Integración de Python con Camunda

disha_umarwani **Sep '18**

Estoy empezando a usar Camunda y soy un desarrollador de Python. ¿Algún recurso para integrar Camunda con scripts externos de Python?

¡Gracias!

---

thorben **Camunda Developer** **Sep '18**

Hola **@disha_umarwani** ,

En general, hay dos enfoques para usar el código Python en un proceso:

1. **Tarea externa** trabajadores que se implementan en Python y que se ejecutan independientemente del proceso de Camunda. No conozco un ejemplo de Python, pero **este artículo** describe muy bien el principio general y, con un cliente REST adecuado, debe comenzar de inmediato.
2. **Tareas de script** integradas en el modelo de proceso que se ejecutan a través de Jython desde el proceso de Camunda. Puede editarlos en Camunda Modeler, solo configúrelos `python`como el idioma de la descripción. Además, asegúrese de que el motor de secuencias de comandos Jython esté en el classpath de Camunda.

En general, recomiendo el enfoque 1, ya que debería ser más fácil de usar y le brinda total flexibilidad sobre cómo escribir y construir su código Python. El Enfoque 2 podría ser adecuado para secuencias de comandos y lógica bastante simples.

Saludos,
Thorben

Nele **Sep '18**

Hola Disha,
recientemente trabajé en un trabajador externo de Python. Aquí están mis fragmentos de código. Puede que no sea perfecto, pero está funcionando y podría ayudarte.

```
import requests
import json
import time

#Define API Endpoint
url = 'http://localhost:8080/engine-rest/external-task/fetchAndLock'
```

```
# Define Json
task= {"workerId":"XXX",
  "maxTasks":2,
  "usePriority":"true",
  "topics":
      [{"topicName": "coolStuff",
      "lockDuration": 10000,
      "variables": ["orderId"]
      }]
}

# Connect to the API (Fetch and Lock)
try:
    res = requests.post(url, json=task)
    print(res.status_code)

#Get Body Text

    body = res.text
    print(body)

except:
    print('Engine is down')


while body == '[]':
    res = requests.post(url, json=task)
    time.sleep(5)

    if body !='[]':
        break

#here goes your business logic!

# Connect to the API (Complete)
response = json.loads(body)
x = response[0]['id']


#convert into a string
x = str(x)

complete_url = 'http://localhost:8080/engine-rest/external-task/'+x+'/complete



response= {
  "workerId": "XXX",
  "variables":
```

```
      {"aVariable": {"value": "aStringValue"},
      "anotherVariable": {"value": 42},
      "aThirdVariable": {"value": "true"}}
}

complete = requests.post(complete_url, json=response)
```

Daño **Oct '18**

Hola Nele, gracias por tu código. Funciona bien, pero tengo un problema. Lo que esperaría es lo siguiente:

1. el programa comienza a sondear para que algo suceda (lo hace)
2. entra en acción en el momento en que comienzo un proceso en Camunda que tiene el token de este trabajador.

Pero es al revés: el trabajador solo recoge el proceso de Camunda cuando comienzo al trabajador después de comenzar el proceso para ganar Camunda. Esto es al revés de lo que esperaría.

¿Sabes lo que podría estar haciendo mal?

Aquí está mi código (en realidad su código):

```
import requests
import json
import time

# Define API Endpoint
url = 'http://localhost:8080/engine-rest/external-task/fetchAndLock'

# Define Json
task = {"workerId": "XXX",
        "maxTasks": 10,
        "usePriority": "true",
        "topics":
            [{"topicName": "loadData",
              "lockDuration": 10000,
              "variables": ["orderId"]
              }]
        }
body = []
# Connect to the API (Fetch and Lock)
try:
    res = requests.post(url, json=task)
    print('res.status_code', res.status_code)
    # Get Body Text
    body = res.text
```

```
    print('res.text', body)
except:
    print('Engine is down')

while body == '[]':
    print('wait')
    time.sleep(5)
    res = requests.post(url, json=task)
    print('result', res)
    if body != '[]':
        break

# here goes your business logic!
print("Connection succeeded")

# Connect to the API (Complete)
response = json.loads(body)
x = response[0]['id']

print('id', x)

# convert into a string
x = str(x)

complete_url = 'http://localhost:8080/engine-rest/external-task/' + x + '/comp

response = {
    "workerId": "XXX",
    "variables":
        {"aVariable": {"value": "aStringValue"},
         "anotherVariable": {"value": 42},
         "aThirdVariable": {"value": "true"}}
}

print('response', response)

complete = requests.post(complete_url, json=response)

print('complete', complete)
```

Nele **Oct '18**

Hola **@Harm**
Hubo un error en mi fragmento. Lo siento por eso.

El cuerpo variable no se asignó nuevamente en el ciclo while. Por lo tanto, el cuerpo se mantiene como está desde la primera llamada de descanso.

Debe cambiar el ciclo while a lo siguiente:

```
while body =='[ ]':
   res = requests.post(url, json =task)
   body = res.text
   time.sleep(5)

    if body !='[ ]':
       break
```

Eso debería hacer el truco.

De todos modos, el trabajador terminará tan pronto como haya obtenido, bloqueado y completado una Tarea.
Por lo tanto, es posible que deba cambiarlo si desea que el trabajador comience nuevamente después de obtener una tarea🙂

Saludos Nele

Daño **Oct '18**

Hola nele

Gracias, ya lo tengo funcionando.

Saludos,
Daño

Daño **Oct '18**

Hola disha

Este es un ejemplo de un script que ejecuta un script de Python y devuelve el valor de retorno que se evaluará en el proceso. De esta manera, puede ejecutar Python desde un paso del proceso Camunda.

```
def pmdCommand = '/usr/local/bin/python3.7 /users/user/Documents/FTKDemo/DEXRS

def sout = new StringBuffer()
def serr = new StringBuffer()

def process = pmdCommand.execute()
process.consumeProcessOutput(sout, serr)
process.waitForProcessOutput()
execution.setVariable("loadReturnValue", process.exitValue())
```

---

olekslev **Nov '18**

> Daño:
>
> Este es un ejemplo de un script que ejecuta un script de Python y devuelve el valor de retorno que se evaluará en el proceso. De esta manera, puede ejecutar Python desde un paso del proceso Camunda.

Gracias por compartir algunos ejemplos de Python. ¡Es exactamente lo que estoy buscando ahora mismo!

Daño **Nov '18**

¡El gusto es mio!