“Otro trabajo de YAML más”

Autor: Manuel Ortega Cabrera
Módulo: “Lenguajes de Marcas y Sistemas de Gestión de Información”
Ciclo: C.F.G.S. “Administración de Sistemas Informáticos en Red”
Centro: I.E.S. Pablo Ruiz Picasso (Chiclana de la Frontera, Cádiz)
Curso: 2023/24

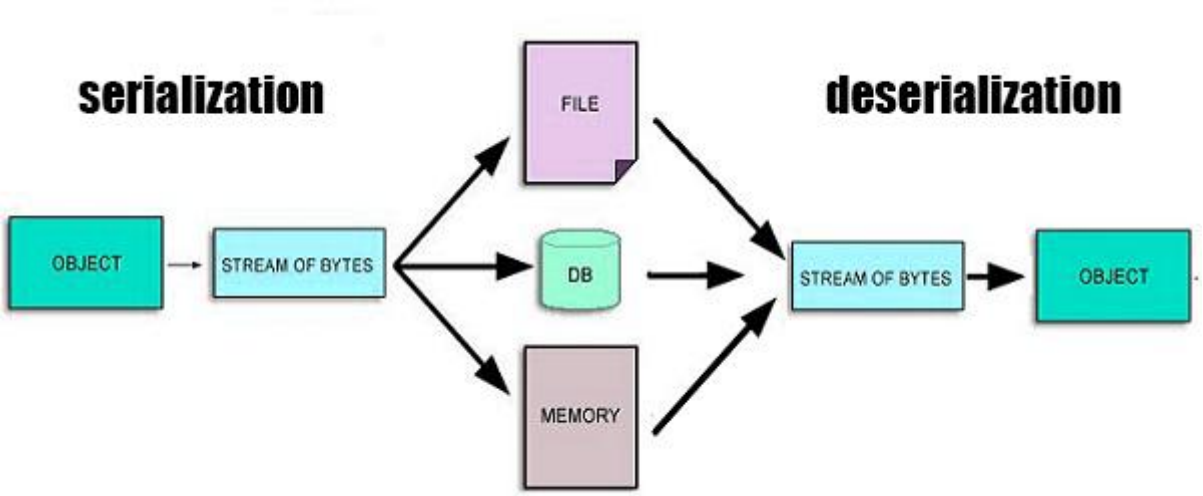
ÍNDICE

INTRODUCCIÓN: ¿QUÉ ES LA SERIALIZACIÓN?

Para entender bien lo que es YAML es necesario conocer el concepto de “serialización”.

La serialización consiste en codificar un objeto en un medio de almacenamiento (software o hardware) como una serie de bytes o en un formato humanamente más legible (como XML o JSON) con el fin de transmitirlo. El resultado puede ser usado para crear un nuevo objeto que es idéntico en todo al original, incluido su estado interno.

Se trata de un mecanismo ampliamente usado para transportar objetos a través de una red, para hacer persistente un objeto en un archivo o base de datos, o para distribuir objetos idénticos a varias aplicaciones o localizaciones.



¿QUÉ ES YAML?

YAML es un lenguaje de serialización de datos legible por humanos que suele utilizarse en el diseño de archivos de configuración.

Las siglas “YAML” corresponden a “**Y**AML **A**in't **M**arkup **L**anguage” (“YAML no es un lenguaje de marcado”). A comienzos de su desarrollo, “YAML” significaba “**Y**et **A**nother **M**arkup **L**anguage” (“otro lenguaje de marcado más”) para distinguir su propósito centrado en los datos en lugar del marcado de documentos.

Debido a que se usa frecuentemente XML para serializar datos y XML es un auténtico lenguaje de marcado de documentos, se suele considerar a YAML como un lenguaje de marcado ligero.

YAML fue creado bajo la creencia de que todos los datos pueden ser representados adecuadamente como combinaciones de listas, hashes (mapeos) y datos escalares (valores simples).

Se diseñó teniendo en cuenta que fuera muy legible pero que a la vez fuese fácilmente mapeable a los tipos de datos más comunes en la mayoría de los lenguajes de alto nivel.

Basa su funcionalidad en JSON (conjuntos de pares clave-valor), con la adición de líneas nuevas e indentación inspirada en Python. A diferencia de Python, YAML no permite tabulaciones literales, en su lugar usa espacios.

Nota: "Indentar" es poner espacios hacia la derecha para mover una línea de código (sangría), lo que puedes hacer usando la barra espaciadora o la tecla de tabulación. Se suele hacer para mejorar la legibilidad del código aunque hay lenguajes (como Python y YAML) que lo usan para especificar la estructura.

APRENDEMOS YAML

Comentarios

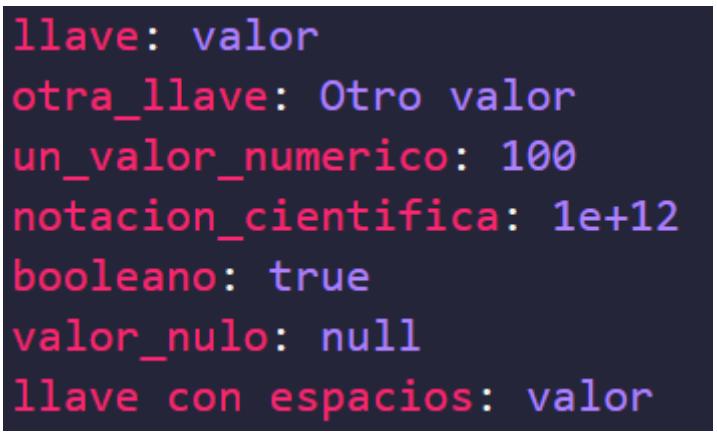
Los comentarios vienen encabezados por la almohadilla (**#**) y continúan hasta el final de la línea.

```
# Los comentarios en YAML se ven así.
```

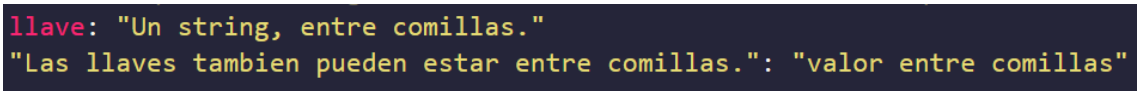
Tipos simples o escalares

Nuestro objeto raíz (el cual es el mismo a lo largo de todo el documento) será un mapa (equivalente a un diccionario, hash, u objeto en otros lenguajes).

A continuación algunos ejemplos de pares clave-valor:

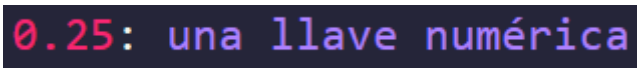


Los valores y las llaves por lo general aparecen sin entrecomillar, pero pueden incluirse entre comillas dobles (**"**), o comillas simples (**'**). Nótese que los strings no tienen porqué estar necesariamente entre comillas:



En las comillas dobles, los caracteres especiales se pueden representar con secuencias de escape similares a las del lenguaje de programación C, que comienzan con una barra invertida (****).

Nótese también que las llaves de los mapas no requieren ser strings necesariamente:



Los strings de líneas múltiples pueden ser escritos como un “bloque literal” (usando pipes **|**) el cual mantiene retornos de línea o como un ”bloque doblado” (usando **>**) que ignora los retornos de línea:

```yaml
bloque_literal: |
    Este bloque completo de texto será preservado como el valor de la llave
    'bloque_literal', incluyendo los saltos de línea.

    Se continúa guardando la literal hasta que se cese la indentación.
        Cualquier línea que tenga más indentación, mantendrá los espacios dados
        (por ejemplo, estas líneas se guardarán con cuatro espacios)

bloque_doblado: >
    De la misma forma que el valor de 'bloque_literal', todas estas
    líneas se guardarán como una sola literal, pero en esta ocasión todos los
    saltos de línea serán reemplazados por espacio.

    Las líneas en blanco, como la anterior, son convertidas a un salto de línea.

        Las líneas con mayor indentación guardan sus saltos de línea.
        Esta literal ocuparán dos líneas.
```

```yaml
--- |
 El texto mantiene su estructura,
 en el sentido que preserva los retornos de línea.

 Esto incluye también líneas en blanco.
```

```yaml
SAYO
 --- >
  El texto rodeado
  será formateado
  como un único
  párrafo

  Las líneas en blanco
  denotan saltos de párrafo.
```

La estructura del documento se denota indentando con espacios en blanco (recuerda que en YAML no se permite el uso de caracteres de tabulación para sangrar). La indentación se usa para anidar elementos:

```yaml
un_mapa_indentado:
    llave: valor
    otra_llave: otro valor
    otro_mapa_indentado:
        llave_interna: valor_interno
```

Las llaves también pueden ser de múltiples líneas, usando **?** para indicar el inicio de una llave

```yaml
? |
    Esto es una llave
    que tiene múltiples líneas
: y este es su valor
```

Tipos de colecciones

Las colecciones en YAML usan la indentación para delimitar el alcance.

Los miembros de las listas se pueden denotar de dos formas distintas:

- Encabezados por un guión (**-**) con un elemento por cada línea
- Todos los elementos en la misma línea entre corchetes (**[ ]**) y separados por coma y espacio (**,**):

```yaml
--- # Películas favoritas, formato de bloque
- BotijoAzul
- BotijoVerde
- Viridiana
- Psicosis
...
--- # Lista de la compra, formato en línea
[leche, pan, huevos]
[chorizo, morcilla, botijo, pollo]
```

Y se pueden combinar ambos formatos:

```
- [Uno, Dos, Tres]
- [Domingo, Lunes, Martes]
- [Luna, Marte, Tierra]
```

Una lista puede ser el valor de una llave:

```
secuencia:
    - Elemento 1
    - Elemento 2
    - Elemento 3
    - Elemento 4
```

Las listas pueden tener distintos tipos en su contenido:

```
secuencia_combinada:
    - texto
    - 5
    - 0.6
    - llave: valor # se convierte en un json dentro de la secuencia
    -
        - Esta es una secuencia
        - ...dentro de otra secuencia
```

Dado que todo JSON está incluído dentro de YAML, también puedes escribir mapas con la sintaxis de JSON:

```
mapa_de_json_1: {"llave": "valor"}
mapa_de_json_2:
  llave: valor
```

```yaml
secuencia_de_json_1: [3, 2, 1, "despegue"]
secuencia_de_json_2:
  - 3
  - 2
  - 1
  - "despegue"
```

Los vectores asociativos se representan:

- En bloque: cada elemento del vector en una línea en formato clave-valor.
- En línea: todos los elementos en una misma línea entre llaves (**{ }**), en formato clave-valor y separados por coma seguida de espacio (**,**).

```yaml
--- # Bloque
nombre: Pepe López
edad: 33
--- # En  línea
{nombre: Pepe López, edad: 33}
```

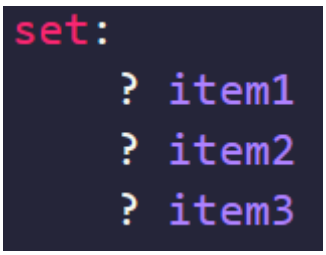
Puedes combinar listas y vectores, creando así listas de vectores:

```yaml
- {nombre: Pepe López, edad: 33}
- nombre: Maria Garcia
  edad: 27
```

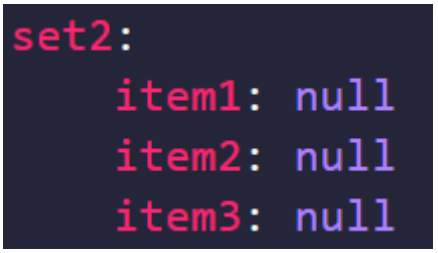
O vectores de listas:

```yaml
hombres: [Pepe Lopez, Guillermo Garcia]
mujeres:
 - María García
 - Susana Márquez
```

YAML también soporta conjuntos usando el símbolo (**?**). Al poner esto precediendo a un valor de un vector asociativo permite que se construyan claves complejas sin ambigüedad y se ven de la siguiente forma:



Al igual que Python, los conjuntos sólo son mapas con valores nulos. El ejemplo de arriba es equivalente a:

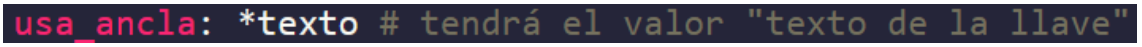


Características extras

Se pueden incluir múltiples documentos dentro de un único flujo, separándolos por tres guiones (**---**). Los tres puntos (**...**) indican el fin de un documento dentro de un flujo.

YAML tiene funciones útiles llamadas 'anchors' (anclas), que te permiten duplicar fácilmente contenido a lo largo de tu documento. El (**&**) indica la declaración del ancla:

El asterisco (*****) indica el uso de dicha ancla:



Hay dos caracteres adicionales que están reservados en YAML para su posible estandarización en un futuro: la arroba (**@**) y el acento grave (`).

Tags

En YAML, los nodos que no tienen un tag obtienen su tipo según la aplicación que los use, al usar un tag (Con **!!** seguido de una cadena que indica el tipo) se pueden declarar tipos explícitamente.

```yaml
string_explicito: !!str 0.5 # !!str para declarar un string
integer_explicito: !!int 5 # !!int para declarar un integer
float_explicito: !!float 1.2 # !!float para declarar un float
conjunto_explicito: !!set # !!set para declarar un conjunto
  ? Uno
  ? Dos
  ? Tres
mapa_ordenado_explicito: !!omap # !!omap para declarar un mapa ordenado
- Primero: 1
- Segundo: 2
- Tercero: 3
- Cuarto: 4
```

Documentos

La extensión .yml es la utilizada por los archivos que contienen YAML.

Los documentos YAML pueden ser precedidos por directivas compuestas por un signo de porcentaje (**%**) seguidos de un nombre y parámetros delimitados por espacios. Hay definidas dos directivas en YAML:

- La directiva **%YAML** se utiliza para identificar la versión de YAML en un documento dado.
- La directiva **%TAG** se utiliza como atajo para los prefijos de URIs. Estos atajos pueden ser usados en las etiquetas de tipos de nodos.

LENGUAJES QUE SOPORTAN YAML

- JavaScript
- XML
- Python
- PHP
- Java
- Ruby
- Haskell
- Tcl
- Perl
- Objective-C

SITIOS WEB DE INTERÉS

- Sitio web oficial de YAML: https://yaml.org/
- Validador online de YAML: https://codebeautify.org/yaml-validator
- Conversor online de YAML a JSON, Python y YAML Canónico: http://yaml-online-parser.appspot.com/

BIBLIOGRAFÍA

https://es.wikipedia.org/wiki/YAML

https://es.wikipedia.org/wiki/Serializaci%C3%B3n

https://learnxinyminutes.com/docs/es-es/yaml-es/