JSON

ADOLFO SANZ DE DIEGO

OCTUBRE 2017

1 ACERCA DE

1.1 AUTOR

- **Adolfo Sanz De Diego**
  - Blog: asanzdiego.blogspot.com.es
  - Correo: asanzdiego@gmail.com
  - GitHub: github.com/asanzdiego
  - Twitter: twitter.com/asanzdiego
  - LinkedIn: in/asanzdiego
  - SlideShare: slideshare.net/asanzdiego

1.2 LICENCIA

- **Copyright:**
  - Antonio Sarasa Cabezuelo <antoniosarasa@campusciff.net>

1.3 FUENTE

- Las slides y sus fuentes las podéis encontrar en:
  - https://github.com/asanzdiego/curso-intro-linux-web-sql-2016

2 INTRODUCCIÓN A JSON

2.1 ¿QUÉ ES?

- JSON (**JavaScript Object Notation**) es un formato de datos que se caracteriza:
  - Está basado en JavaScript.
  - Es utilizado para el intercambio de datos.
  - Es utilizado por muchas APIs de sitios web tales como Facebook, Twitter,... para devolver su contenido.
  - Es independiente del lenguaje
  - Los archivos tienen extensión .json

2.2 PAREJAS CLAVE=VALOR

- JSON representa objetos de manera textual mediante **parejas clave=valor**,

```
{
    "libro": [
        {
            "id": "01",
            "lenguaje": "Java",
            "edición": "tercera",
            "autor": "Herbert Schildt"
        },
        {
            "id": "07",
            "lenguaje": "C++",
            "edición": "seguna",
            "autor": "E.Balagurusamy"
        }
    ]
}
```

Ejemplo JSON

2.3 SINTAXIS JSON

- Un objeto se representa como una secuencia de parejas clave=valor encerradas entre llaves { y }.
- Las claves son cadenas de texto entre comillas " y ".

2.4 VALORES JSON

- Tipos básicos: **cadena, número, booleano, null**
- **Arrays** de valores: entre corchetes [y]
- Otros **objetos** JSON: entre llaves { y }

2.5 EJEMPLO JSON (I)

- Considerar el siguiente ejemplo dónde se quiere representar la ficha de un estudiante con sus datos personales y asignaturas matriculadas:
  - Nombre="Pepito Pérez"
  - DNI="517899R"
  - Edad="22"

2.6 EJEMPLO JSON (II)

- Asignaturas matriculadas:
  - Obligatorias: Sistemas Operativos, Compiladores, y Bases de Datos.
  - Optativas: Bases de Datos NoSQL, Minería de Datos, Programación Lógica.
  - Libre Elección: Ajedrez, Música Clásica

2.7 EJEMPLO JSON (III)

- La ficha de información se puede representar en un documento JSON de la siguiente manera:

```json
{
    "Nombre":"Pepito Pérez",
    "DNI": "5167778E",
    "Edad":22,
    "Asignaturas": {
        "Obligatorias":{
            "Sistemas Operativos",
            "Compiladores",
            "Bases de Datos"
        },
        "Optativas":{
            "Bases de Datos NoSQL",
            "Minería de Datos",
            "Programación Lógica"
        },
        "LibreElección":{
            "Ajedrez",
            "Música Clásica"
        }
    }
}
```

Ejemplo JSON

2.8 EJEMPLO API

- http://jsonplaceholder.typicode.com/posts
- http://jsonplaceholder.typicode.com/posts/1/comments

3 JSON DESDE PYTHON

3.1 MÓDULO JSON

- Para gestionar Documentos JSON desde Python se usa el **modulo JSON** que permite la traducción de datos JSON en valores de Python.

3.2 TIPOS PERMITIDOS

- JSON no puede almacenar cualquier tipo de valor Python, únicamente **cadenas, enteros, reales, booleanos, listas, diccionarios y el tipo None**.

3.3 TIPOS NO PERMITIDOS

- JSON no puede representar objetos específicos de Python tales como **Ficheros, expresiones regulares, ...**

3.4 JSON DESDE CADENA

- Para traducir una cadena que contiene datos JSON en un valor de Python se utiliza el **método json.loads()**.

```
JsonDatos='{"nombre": "Sofia", "matriculado":true, "asignaturas":34, "ID":null}'
import json
PythonDatos=json.loads(JsonDatos)
print PythonDatos
```

Cargar JSON

3.5 DICCIONARIO

- La llamada al método loads() del módulo json permite cargar una cadena de datos JSON en valores de Python, retornando como **resultado un diccionario**.
- Si se quiere acceder a los distintos elementos del diccionario **se usan los índices**. La cadena JSON utiliza dobles comillas para las claves.

3.6 SIN ORDEN

- Observar que **los valores en los diccionarios no están ordenados**, por lo que los pares clave-valor pueden aparecer en orden diferente a como aparecían en la cadena original.

3.7 ESCRIBIR JSON

- Para escribir un valor de Python como una cadena de datos JSON se usa el método json.dumps().

```
PythonDatos={'nombre': 'Sofia', 'matriculado':True, 'asignaturas':34, 'ID':None}
import json
JSONDatos=json.dumps(PythonDatos)
print JSONDatos
```

Escribir JSON

3.8 EJEMPLOS DE PROCESAMIENTO

- Para comprender la utilidad y aplicación de los Documentos JSON se van a ver un par de ejemplos de procesamiento de documentos JSON desde el lenguaje de programación Python.

3.9 ENUNCIADO EJEMPLO 1

- Considerar un programa que permita:
  - Leer desde teclado una ciudad
  - Llamar a la API de geocodificación de Google
  - Extraer la información en formato JSON que nos devuelve.

3.10 RECOGER JSON

- Mediante urllib se recupera el texto en JSON que la API de geocodificación de Google devuelve.

```
import urllib
import json

serviceurl = 'http://maps.googleapis.com/maps/api/geocode/json?'

while True:
    address = raw_input('Entrar ciudad: ')
    if len(address) < 1 : break

    url = serviceurl + urllib.urlencode({'sensor':'false', 'address': address})
    print 'Recuperando', url
    uh = urllib.urlopen(url)
    data = uh.read()
    print 'Recuperados',len(data),'characters'
```

Ejemplo 1 - recoger JSON

3.11 PROCESAR JSON

- Una vez recuperados los datos JSON se analizan y se muestran.

```
try: js = json.loads(str(data))
except: js = None
if 'status' not in js or js['status'] != 'OK':
    print '==== Fallo de recuperación ===='
    print data
    continue

print json.dumps(js, indent=4)

lat = js["results"][0]["geometry"]["location"]["lat"]
lng = js["results"][0]["geometry"]["location"]["lng"]
print 'lat',lat,'lng',lng
location = js['results'][0]['formatted_address']
print location
```

Ejemplo 1 - procesar JSON

3.12 EJEMPLOS 1

```
import urllib
import json

serviceurl = 'http://maps.googleapis.com/maps/api/geocode/json?'

while True:
    address = raw_input('Entrar ciudad: ')
    if len(address) < 1 : break

    url = serviceurl + urllib.urlencode({'sensor':'false', 'address': address})
    print 'Recuperando', url
    uh = urllib.urlopen(url)
    data = uh.read()
    print 'Recuperados',len(data),'characters'

    try: js = json.loads(str(data))
    except: js = None
    if 'status' not in js or js['status'] != 'OK':
        print '==== Fallo de recuperación ===='
        print data
        continue

    print json.dumps(js, indent=4)

    lat = js["results"][0]["geometry"]["location"]["lat"]
    lng = js["results"][0]["geometry"]["location"]["lng"]
    print 'lat',lat,'lng',lng
    location = js['results'][0]['formatted_address']
    print location
```

Ejemplo 1 - completo

3.13 ENUNCIADO EJEMPLO 2

- Twitter tiene una disponible una API con servicios para los usuarios. Para poder utilizar dicha API es necesario el uso de firmas Oauth (es una tecnología para firmar peticiones en Internet) en cada solicitud.

3.14 CREAR CUENTA

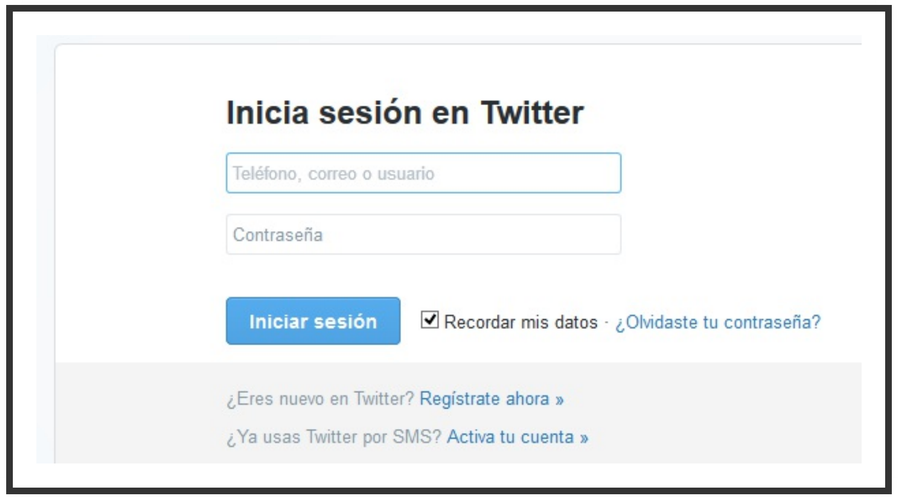
- Si no se dispone de cuenta en twitter, entonces hay que crearse una cuenta en https://twitter.com/

3.15 ACEDER A APPS

- A continuación se navega a la dirección https://apps.twitter.com/, dónde habrá que autenticarse. Esta página da acceso a la api de twitter para desarrolladores.

3.16 AUTENTICARSE

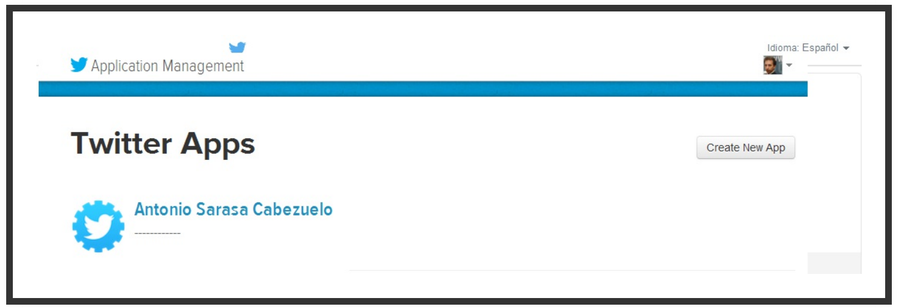
- Nos autenticamos:



Ejemplo 2 - autenticarse en twitter

3.17 DENTRO DE APPS

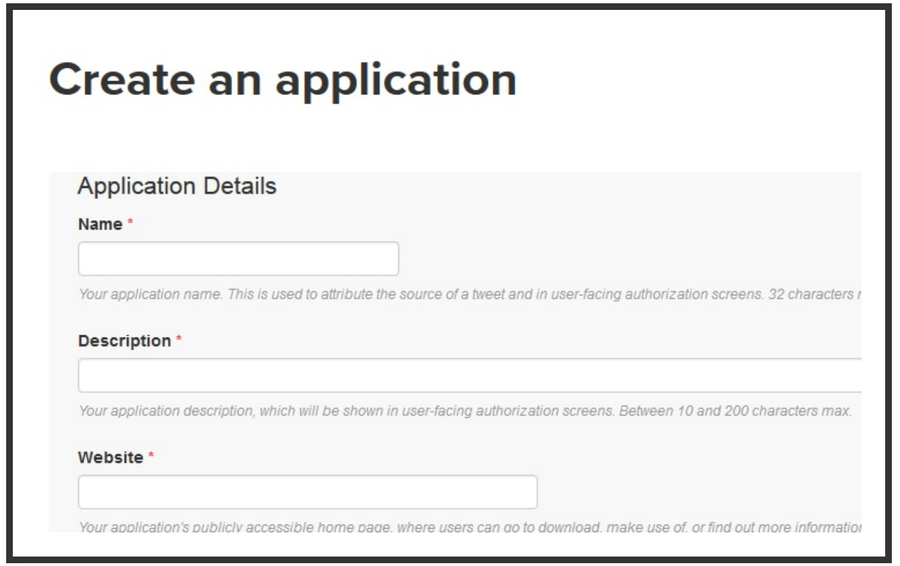
- Una vez autenticados, se entra en la página principal:



Ejemplo 2 - dentro de apps de twitter

3.18 CREAR NUEVA APP

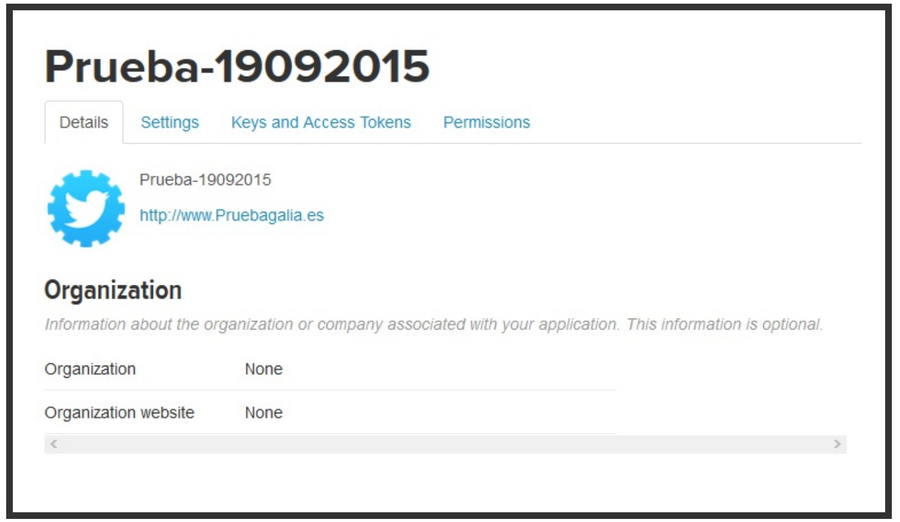
- Se selecciona "Create New App", lo que abre un formulario que hay que rellenar.



Ejemplo 2 - dentro de apps de twitter

3.19 INFORMACIÓN APP

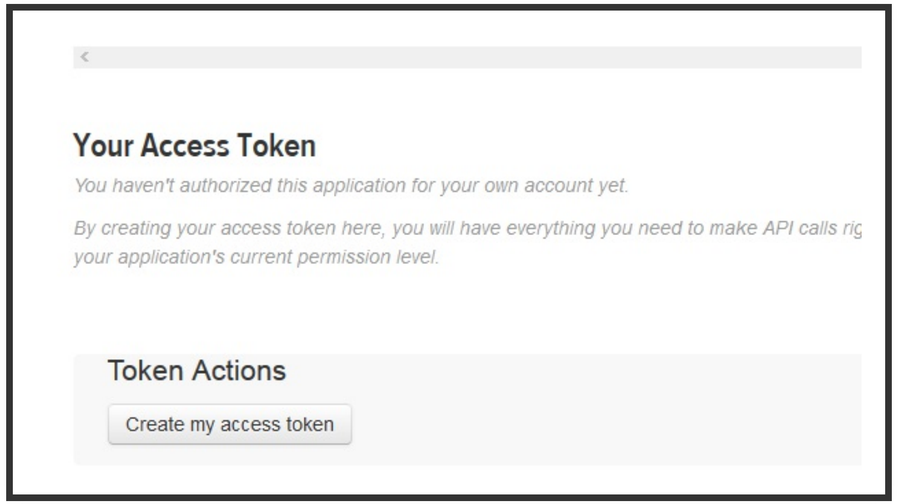
- Cuando se pulsa sobre "Create your Twitter application" aparece una nueva página con la información de la aplicación.



Ejemplo 2 - información de apps de twitter

3.20 CREAR ACCESS TOKEN

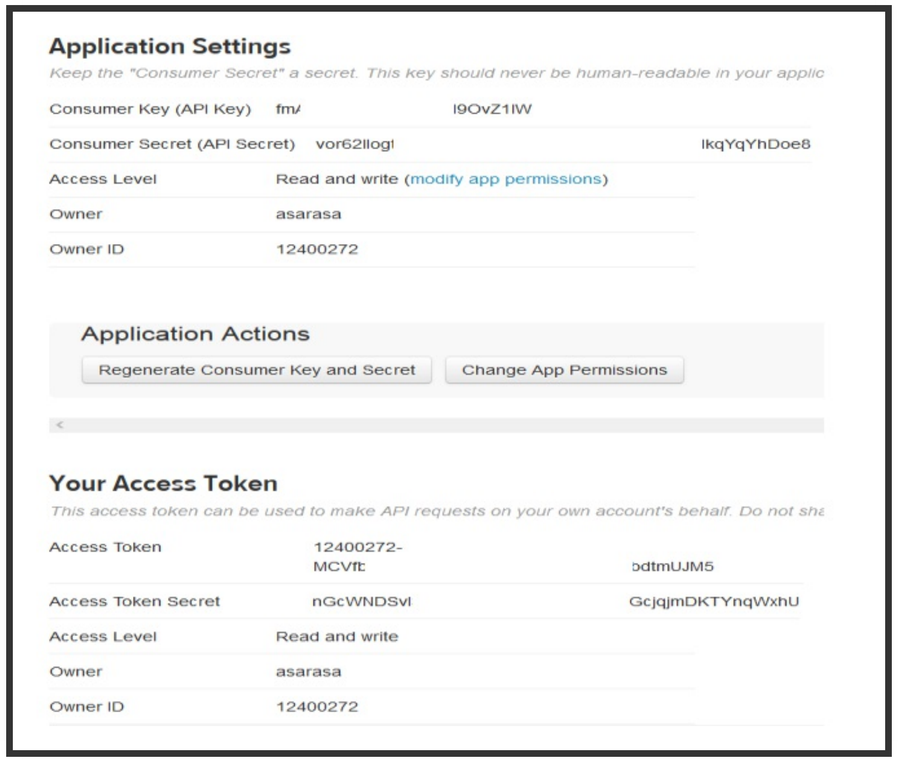
- Se pulsa sobre la solapa "Keys and Access content", y en dicha página se busca la sección "Your access token" dónde se pulsa sobre la opción "create my access token".



Ejemplo 2 - crear access token de twitter

3.21 VALORES APP

- Después de pulsar sobre la opción "create my access token" se han generado un conjunto de valores en la página.



Ejemplo 2 - valores app de twitter

3.22 VALORES IMPORTANTES

- Los valores que son necesarios utilizar para acceder a los datos de twitter son:
  - api_key = "Valor de la api key"
  - api_secret = "Valor de la api secret"
  - access_token_key = "Valor de la access token"
  - access_token_secret = "Valor de la access token secret"
- Una vez que se disponen de estos valores, se crea un programa en python para recuperar datos de Twitter.

3.23 PROGRAMA PYTHON

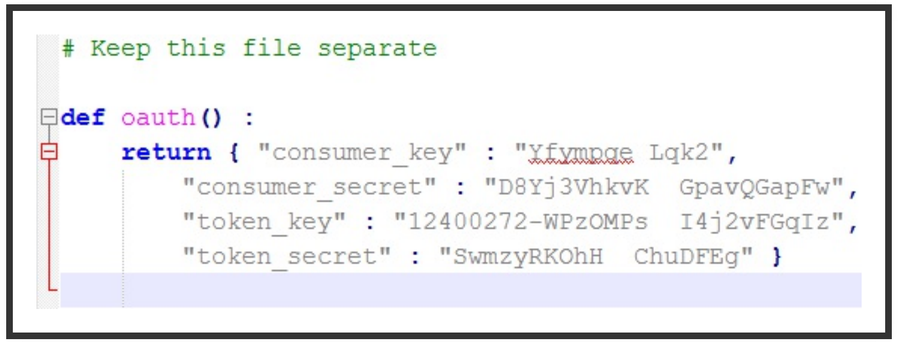
- Se van a necesitar 4 programas:
  - oauth.py
  - twurl.py
  - hidden.py
  - twitter.py

3.24 OAUTH.PY

- El programa oauth.py contiene una implementación del protocolo de firmas oauth.

3.25 HIDDEN.PY

- El programa hidden.py contiene los parámetros para acceder a la API de twitter que se han debido copiar de la app creada en la API en la sección de "Keys and Access content".



Ejemplo 2 - hidden.py

3.26 TWURL.PY

- El programa twurl.py es un programa auxiliar necesario para realizar la conexión.

```
import urllib
import oauth
import hidden

def augment(url, parameters) :
    secrets = hidden.oauth()
    consumer = oauth.OAuthConsumer(secrets['consumer_key'], secrets['consumer_secret'])
    token = oauth.OAuthToken(secrets['token_key'],secrets['token_secret'])

    oauth_request = oauth.OAuthRequest.from_consumer_and_token(consumer,
        token=token, http_method='GET', http_url=url, parameters=parameters)
    oauth_request.sign_request(oauth.OAuthSignatureMethod_HMAC_SHA1(), consumer, token)
    return oauth_request.to_url()
```

Ejemplo 2 - twurl.py

3.27 TWITTER.PY

- El programa twitter.py contiene el procesamiento buscado.
- En este caso se recuperan los amigos de un usuario en Twitter, se analiza el JSON devuelto y se extrae parte de la información sobre esos amigos.

3.28 OTROS DATOS

- También recupera información de las cabeceras de respuesta HTTP.
- En particular de x-rate-limit-remaining que informa sobre cuántas peticiones se pueden hacer antes de que ser bloqueados por un corto periodo de tiempo.

3.29 CÓDIGO TWITTER.PY

```
import urllib
import twurl
import json

TWITTER_URL = 'https://api.twitter.com/1.1/friends/list.j

while True:
    print ''
    acct = raw_input('Entrar cuenta de twitter:')
    if ( len(acct) < 1 ) : break
    url = twurl.augment(TWITTER_URL,
        {'screen_name': acct, 'count': '5'} )
    print 'Recuperando', url
    connection = urllib.urlopen(url)
    data = connection.read()
    headers = connection.info().dict
    print 'Restante', headers['x-rate-limit-remaining']
    js = json.loads(data)
    print json.dumps(js, indent=4)
```

Ejemplo 2 - twitter.py