Prácticas con json
Servicios Telemáticos 2015-2016
Grado en Ingeniería Telemática
Universidad Rey Juan Carlos
`http://gsyc.urjc.es`

Observaciones

- Crea el fichero `~/st/practica04.txt`, que contendrá la memoria que escribas sobre este segundo bloque de prácticas.

  Cuando una práctica consista solamente en preparar un script, no es necesario que escribas nada en la memoria. Cuando la práctica consista en revisar ficheros o darles formato, sí.

- Crea el directorio `~/st/practica04` donde escribirás el resto de ficheros de esta práctica. Recuerda que la recogida de las prácticas se hará automáticamente el dia del examen, así que es muy importante que respetes al pie de la letra todos los nombres de ficheros y directorios que indiquen todos los enunciados.

Práctica 4.1 jsoncheck, jsonpp

En esta práctica probarás jsoncheck, un pequeño programa que comprueba que un fichero contiene un valor json válido, y jsonpp, que hace *pretty printing* de un fichero con valores json. Ambor trabajan a partir de un fichero leído por stdin

1. Descarga jsoncheck[1] y jsonpp[2]

   y guárdalo en `~/bin`

2. Escribe un objeto json de tipo *holamundo* con un par de pares valor-clave y úsalo para probar jsoncheck y jsonpp

   Copia y pega el resultado en el fichero con la memoria de la práctica

Práctica 4.2 coches.json

Escribe un fichero json `~/st/practica04/coches.json` con la misma información que tu fichero `~/st/practica02/coches.xml`
Compruébalo con jsoncheck, dale formato con jsonpp
Copia y pega el resultado en el fichero con la memoria de la práctica

Práctica 4.3 discoteca.json

Escribe un fichero json `~/st/practica04/discoteca.json` con la misma información que tu fichero `~/st/practica02/discoteca.xml`
Compruébalo con jsoncheck, dale formato con jsonpp
Copia y pega el resultado en el fichero con la memoria de la práctica

[1] `http://gsyc.es/~mortuno/st/jsoncheck`
[2] `http://gsyc.es/~mortuno/st/jsonpp`

Práctica 4.4 Fechas y horas en json

Modifica el script
`~/st/practica03/fechas.py`
para que en caso de recibir la opción `-j --json` muestre la salida en formato json. Deberá generar una lista de objetos. La clave de cada objeto será el número de línea (1,2,3...) y el valor será o bien una cadena con la fecha y la hora o bien el timestamp como número (no como cadena).

Observa que:

- Solo cambia el formato de la salida, la información contenida y el resto del comportamiento del script será idéntico al comportamiento en la práctica 3
- Aunque esta es la práctica 4, estás modificando un fichero del directorio `~/st/practica03`

Práctica 4.5 papeleria.py

Escribe el script `~/st/practica04/papeleria.py` (sin tilde en la i) que aceptará por entrada estándar un fichero en formato json según la siguiente especificación

- El fichero contendrá un array de valores
- Cada valor será un objeto
- Cada objeto tendrá tres claves: *ref*, cuyo valor será un número, *descripcion* (sin tilde en la 'o'), cuyo valor será una cadena y *precio*, cuyo valor será un número
- El script deberá mostrar en la salida estándar la misma información, en formato tabular
-

Ejemplo de entrada:

```
[{"descripcion": "Caja 12 l\u00e1pices", "precio": 7.54, "ref": 869468}, {"descripcion": "Papelera negra", "precio": 6.1, "r
```

Descarga este ejemplo[3] para probar tu práctica
La salida deberá tener un aspecto similar a este:

```
Ref.     Precio   Descripción
---------------------------------------------------------------
869468   7.54     Caja 12 lápices
968726   6.10     Papelera negra
518196   5.40     Paquete 500 hojas A4
```

Práctica 4.6 inflacion.py

Escribe el script `~/st/practica04/inflacion.py` que

- Aceptará desde stdin un fichero json con el formato descrito en el apartado anterior
- Tendrá definida una constante llamada INFLACION. Será un objeto (una variable) que aparecerá al principio del código, inmediatamente después de las sentencias import
- Mostrará en stdout el mismo fichero que a la entrada, con el mismo formato, pero con los precios multiplicados por 1+(INFLACION*0.01)

  Ejemplo: si INFLACION vale 3 y el precio era 7.54, el nuevo precio será 7.7662

[3] `http://gsyc.es/~mortuno/st/libreria.json`

Práctica 4.7 papelería en xml

Escribe un script con el nombre `~/st/practica04/papeleria2xml.py` (sin tilde en la i) que reciba por stdin un fichero en json como el especificado en la práctica 4.5 y escriba en stdout un fichero con la misma información en xml. Tienes libertad para definir el formato de este xml, con tal de que contenga la misma información y que sea sintácticamente correcto (compruébalo con xmlcheck)

Práctica 4.8 Fechas en xml

Una vez que tu script `~/st/practica03/fechas.py` funcione correctamente como indica la práctica 4.4, hazle una copia con nombre `~/st/practica03/fechas0.py`

Mueve `~/st/practica03/fechas.py` a `~/st/practica04/fechas.py` y hazle las siguientes modificaciones

(Observa que antes estaba en el directorio practica03 y ahora en practica04)

1. Si el script recibe la opción `-a --ascii`, mostrará la salida en texto plano, igual que `~/st/practica03/fechas0.py`
2. Si el script recibe la opción `-j --json`, muestra la salida en json, como que hasta ahora
3. Si el script no recibe la opción `-a --ascii` ni la opción `-j --json`, mostrará la salida en formato XML
4. El elemento raiz se llamará *instants* y cada uno de sus elementos, *instant*
5. Si la fecha corresponde a una ciudad, el elemento raíz tendrá un atributo llamado *city* cuyo valor será esa ciudad

   Si la fecha es un timestamp, el elemento raíz tendrá un atributo llamado *city* cuyo valor será la cadena de texto *timestamp*
6. Cada uno de los elementos tendrá un atributo llamado *date* con la fecha, en el mismo formato especificado en la práctica 3.1 así como un atributo llamado *ordinal* con el número de línea

Ejemplo de salida para Madrid:

```
<instants city="Madrid">
    <instant date="2004-10-18 23:06:49+02:00" ordinal="1" />
    <instant date="2015-08-21 05:32:06+02:00" ordinal="2" />
    <instant date="2005-07-26 08:12:25+02:00" ordinal="3" />
[...]
    <instant date="2001-02-27 05:59:39+01:00" ordinal="40" />
</instants>
```

Ejemplo de salida para timestamp

```
<instants city="timestamp">
    <instant date="1098133609" ordinal="1" />
    <instant date="1440127926" ordinal="2" />
    <instant date="1122358345" ordinal="3" />
[...]
    <instant date="983249979" ordinal="40" />
</instants>
```

- La salida de tu script no es necesario que incorpore pretty printing, pero usa xmlpp y xmlcheck para comprobar que es correcta
- Estos ejemplos ilustran el formato de la salida, como podrás observar estas fechas no se corresponden a las del ejemplo mostrado en 3.1