

Python - Cridant una API II

Un cop completat l'exercici [Python - Cridant una API](#) aprendrem a tractar el resultat de la crida.

Per a l'exercici esmentat el resultat a tractar és la confirmació que l'event s'ha creat:

```
{  
  "created": true  
}
```

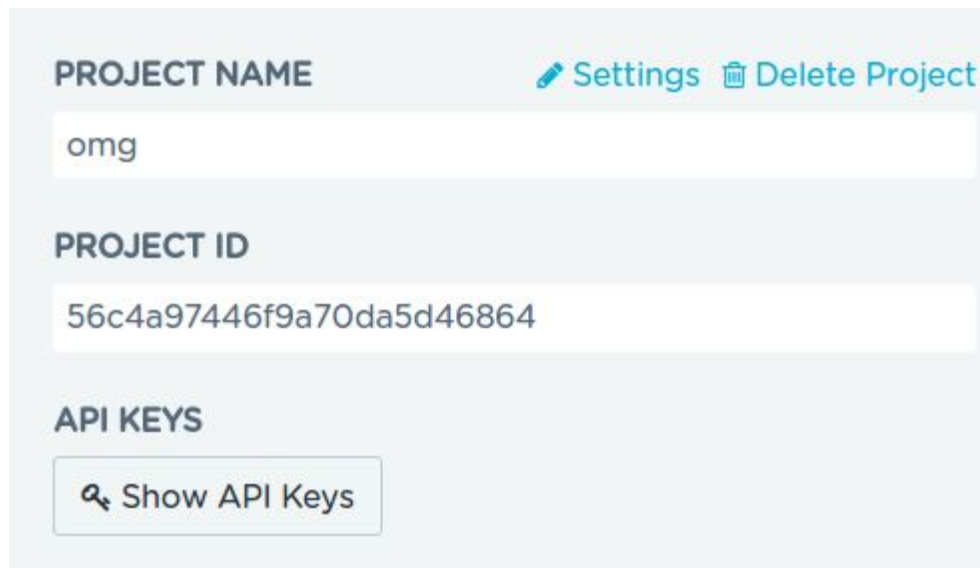
A l'exercici anterior pintavem aquest resultat com una cadena de caràcters, però hem d'aprendre a representar-ho amb les estructures del propi llenguatge.

Exercici

- Llegeix i compren [JSON encoder and decoder](#) l'apartat *Decoding JSON*, és tan fàcil com sembla. Fixat en que el resultat no és un string sino una llista amb un string i un diccionari que, a la seva vegada, conté una llista com a valor.
- Decodifica el json que t'envia *keen.io*. I accedeix al valor de 'created'
- Apren a distingir `stream` de `string`. Fixat que la resposta la rebem en un *stream* i que python pot llegir-lo sense haver de carregar-lo en un *string*, mitjançant el mètode `load` (en compte de *loads*).
- Fes crides a l'API per obtenir [recomptes](#), sumes, mitjanes i mostra els seus valors.
Exemple https://api.keen.io/3.0/projects/PROJECT_ID/queries/count?api_key=REAL_KEY&event_collection=COLLECTION_NAME&group_by=vista
- Fes una crida complexa i recorre el resultat. Per exemple, [request with interval](#)
- Fes una crida a google maps per aconseguir el codi postal d'una adreça. Mira el codi amb php d'exemple.

Realització de la pràctica

Creem el nostre projecte a keen.io:



The screenshot shows the Keen IO project creation interface. It has a light blue header with 'PROJECT NAME' on the left and 'Settings' (with a pencil icon) and 'Delete Project' (with a trash icon) on the right. Below the header, there is a text input field containing 'omg'. Underneath that is the 'PROJECT ID' section, which displays '56c4a97446f9a70da5d46864'. At the bottom, there is an 'API KEYS' section with a button that says 'Show API Keys' (with a magnifying glass icon).

A la vista on vulguem utilitzar el recompte, introduïrem el codi següent, el qual s'explica a continuació:

```
models.py    x    urls.py    x    views.py    +
39
40     # -- KEEN IO -- #
41
42     url = 'https://api.keen.io/3.0/projects/56c4a97446f9a70da5d46864/e
43
44     values_dict = {
45         "Accio": "Afegir/Editar moviment",
46         "Element": moviment.element.nom,
47         "Quantitat": moviment.quantitat,
48     }
49     values = json.dumps( values_dict )
50     headers = { 'Content-Type': 'application/json' }
51     req = urllib2.Request(url, values, headers)
52     response = urllib2.urlopen(req)
53     print response.read()
54
55     # -- END KEEN IO -- #
56     return HttpResponseRedirect(reverse('inventari:mostramov'))
```

La url, és la url del nostre projecte que té la ID del projecte, la api key, y la write key. Ja que ens interessa enviarle un event per json.

Cal dir que és necessari importar:

```
12 import json
13 import urllib2
```

El següent valor, values_dict és el diccionari de dades que utilitzarem en forma de clau->valor. En el nostre cas volem saber que l'usuari esta afegint o editant un moviment, i saber quin és, i quina quantitat té.

Les següents línees serveixen per poder enviar correctament el json al keen.io. On es processas el json, es defineixen els headers i s'envia el request.

D'aquesta manera, cada vegada que un usuari crei o editi un moviment a la nostra aplicació, en podrem guardar un registre, tal i com es veu aquí:

EVENT EXPLORER		Refresh	Delete Collection
test ▼			
Last 10 Events		Event Properties	
#	Created At		
1	2016-02-19 @ 19:19:51 +0100		
2	2016-02-19 @ 19:04:53 +0100		

Ara, si volem notificar a l'usuari de que l'event s'ha enviat correctament a Keen.io, haurem d'afegir-hi el codi següent:

- Moviment importat correctament
- Event enviat a Keen.io correctament

```

resposta = json.loads(response.read())
print type ( resposta )
creat = resposta.get( 'created', False )
if creat :
    messages.success(request, 'Event enviat a Keen.io correctament')
else:
    messages.error(request, 'Error enviant event a Keen.io')

# -- END KEEN IO -- #

```

Agafem la resposta del keen a la variable resposta, agafem la clau “created”, i si és true, ens imprimirà per pantalla “Event enviat...”.

Si volem ensenyar els resultats a una vista propia, la creem i utilitzarem un altre template també. Com per exemple:

Vista:

```

@login_required
def resum(request):

    url = 'https://api.keen.io/3.0/projects/56c4a97446f9a70da5d46864/queries/count?api_key='
    req = urllib2.Request(url)
    response = urllib2.urlopen(req)
    resposta = json.load(response)
    return render (request, 'inventari/grafics.html', {'dades': resposta['result']})

```

Template:

```

1  {% extends 'base.html' %}
2
3  {% block title %}Gràfics{% endblock %}
4
5  {% block content %}
6  |
7  {%for r in dades%}
8
9  <table class="table">
10     <tr>
11         <th>Element</th>
12         <th>Quantitat afegida</th>
13     </tr>
14     <tr>
15         <td>{{r.Element}}</td>
16         <td>{{r.result}}</td>
17     </tr>
18 </table>
19
20 {%endfor%}
21
22 {% endblock %}

```

Resultat:

Element	Quantitat afegida
Monitor	2

Element	Quantitat afegida
Portàtil	1

Element	Quantitat afegida
SO Windows	1