

Celery

Eine asynchrone Task Queue
(nicht nur) für Django

Markus Zapke-Gründemann

www.keimlink.de

Leipzig Python User Group 12.10.2010

Überblick

- Warum eine Task Queue?
- Celery
- Python Task
- Zeitgesteuerter Task
- Django Task
- Webhook

Warum eine Task Queue?

- Entkoppeln von Informationsproduzenten und -konsumenten
- Asynchrone Verarbeitung
- Skalierbarkeit

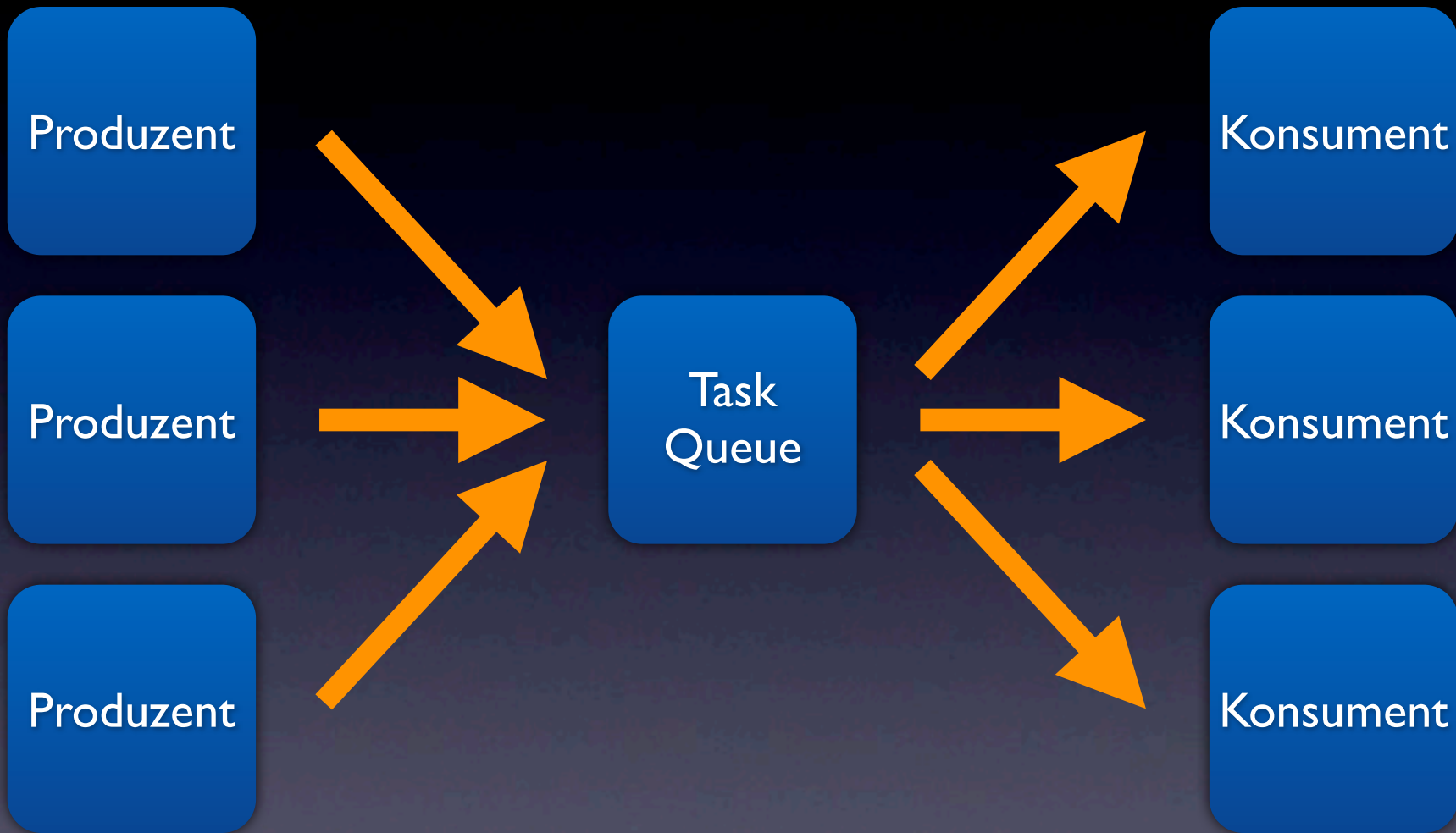
Produzent

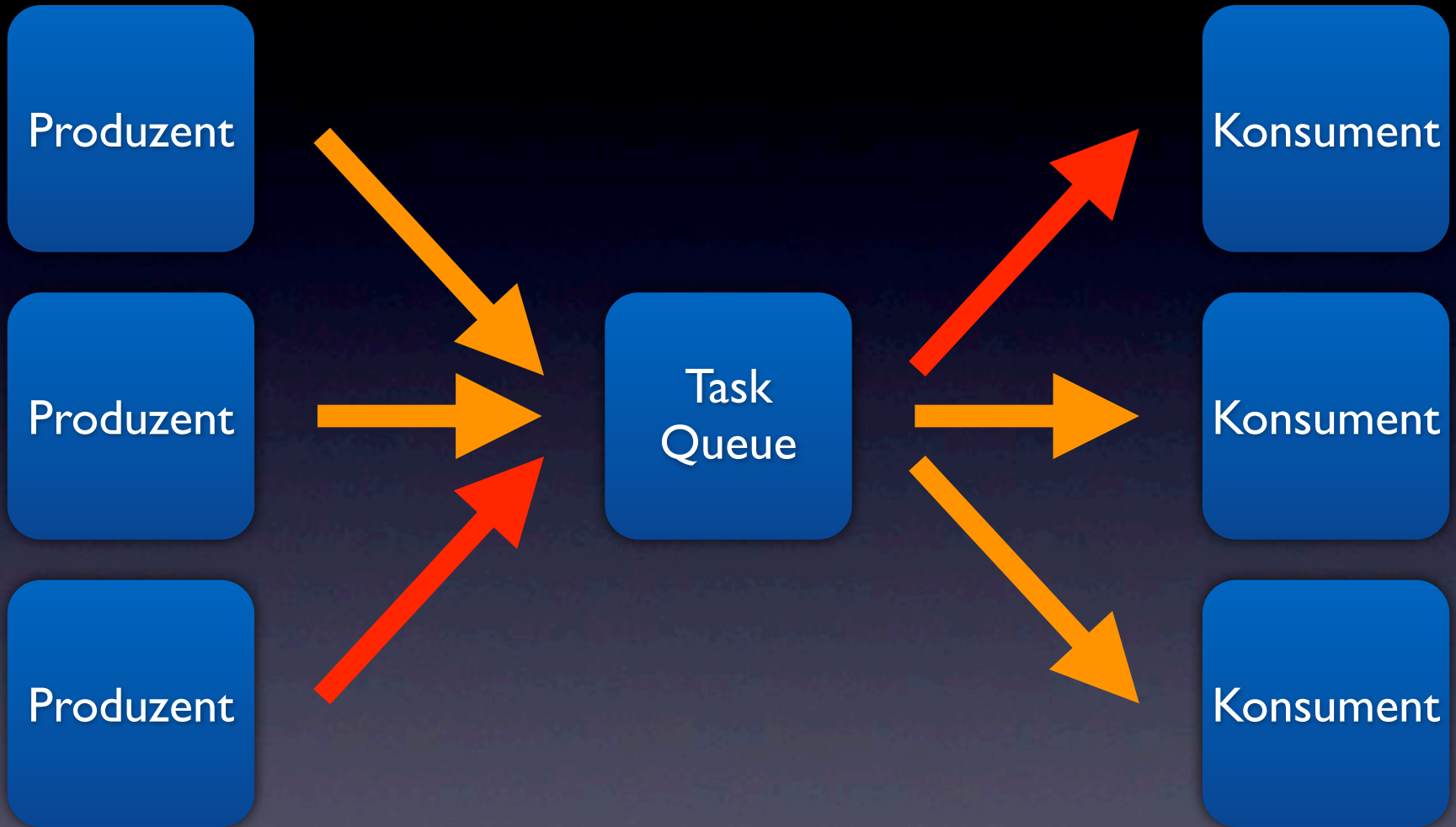


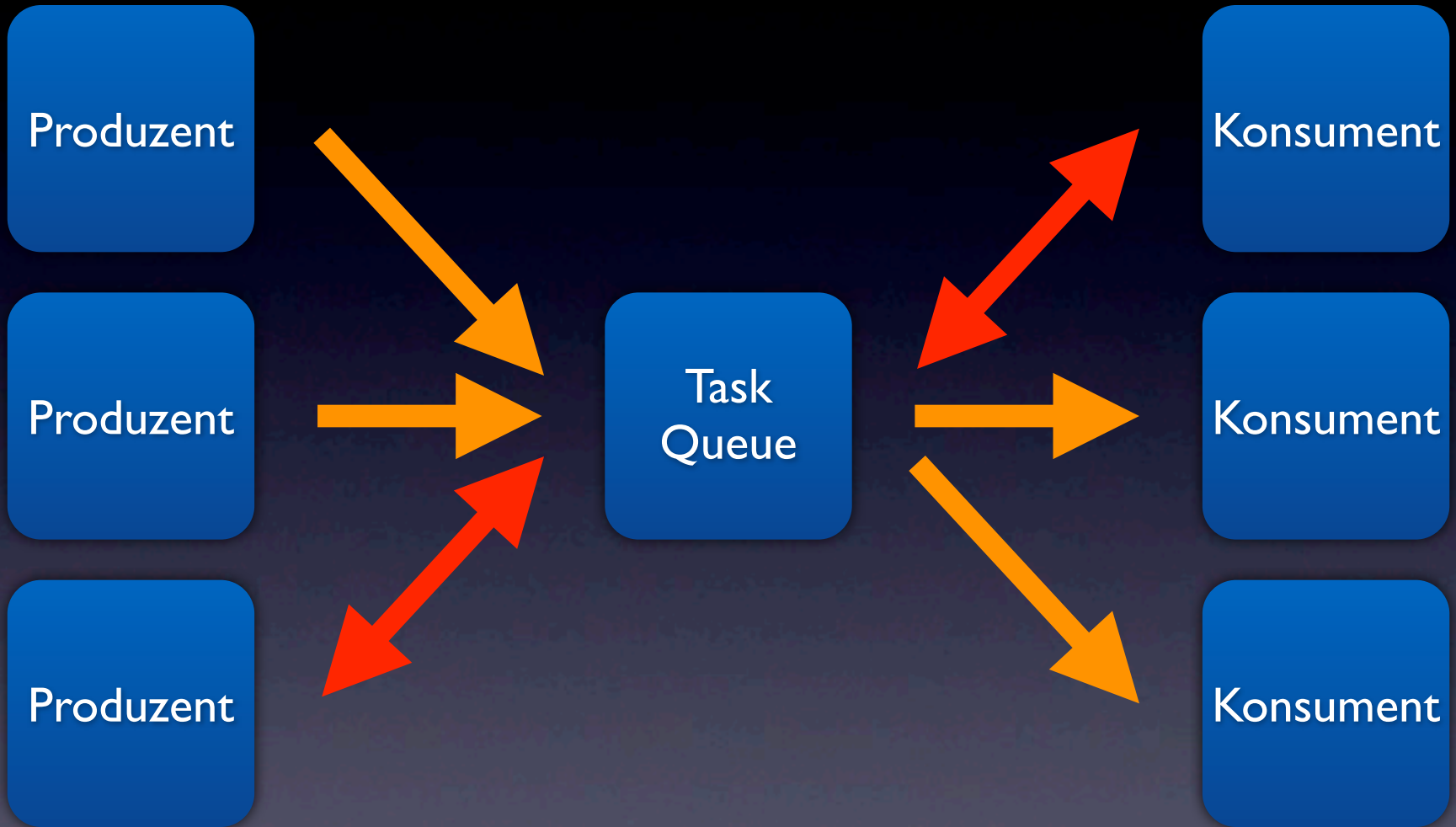
Task
Queue



Konsument







Celery

- Autor: Ask Solem Hoel
- RabbitMQ, Stomp, Redis und Ghetto Queue
- Clustering mit RabbitMQ
- Parallele Ausführung
- Zeitgesteuerte Ausführung

Celery

- Rückgabewert sofort oder später verarbeiten
- Speichern der Rückgabe in „Result Stores“
- Webhooks
- Serialisierung (Pickle, JSON, YAML)
- Wiederholen abgebrochener Tasks

Celery

- Callbacks
- Task Sets
- Logging
- Bei Fehler E-Mail versenden
- Ausführliche Dokumentation
- BSD Lizenz

Celery Komponenten

- celeryd
- celerybeat
- celeryctl
- celeryev
- camqadm
- celerymon
- carrot
- anyjson
- SQLAlchemy (optional)

Celery Installation

```
$ pip install celery
```

```
$ easy_install celery
```

Python Task

```
# Task als Klasse
from celery.task import Task
from myproject.models import User

class CreateUserTask(Task):
    def run(self, username, password):
        User.create(username, password)

>>> from tasks import CreateUserTask
>>> CreateUserTask().delay('john', 'secret')
```



```
# Task als Funktion mit Decorator
from celery.decorators import task
from myproject.models import User

@task
def create_user(username, password):
    User.create(username, password)

>>> from tasks import create_user
>>> create_user.delay('john', 'secret')
```

Python Task

```
@task() # Benutzt pickle, um das Objekt zu serialisieren.
def check_means(user):
    return user.has_means()

>>> from tasks import check_means
>>> result = check_means.delay(user)
>>> result.ready() # Gibt True zurück wenn der Task beendet ist.
False
>>> result.result # Task ist noch nicht beendet, kein Ergebnis verfügbar.
None
>>> result.get() # Warten bis der Task fertig ist und Ergebnis zurückgeben.
93.27
>>> result.result # Jetzt ist ein Ergebnis da.
93.27
>>> result.successful() # War der Task erfolgreich?
True
```

Python Task Konfiguration

```
# celeryconfig.py
BROKER_HOST = "localhost"
BROKER_PORT = 5672
BROKER_USER = "myuser"
BROKER_PASSWORD = "mypassword"
BROKER_VHOST = "myvhost"

CELERY_RESULT_BACKEND = "database"
CELERY_RESULT_DBURI = "mysql://user:password@host/dbname"

CELERY_IMPORTS = ("tasks", )
```

```
$ celeryd --loglevel=INFO
```

Zeitgesteuerter Task

```
# Periodic task
from celery.decorators import periodic_task
from datetime import timedelta

@periodic_task(run_every=timedelta(seconds=30))
def every_30_seconds():
    print("Running periodic task!")

# crontab
from celery.task.schedules import crontab
from celery.decorators import periodic_task

@periodic_task(run_every=crontab(hour=7, minute=30, day_of_week=1))
def every_monday_morning():
    print("Execute every Monday at 7:30AM.")
```

```
$ celerybeat
```

```
$ celeryd -B
```

celeryev

celeryev 1.1.1

UUID	TASK	WORKER	TIME	STATE
63aa2f21-433e-4cae-8882-9ffffc2c09d6	tasks.sleeptask	casper.local	10:02:30	SUCCESS
fcca35b5-8b52-49a4-a79e-31a0747aca98	tasks.sleeptask	casper.local	10:02:27	SUCCESS
44d58060-833e-45fc-a291-11abc1ee44a4	tasks.sleeptask	casper.local	10:02:25	SUCCESS
bed79a28-3819-4904-975f-9eb5a7aae2d5	tasks.sleeptask	casper.local	10:02:23	SUCCESS
2599b117-3c10-45a3-8544-2e63b284c96f	tasks.sleeptask	casper.local	10:02:21	SUCCESS
7a07fcc1-7a13-4878-82a6-738673e4c3d9	tasks.sleeptask	casper.local	10:02:18	RECEIVED
75486d0d-aae4-4129-bc55-feba0a2abe03	tasks.sleeptask	casper.local	10:02:18	RECEIVED
e47e2069-a2bf-4af3-a93d-c3ef96ffd12c	tasks.sleeptask	casper.local	10:02:18	RECEIVED
3a7a6759-7fa8-48ec-9f89-b222acd3b49f	tasks.sleeptask	casper.local	10:02:18	RECEIVED
01fec1b6-6996-41f9-a337-909adec5183d	tasks.sleeptask	casper.local	10:02:18	RECEIVED
fda219d3-c24b-492c-b948-9f09b1945e8d	tasks.sleeptask	casper.local	10:02:18	RECEIVED
627428a6-a9ed-4c3b-ad64-a869b582e068	tasks.sleeptask	casper.local	10:02:18	RECEIVED
872052d0-71b6-4287-a24d-d60fda0e8ebc	tasks.sleeptask	casper.local	10:02:18	RECEIVED
c8d0a21e-aac2-4f3a-90d6-fee3b94caaca	tasks.sleeptask	casper.local	10:02:18	RECEIVED
1c9d67d8-0b8f-4fd0-8d30-e72694526df3	tasks.sleeptask	casper.local	10:02:18	RECEIVED
6b179f86-4be5-4b0e-a81b-e25525c3a02a	tasks.sleeptask	casper.local	10:02:18	RECEIVED
c02ffd1d-36a8-40c4-a5a1-9aedfcba5eeb	tasks.sleeptask	casper.local	10:02:18	RECEIVED
3795b272-b5e4-429e-84e3-583d0e02261b	tasks.sleeptask	casper.local	10:02:18	STARTED
6410ee9b-0ea7-4ff8-b40d-4ca023038fe1	tasks.sleeptask	casper.local	10:02:18	STARTED
6d14daf2-5025-48ea-b445-ca4b9fcc9369	tasks.sleeptask	casper.local	10:02:18	SUCCESS

Selected: runtime=3.01s eta=2010-06-04T10:02:21.513155 args=[3] result=3 kwargs={}

Workers online: casper.local

Info: events:43 tasks:20 workers:1/1

Keys: j:up k:down i:info t:traceback r:result c:revoke ^c: quit

\$ celeryev

Django Task Installation

```
$ pip install django-celery
```

```
$ easy_install django-celery
```

Django Task Spamfilter

Django Task Spamfilter



Produzent

```
blog.views.add_comment
```

Django Task Spamfilter

Produzent

```
blog.views.add_comment
```

Comment
Model

Django Task Spamfilter

Produzent

`blog.views.add_comment`

`comment.save()`

Comment
Model

A diagram illustrating a Django task. It shows a blue rounded square labeled 'Produzent' at the top left. Below it, the code `blog.views.add_comment` is written. A large orange arrow points from this code down to a blue rounded square labeled 'Comment Model' at the bottom right. To the left of the arrow, the code `comment.save()` is written, indicating the specific task being performed.

Django Task Spamfilter

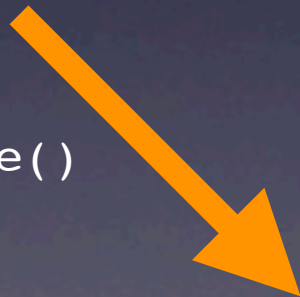
Produzent

Task
Queue

```
blog.views.add_comment
```

```
comment.save()
```

Comment
Model



Django Task Spamfilter

Produzent

Task
Queue

Konsument

`blog.views.add_comment`

`blog.tasks.spam_filter`

`comment.save()`

Comment
Model



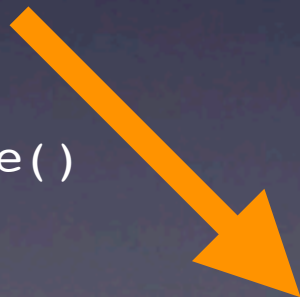
```
graph LR; P[Produzent] --> TQ[Task Queue]; TQ --> K[Konsument]; P --- PCode[blog.views.add_comment]; K --- KCode[blog.tasks.spam_filter]; P --- CM[Comment Model]; CM --- CMCode[comment.save()];
```

Django Task Spamfilter

```
blog.tasks.spam_filter.delay(comment.id,  
remote_addr)
```



```
blog.views.add_comment
```



```
comment.save()
```



```
blog.tasks.spam_filter
```

Django Task Spamfilter

```
blog.tasks.spam_filter.delay(comment.id,  
remote_addr)
```



```
blog.views.add_comment
```

```
blog.tasks.spam_filter
```

```
comment.save()
```

```
graph TD; A[blog.views.add_comment] --> B[comment.save()]; B --> C[Comment Model]
```

The diagram shows a blue rounded square labeled 'Comment Model'. An orange arrow points from the text 'comment.save()' to the 'Comment Model' box. This 'comment.save()' text is positioned below the 'blog.views.add_comment' text from the previous block.

Django Task Spamfilter

```
blog.tasks.spam_filter.delay(comment.id,  
remote_addr)
```



```
blog.views.add_comment
```

```
blog.tasks.spam_filter
```

```
comment.save()
```

```
comment.is_spam = True  
comment.save()
```

```
graph TD; A[blog.views.add_comment] --> B[comment.save()]; B --> C[Comment Model]; C --> D[blog.tasks.spam_filter]; D --> E[comment.is_spam = True, comment.save()];
```

The diagram shows the interaction between the Comment Model and the spam filter task. It starts with a blue rounded square labeled 'Comment Model'. An orange arrow points from the Comment Model to the left, towards the text 'comment.save()'. Another orange arrow points from the Comment Model to the right, towards the text 'blog.tasks.spam_filter'. A third orange arrow points from the text 'blog.tasks.spam_filter' to the right, towards the text 'comment.is_spam = True' and 'comment.save()'.

blog/tasks.py

```
@task
```

```
def spam_filter(comment_id, remote_addr=None, **kwargs):  
    logger = spam_filter.get_logger(**kwargs)  
    logger.info("Running spam filter for comment %s" % comment_id)  
  
    comment = Comment.objects.get(pk=comment_id)  
    current_domain = Site.objects.get_current().domain  
    akismet = Akismet(settings.AKISMET_KEY, "http://%s" % domain)  
    if not akismet.verify_key():  
        raise ImproperlyConfigured("Invalid AKISMET_KEY")  
  
    is_spam = akismet.comment_check(user_ip=remote_addr,  
                                    comment_content=comment.comment,  
                                    comment_author=comment.name,  
                                    comment_author_email=comment.email_address)  
  
    if is_spam:  
        comment.is_spam = True  
        comment.save()  
  
    return is_spam
```

Django Task Konfiguration

```
# settings.py
INSTALLED_APPS += ("djcelery", )
import djcelery
djcelery.setup_loader()
```

```
BROKER_HOST = "localhost"
BROKER_PORT = 5672
BROKER_USER = "myuser"
BROKER_PASSWORD = "mypassword"
BROKER_VHOST = "myvhost"
```

```
$ python manage.py syncdb
```

```
$ python manage.py celeryd -l info
```

Django Admin Monitor

Django administration Welcome, askadmin. Change password / Log out

Home > Djangery > Tasks

Tasks

2010

Action: ----- : Go 0 of 172 selected

<input type="checkbox"/>	UUID	State	Name	Args	Kwargs	ETA	Worker
<input type="checkbox"/>	5456f8ea-f8c8-4778-91b1-a468abc4f91f	SUCCESS	[.]statuses_twitter_update	('e97c687013087f614a6081708087308e', '96376157-e6ca-48d1-8804-8b...)	()	None	h9
<input type="checkbox"/>	b6fa561c-8f8a-4e39-9894-58403a21fb81	SUCCESS	portaloperacom[.]refresh_portal_blog	()	()	None	h10
<input type="checkbox"/>	ef256274-7a73-4834-90fb-03f89e15aca6	SUCCESS	[.]statuses_twitter_update	('e97c687013087f614a6081708087308e', '96376157-e6ca-48d1-8804-8b...)	()	None	h9
<input type="checkbox"/>	3359ba9e-73b7-4d8f-ba7e-1fb6269ba396	SUCCESS	d[.]refresh_feed	()	('feed_id': 232L, 'feed_url': 'http://www.cbc.com/16/18746125/...')	None	h6
<input type="checkbox"/>	51681ec8-842b-46e4-9998-7632460b40b3	SUCCESS	[.]statuses_twitter_update	('e97c687013087f614a6081708087308e', '96376157-e6ca-48d1-8804-8b...)	()	None	h10
<input type="checkbox"/>	ce47e922-c76c-45aa-8805-c2d853e05dbb	SUCCESS	[.]statuses_twitter_update	('e97c687013087f614a6081708087308e', '96376157-e6ca-48d1-8804-8b...)	()	None	h9
<input type="checkbox"/>	2ebb8254-6891-4b6c-a8e7-	SUCCESS	[.]statuses_twitter_update	('e97c687013087f614a6081708087308e', '96376157-e6ca-48d1-8804-8b...)	()	None	h10

Filter

By state

- All
- RETRY
- REVOKED
- SUCCESS
- STARTED
- FAILURE
- PENDING

By name

All

- djangofeeds.tasks.refresh_feed
- opalfmevents.lastfm_events_up
- opaltweets.statuses_twitter_up
- portaloperacom.tasks.refresh_

By event received at

Any date

- Today
- Past 7 days
- This month
- This year

By ETA

Any date

- Today
- Past 7 days
- This month
- This year

By worker

All

- h10
- h8
- h6

```
$ python manage.py celerycam
oder
$ djcelerymon
```

Webhook

```
# POST
```

```
>>> from celery.task.http import URL
>>> res = URL("http://example.com/multiply").get_async(x=10, y=10)
>>> res.get() # {"status": "success", "retval": 100}
100
```

```
# GET
```

```
>>> from celery.task.http import HttpDispatchTask
>>> url = "http://example.com/multiply"
>>> res = HttpDispatchTask.delay(url, method="GET", x=10, y=10)
>>> res.get() # {"status": "success", "retval": 100}
100
```

Links

- <http://celeryproject.org/>
- <http://github.com/ask>
- <http://pypi.python.org/pypi/celery>
- <http://pypi.python.org/pypi/django-celery>
- <http://pypi.python.org/pypi/celerymon>

Lizenz

Dieses Werk ist unter einem Creative Commons Namensnennung-Weitergabe unter gleichen Bedingungen 3.0 Unported Lizenzvertrag lizenziert.

Um die Lizenz anzusehen, gehen Sie bitte zu <http://creativecommons.org/licenses/by-sa/3.0/> oder schicken Sie einen Brief an Creative Commons, 171 Second Street, Suite 300, San Francisco, California 94105, USA.

