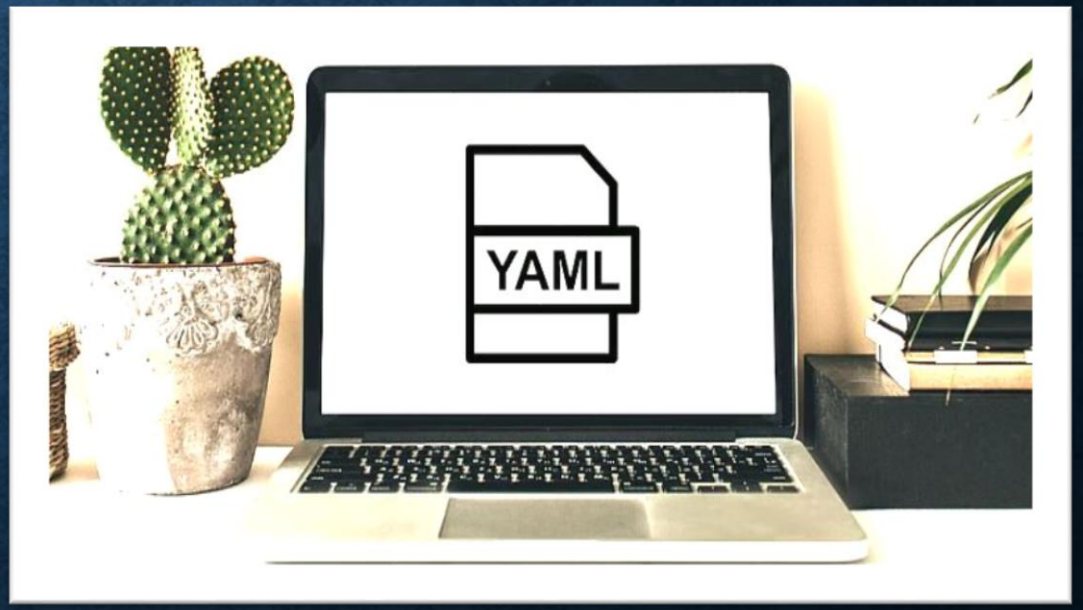
UT05_04 YAML

Lenguaje de serialización de datos



Francisco José García Cutillas

1. INTRODUCCIÓN A YAML

- YAML es un lenguaje de serialización de datos, propuesto como alternativa a XML.
- No se considera un lenguaje de marcado, de ahí su nombre (Ain't A Markup Language).
- Suele utilizarse en el diseño de archivos de configuración.
- Al no considerarse un lenguaje de marcado, se enfatiza la idea de que su utilización está orientada para los datos, no para los documentos.
- Fue creado por Clark Evans en 2001.

2. CARACTERÍSTICAS

- YAML utiliza la extensión .yml o .yaml.
- Tiene características que provienen de los lenguajes Perl, C, XML o HTML entre otros.
- Se basa en JSON, por lo que archivos correctamente formados JSON son compatibles con YAML.
- No dispone de llaves, corchetes, etiquetas de cierre u otros símbolos de formato habituales. Por lo que su lectura es más sencilla, ya que utiliza la sangría al estilo Python.
- Está diseñado para que no se admita el carácter de tabulación y así mantener la portabilidad en todos los sistemas. Utiliza el carácter espacio.
- Los comentarios en YAML van precedidos de almohadilla (#). No existen comentarios de varias líneas, por lo que para cada línea de comentario, se debe preceder de #.
- Un documento YAML comienza con "---" que indica el inicio del archivo YAML.

3. SINTAXIS

- Con respecto a la sintaxis de YAML tenemos dos tipos de estructuras principales:
  - Mapas.
  - Listas.

3.1 Mapas

- El objeto raíz de un fichero YAML será un mapa, equivale a un objeto en otros lenguajes.
- La forma de representar un mapa es "identificador: valor". Ejemplos:

```yaml
llave: valor
valor_numerico: 12
notacion_cientifica: 1e+10
booleano: true
nulo: null
llave con espacio: valor
```

- Los tipo String no deben ir entre comillas, aunque también es válido.

```
llave: "String entre comillas"
"la llave tambien puede ir entre comillas": "valor entre comillas"
```

- Hay también dos formas de escribir bloques de String.

```
bloque_literal: |
    Este bloque se guardará tal cual, incluyendo saltos de línea.

    Se continúa guardando hasta que cese la indentación.
        Cualquier línea que tenga más indentación, mantendrá los espacios dados.

bloque_doblado: >
    En este caso, todas las líneas se guardarán como una sola literal. En este caso
    los saltos de línea se guardarán como espacio.

    Las líneas en blanco, como la anterior, se considera salto de línea.

        Las líneas con mayor indentación guardan sus saltos de línea.
        Por ejemplo estas dos líneas ocuparán dos líneas.
```

- Mapas indentados, para anidar elementos.

```
mapa_indentado:
    llave: valor
    numero: 3
    otro_mapa_indentado:
        llave_interna: valor_interno
```

- Las llaves de los mapas no tienen por qué ser de tipo String. Pueden ser numéricas, u objetos de diferentes líneas utilizando ?|

```
0.25: llave numérica

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

3.2 Listas

- Las listas en YAML usan la indentación para delimitar el alcance, y cada elemento de la lista inicia en su propia línea.

```
- Amarillo
- Verde
- Azul
```

- Se puede usar una secuencia como valor para una llave.

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

- Las secuencias pueden contener distintos tipos de contenido.

```yaml
secuencia_combinada:
  - texto
  - 5
  - 0.6
  - llave: valor
```

- Dado que JSON está incluido dentro de YAML, podemos escribir listas como si se tratara de un JSON.

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

- También podemos usar los arrays de JSON.

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

- YAML también soporta los conjuntos usando ?

```yaml
set:
  ? item1
  ? item2
  ? item3
```

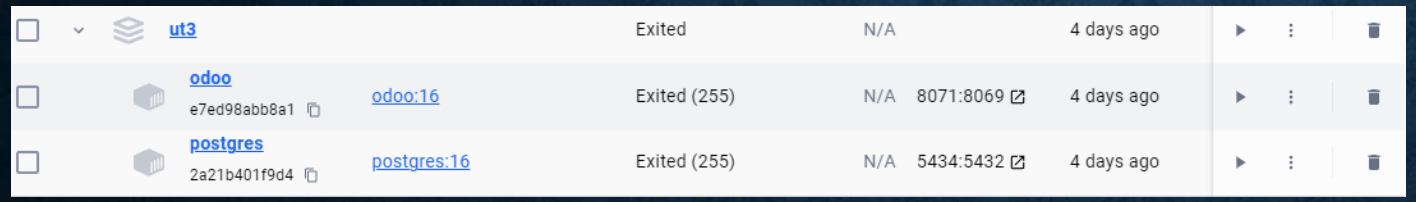
```yaml
#Equivale a
set2:
  item1: null
  item2: null
  item3: null
```

4. USOS

- El uso principal de un archivo YAML es la creación de archivos de configuración usados en:
  - Herramientas.
  - Servicios de automatización.
  - DevOps.
  - Infraestructura como código (IaC).
- Se recomienda utilizar YAML en lugar de JSON ya que es más fácil de comprender, aunque se pueden utilizar ambos de manera indistinta.

5. HERRAMIENTAS QUE UTILIZAN YAML

- Con respecto a las herramientas que utilizan YAML, vamos a comentar alguna de ellas. Como son:
  - Docker.
  - Kubernetes.
  - Ansible.
  - Chef
- Ejemplo de archivo de configuración YAML para Docker.




```yaml
version: "3.8"
name: ut3
services:
    odoo:
        container_name: odoo
        image: odoo:16
        depends_on:
            - db
        ports:
            - "8071:8069"
        volumes:
            - data:/var/lib/odoo
            - .\logs:/var/log/odoo
            - .\complementos:/mnt/extra-addons
            - .\configuracion:/etc/odoo
    db:
        container_name: postgres
        image: postgres:16
        ports:
            - "5434:5432"
        environment:
            - POSTGRES_DB=postgres
            - POSTGRES_PASSWORD=odoo
            - POSTGRES_USER=odoo
        volumes:
            - postgres:/var/lib/postgresql/data
volumes:
    data:
    postgres:
```

6. BIBLIOGRAFÍA

- https://learnxinyminutes.com/docs/es-es/yaml-es/
- https://keepcoding.io/blog/que-es-y-para-que-sirve-el-fichero-yaml/
- https://www.redhat.com/es/topics/automation/what-is-yaml
- https://onthedock.github.io/post/170525-introduccion-a-yaml/
- https://yaml.org/
- https://geekflare.com/es/what-is-yaml/
- https://tecnoyfoto.com/integracion-de-yaml-con-lenguajes-y-herramientas
- Enlace al vídeo explicativo. https://youtu.be/d5npcBRjJgA