

---

# **django-birthday Documentation**

***Release 0.1.1***

**Jonas Obrist**

**Oct 29, 2021**



**CONTENTS**

**1 Usage 3**  
1.1 Method References . . . . . 3

**2 Limitations 5**

**3 Indices and tables 7**

**Index 9**



django-birthday is a helper library for Django to work with birthdays in models.

**Warning:** This library is pretty hacky. Use it if you're lazy, not if you're concerned about your code quality!

Contents:



## USAGE

django-birthday provides a `birthday.fields.BirthdayField` model field type which is a subclass of `django.db.models.DateField` and thus has the same characteristics as that. It also internally adds a second field to your model holding the day of the year for that birthday, this is used for the extra functionality exposed by `birthday.managers.BirthdayManager` which you should use as the manager on your model.

A model could look like this:

```
from django.db import models
from django.conf import settings

from birthday import BirthdayField, BirthdayManager

class UserProfile(models.Model):
    user = models.ForeignKey(settings.AUTH_USER_MODEL)
    birthday = BirthdayField()

    objects = BirthdayManager()
```

Get all user profiles within the next 30 days:

```
UserProfile.objects.get_upcoming_birthdays()
```

Get all user profiles which have their birthday today:

```
UserProfile.objects.get_birthdays()
```

Or order the user profiles according to their birthday:

```
UserProfile.objects.order_by_birthday()
```

## 1.1 Method References

`birthday.managers.BirthdayManager.get_upcoming_birthdays()`

Returns a queryset containing objects that have an upcoming birthday.

### Parameters

- **days** – *Optional*. Amount of days that still count as ‘upcoming’, defaults to 30.
- **after** – *Optional*. Start day to use, defaults to ‘today’.

- **include\_day** – *Optional*. Include the ‘after’ day for lookups.
- **order** – *Optional*. Whether the queryset should be ordered by birthday, defaults to True.
- **reverse** – *Optional*. Only applies when *order* is True. Apply reverse ordering.

**Return type** Instance of `django.db.models.query.QuerySet`.

`birthday.managers.BirthdayManager.get_birthdays()`

Returns a queryset containing objects which have the birthday on a specific day.

**Parameters** **day** – *Optional*. What day to get the birthdays of. Defaults to ‘today’.

**Return type** Instance of `django.db.models.query.QuerySet`.

`birthday.managers.BirthdayManager.order_by_birthday()`

Returns a queryset ordered by birthday (not age!).

**Parameters** **reverse** – *Optional*. Defaults to *False*. Whether or not to reverse the results.

**Return type** Instance of `django.db.models.query.QuerySet`.



## LIMITATIONS

There are a couple of limitations for django-birthday:

- You can only have **one** `birthday.fields.BirthdayField` field on a single model.
- You cannot chain the custom methods provided by the manager.
- Ordering by a `birthday.fields.BirthdayField` while not using `birthday.managers.BirthdayManager.order_by_birthday()` will order by **age**, not when their birthday is in a year.



## INDICES AND TABLES

- `genindex`
- `modindex`
- `search`



## INDEX

### G

`get_birthdays()` (*birthday.managers.BirthdayManager* method),  
4

`get_upcoming_birthdays()` (*birthday.managers.BirthdayManager* method),  
3

### O

`order_by_birthday()` (*birthday.managers.BirthdayManager* method),  
4