



# **django***celeryresultsDocumentation*

***Release 1.0.4***

**Ask Solem**

**Nov 13, 2018**



---

## Contents

---

<b>1</b>	<b>About</b>	<b>3</b>
<b>2</b>	<b>Installing</b>	<b>5</b>
<b>3</b>	<b>Contents</b>	<b>7</b>
<b>4</b>	<b>Indices and tables</b>	<b>13</b>
	<b>Python Module Index</b>	<b>15</b>



**Version** 1.0.4

**Web** <http://django-celery-results.readthedocs.io/>

**Download** <http://pypi.python.org/pypi/django-celery-results>

**Source** <http://github.com/celery/django-celery-results>

**Keywords** django, celery, database, results



# CHAPTER 1

---

## About

---

This extension enables you to store Celery task results using the Django ORM.

It defines a single model (`django_celery_results.models.TaskResult`) used to store task results, and you can query this database table like any other Django model.





## CHAPTER 2

---

### Installing

---

The installation instructions for this extension is available from the [Celery documentation](#):

<http://docs.celeryproject.org/en/latest/django/first-steps-with-django.html#django-celery-results-using-the-django-orm-cache-as-a-resu>



### 3.1 Copyright

*django-celery-results User Manual*

by Ask Solem

Copyright © 2016, Ask Solem

All rights reserved. This material may be copied or distributed only subject to the terms and conditions set forth in the *Creative Commons Attribution-ShareAlike 4.0 International* <<http://creativecommons.org/licenses/by-sa/4.0/legalcode>>‘\_ license.

You may share and adapt the material, even for commercial purposes, but you must give the original author credit. If you alter, transform, or build upon this work, you may distribute the resulting work only under the same license or a license compatible to this one.

---

**Note:** While the *django-celery-results documentation* is offered under the *Creative Commons Attribution-ShareAlike 4.0 International* license the *django-celery-results software* is offered under the [BSD License \(3 Clause\)](#)

---

### 3.2 API Reference

**Release** 1.0

**Date** Nov 13, 2018

#### 3.2.1 `django_celery_results.backends`

**class** `django_celery_results.backends.CacheBackend(*args, **kwargs)`  
Backend using the Django cache framework to store task metadata.

**cache\_backend**

**decode** (*data*)

**delete** (*key*)

**encode** (*data*)

**get** (*key*)

**set** (*key*, *value*)

```
class django_celery_results.backends.DatabaseBackend(app,
                                                    serializer=None,
                                                    max_cached_results=None,
                                                    accept=None, expires=None,
                                                    expires_type=None, url=None,
                                                    **kwargs)
```

The Django database backend, using models to store task state.

**TaskModel**

alias of `django_celery_results.models.TaskResult`

**cleanup** ()

Delete expired metadata.

**decode\_content** (*obj*, *content*)

**encode\_content** (*data*)

**subpolling\_interval** = 0.5

### 3.2.2 django\_celery\_results.backends.database

```
class django_celery_results.backends.database.DatabaseBackend(app,
                                                             serial-izer=None,
                                                             max_cached_results=None,
                                                             accept=None, ex-pires=None, ex-pires_type=None,
                                                             url=None,
                                                             **kwargs)
```

The Django database backend, using models to store task state.

**TaskModel**

alias of `django_celery_results.models.TaskResult`

**cleanup** ()

Delete expired metadata.

**decode\_content** (*obj*, *content*)

**encode\_content** (*data*)

**subpolling\_interval** = 0.5

### 3.2.3 django\_celery\_results.backends.cache

Celery cache backend using the Django Cache Framework.

```
class django_celery_results.backends.cache.CacheBackend(*args, **kwargs)
    Backend using the Django cache framework to store task metadata.
```

```

cache_backend
decode (data)
delete (key)
encode (data)
get (key)
set (key, value)

```

### 3.2.4 django\_celery\_results.models

Database models.

```

class django_celery_results.models.TaskResult (*args, **kwargs)
    Task result/status.

    exception DoesNotExist
    exception MultipleObjectsReturned
    as_dict ()
    content_encoding
        A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is
        executed.
    content_type
        A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is
        executed.
    date_done
        A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is
        executed.
    get_next_by_date_done (**morekwargs)
    get_previous_by_date_done (**morekwargs)
    get_status_display (**morekwargs)
    hidden
        A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is
        executed.
    id
        A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is
        executed.
    meta
        A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is
        executed.
    objects = <django_celery_results.managers.TaskResultManager object>
    result
        A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is
        executed.
    status
        A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is
        executed.

```

**task\_args**

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

**task\_id**

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

**task\_kwargs**

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

**task\_name**

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

**traceback**

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

### 3.2.5 django\_celery\_results.managers

Model managers.

**class** django\_celery\_results.managers.TaskResultManager

Manager for celery.models.TaskResult models.

**connection\_for\_read()**

**connection\_for\_write()**

**current\_engine()**

**delete\_expired(*expires*)**

Delete all expired results.

**get\_all\_expired(*expires*)**

Get all expired task results.

**get\_task(*task\_id*)**

Get result for task by `task_id`.

**Keyword Arguments** **exception\_retry\_count** (*int*) – How many times to retry by transaction rollback on exception. This could happen in a race condition if another worker is trying to create the same task. The default is to retry once.

**store\_result(\*\**kwargs*)**

Store the result and status of a task.

**Parameters**

- **content\_type** (*str*) – Mime-type of result and meta content.
- **content\_encoding** (*str*) – Type of encoding (e.g. binary/utf-8).
- **task\_id** (*str*) – Id of task.
- **task\_name** (*str*) – Celery task name.
- **task\_args** (*str*) – Task arguments.
- **task\_kwargs** (*str*) – Task kwargs.

- **result** (*str*) – The serialized return value of the task, or an exception instance raised by the task.
- **status** (*str*) – Task status. See `celery.states` for a list of possible status values.

#### Keyword Arguments

- **traceback** (*str*) – The traceback string taken at the point of exception (only passed if the task failed).
- **meta** (*str*) – Serialized result meta data (this contains e.g. children).
- **exception\_retry\_count** (*int*) – How many times to retry by transaction rollback on exception. This could happen in a race condition if another worker is trying to create the same task. The default is to retry twice.

**warn\_if\_repeatable\_read**()

**exception** `django_celery_results.managers.TxIsolationWarning`  
Warning emitted if the transaction isolation level is suboptimal.

`django_celery_results.managers.transaction_retry` (*max\_retries=1*)  
Decorate a function to retry database operations.

For functions doing database operations, adding retrying if the operation fails.

**Keyword Arguments** **max\_retries** (*int*) – Maximum number of retries. Default one retry.

### 3.2.6 django\_celery\_results.utils

Utilities.

`django_celery_results.utils.now`()  
Return the current date and time.

## 3.3 Change history

### 3.3.1 1.0.4

**release-date** 2018-11-12 19:00 p.m. UTC+2:00

**release-by** Omer Katz

1.0.3 is broken. Use 1.0.4

- Revert renaming label as it is a breaking change.

### 3.3.2 1.0.3

**release-date** 2018-11-12 18:00 p.m. UTC+2:00

**release-by** Omer Katz

- Revert renaming label as it is a breaking change.

### 3.3.3 1.0.2

**release-date** 2018-11-12 18:00 p.m. UTC+2:00

**release-by** Omer Katz

- **Store task name, args, kwargs as part of the task results in database.** Contributed by :github\_user: *wardal*.
- **Admin screen changes - task name filter, search on task\_name, task\_id, status.** Contributed by :github\_user: *jaylynch*.
- Added default\_app\_config.
- Added missing migration.
- Fix MySQL max length issue.
- Drop support for Django<1.11.

### 3.3.4 1.0.1

**release-date** 2016-11-07 02:00 p.m. PST

**release-by** Ask Solem

- Migrations were not being installed as part of the distribution (Issue #4).
- Now includes simple task result admin interface.  
Contributed by @zeezdev.
- Now depends on Celery 4.0.0.

### 3.3.5 1.0.0

**release-date** 2016-09-08 03:19 p.m. PDT

**release-by** Ask Solem

- Initial release

## 3.4 Glossary

**term** Description of term



## CHAPTER 4

---

### Indices and tables

---

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



### d

`django_celery_results.backends`, [7](#)  
`django_celery_results.backends.cache`, [8](#)  
`django_celery_results.backends.database`,  
[8](#)  
`django_celery_results.managers`, [10](#)  
`django_celery_results.models`, [9](#)  
`django_celery_results.utils`, [11](#)



## A

`as_dict()` (`django_celery_results.models.TaskResult` method), 9

## C

`cache_backend` (`django_celery_results.backends.cache.CacheBackend` attribute), 8

`cache_backend` (`django_celery_results.backends.CacheBackend` attribute), 7

`CacheBackend` (class in `django_celery_results.backends`), 7

`CacheBackend` (class in `django_celery_results.backends.cache`), 8

`cleanup()` (`django_celery_results.backends.database.DatabaseBackend` method), 8

`cleanup()` (`django_celery_results.backends.DatabaseBackend` method), 8

`connection_for_read()` (`django_celery_results.managers.TaskResultManager` method), 10

`connection_for_write()` (`django_celery_results.managers.TaskResultManager` method), 10

`content_encoding` (`django_celery_results.models.TaskResult` attribute), 9

`content_type` (`django_celery_results.models.TaskResult` attribute), 9

`current_engine()` (`django_celery_results.managers.TaskResultManager` method), 10

## D

`DatabaseBackend` (class in `django_celery_results.backends`), 8

`DatabaseBackend` (class in `django_celery_results.backends.database`), 8

`date_done` (`django_celery_results.models.TaskResult` attribute), 9

`decode()` (`django_celery_results.backends.cache.CacheBackend` method), 9

`decode()` (`django_celery_results.backends.CacheBackend` method), 8

`decode_content()` (`django_celery_results.backends.database.DatabaseBackend` method), 8

`decode_content()` (`django_celery_results.backends.DatabaseBackend` method), 8

`delete()` (`django_celery_results.backends.cache.CacheBackend` method), 9

`delete()` (`django_celery_results.backends.CacheBackend` method), 8

`delete_expired()` (`django_celery_results.managers.TaskResultManager` method), 10

`django_celery_results.backends` (module), 7

`django_celery_results.backends.cache` (module), 8

`django_celery_results.backends.database` (module), 8

`django_celery_results.managers` (module), 10

`django_celery_results.models` (module), 9

`django_celery_results.utils` (module), 11

## E

`encode()` (`django_celery_results.backends.cache.CacheBackend` method), 9

`encode()` (`django_celery_results.backends.CacheBackend` method), 8

`encode_content()` (`django_celery_results.backends.database.DatabaseBackend` method), 8

`encode_content()` (`django_celery_results.backends.DatabaseBackend` method), 8

## G

`get()` (`django_celery_results.backends.cache.CacheBackend` method), 9

`get()` (`django_celery_results.backends.CacheBackend` method), 8

`get_all_expired()` (`django_celery_results.managers.TaskResultManager` method), 10

`get_next_by_date_done()` (`django_celery_results.models.TaskResult` method), 9

[get\\_previous\\_by\\_date\\_done\(\)](#) (django-celery-results.models.TaskResult method), [9](#)  
[get\\_status\\_display\(\)](#) (django-celery-results.models.TaskResult method), [9](#)  
[get\\_task\(\)](#) (django-celery-results.managers.TaskResultManager method), [10](#)  
**H**  
[hidden](#) (django-celery-results.models.TaskResult attribute), [9](#)  
**I**  
[id](#) (django-celery-results.models.TaskResult attribute), [9](#)  
**M**  
[meta](#) (django-celery-results.models.TaskResult attribute), [9](#)  
**N**  
[now\(\)](#) (in module django-celery-results.utils), [11](#)  
**O**  
[objects](#) (django-celery-results.models.TaskResult attribute), [9](#)  
**R**  
[result](#) (django-celery-results.models.TaskResult attribute), [9](#)  
**S**  
[set\(\)](#) (django-celery-results.backends.cache.CacheBackend method), [9](#)  
[set\(\)](#) (django-celery-results.backends.CacheBackend method), [8](#)  
[status](#) (django-celery-results.models.TaskResult attribute), [9](#)  
[store\\_result\(\)](#) (django-celery-results.managers.TaskResultManager method), [10](#)  
[subpolling\\_interval](#) (django-celery-results.backends.database.DatabaseBackend attribute), [8](#)  
[subpolling\\_interval](#) (django-celery-results.backends.DatabaseBackend attribute), [8](#)  
**T**  
[task\\_args](#) (django-celery-results.models.TaskResult attribute), [9](#)  
[task\\_id](#) (django-celery-results.models.TaskResult attribute), [10](#)  
[task\\_kwargs](#) (django-celery-results.models.TaskResult attribute), [10](#)  
[task\\_name](#) (django-celery-results.models.TaskResult attribute), [10](#)  
[TaskModel](#) (django-celery-results.backends.database.DatabaseBackend attribute), [8](#)  
[TaskModel](#) (django-celery-results.backends.DatabaseBackend attribute), [8](#)  
[TaskResult](#) (class in django-celery-results.models), [9](#)  
[TaskResult.DoesNotExist](#), [9](#)  
[TaskResult.MultipleObjectsReturned](#), [9](#)  
[TaskResultManager](#) (class in django-celery-results.managers), [10](#)  
[term](#), [12](#)  
[traceback](#) (django-celery-results.models.TaskResult attribute), [10](#)  
[transaction\\_retry\(\)](#) (in module django-celery-results.managers), [11](#)  
[TxIsolationWarning](#), [11](#)  
**W**  
[warn\\_if\\_repeatabe\\_read\(\)](#) (django-celery-results.managers.TaskResultManager method), [11](#)