currency field when using the frozen ORM. (rach)
- Disallow override of objects manager if not setting it up with an instance. (rach)

2.1.53 0.3.1 - 2012-10-11

Fixed

- Fix `AttributeError` when Model inherit a manager. (rach)
- Correctly serialize the field. (akumria)

2.1.54 0.3 - 2012-09-30

Added

- Allow django-money to be specified as read-only in a model. (akumria)
- South support: Declare default attribute values. (pjdelport)

2.1.55 0.2 - 2012-04-10

- Initial public release

2.2 Contributing to Django money

Django-money is contribute-friendly project. Contributions are highly welcomed and appreciated. There is a guideline for contributing.

2.2.1 Quickstart

- All code contributions should be tested
- Code should conform to syntax conventions. There's a `tox` command to help fixing it: `tox -e fix-lint`
- Documentation should be updated if it is required
- Put a note to the changelog.

2.2.2 Syntax and conventions

The source code should conform to PEP8 with following notice:

- Line length should not exceed **120** characters.

2.2.3 Running the tests

We use `tox` to run the tests:

```
$ tox -e lint,django111-py36,django111-py27

The test environments above are usually enough to cover most cases locally.
```

2.2.4 Report bugs

Report bugs in the issue tracker.

If you are reporting a bug, please include:

- Any details about your local setup that might be helpful in troubleshooting, specifically the Python interpreter version, Django & django-money versions.
- Detailed steps to reproduce the bug.

If you can write a demonstration test that currently fails but should pass (xfail), that is a very useful commit to make as well, even if you cannot fix the bug itself.

CHAPTER 3

Indices and tables

- genindex
- modindex
- search