



EBook Gratuito

APPENDIMENTO

celery

Free unaffiliated eBook created from
Stack Overflow contributors.

#celery

Sommario

Di.....	1
Capitolo 1: Iniziare con il sedano.....	2
Osservazioni.....	2
Examples.....	2
Installazione o configurazione.....	2
Sedano + Redis.....	2
Installazione.....	2
Configurazione.....	2
Applicazione.....	3
Esecuzione del server di Celery.....	3
Chiamare l'attività.....	3
Mantenere i risultati.....	3
Capitolo 2: Monitoraggio del sedano con Fiore.....	5
Examples.....	5
Pronto e funzionante con Flower.....	5
Titoli di coda.....	6

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: [celery](#)

It is an unofficial and free celery ebook created for educational purposes. All the content is extracted from [Stack Overflow Documentation](#), which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official celery.

The content is released under Creative Commons BY-SA, and the list of contributors to each chapter are provided in the credits section at the end of this book. Images may be copyright of their respective owners unless otherwise specified. All trademarks and registered trademarks are the property of their respective company owners.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Capitolo 1: Iniziare con il sedano

Osservazioni

"Celery è una coda di attività / coda di lavoro asincrona basata sul passaggio di messaggi distribuiti." - <http://www.celeryproject.org/>

Celery è ottimo per le attività in background asincrone e pianificate. Viene comunemente utilizzato per attività di lunga durata che fanno parte di un'applicazione Django o Flask.

Examples

Installazione o configurazione

È possibile installare Celery tramite il Python Package Index (PyPI) o dalla sorgente.

Per installare l'ultima versione usando `pip` :

```
$ pip install celery
```

Per installare utilizzando `easy_install` :

```
$ easy_install celery
```

Download e installazione dalla fonte

Scarica l'ultima versione di Celery da <http://pypi.python.org/pypi/celery/>

Puoi installarlo effettuando le seguenti operazioni:

```
$ tar xvfz celery-0.0.0.tar.gz
$ cd celery-0.0.0
$ python setup.py build
# python setup.py install # as root
```

Sedano + Redis

Installazione

Sono necessarie dipendenze aggiuntive per il supporto Redis. Installa contemporaneamente Celery e le dipendenze usando il pacchetto di `celery[redis]` :

```
$ pip install -U celery[redis]
```

Configurazione

Configura la posizione del tuo database Redis:

```
BROKER_URL = 'redis://localhost:6379/0'
```

L'URL dovrebbe essere nel formato di:

```
redis://:password@hostname:port/db_number
```

Applicazione

Crea il file `tasks.py`:

```
from celery import Celery

BROKER_URL = 'redis://localhost:6379/0'
app = Celery('tasks', broker=BROKER_URL)

@app.task
def add(x, y):
    return x + y
```

Il primo argomento su `Celery` è il nome del modulo corrente. In questo modo i nomi possono essere generati automaticamente. Il secondo argomento è la parola chiave `broker` che specifica l'URL del broker dei messaggi.

Esecuzione del server di Celery

Esegui il worker eseguendo con l'argomento `worker`:

```
$ celery -A tasks worker --loglevel=info
```

Chiamare l'attività

Per chiamare l'attività, utilizzare il metodo `delay()`.

```
>>> from tasks import add
>>> add.delay(4, 4)
```

La chiamata di un'attività restituisce un'istanza [AsyncResult](#), che può controllare lo stato dell'attività, attendere il completamento dell'attività o ottenere il valore restituito. (Se l'attività non è riuscita, ottiene l'eccezione e traceback).

Mantenere i risultati

Per tenere traccia degli stati dell'attività, Celery ha bisogno di memorizzare o inviare gli stati da qualche parte. Usa Redis come back-end dei risultati:

```
BROKER_URL = 'redis://localhost:6379/0'
BACKEND_URL = 'redis://localhost:6379/1'
app = Celery('tasks', broker=BROKER_URL, backend=BACKEND_URL)
```

Per saperne di più sui backend dei risultati, vedere i backend dei [risultati](#) .

Ora con il backend risultato configurato, richiamare nuovamente l'attività. Questa volta tieni premuto [sull'istanza AsyncResult](#) restituita dall'attività:

```
>>> result = add.delay(4, 4)
```

Il metodo `ready()` restituisce se l'attività ha terminato l'elaborazione o meno:

```
>>> result.ready()
False
```

È possibile attendere il completamento del risultato, ma questo è usato raramente poiché trasforma la chiamata asincrona in una sincrona:

```
>>> result.get(timeout=1)
8
```

Basato sul documento ufficiale di sedano

Leggi [Iniziare con il sedano online](https://riptutorial.com/it/celery/topic/6987/iniziare-con-il-sedano): <https://riptutorial.com/it/celery/topic/6987/iniziare-con-il-sedano>

Capitolo 2: Monitoraggio del sedano con Fiore

Examples

Pronto e funzionante con Flower

Flower è uno strumento basato sul web per monitorare Celery.

Per installare Flower, possiamo utilizzare `pip` come segue:

```
pip install flower
```

Per eseguire Flower for `proj` :

```
celery -A proj flower
```

Ora, il fiore è accessibile in:

```
http://localhost:5555
```

Leggi [Monitoraggio del sedano con Fiore online](https://riptutorial.com/it/celery/topic/7477/monitoraggio-del-sedano-con-fiore):

<https://riptutorial.com/it/celery/topic/7477/monitoraggio-del-sedano-con-fiore>

Titoli di coda

S. No	Capitoli	Contributors
1	Iniziare con il sedano	4444 , Community , jamjar , jegesh , Majid
2	Monitoraggio del sedano con Fiore	ettanany