Web2py + Celery (async tasks)

First of all, we need to install the dependencies.

```
pip3 install celery redis pyDAL
```

In your web2py/applications/YourApplication/modules create the “tasks.py” file with the following content:

```
"""
To use celery tasks:
1) pip install -U "celery[redis]"
2) In settings.py:
   USE_CELERY = True
   CELERY_BROKER = "redis://localhost:6379/0"
3) Start "redis-server"
4) Start "celery -A apps.{YourAppNameHere}.modules.tasks beat" for intervals
5) Start "celery -A applications.{YourAppNameHere}.modules.tasks worker --loglevel=info" for each worker

"""
from pydal import DAL, Field
from celery import Celery
import os, time

dirpath = os.getcwd()
scheduler = Celery(
        "applications.YourAppNameHere.modules.tasks", broker="redis://localhost:6379/0", backend='redis://localhost'
    )

#connection to the existing database.
db = DAL('sqlite://database.sqlite',
        folder=dirpath + '/applications/YourAppNameHere/databases',
        pool_size=1,
        migrate=False,
        fake_migrate=False,
        )
#define tables like in your model (db.py), for this example there is no need to declare the validators, this table definition is
#only made for pydal can be aware of the tables and columnns
#in case you need validators you have to import it from pydal.validators. ex: from pydal.validators import IS_NOT_EMPTY()
db.define_table('dummytable',
                Field('dummy_field', 'string'),
                )


#simulating a long running process
@scheduler.task()
def MyLongRunningTask(arg1, arg2):
    time.sleep(5)
    #using db
    #db._adapter.reconnect()
    #db.dummytable.insert(dummy_field="i'm a dummy content")
    #db.commit()

    return "Task finished! Args: {} – {}".format(arg1, arg2)

# run MyLongRunningTask every 10 seconds. For this celery need to be started with “beat” option
```

```
scheduler.conf.beat_schedule = {
    "my_first_task": {
        "task": "applications.YourAppNameHere.modules.tasks.MyLongRunningTask",
        "schedule": 10.0,
        "args": ('hello', 'world'),
    },
}
```

Go to web2py directory and run celery:

```
celery -A applications.YourApp.modules.tasks worker --loglevel=info
```

apps –> applications folder in web2py directory.
YourApp –> your application name inside py4web/apps/.
modules –> modules folder inside your application
tasks –> refers to tasks.py file under web2py/applications/YourApp/modules/tasks.py

Note: don't forget to start it inside web2py folder, or some errors can arise.

On your controller, create the functions to call the previous task

```
##on the controller
from tasks import MyLongRunningTask
def call_task():
    #use .delay() method to pass the task to celery, also the arguments need to be passed inside
.delay('hello', 'world')
    result = MyLongRunningTask.delay('Hello', 'World')

    return 'task id: {} - task status {}'.format(result.id, result.status)

## to check if task has finished
#http://127.0.0.1:8000/yourApp/default/check_task/baf05db6-0264-4e68-917b-c7dfab45ab0d
from celery.result import AsyncResult
def check_task():
    res = AsyncResult(request.args(0))
    r = res.status
    print(r)
    if r == 'SUCCESS':
        r = res.get()
        print(r)
    # use .get() only when status is success, if you call .get() (to retrieve results) and the task is still
running,
    #the task will be converted from async to sync.

    return r
```

With the previous, you can have celery working with web2py for asynchronous tasks. Adapt the tasks.py file at your needs.