

Topic: Custom Management Command 8: Importing Large Data Using Celery & Redis

Speaker: Udemy Instructor Rathan Kumar / Notebook: Django: Automating Common Tasks



1. After we installed REDIS, we installed REDIS as a package in our VIRTUAL ENVIRONMENT.

```
(env) Rosalie@DELL MINGW64 C:/Users/Rosalie/AppData/Local/Programs/Microsoft VS Code (main)
$ pip install redis
Collecting redis
  Downloading redis-5.0.8-py3-none-any.whl (255 kB)
    |██████████| 255 kB 1.7 MB/s
Collecting async-timeout>=4.0.3
  Downloading async_timeout-4.0.3-py3-none-any.whl (5.7 kB)
Installing collected packages: async-timeout, redis
Successfully installed async-timeout-4.0.3 redis-5.0.8
WARNING: You are using pip version 21.2.4; however, version 24.2 is available.
You should consider upgrading via the 'C:\Users\Rosalie\OneDrive\Desktop\LEARNING DJANGO PROJECTS\AutomatingCommonTasks\env\Scripts\python.exe -m pip install --upgrade pip' command.
(env)
Rosalie@DELL MINGW64 C:/Users/Rosalie/AppData/Local/Programs/Microsoft VS Code (main)
$ * History restored
```

2. You installed CELERY.

```
$ pip install celery
```

3. Run this to create a Celery Worker:

windows:

```
$ celery -A autocommontasks_main worker --loglevel=info --pool=solo
```

macos:

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

This results to this with an error:

```

(celery)
Rosilie@DELL MINGW64 C:/Users/Rosilie/AppData/Local/Programs/Microsoft VS Code (main)
$ celery -A autocommontasks_main worker --loglevel=Info --pool=solo
[2024-08-13 15:52:03,052: WARNING/MainProcess] No hostname was supplied. Reverting to default 'localhost'

----- celery@DELL v5.4.0 (opalentes)
-- ****
-- ***** Windows-10-10.0.22631-SP0 2024-08-13 15:52:03
-- *** * --
-- ** ----- [config]
-- **      > app:      autocommontasks_main:0x278ef7c2a60
-- **      > transport: amqp://guest:**@localhost:5672//
-- **      > results:  disabled://
-- *** * > concurrency: 8 (solo)
-- ***** > task events: OFF (enable -E to monitor tasks in this worker)
-- ****

----- [queues]
-- > celery      exchange=celery(direct) key=celery

[tasks]
. autocommontasks_main.celery.debug_task

[2024-08-13 15:52:03,097: WARNING/MainProcess] C:/Users/Rosilie/AppData/Local/Programs/Python/Python39/lib/site-packages/celery/worker/consumer/consumer.py:508: CPendingDeprecationWarning: The broker_connection_retry configuration setting will no longer determine whether broker connection retries are made during startup in Celery 6.0 and above. If you wish to retain the existing behavior for retrying connections on startup, you should set broker_connection_retry_on_startup to True.
warnings.warn(
[2024-08-13 15:52:05,130: ERROR/MainProcess] consumer: Cannot connect to amqp://guest:**@127.0.0.1:5672//: [WinError 10061] No connection could be made because the target machine actively refused it.
Trying again in 2.00 seconds... (1/100)
[2024-08-13 15:52:09,168: ERROR/MainProcess] consumer: Cannot connect to amqp://guest:**@127.0.0.1:5672//: [WinError 10061] No connection could be made because the target machine actively refused it.
Trying again in 4.00 seconds... (2/100)
[2024-08-13 15:52:15,252: ERROR/MainProcess] consumer: Cannot connect to amqp://guest:**@127.0.0.1:5672//: [WinError 10061] No connection could be made because the target machine actively refused it.
Trying again in 6.00 seconds... (3/100)
[2024-08-13 15:52:23,339: ERROR/MainProcess] consumer: Cannot connect to amqp://guest:**@127.0.0.1:5672//: [WinError 10061] No connection could be made because the target machine actively refused it.
Trying again in 8.00 seconds... (4/100)
[2024-08-13 15:52:33,452: ERROR/MainProcess] consumer: Cannot connect to amqp://guest:**@127.0.0.1:5672//: [WinError 10061] No connection could be made because the target machine actively refused it.
Trying again in 10.00 seconds... (5/100)

```

4. To correct this, we have to update our SETTINGS.PY

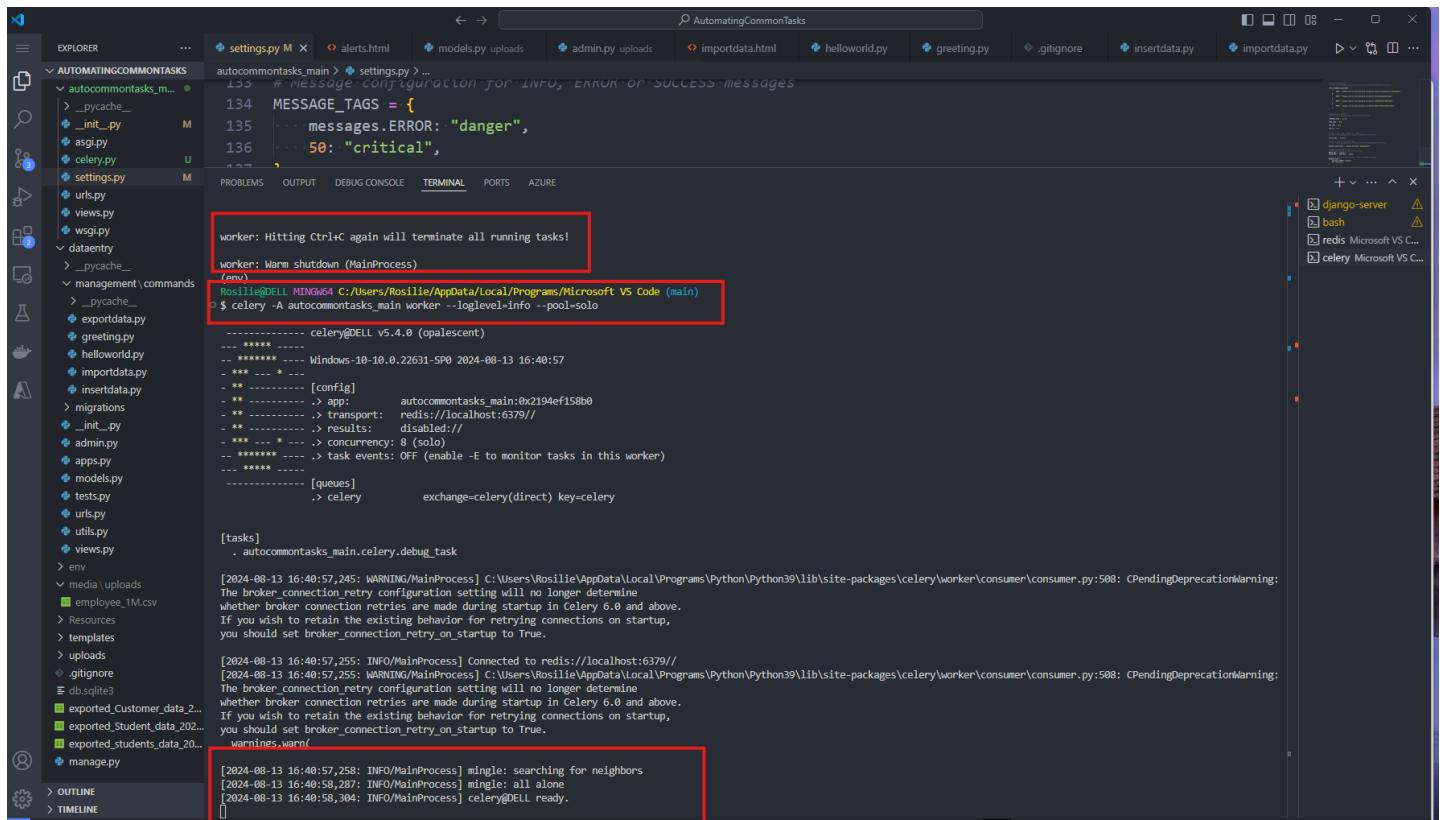


```

autocommontasks_main > settings.py > ...
133 # Message configuration for INFO, ERROR or SUCCESS messages
134 MESSAGE_TAGS = {
135     messages.ERROR: "danger",
136     50: "critical",
137 }
138
139 # Celery-related configuration; setting the message broker
140 CELERY_BROKER_URL = 'redis://localhost:6379'
141

```

In the bash terminal, press **CTRL + C** to terminate the process and try again. Your celery should be ready.



```

    worker: Hitting Ctrl+C again will terminate all running tasks!
    worker: Warm shutdown [MainProcess]
    celery@DELL MINGW64 ~ /Users/Rosilie/AppData/Local/Programs/Microsoft VS Code (main)
    $ celery -A autocommontasks_main worker --loglevel=info --pool=solo
    celery@DELL v5.4.0 (opalensemble)
    ****
    Windows-10-10.0.22631-SP0 2024-08-13 16:40:57
    *** * ...
    ** [config]
    ** > app: autocommontasks_main:0x2194ef158b0
    ** > transport: redis://localhost:6379//
    ** > results: disabled://
    ** * ... > concurrency: 8 (solo)
    ** * ... > task events: OFF (enable -E to monitor tasks in this worker)
    ** ...
    [queues]
    > celery exchange=celery(direct) key=celery

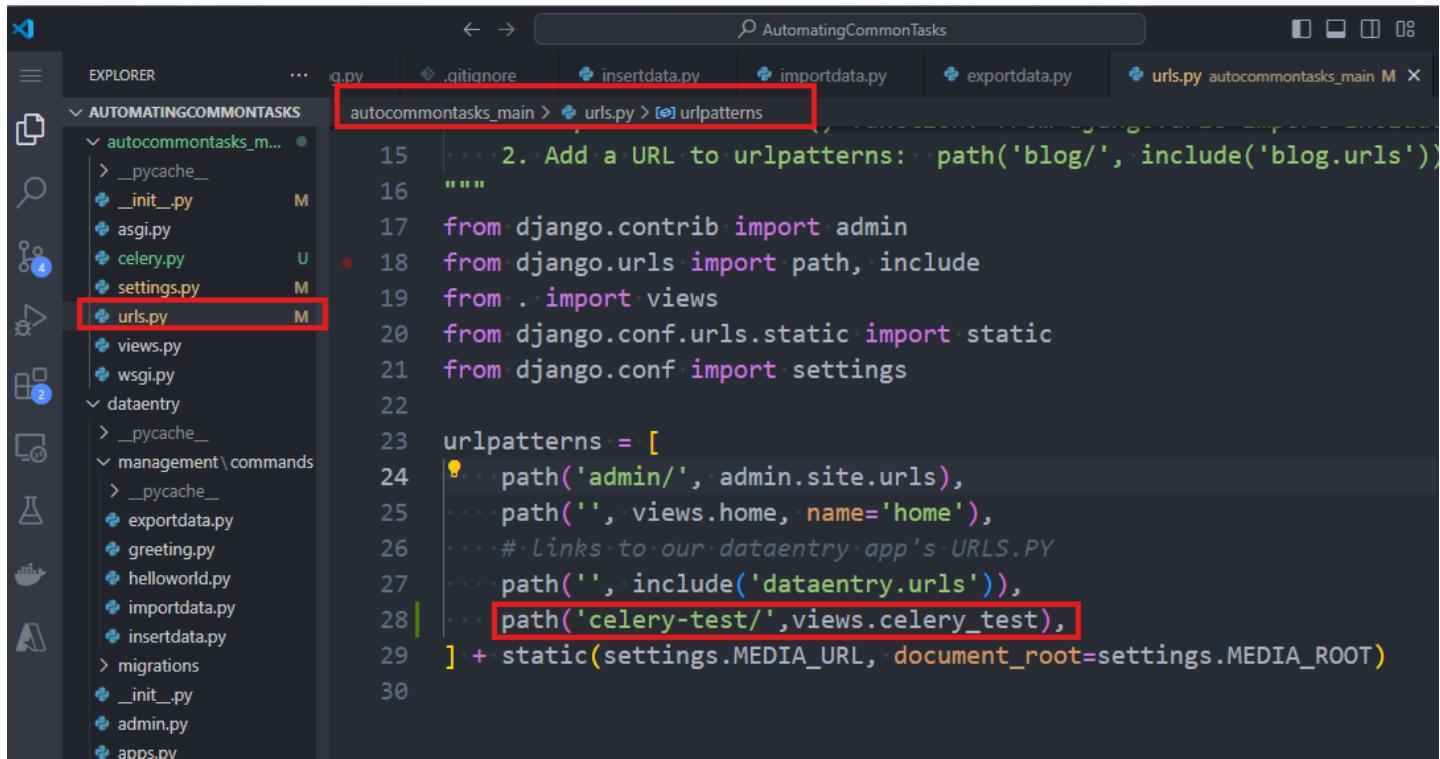
    [tasks]
    . autocommontasks_main.celery.debug_task

    [2024-08-13 16:40:57,245: WARNING/MainProcess] Connected to redis://localhost:6379//
    [2024-08-13 16:40:57,255: WARNING/MainProcess] C:\Users\Rosilie\AppData\Local\Programs\Python\Python39\lib\site-packages\celery\worker\consumer\consumer.py:508: CPendingDeprecationWarning:
    The broker_connection_retry configuration setting will no longer determine
    whether broker connection retries are made during startup in Celery 6.0 and above.
    If you wish to retain the existing behavior for retrying connections on startup,
    you should set broker_connection_retry_on_startup to True.

    [2024-08-13 16:40:57,255: INFO/MainProcess] Connected to redis://localhost:6379//
    [2024-08-13 16:40:58,207: INFO/MainProcess] C:\Users\Rosilie\AppData\Local\Programs\Python\Python39\lib\site-packages\celery\worker\consumer\consumer.py:508: CPendingDeprecationWarning:
    The broker_connection_retry configuration setting will no longer determine
    whether broker connection retries are made during startup in Celery 6.0 and above.
    If you wish to retain the existing behavior for retrying connections on startup,
    you should set broker_connection_retry_on_startup to True.

    [2024-08-13 16:40:57,258: INFO/MainProcess] mingle: searching for neighbors
    [2024-08-13 16:40:58,207: INFO/MainProcess] mingle: all alone
    [2024-08-13 16:40:58,304: INFO/MainProcess] celery@DELL ready.
  
```

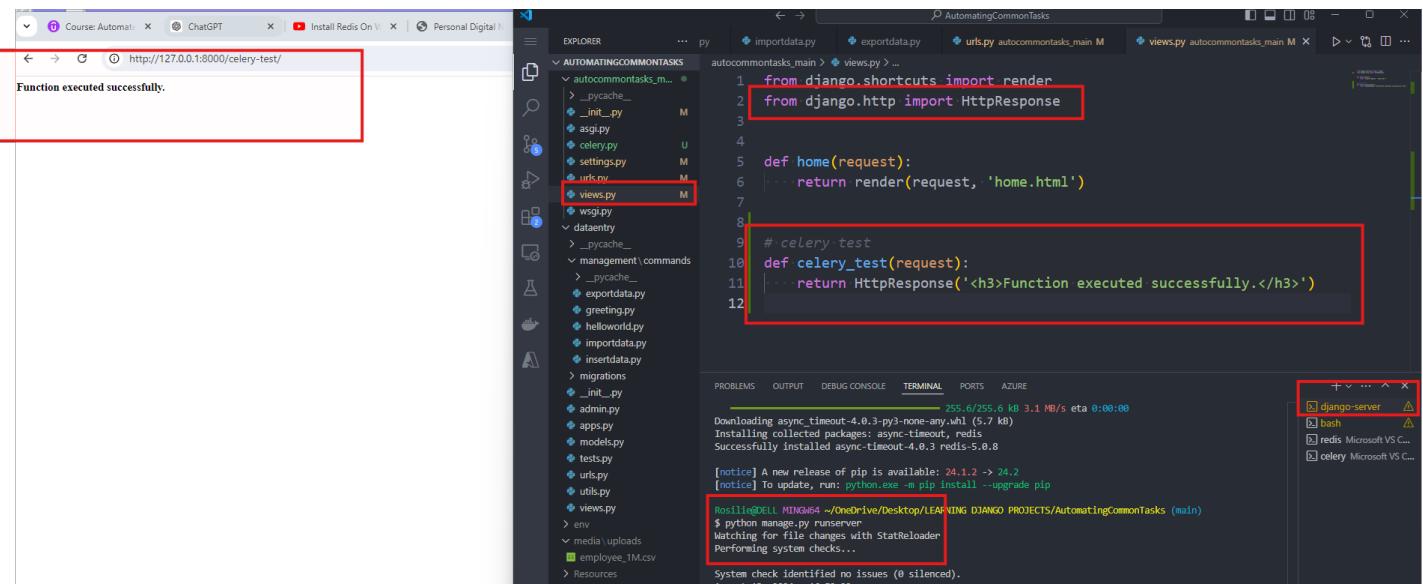
5. To use CELERY for testing in our project, create a new URL in our root project's URLs.PY



```

    autocommontasks_main > urls.py > urlpatterns
    15 2. Add a URL to urlpatterns: path('blog/', include('blog.urls'))
    16 """
    17 from django.contrib import admin
    18 from django.urls import path, include
    19 from . import views
    20 from django.conf.urls.static import static
    21 from django.conf import settings
    22
    23 urlpatterns = [
    24     path('admin/', admin.site.urls),
    25     path('', views.home, name='home'),
    26     # Links to our dataentry app's URLs.PY
    27     path('', include('dataentry.urls')),
    28     path('celery-test/', views.celery_test),
    29 ] + static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
    30
  
```

We run our django server and att the url in our browser: <http://127.0.0.1:8000/celery-test/>



```

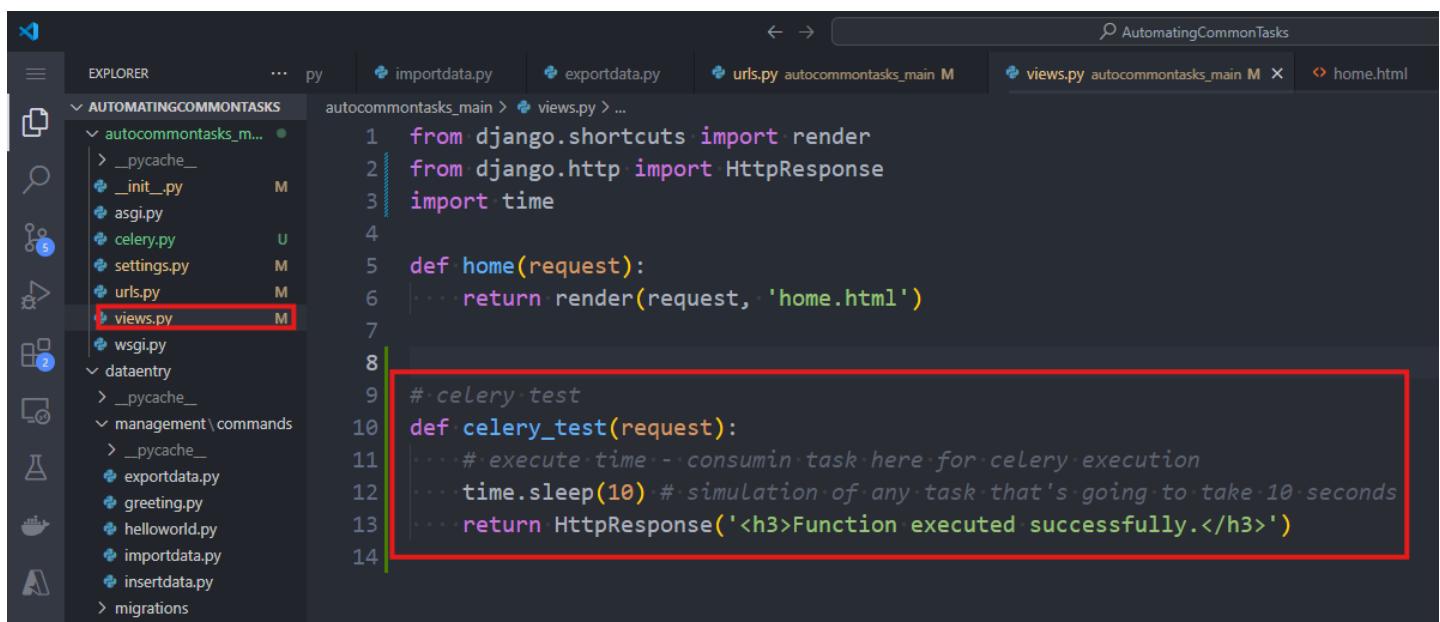
from django.shortcuts import render
from django.http import HttpResponse

def home(request):
    return render(request, 'home.html')

# celery test
def celery_test(request):
    return HttpResponse('<h3>Function executed successfully.</h3>')

```

6. Now update our CELERY_TEST FUNCTION to perform time-consuming operations.



```

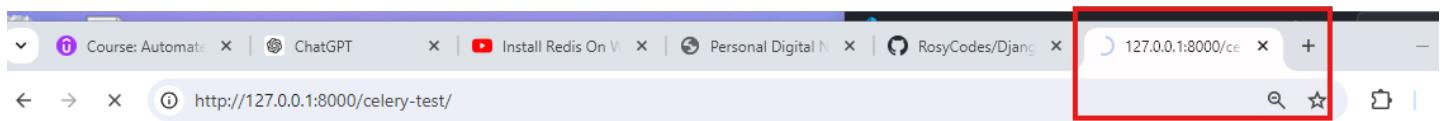
from django.shortcuts import render
from django.http import HttpResponse
import time

def home(request):
    return render(request, 'home.html')

# celery test
def celery_test(request):
    # execute time-consuming task here for celery execution
    time.sleep(10) # simulation of any task that's going to take 10 seconds
    return HttpResponse('<h3>Function executed successfully.</h3>')

```

So, when you reload your page, in the background, it will load 10 seconds (Sleep)



127.0.0.1:8000/celery-test/

Function executed successfully.

7. We want to pass then the SLEEP(10) TASK to celery, so that our project or user can do or see other things while Celery is working on something else. To do this, in the DATAENTRY folder, create a new file, TASKS.PY and update as

The screenshot shows the VS Code interface with the following details:

- EXPLORER:** Shows the project structure for "AUTOMATINGCOMMONTASKS" with files like `autocommontasks_main.py`, `celery.py`, `dataentry`, `settings.py`, and `tasks.py`.
- CODE EDITOR:** The `tasks.py` file is open, containing the following code:


```
1 # from the project folder in CELERY.PY, we load the variable APP
2 from autocommontasks_main.celery import app
3 import time
4
5
6 @app.task # uses a decorator as a celery task
7 def celery_test_task():
8     time.sleep(10) # simulation of any task that's going to take 10s
9     return 'Task executed successfully.'
10
```

 A red box highlights the task definition.
- TERMINAL:** Shows the output of a celery worker:


```
[2024-08-13 17:19:13,983: INFO/MainProcess] Connected to redis://localhost:6379/
[2024-08-13 17:19:13,984: WARNING/MainProcess] C:\Users\Rosilie\AppData\Local\Programs\Python\Python39\lib
\site-packages\celery\worker\consumer\consumer.py:508: PendingDeprecationWarning: The broker_connection_r
etry configuration setting will no longer determine
whether broker connection retries are made during startup in Celery 6.0 and above.
If you wish to retain the existing behavior for retrying connections on startup,
you should set broker_connection_retry_on_startup to True.
warnings.warn(
[2024-08-13 17:19:13,987: INFO/MainProcess] mingle: searching for neighbors
[2024-08-13 17:19:15,005: INFO/MainProcess] mingle: all alone
[2024-08-13 17:19:15,028: INFO/MainProcess] celery@DELL ready.
[2024-08-13 17:19:29,272: INFO/MainProcess] Task dataentry.tasks.celery_test_task[4eaac948-96de-4f37-9249-
3cb65f25cc31] received
[2024-08-13 17:19:39,285: INFO/MainProcess] Task dataentry.tasks.celery_test_task[4eaac948-96de-4f37-9249-
3cb65f25cc31] succeeded in 10.01600000000326s: 'Task executed successfully.'
```

 A red box highlights the success message.

8. To test all this, LOAD your URL `http://127.0.0.1:8000/celery-test/` again, click on your CELERY TERMINAL, and it should RECEIVE and DISPLAY the message 'TASK EXECUTED SUCCESSFULLY' while we see our webpage with H3 tag showing the message right away.

The screenshot shows the VS Code interface with the following details:

- EXPLORER:** Shows the project structure for "AUTOMATINGCOMMONTASKS" with files like `autocommontasks_main.py`, `celery.py`, `dataentry`, `settings.py`, and `views.py`.
- CODE EDITOR:** The `views.py` file is open, containing the following code:


```
1 from django.shortcuts import render
2 from django.http import HttpResponse
3 # imports the function from the TASKS.PY of DATAENTRY folder
4 from dataentry.tasks import celery_test_task
```

 A red box highlights the import statement.
- CODE EDITOR:** The `views.py` file continues with:


```
5
6
7 def home(request):
8     return render(request, 'home.html')
9
10
11 # celery test
12 def celery_test(request):
13     # execute time-consuming task here for celery execution
14     celery_test_task.delay() # calls the celery_test_task function and
15     # execute this asynchronously or in the background or not at the
16     # but we get the H3 message displayed right away.
17     return HttpResponse('<h3>Function executed successfully.</h3>')
18
```

 A red box highlights the celery task call.
- TERMINAL:** Shows the output of a celery worker and a browser request:


```
attempted on something that is not a socket.
(env)
Rosilie@DELL MINGW64 C:/Users/Rosilie/AppData/Local/Programs/Python/Python39/Scripts (main)
$ netstat -an | findstr :6379
  TCP [::]:6379               [::]:*                LISTENING      38012
  TCP [::]:6379               [::]:65497          ESTABLISHED   38012
  TCP [::]:6379               [::]:65498          ESTABLISHED   38012
  TCP [::]:65497              [::]:6379          ESTABLISHED   16768
  TCP [::]:65498              [::]:6379          ESTABLISHED   16768
(env)
Rosilie@DELL MINGW64 C:/Users/Rosilie/AppData/Local/Programs/Python/Python39/Scripts (main)
$ redis-cli ping
PONG
(env)
Rosilie@DELL MINGW64 C:/Users/Rosilie/AppData/Local/Programs/Python/Python39/Scripts (main)
$ curl -s http://127.0.0.1:8000/celery-test/ | grep 'Function executed successfully.'
<h3>Function executed successfully.</h3>
```

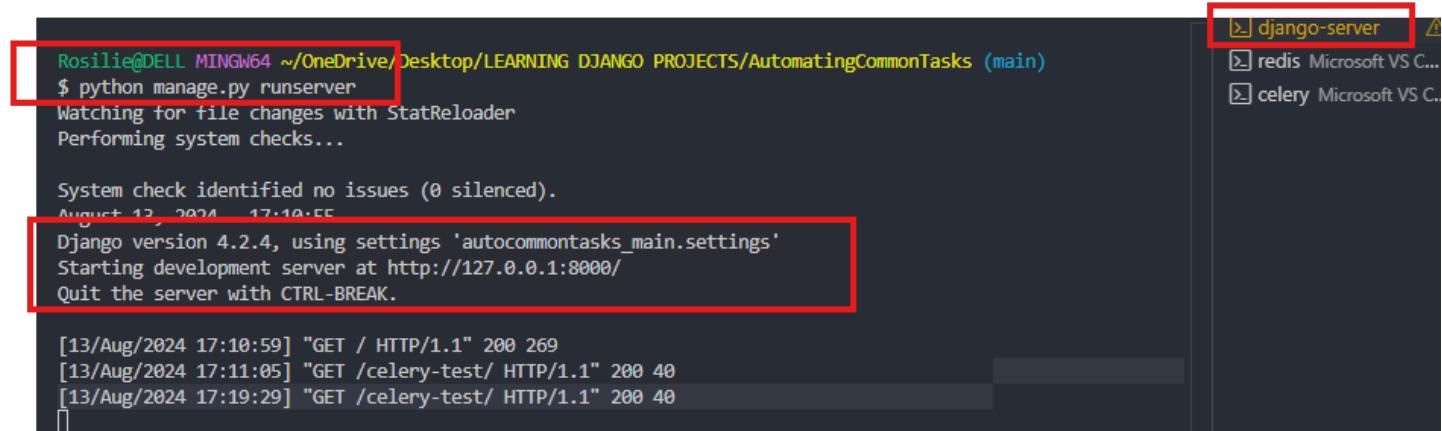
 A red box highlights the browser response.

IMPORTANT REMINDER:

So, when you run your django project with celery and redis, THERE SHOULD BE 3 BASH TERMINALS AND YOU NEED TO NAME THEM APPROPRIATELY:

1. DJANGO-SERVER- this is where you will run your python server to run your Django project:

```
$ python manage.py runserver
```



```
Rosilie@DELL MINGW64 ~/OneDrive/Desktop/LEARNING DJANGO PROJECTS/AutomatingCommonTasks (main)
$ python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
August 13, 2024 - 17:10:55
Django version 4.2.4, using settings 'autocommonTasks_main.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.

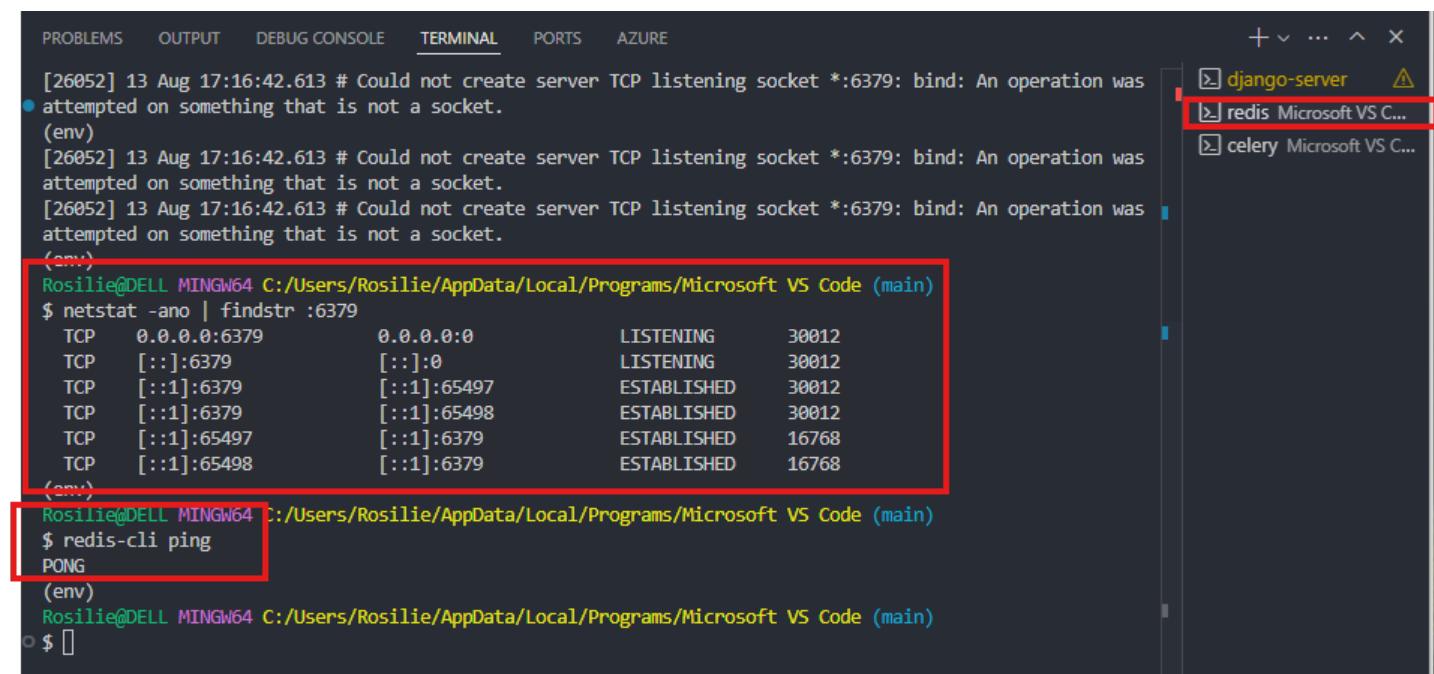
[13/Aug/2024 17:10:59] "GET / HTTP/1.1" 200 269
[13/Aug/2024 17:11:05] "GET /celery-test/ HTTP/1.1" 200 40
[13/Aug/2024 17:19:29] "GET /celery-test/ HTTP/1.1" 200 40
```

2. REDIS - this is where you run your REDIS-SERVER. Every time you start this, it will create a new process of REDIS, so you can simply kill it or ignore it. As long as when you PING your REDIS SERVER, it returns a PONG message, then you are fine. You have multiple processes (TCP) here because you have started your REDIS-SERVER multiple times.

```
$ redis-server
```

```
$ netstat -ano | findstr :6379 (this is to see what process is listening to the port 6379 (Redis)
```

```
$ redis-cli ping (Run this before you run the REDIS-SERVER so you wont have several TCPs)
```



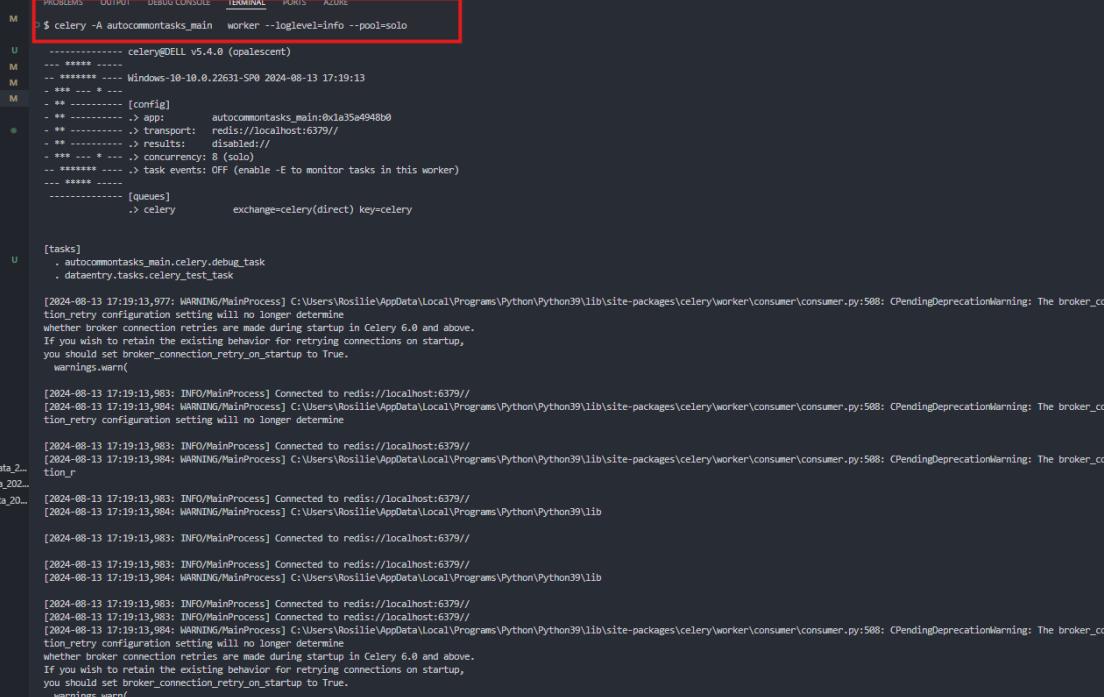
```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS AZURE + - x

[26052] 13 Aug 17:16:42.613 # Could not create server TCP listening socket *:6379: bind: An operation was attempted on something that is not a socket.
● (env)
[26052] 13 Aug 17:16:42.613 # Could not create server TCP listening socket *:6379: bind: An operation was attempted on something that is not a socket.
[26052] 13 Aug 17:16:42.613 # Could not create server TCP listening socket *:6379: bind: An operation was attempted on something that is not a socket.
● (env)
Rosilie@DELL MINGW64 C:/Users/Rosilie/AppData/Local/Programs/Microsoft VS Code (main)
$ netstat -ano | findstr :6379
  TCP  0.0.0.0:6379      0.0.0.0:0          LISTENING      30012
  TCP  [::]:6379        [::]:0            LISTENING      30012
  TCP  [::1]:6379       [::1]:65497        ESTABLISHED    30012
  TCP  [::1]:6379       [::1]:65498        ESTABLISHED    30012
  TCP  [::1]:65497      [::1]:6379        ESTABLISHED    16768
  TCP  [::1]:65498      [::1]:6379        ESTABLISHED    16768
● (env)
Rosilie@DELL MINGW64 C:/Users/Rosilie/AppData/Local/Programs/Microsoft VS Code (main)
$ redis-cli ping
PONG
● (env)
Rosilie@DELL MINGW64 C:/Users/Rosilie/AppData/Local/Programs/Microsoft VS Code (main)
$ 
```

3. CELERY- this is where you will run this code whenever you need to launch a new task or when you reload your Django page. Meaning, you have to launch this command again if you have made new changes to your CELERY TASKS like changing 10 seconds to 5 seconds to see the effect.

*autocommonTasks_main - this is your Django Project Name

```
$ celery -A autocommonTasks_main worker --loglevel=info --pool=solo
```



```
$ celery -A autocommontasks_main worker --logLevel=info --pool=solo
...
[2024-08-13 17:19:13,987: INFO/MainProcess] Connected to redis://localhost:6379/
[2024-08-13 17:19:13,988: INFO/MainProcess] mingle: searching for neighbors
[2024-08-13 17:19:15,005: INFO/MainProcess] mingle: all alone
[2024-08-13 17:19:15,028: celery@DELL ready.
[2024-08-13 17:19:29,272: INFO/MainProcess] Task dataentry.tasks.celery_test_task[4eaaac948-96de-4f37-9249-3cb65f25cc31] received
[2024-08-13 17:19:39,285: INFO/MainProcess] Task dataentry.tasks.celery_test_task[4eaaac948-96de-4f37-9249-3cb65f25cc31] succeeded in 10.01600000000326s: 'Task executed successfully.'
```

Copyright © Personal Digital Notebooks | By Rosilie | Date Printed: Feb. 4, 2026, 11:29 a.m.