

13 Python y servicios web

Uso de servicios web

Una vez que fue fácil recuperar documentos y analizarlos a través de HTTP usando programas, no tomó mucho tiempo desarrollar un enfoque en el que comenzó a producir documentos que fueron diseñados específicamente para ser consumidos por otros programas (es decir, no HTML para mostrar en un navegador).

Hay dos formatos comunes que utilizamos cuando intercambiamos datos a través de la web. El "lenguaje de marcado extensible" o XML ha estado en uso durante mucho tiempo y es el más adecuado para intercambiar datos de tipo documento. Cuando los programas solo desean intercambiar diccionarios, listas u otra información interna entre ellos, utilizan la notación de objetos JavaScript o JSON (consulte [www.json.org] (<http://www.json.org>)). Veremos ambos formatos.

Lenguaje de marcado extensible - XML

XML se ve muy similar a HTML, pero XML es más estructurado que HTML. Aquí hay una muestra de un documento XML:

```
<person>
  <name>Chuck</name>
  <phone type="intl">
    +1 734 303 4456
  </phone>
  <email hide="yes"/>
</person>
```

A menudo es útil pensar en un documento XML como una estructura de árbol donde hay una etiqueta superior "persona" y otras etiquetas como "teléfono" se crean como **hijos** de sus nodos padres.

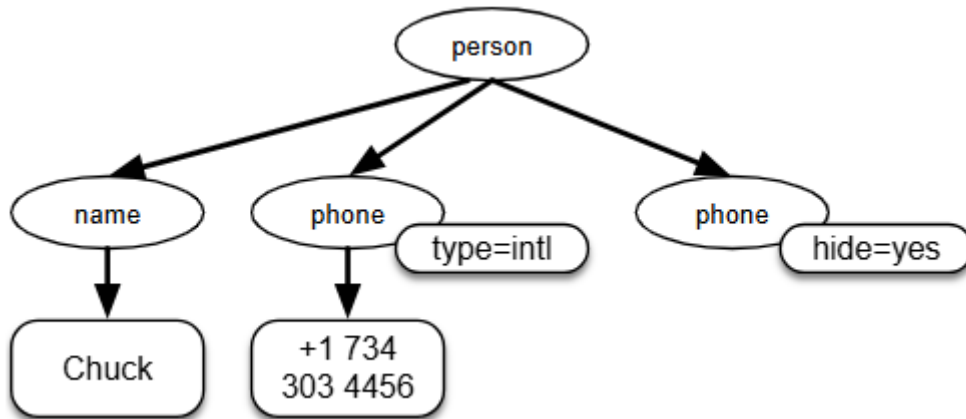


Imagen - Una representación en árbol de XML

Una representación en árbol de XML

Analizando XML

Aquí hay una aplicación simple que analiza algunos XML y extrae algunos elementos de datos del XML:

<https://trinket.io/embed/python3/b4dfccee06>

`fromstring` convierte la representación de cadena del XML en un "árbol" de nodos XML. Cuando el XML está en un árbol, tenemos una serie de métodos a los que podemos llamar para extraer partes de datos del XML.

La función `find` busca en el árbol XML y recupera un **nodo** que coincide con la etiqueta especificada. Cada nodo puede tener algún texto, algunos atributos (como ocultar) y algunos nodos "secundarios". Cada nodo puede ser la parte superior de un árbol de nodos.

Name: Chuck
Attr: yes

El uso de un analizador XML como `ElementTree` tiene la ventaja de que si bien el XML en este ejemplo es bastante simple, resulta que hay muchas reglas con respecto a XML válido y el uso de `ElementTree` nos permite extraer datos de XML sin preocuparnos por las reglas de la sintaxis XML.



Bucle a través de nodos

A menudo, el XML tiene varios nodos y necesitamos escribir un bucle para procesar todos los nodos. En el siguiente programa, recorremos todos los nodos `user`:

<https://trinket.io/embed/python3/80ba384fbd>

El método `findall` recupera una lista Python de subárboles que representan las estructuras `user` en el árbol XML. Luego podemos escribir un bucle `for` que mira cada uno de los nodos de usuario e imprime los elementos de texto `name` e `id`, así como el atributo `x` del nodo `user`.

```
User count: 2
Name Chuck
Id 001
Attribute 2
Name Brent
Id 009
Attribute 7
```

Notación de objetos de JavaScript - JSON

El formato JSON se inspiró en el formato de objeto y matriz utilizado en el lenguaje JavaScript. Pero como Python se inventó antes de JavaScript, la sintaxis de Python para diccionarios y listas influyó en la sintaxis de JSON. Por lo tanto, el formato de JSON es casi idéntico a una combinación de listas y diccionarios de Python.

Aquí hay una codificación JSON que es aproximadamente equivalente al XML simple de arriba:

```
{
  "name" : "Chuck",
  "phone" : {
    "type" : "intl",
```

```
"number" : "+1 734 303 4456"
},
"email" : {
  "hide" : "yes"
}
}
```

Notarás algunas diferencias. Primero, en XML, podemos agregar atributos como "intl" a la etiqueta "phone". En JSON, simplemente tenemos pares clave-valor. Además, la etiqueta XML "persona" ha sido reemplazada por un conjunto de llaves exteriores.

En general, las estructuras JSON son más simples que las de XML porque JSON tiene menos prestaciones que le XML. Pero JSON tiene la ventaja de que mapea **directamente** a una combinación de diccionarios y listas. Y dado que casi todos los lenguajes de programación tienen algo equivalente a los diccionarios y listas de Python, JSON es un formato muy natural para que dos programas intercambien datos.

JSON se está convirtiendo rápidamente en el formato de elección para casi todo el intercambio de datos entre aplicaciones debido a su relativa simplicidad en comparación con XML.

Parsing JSON

Construimos nuestro JSON al anidar diccionarios (objetos) y listas según sea necesario. En este ejemplo, representamos una lista de usuarios donde cada usuario es un conjunto de pares clave-valor (es decir, un diccionario). Así que tenemos una lista de diccionarios.

En el siguiente programa, usamos la biblioteca incorporada **json** para analizar el JSON y leer los datos. Compare esto de cerca con el código y los datos XML equivalentes de arriba. El JSON tiene menos detalles, por lo que debemos saber de antemano que estamos obteniendo una lista y que la lista es de usuarios y cada usuario es un conjunto de pares clave-valor. El JSON es más sucinto (una ventaja) pero también es menos autodescriptivo (una desventaja).

<https://trinket.io/embed/python3/b540d86a7f>

Si comparas el código para extraer datos de JSON y XML analizados, verás que lo que obtenemos de **json.loads()** es una lista de Python que recorreremos con un bucle `for`, y cada elemento dentro de esa lista es un diccionario de Python. Una vez que se hayas analizado el JSON, podemos utilizar



el operador de índice de Python para extraer los distintos datos de cada usuario. No tenemos que usar la biblioteca JSON para explorar el JSON analizado, ya que los datos devueltos son simplemente estructuras nativas de Python.

La salida de este programa es exactamente la misma que la versión XML anterior.

```
User count: 2
Name Chuck
Id 001
Attribute 2
Name Brent
Id 009
Attribute 7
```

En general, hay una tendencia de la industria a alejarse del XML y apostar por JSON para servicios web. Debido a que el JSON es más simple y se asigna más directamente a las estructuras de datos nativos que ya tenemos en los lenguajes de programación, el código de análisis y extracción de datos suele ser más simple y más directo cuando se utiliza JSON. Pero XML es más autodescriptivo que JSON y, por lo tanto, hay algunas aplicaciones en las que XML conserva una ventaja. Por ejemplo, la mayoría de los procesadores de texto almacenan documentos internamente utilizando XML en lugar de JSON.

Interfaces de programación de aplicaciones

Ahora tenemos la capacidad de intercambiar datos entre aplicaciones mediante el Protocolo de transporte de hipertexto (HTTP) y una forma de representar datos complejos que enviamos de un lado a otro entre estas aplicaciones mediante el lenguaje de marcado extensible (XML) o la notación de objetos de JavaScript (JSON).

El siguiente paso es comenzar a definir y documentar los "contratos" entre las aplicaciones que utilizan estas técnicas. El nombre general de estos contratos de aplicación a aplicación es API (Application Programming Interfaces o *Interfaces de Programación de Aplicaciones*). Cuando usamos una API, generalmente un programa hace que un conjunto de **servicios** esté disponible para su uso por parte de otras aplicaciones, publicando las API (es decir, las "reglas") que deben seguirse para acceder a los servicios proporcionados por el programa.

Cuando comenzamos a construir nuestros programas donde la funcionalidad de nuestro programa incluye el acceso a los servicios proporcionados por otros programas, llamamos a este enfoque **Arquitectura Orientada a ServiciosAPI** o SOA. Un enfoque SOA es uno en el que nuestra aplicación general hace uso de los servicios de otras aplicaciones. Un enfoque que no es SOA es cuando la aplicación es una única aplicación independiente que contiene todo el código necesario para implementar la aplicación.

Vemos muchos ejemplos de SOA cuando usamos la web. Podemos ir a un solo sitio web y reservar viajes aéreos, hoteles y automóviles desde un solo sitio. Los datos de los hoteles no se almacenan en los ordenadores de la aerolínea. En su lugar, los ordenadores de la línea aérea se ponen en contacto con los servicios de los ordenadores del hotel, recuperan los datos del hotel y los presentan al usuario. Cuando el usuario acepta hacer una reserva de hotel utilizando el sitio de la aerolínea, el sitio de la línea aérea utiliza otro servicio web en los sistemas del hotel para realizar la reserva. Y cuando llega el momento de cargar su tarjeta de crédito por toda la transacción, aún otros ordenadores se involucran en el proceso.

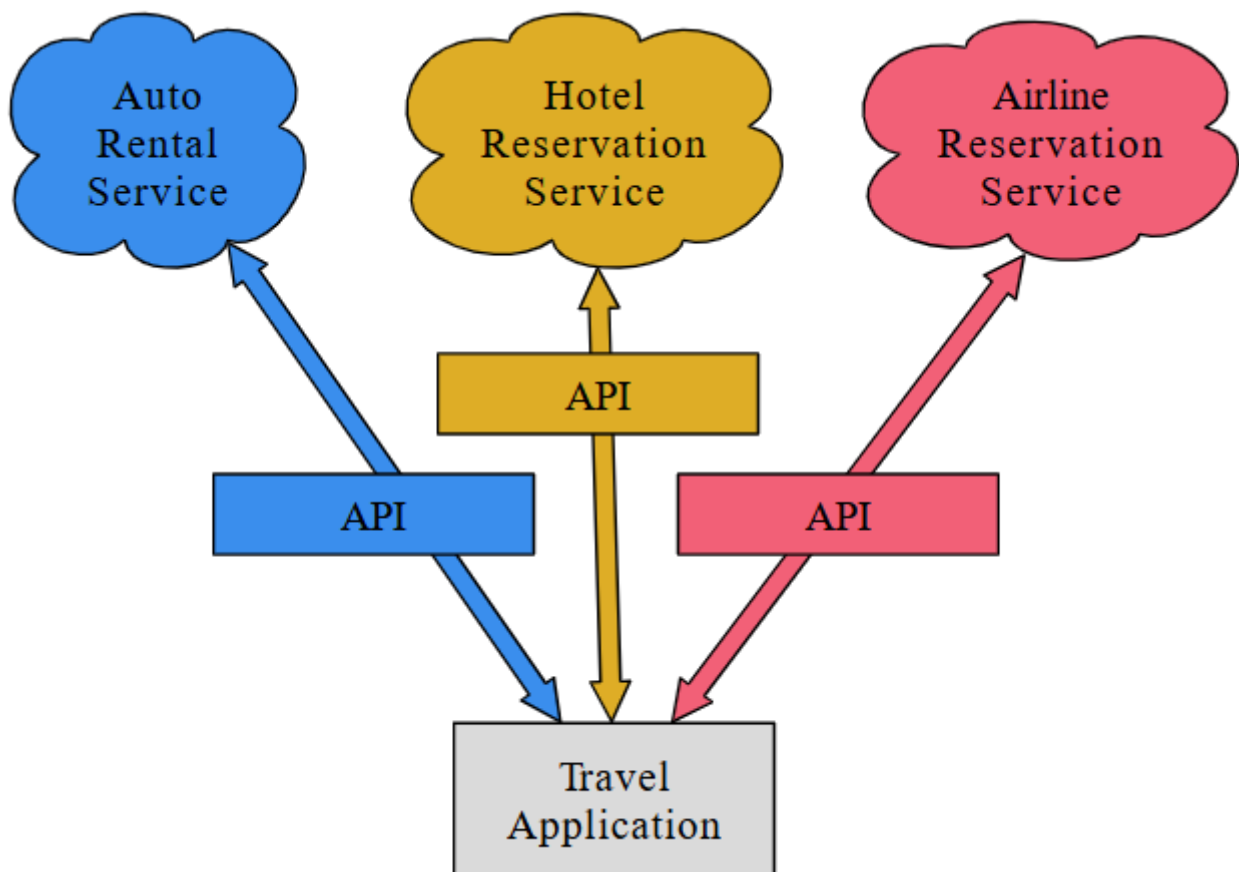


Imagen - Arquitectura orientada a Servicios

Una arquitectura orientada a servicios tiene muchas ventajas, entre ellas: (1) siempre mantenemos una sola copia de datos (esto es particularmente importante para cosas como reservas de hoteles donde no queremos comprometernos en exceso) y (2) los propietarios de los datos pueden



establecer las reglas sobre el uso de sus datos. Con estas ventajas, un sistema SOA debe diseñarse cuidadosamente para tener un buen rendimiento y satisfacer las necesidades del usuario.

Cuando una aplicación hace que un conjunto de servicios en su API esté disponible en la web, llamamos a éstos **servicios web**.

Seguridad y uso de API

Es bastante común que necesites algún tipo de "clave API" para hacer uso de la API de un proveedor. La idea general es que quieren saber quién está usando sus servicios y cuánto está usando cada usuario. Tal vez tengan niveles de servicios gratuitos y de pago, o tengan una política que limite la cantidad de solicitudes que una sola persona puede realizar durante un período de tiempo determinado.

A veces, una vez que obtienes tu clave de API, simplemente incluyes la clave como parte de los datos POST o tal vez como un parámetro en la URL al llamar a la API.

Otras veces, el proveedor desea una mayor seguridad de la fuente de las solicitudes y, por lo tanto, agregan que esperan que usted envíe mensajes firmados criptográficamente con claves y secretos compartidos. Una tecnología muy común que se utiliza para firmar solicitudes a través de Internet se llama **OAuth**. Puede leer más sobre el protocolo OAuth en [www.oauth.net] (

<http://www.oauth.net>).

A medida que la API de Twitter se hizo cada vez más valiosa, Twitter pasó de una API abierta y pública a una API que requería el uso de firmas OAuth en cada solicitud de API. Afortunadamente, todavía hay una serie de bibliotecas OAuth convenientes y gratuitas para que pueda evitar escribir una implementación de OAuth desde cero leyendo la especificación. Estas bibliotecas son de complejidad variable y tienen diversos grados de riqueza. El sitio web de OAuth tiene información sobre varias bibliotecas de OAuth.

Para este próximo programa de muestra, descargaremos los archivos **twurl.py**, **hidden.py**, **oauth.py** y **twitter1.py** de www.py4e.com/code3 y póngalos en una carpeta en su ordenador.

Para utilizar estos programas, necesitarás tener una cuenta de Twitter y autorizar su código Python como una aplicación, configurar una clave, un secreto, un token y un token secreto. Editarás el archivo **hidden.py** y colocarás estas cuatro cadenas en las variables apropiadas en el archivo:

<https://trinket.io/embed/python3/ab6352e0a3>

Se accede al servicio web de Twitter usando una URL como esta:

https://api.twitter.com/1.1/statuses/user_timeline.json

Pero una vez que se haya agregado toda la información de seguridad, la URL se verá más como:

```
https://api.twitter.com/1.1/statuses/user_timeline.json?count=2
&oauth_version=1.0&oauth_token=101...SGI&screen_name=drchuck
&oauth_nonce=09239679&oauth_timestamp=1380395644
&oauth_signature=rLK...BoD&oauth_consumer_key=h7Lu...GNg
&oauth_signature_method=HMAC-SHA1
```

Puedes leer la especificación de OAuth si deseas saber más sobre el significado de los diversos parámetros que se agregan para cumplir con los requisitos de seguridad de OAuth.

Para los programas que ejecutamos con Twitter, ocultamos toda la complejidad de los archivos **oauth.py** y **twurl.py**. Simplemente configuramos los secretos en **hidden.py** y luego enviamos la URL deseada a la función **twurl.augment()** y el código de la biblioteca agrega todos los parámetros necesarios a la URL para nosotros.

Este programa recupera la línea de tiempo para un usuario de Twitter en particular y nos la devuelve en formato JSON en una cadena. Simplemente imprimimos los primeros 250 caracteres de la cadena:

```
import urllib.request, urllib.parse, urllib.error
import twurl
import ssl

# https://apps.twitter.com/
# Create App and get the four strings, put them in hidden.py

TWITTER_URL = 'https://api.twitter.com/1.1/statuses/user_timeline.json'

# Ignore SSL certificate errors
ctx = ssl.create_default_context()
ctx.check_hostname = False
ctx.verify_mode = ssl.CERT_NONE
```



```
while True:
    print('')
    acct = input('Enter Twitter Account:')
    if (len(acct) < 1): break
    url = twurl.augment(TWITTER_URL,
                        {'screen_name': acct, 'count': '2'})
    print('Retrieving', url)
    connection = urllib.request.urlopen(url, context=ctx)
    data = connection.read().decode()
    print(data[:250])
    headers = dict(connection.getheaders())
    # print headers
    print('Remaining', headers['x-rate-limit-remaining'])

# Code: http://www.py4e.com/code3/twitter1.py
# Or select Download from this trinket's left-hand menu
```

Cuando el programa se ejecuta produce el siguiente resultado:

```
Enter Twitter Account:drchuck
Retrieving https://api.twitter.com/1.1/ ...
[{"created_at": "Sat Sep 28 17:30:25 +0000 2013", "
id": 384007200990982144, "id_str": "384007200990982144",
"text": "RT @fixpert: See how the Dutch handle traffic
intersections: http://t.co/tIiVtEhj4\n#brilliant",
"source": "web", "truncated": false, "in_rep
Remaining 178

Enter Twitter Account:fixpert
Retrieving https://api.twitter.com/1.1/ ...
[{"created_at": "Sat Sep 28 18:03:56 +0000 2013",
"id": 384015634108919808, "id_str": "384015634108919808",
"text": "3 months after my freak bocce ball accident,
my wedding ring fits again! :)\n\nhttps://t.co/2XmHPx7kgX",
"source": "web", "truncated": false,
Remaining 177
```

Enter Twitter Account:

Junto con los datos de línea de tiempo devueltos, Twitter también devuelve metadatos sobre la solicitud en los encabezados de respuesta HTTP. Un encabezado en particular, **x-rate-limit-remaining**, nos informa cuántas solicitudes más podemos hacer antes de que podamos cerrarnos por un corto período de tiempo. Puedes ver que nuestras recuperaciones restantes se reducen en una cada vez que realizamos una solicitud a la API.

En el siguiente ejemplo, recuperamos los amigos de Twitter de un usuario, analizamos el JSON devuelto y extraemos parte de la información sobre los amigos. También volcamos el JSON después de analizarlo y lo "imprimimos" con una sangría de cuatro caracteres para permitirnos analizar los datos cuando queremos extraer más campos.

<https://trinket.io/embed/python3/54ce78fb02>

Dado que JSON se convierte en un conjunto de listas y diccionarios Python anidados, podemos usar una combinación de la operación de índice y los bucles `for` para recorrer las estructuras de datos devueltos con muy poco código Python.

El resultado del programa es el siguiente (algunos de los elementos de datos se acortan para ajustarse a la página):

```
Enter Twitter Account:drchuck
Retrieving https://api.twitter.com/1.1/friends ...
Remaining 14
```

```
{
  "next_cursor": 1444171224491980205,
  "users": [
    {
      "id": 662433,
      "followers_count": 28725,
      "status": {
        "text": "@jazzzychad I just bought one .__.",
        "created_at": "Fri Sep 20 08:36:34 +0000 2013",
        "retweeted": false,
```



```

    },
    "location": "San Francisco, California",
    "screen_name": "leahculver",
    "name": "Leah Culver",
  },
  {
    "id": 40426722,
    "followers_count": 2635,
    "status": {
      "text": "RT @WSJ: Big employers like Google ...",
      "created_at": "Sat Sep 28 19:36:37 +0000 2013",
    },
    "location": "Victoria Canada",
    "screen_name": "_valeriei",
    "name": "Valerie Irvine",
  },
],
"next_cursor_str": "1444171224491980205"
}

```

```

leahculver
  @jazzychad I just bought one .__.
_valeriei
  RT @WSJ: Big employers like Google, AT&T are h
ericbollens
  RT @lukew: sneak peek: my LONG take on the good &a
halherzog
  Learning Objects is 10. We had a cake with the L0,
scweeker
  @DeviceLabDC love it! Now where so I get that "etc

Enter Twitter Account:

```

El último bit de la salida es donde vemos el bucle for leyendo los cinco "amigos" más recientes de la cuenta de Twitter **drchuck** e imprimiendo el estado más reciente de cada amigo. Hay muchos más datos disponibles en el JSON devuelto. Si observas la salida del programa, también puedes ver que el "buscar amigos" de una cuenta en particular tiene una limitación de tasa diferente a la cantidad de consultas en la línea de tiempo que podemos ejecutar por período de tiempo.



Estas claves de API seguras permiten que Twitter tenga una sólida confianza de que saben quién está usando su API y sus datos y en qué nivel. El enfoque de limitación de velocidad nos permite realizar recuperaciones de datos personales simples, pero no nos permite crear un producto que extraiga datos de su API millones de veces al día.

Ejercicios

Ejercicio 1: Cambie el www.py4e.com/code3/geojson.py o www.py4e.com/code3/geoxml.py para imprimir el código de país de dos caracteres a partir de los datos recuperados. Agregue la verificación de errores para que su programa no se registre si el código de país no está allí. Una vez que lo tenga funcionando, busque "Océano Atlántico" y asegúrese de que pueda manejar ubicaciones que no estén en ningún país.

Revision #1

Created 5 April 2025 11:16:51 by Javier Quintana

Updated 5 April 2025 11:18:10 by Javier Quintana