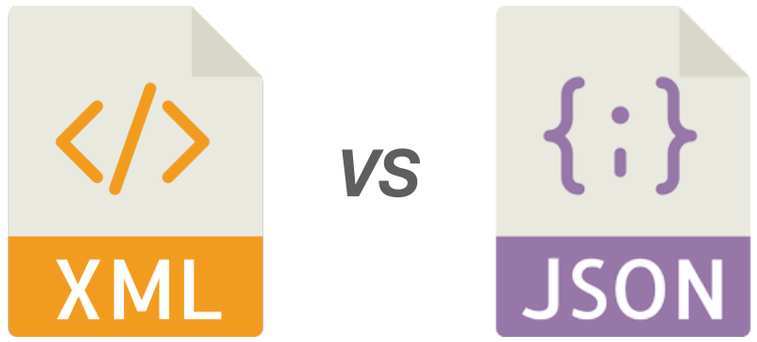


Unidad 5

Tratamiento y transformación de datos

Francisco Javier Sánchez García *LMSGI*

Contenidos

1. JSON

1. JSON

JavaScript Object Notation

JSON significa Notación de Objetos JavaScript, es decir, es el formato en el que JavaScript representa sus objetos, incluidos los arrays.

Es "autodescriptible" y fácil de entender

Pese a su naturaleza tan específica, JSON se ha universalizado como **alternativa a XML** para almacenar e intercambiar datos entre sistemas independientemente de la plataforma y del lenguaje de programación.

Casi todos los lenguajes que se usan en la actualidad disponen de analizadores JSON.

La extensión de los ficheros JSON es *.json*

1. JSON

Sintaxis

La sintaxis JSON se deriva de la sintaxis de notación de objetos JavaScript:

- Los datos están en pares de nombre/valor ("clave":"valor")
- Los datos están separados por comas ,
- Las llaves contienen objetos { }
- Los corchetes tienen arrays, es decir, listas de elementos [ ]

1. JSON

Tipos de datos

En **JSON**, los valores deben ser uno de los siguientes tipos de datos:

- una cadena de caracteres
- un número
- un objeto
- una matriz (array)
- un booleano
- NULO

```
{name:"John", age:31, city:"New York"}
```

```
{
"employees":["John", "Anna", "Peter"]
}
```

```
{"sale":true}
```

```
{"middlename":null}
```

1. JSON

Ejemplo 1: Coche

Queremos guardar la información de un coche (marca, modelo y color).

```json
{
  "coche":{
    "marca":"BMW",
    "modelo":"Serie 1",
    "color":"negro"
  }
}
```

1. JSON

Ejemplo 2: Cine

Un pequeño cine requiere guardar un registro de las películas que tiene.

```json
{
  "peliculas":[
    {
      "titulo":"El padrino",
      "anio":1972,
      "protagonista":"Al Pacino"
    },
    {
      "titulo":"The Batman",
      "anio":2022,
      "protagonista":"Robert Pattinson"
    },
    {
      "titulo":"Avatar",
      "anio":2009,
      "protagonista":"Sam Worthington"
    },
    {
      "titulo":"Black Panther: Wakanda Forever",
      "anio":2022,
      "protagonista":"Letitia Wright"
    },
    {
      "titulo":"Black Panther: Wakanda Forever",
      "anio":2022,
      "protagonista":"Letitia Wright"
    }
  ]
}
```

Ejercicio 1. Realiza un JSON para estructurar la información de la tabla siguiente:

	Libro 1	Libro 2	Libro 3
Título	El programador prágmatico	Código Limpio	Aprendiendo JavaScript
Autor	David Thomas y Andrew Hunt	Robert C. Martin	Carlos Azaustre
Año de publicación	2022	2012	2016
Editorial	Anaya Multimedia	Anaya Multimedia	Publicación independiente
Páginas	339	463	99

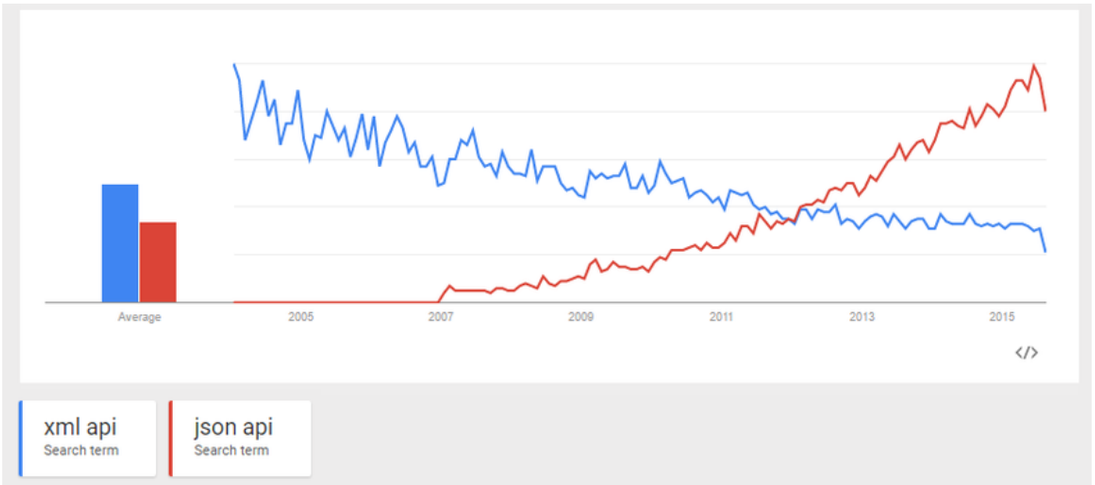
Ejercicio 2. Escribir un documento JSON que almacene la siguiente información:

Ciudades			
Nombre	País	Continente	Elevación
Ceuta	España	África	10m
Mykonos	Grecia	Europa	-
Pekín	China	Asia	44m
Ciudad de México	México	América	2240m

1. JSON

vs XML

```
{"employees":[
  { "firstName":"John", "lastName":"Doe" },
  { "firstName":"Anna", "lastName":"Smith" },
  { "firstName":"Peter", "lastName":"Jones" }
]}
```




```
<employees>
   <employee>
      <firstName>John</firstName>
      <lastName>Doe</lastName>
   </employee>
   <employee>
      <firstName>Anna</firstName>
      <lastName>Smith</lastName>
   </employee>
   <employee>
      <firstName>Peter</firstName>
      <lastName>Jones</lastName>
   </employee>
</employees>
```

1. JSON

vs XML

Tanto **JSON** como **XML**:

- Son "autodescriptibles"
- Son jerárquicos (valores dentro de los valores)
- Pueden ser analizados y utilizados por muchos lenguajes de programación
- Se pueden obtener con una solicitud HTTP

1. JSON

vs XML

XML Lenguaje de marcado extensible	JSON Notación de objetos JavaScript
idioma	Formato de texto
Representa elementos de datos	Se utiliza para la representación de objetos
No hay soporte directo de matriz	Admite tipos de datos de texto y numéricos, matrices y objetos
Utiliza etiquetas de apertura y cierre	No utiliza etiquetas de cierre, sino que utiliza corchetes (curúme y cuadrado)
Admite el espacio de nombres	No es compatible con espacios de nombres
Más seguro	Menos seguro
Apoya los comentarios	No apoya los comentarios
Formato de datos independiente que admite diferentes codificaciones	Un lenguaje de intercambio de datos es independiente y solo admite archivos de texto codificados en UTF-8

Ejercicio 3. Escribir un documento JSON que almacene la siguiente información:

Hechos históricos			
Descripción	Fecha		
	Día	Mes	Año
IBM da a conocer el PC	12	8	1981
Se funda Google	4	9	1998
Se funda Facebook	4	2	2004
Steve Jobs presenta el iPhone	29	6	2007

Ejercicio 4. Crear un documento JSON que describa una lista de marcadores de páginas web, sabiendo que se desea que la información de cada ***página*** sea el ***nombre***, una ***descripción*** breve y su ***URL***. Los datos de los marcadores son los descritos en la siguiente tabla:

MDN Web Docs Resources for Developers, by Developers https://developer.mozilla.org/es/
W3Schools Learn to code https://www.w3schools.com
Wikipedia La enciclopedia libre https://es.wikipedia.org/

Ejercicio 5. Escribir un documento JSON que guarde información de dos equipos de fútbol (nombre, ciudad y entrenador) con dos jugadores (nombre, posición y nacionalidad) cada uno. Las posiciones posibles son (portero, defensa, medio, delantero). Utilizar datos reales para los equipos y jugadores. No obstante, no deberá indicarse el nombre del entrenador.

2. XPath

2. XPath

¿Qué es?

- Significa *XML Path Language*
- **XPath** es un elemento importante del estándar **XSLT**, recomendado por el **W3C**.
- Se puede utilizar para navegar a través de elementos y atributos en un documento XML.
- Contiene más de 200 **funciones integradas**
- También se puede utilizar con **JSON**

2. XPath

Expresiones de ruta

Utiliza **expresiones** de ruta para **seleccionar nodos** o conjuntos de nodos en un documento XML.

Estas expresiones de ruta se parecen mucho a las expresiones de ruta que usas con los **sistemas de archivos** informáticos:

Ejercicio 6. XPath es un lenguaje XML que permite:

A) Transformar el formato de los datos de un fichero XML.
B) Definir un vocabulario que ha de cumplir un documento XML.
C) Obtener los datos del fichero XML de una base de datos.
D) Acceder a los datos de un fichero XML.

2. XPath

Funciones estándar

Hay funciones para cadenas, valores numéricos, booleanos, de fecha y hora, manipulación de nodos, de secuencias y mucho más.

Las **expresiones XPath** también se pueden usar en JavaScript, Java, **PHP**, Python, C y C++, y muchos otros lenguajes.

Una expresión XPath se aplica sobre una estructura de árbol de un documento, y podemos obtener cuatro tipos de **resultados**:

- Un conjunto de elementos (nodos).
- Booleano.
- Número.
- Cadena.

2. XPath

Nodos

XPath opera sobre un documento XML como un árbol. El árbol contiene nodos y hay diferentes tipos:

- **Nodo raíz**. No debe confundirse con el elemento raíz del documento, ya que éste último está por debajo de él. Se identifica por “**/**”.

- **Nodos elemento**, son cada uno de los elementos del documento. Todos ellos tienen un padre. Se identifica por “**/**”.

- **Nodos texto**, son aquellos que no están marcados con ninguna etiqueta. Este tipo de nodos no tienen hijos.

2. XPath

Nodos

- **Nodos atributo**, no se consideran hijos las etiquetas añadidas al elemento. Para aquellos atributos definidos con la propiedad **#IMPLIED** en su **DTD** no se crean nodos. Se identifica con **@**.

- **Nodos de comentario y de instrucciones**, son los que se generan para los elementos con comentarios e instrucciones.

- **Nodo actual**, aquél al que nos referimos cuando se evalúa una expresión XPath. Se identifica por “**.**”

- **Tamaño del contexto**, es el número de nodos que se están evaluando en un momento dado en una expresión XPath.

Ejercicio 7. Indica cuáles de los siguientes elementos de un documento XML pueden ser nodos del mismo:
A) Atributos.
B) Comentarios.
C) Etiquetas.
D) Texto.

2. XPath

Nodos

XPath	Descripción
/	El objeto/elemento raíz
.	El objeto/elemento actual
/	El objeto/elemento hijo
..	Operador principal
//	Descenso recursivo
*	Todos los objetos/elementos, independientemente de sus nombres.
@	Acceso a atributos.
[ ]	Operador de subíndice. XPath lo utiliza para iterar sobre colecciones de elementos.
|	El operador de unión en XPath da como resultado una combinación de nodos.
[ ]	Aplica una expresión de filtro (script).

2. XPath

Ruta de localización

Como la representación interna del documento para XPath es un árbol, se puede navegar especificando caminos de una manera similar a como se hace en los directorios de los sistemas operativos. Se corresponde con la ruta que hay que seguir para localizar un nodo.

La evaluación de una **ruta de localización siempre devuelve un conjunto de nodos**, aunque puede estar vacío.

¿Cómo podemos crear una ruta de localización? Mediante la unión de varios pasos de localización.

Lo más importante para tener en cuenta a la hora de crear una expresión es saber el nodo en el que está situado inicialmente (**nodo de contexto**), ya que es desde este que se evaluará.

2. XPath

Ruta de localización

El nodo de contexto al principio es la raíz pero se va moviendo a medida que se van evaluando las expresiones, y por lo tanto podemos expresar los caminos XPath de dos maneras:

- **Caminos absolutos** siempre empiezan en la raíz del árbol. Se pueden identificar porque el primer carácter de la expresión siempre será la raíz "/". No importa el nodo de contexto si con caminos absolutos, porque el resultado siempre será el mismo.

- **Caminos relativos** parten desde el nodo en el que estamos situados.

En el caso de que el nodo contexto no tenga ningún nodo hijo con esa denominación, el valor de la ruta será un elemento vacío.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE clase SYSTEM "clase.dtd">
<clase>
   <asignatura>Lenguajes de Marcas</asignatura>
   <profesor especialidad = "507">
      <nombre>Elena</nombre>
      <apellidos>Escarcena</apellidos>
   </profesor>
   <alumnos>
      <alumno>
         <nombre>Felipe</nombre>
         <apellidos>Rodriguez</apellidos>
      </alumno>
      <alumno>
         <nombre>Maite</nombre>
         <apellidos>Garcia</apellidos>
      </alumno>
   </alumnos>
</clase>
```

Se puede obtener el nodo <nombre> del profesor utilizando la siguiente expresión XPath:

```
/clase/profesor/nombre
```

Hay que tener en cuenta que el resultado de esta expresión no es sólo el contenido del elemento sino todo el elemento <nombre>.

```xml
<nombre>Elena</nombre>
```

Ejercicio 8: Realiza una expresión de XPath para quedarte con el nombre de los alumnos

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE clase SYSTEM "clase.dtd">
<clase>
   <asignatura>Lenguajes de Marcas</asignatura>
   <profesor especialidad = "507">
      <nombre>Elena</nombre>
      <apellidos>Escarcena</apellidos>
   </profesor>
   <alumnos>
      <alumno>
         <nombre>Felipe</nombre>
         <apellidos>Rodriguez</apellidos>
      </alumno>
      <alumno>
         <nombre>Maite</nombre>
         <apellidos>Garcia</apellidos>
      </alumno>
   </alumnos>
</clase>
```

El resultado serán los dos resultados posibles:

```xml
<nombre>Felipe</nombre>
<nombre>Maite</nombre>
```

Ejercicio 9: Realiza una expresión de XPath para quedarte con todos los alumnos

Dará como resultado un subárbol de elementos:

```
<alumnos>
   <alumno>
      <nombre>Felipe</nombre>
      <apellidos>Aranburu</apellidos>
   </alumno>
   <alumno>
      <nombre>Maite</nombre>
      <apellidos>Garcia</apellidos>
   </alumno>
</alumnos>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE clase SYSTEM "clase.dtd">
<clase>
   <asignatura>Lenguajes de Marcas</asignatura>
   <profesor especialidad = "507">
      <nombre>Elena</nombre>
      <apellidos>Escarcena</apellidos>
   </profesor>
   <alumnos>
      <alumno>
         <nombre>Felipe</nombre>
         <apellidos>Rodriguez</apellidos>
      </alumno>
      <alumno>
         <nombre>Maite</nombre>
         <apellidos>Garcia</apellidos>
      </alumno>
   </alumnos>
</clase>
```

2. XPath

Expresiones

¿Qué expresiones podemos utilizar en una expresión XPath?

- **Paréntesis**, “()”; **llaves** , “{}” y **corchetes**, “[ ]”.
- **Atributo**, “@”.
- El nombre de un elemento
- **Tipo de nodo**. Puede haber varios tipos pero el que representa el contenido textual de un elemento o atributo, text() es el más utilizado.

2. XPath

Expresiones - Ejemplo con corchetes “[ ]”

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE clase SYSTEM "clase.dtd">
<clase>
   <asignatura>Lenguajes de Marcas</asignatura>
   <profesor especialidad = "507">
      <nombre>Elena</nombre>
      <apellidos>Escarcena</apellidos>
   </profesor>
   <alumnos>
      <alumno>
         <nombre>Felipe</nombre>
         <apellidos>Rodriguez</ap
      </alumno>
      <alumno>
         <nombre>Maite</nombre>
         <apellidos>Garcia</apell
      </alumno>
   </alumnos>
</clase>
```

```
Si se sabe que una expresión devolverá varios resultados
pero sólo se quiere uno específico se puede usar un número
rodeado por corchetes cuadrados "[]" para indicar qué es
lo que se quiere conseguir.
Para devolver sólo el primer alumno puede hacer lo siguiente:

/clase/alumnos/alumno[1]
```

2. XPath

Expresiones - Ejemplo con corchetes “[ ]”

Devolverá:

```
<alumno>
<nombre>Felipe</nombre>
<apellidos>Aranburu</apellidos>
</alumno>
```

Se pueden usar los corchetes en cualquier lugar de la expresión
para hacer determinar cuál de las ramas se elegirá.
Por ejemplo, se puede obtener sólo el segundo alumno con una expresión como esta:

```
/clase/alumnos/alumno[2]/nombre
```

Ejercicio 10: Realiza una expresión de XPath para seleccionar el cuarto deportista.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE deportistas [...]>
<deportistas>
   <f1 codigo="VET" pais="ALE">Sebastian Vettel</f1>
   <f1 codigo="ALO">Fernando Alonso</f1>
   <f1 codigo="SAI" pais="ESP">Carlos Sainz</f1>
   <tenis>Rafael Nadal</tenis>
</deportistas>
```

2. XPath

Expresiones - Ejemplo atributos

Los valores de los atributos se pueden conseguir especificando el símbolo @ delante del nombre una vez se haya alcanzado el elemento que lo contiene:

```
/clase/profesor/@especialidad
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE clase SYSTEM "clase.dtd">
<clase>
   <asignatura>Lenguajes de Marcas</asignatura>
   <profesor especialidad = "507">
      <nombre>Elena</nombre>
      <apellidos>Escarcena</apellidos>
   rofesor>
   umnos>
   <alumno>
      <nombre>Felipe</nombre>
      <apellidos>Rodriguez</apellidos>
   </alumno>
   <alumno>
      <nombre>Maite</nombre>
      <apellidos>Garcia</apellidos>
   </alumno>
   </alumnos>
</clase>
```

2. XPath

Expresiones - Ejemplo atributos

```
Hay que tener en cuenta que a diferencia de lo que ocurre con los elementos,
al obtener un atributo no tendremos un elemento sino sólo su valor:

especialidad="507"
```

Ejercicio 11: Realiza una expresión de XPath para seleccionar a Alonso.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE deportistas [...]>
<deportistas>
   <f1 codigo="VET" pais="ALE">Sebastian Vettel</f1>
   <f1 codigo="ALO">Fernando Alonso</f1>
   <f1 codigo="SAI" pais="ESP">Carlos Sainz</f1>
   <tenis>Rafael Nadal</tenis>
</deportistas>
```

2. XPath

Expresiones - Ejemplo text( )

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE clase SYSTEM "clase.dtd">
<clase>
   <asignatura>Lenguajes de Marcas</asignatura>
   <profesor especialidad = "507">
      <nombre>Elena</nombre>
      <apellidos>Escarcena</apel
   </profesor>
   <alumnos>
      <alumno>
         <nombre>Felipe</nombre>
         <apellidos>Rodriguez</apellidos>
      </alumno>
      <alumno>
         <nombre>Maite</nombre>
         <apellidos>Garcia</apellidos>
      </alumno>
   </alumnos>
</clase>
```

Para cuando sólo queramos el contenido de un elemento, podemos utilizar la función text() para obtener el contenido:

```
/clase/profesor/nombre/text()
```

2. XPath

Otras expresiones

“*”: si queremos obtener todos los elementos dentro del nodo: ***/clase/profesor/****

Las **"//"** indican que cuadrará con cualquier cosa desde el nodo en el que estamos. Puede ser un solo elemento o un árbol. Por ejemplo, si queremos obtener todos los elementos nombre del archivo independientemente del lugar donde estén dentro del documento: ***//nombre***

Se pueden poner dobles barras en cualquier lugar de la expresión. Aunque facilita la creación de expresiones no es recomendable abusar por motivos de eficiencia. Las expresiones con este comodín requerirán más cálculos para ser evaluadas, y por lo tanto tardarán más en dar resultados. ***/clase/alumnos//nombre***

Operadores: and, or, mod, div, *, /, //, |, +, -, =, !=, <, >, <=, >=.

Separador “::”.

Elemento **actual** "." y elemento **padre** “.."

Coma, “,”.

2. XPath

Otras expresiones

Ejes, son una serie de elementos que permiten hacer referencia a partes del árbol:

- **child::**, selecciona el hijo del elemento actual. Su forma habitual es la barra, **/**, pero puede ponerse: ***/child::***
- **attribute::**, permite seleccionar los atributos del elemento actual.
- **descendant::**, permite seleccionar todos los nodos que descienden del nodo actual. Se corresponde con la doble barra, **//**, aunque se puede usar: ***descendant::***
- **self::**, se refiere al nodo contexto y se corresponde con el punto ”.”.
- **parent::**, selecciona los nodos padre, para referirnos a él usamos los dos puntos, “..”.

Por ejemplo la siguiente expresión: ***/clase/profesor/nombre*** la podríamos escribir utilizando ejes como: ***/child::clase/child::profesor/child::nombre***

Literales, se ponen entre comillas dobles o simples. Pueden anidarse alternando el tipo de comillas.

Números.

Referencias a **variables**, para lo que se utiliza la sintaxis: ***$nombreVariable***

Ejercicio 12. Relaciona los ejes siguientes con sus equivalentes:

A) child::
B) descendant::
C) attribute::
D) parent::

1. @
2. /
3. //
4. ..

2. XPath

Predicado

Es una **expresión booleana** que añade un nivel de verificación al paso de localización.

En estas expresiones podemos incorporar funciones XPath

¿Qué ventajas nos da?

Mediante las rutas de localización se pueden seleccionar varios nodos a la vez, pero el uso de predicados permite seleccionar un nodo con ciertas características.

Los predicados se incluyen dentro de una ruta utilizando los corchetes:

/receta/ingredientes/ingrediente[@codigo="1"]/nombre

En este caso se está indicando al intérprete que escoja, dentro de un fichero XML de recetas, el nombre del ingrediente cuyo código tiene el valor "1".

Ejercicio 13. Partiendo del xml visto en ejercicios anteriores, realiza una expresión con predicado que seleccione sólo los profesores que tengan un elemento `<nombre>` como hijo de `<profesor>`:

2. XPath

Predicado

En el valor de la expresión se pueden especificar caminos relativos desde el nodo que tenga la condición.

Utilizando condiciones se puede hacer una expresión que sólo devuelva el profesor si tiene alumnos.

/clase/profesor[../alumnos/alumno]

Ejercicio 14. Partiendo del xml visto en ejercicios anteriores, realiza una expresión con predicado para obtener los profesores que se llamen “Elena”:

Ejercicio 15. Partiendo del xml visto en ejercicios anteriores, realiza una expresión con predicado para obtener el apellido de la profesora que se llama “Elena"

2. XPath

Predicado

Del mismo modo que con los elementos, se pueden poner condiciones a los atributos para saber si su valor tiene un determinado valor, etc.

/clase/profesor[@especialidad=“507”]

Se pueden mezclar las expresiones con condiciones sobre atributos y sobre elementos para conseguir expresiones más complejas especificando una al lado de la otra.

Ejercicio 16. Partiendo del xml visto en ejercicios anteriores, realiza una expresión con predicado para obtener el profesor que tiene el atributo especialidad en "507" y que se llama "Elena":

2. XPath

Funciones

Entre las más importantes en XPath destacan:

boolean(), al aplicarla sobre un conjunto de nodos devuelve true si no es vacío.

not(), al aplicarla sobre un predicado devuelve true si el predicado es falso, y false si el predicado es verdadero.

true(), devuelve el valor true.

false(), devuelve el valor false.

count(), devuelve el número de nodos que forman un conjunto de nodos.

name(), devuelve un nombre de un nodo.

local-name(), devuelve el nombre del nodo actual o del primer nodo de un conjunto.

namespace-uri(), devuelve el URI del nodo actual o del primer nodo de un conjunto.

2. XPath

Funciones

position(), devuelve la posición de un nodo en su contexto comenzando en 1.

last(), Devuelve el último elemento del conjunto dado.

normalize-space(), permite normalizar los espacios de una cadena de texto, es decir, si en una cadena hay varios espacios consecutivos, esta función lo sustituye por uno solo.

string(), es una función que convierte un objeto en una cadena. Los valores numéricos se convierten en la cadena que los representa teniendo en cuenta que los positivos pierden el signo. Los valores booleanos se convierten en una cadena con su valor, “true” o “false”.

concat(), devuelve dos cadenas de texto concatenadas.

string-length(), devuelve la cantidad de caracteres que forman una cadena de caracteres.

sum(), devuelve la suma de los valores numéricos de cada nodo en un conjunto de nodos determinado.

2. XPath

Funciones - Ejemplos

Para seleccionar los dos primeros elementos de tipo elemento de un fichero XML pondremos:

//elemento[position()<=2]

El ejemplo siguiente devuelve “Xpath permite obtener datos de un fichero XML”.

concat('XPath', 'permite obtener datos de un fichero XML’)

La función **not ()** se utiliza para negar las condiciones:

/clase/profesor[not(@especialidad>”507")]

Si queremos saber cuántos alumnos tenemos:

count(/clase/alumnos/alumno)

Ejercicio 17. Partiendo del xml de las siguientes diapositiva, realiza las siguientes consultas:

1. Nombre del propietario de la agenda.
2. Teléfono de casa del propietario.
3. Nombres y apellidos de los contactos de la agenda.
4. Nombre e identificador de cada contacto.
5. Datos del contacto con identificador “p02".
6. Identificadores de los contactos que tienen móvil.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<agenda>
   <propietario>
      <identificadores>
         <nombre>Blanca</nombre>
         <apellidos>Arnau Gonzalez</apellidos>
      </identificadores>
      <direccion>
         <calle>Calle Real, nº16</calle>
         <localidad>Ceuta</localidad>
         <cp>51002</cp>
      </direccion>
      <telefonos>
         <movil>673898765</movil>
         <casa>956124567</casa>
         <trabajo>628983456</trabajo>
      </telefonos>
   </propietario>
   <contactos>
      <persona id="p01">
         <identificadores>
            <nombre>Ildefonso</nombre>
            <apellidos>Fernandez Bermejo</apellidos>
         </identificadores>
         <direccion>
            <calle>Calle Beatriz de Silva 24, 6B</calle>
            <localidad>Ceuta</localidad>
            <cp>51001</cp>
         </direccion>
         <telefonos>
            <movil>670123123</movil>
         </telefonos>
      </persona>
      <persona id="p02">
         <identificadores>
            <nombre>Roberto</nombre>
            <apellidos>Leria Olmedo</apellidos>
         </identificadores>
         <direccion>
```

```
      </direccion>
      <telefonos>
         <movil>670123123</movil>
      </telefonos>
   </persona>
   <persona id="p02">
      <identificadores>
         <nombre>Roberto</nombre>
         <apellidos>Leria Olmedo</apellidos>
      </identificadores>
      <direccion>
         <calle>Avenida de España 4, 2F</calle>
         <localidad>Ceuta</localidad>
         <cp>51002</cp>
      </direccion>
      <telefonos>
         <movil>670987456</movil>
         <casa>956333323</casa>
      </telefonos>
   </persona>
   <persona id="p03">
      <identificadores>
         <nombre>Nabil</nombre>
         <apellidos>Melouki Mohamed</apellidos>
      </identificadores>
      <direccion>
         <calle>Ciudad de Acicatena 1</calle>
         <localidad>Ceuta</localidad>
         <cp>51001</cp>
      </direccion>
      <telefonos>
         <movil>697564343</movil>
         <casa>956987974</casa>
         <trabajo>677899234</trabajo>
      </telefonos>
   </persona>
</contactos>
</agenda>
```

2. XPath

Uso en JSON - JSONPath

Una **ventaja** que se enfatiza con frecuencia de XML es la disponibilidad de muchas **herramientas** para analizar, transformar y extraer datos de los documentos XML. **XPath** es una de ellas.

Es por ello que se han creado soluciones como **JSONPath** que puede:

- Encontrar y extraer de forma interactiva de las estructuras JSON en el cliente sin scripts especiales.

- Los datos JSON solicitados por el cliente se pueden reducir a las partes relevantes del servidor, reduciendo el uso del ancho de banda en la respuesta del servidor.

2. XPath

Uso en JSON

Debido al hecho de que **JSON** es una **representación natural** de datos para **muchos lenguajes** de programación, las posibilidades son altas de que el lenguaje en particular tenga elementos de **sintaxis** nativos para **acceder** a una estructura JSON.

La siguiente expresión de **XPath**

/store/book[1]/title

se vería para **JSONPath** como

x.store.book[0].title

O bien

x['store']['book'][0]['title']

2. XPath

Uso en JSON

Aquí hay una descripción y una comparación de los elementos de sintaxis JSONPath con sus contrapartes XPath.

XPath	JSONPath	Descripción
/	$	El objeto/elemento raíz
.	@	El objeto/elemento actual
/	. o [ ]	El objeto hijo
..	N/a	Operador principal
//	..	Descenso recursivo
*	*	Comodín. Todos los objetos/elementos, independientemente de sus nombres.
@	N/a	Acceso a atributos. Las estructuras JSON no tienen atributos.
[ ]	[ ]	Operador de subíndice. XPath lo utiliza para iterar sobre colecciones de elementos y para predicados. En JSON es el operador de array nativo.
|	[,]	El operador de unión en XPath da como resultado una combinación de nodos. JSONPath permite nombres alternativos o índices de matriz como un conjunto.
[ ]	?( )	Aplica una expresión de filtro (script).
N/a	( )	Expresión de script, utilizando el motor de script subyacente.
( )	N/a	Agrupación en Xpath

2. XPath

Uso en JSON

Practiquemos las expresiones JSONPath con algunos ejemplos:

```json
{
  "store": {
    "book": [
      {
        "category": "reference",
        "author": "Nigel Rees",
        "title": "Sayings of the Century",
        "price": 8.95
      },
      {
        "category": "fiction",
        "author": "Evelyn Waugh",
        "title": "Sword of Honour",
        "price": 12.99
      },
      {
        "category": "fiction",
        "author": "Herman Melville",
        "title": "Moby Dick",
        "isbn": "0-553-21311-3",
        "price": 8.99
      },
      {
        "category": "fiction",
        "author": "J. R. R. Tolkien",
        "title": "The Lord of the Rings",
        "isbn": "0-395-19395-8",
        "price": 22.99
      }
    ],
    "bicycle": {
      "color": "red",
      "price": 19.95
    }
  }
}
```

2. XPath

Uso en JSON

Practiquemos las expresiones JSONPath con algunos ejemplos:

XPath	JSONPath	Descripción
/store/book/author	$.store.book[*].author	Los autores de todos los libros de la tienda
//author	$..author	Todos los autores
/store/*	$.store.*	Todas las cosas que hay reservado, que son algunos libros y una bicicleta.
/store//price	$.store..price	El precio de todo en la tienda.
//book[3]	$..book[2]	El tercer libro
//book[last()]	$..book[(@.length-1)] $..book[-1:]	El último libro en orden.
//book[position()<3]	$..book[0,1] $..book[:2]	Los dos primeros libros
//book[isbn]	$..book[?(@.isbn)]	Filtrar todos los libros con el isbn
//book[price<10]	$..book[?(@.price<10)]	Filtrar todos los libros más baratos que 10
//*	$..*	Todos los elementos del documento.

Ejercicio 18. Partiendo de los xml vistos en ejercicios de XPath anteriores, pásalos a formato JSON y repite la realización de consultas.

Instalar la siguiente extensión en VSCode: https://marketplace.visualstudio.com/items?itemName=weijunyu.vscode-json-path

Recursos online

Recursos online

- W3Schools XML XPath
- W3Schools JSON
- Tutorialspoint - XPath
- Tutorialspoint - JSON
- JSONPath - XPath for JSON

Bibliografía

Bibliografía

- Lenguajes de Marcas y Sistemas de Gestión de Información. Síntesis.

- Lenguajes de Marcas y Sistemas de Gestión de Información. Paraninfo.

- Lenguajes de Marcas y Sistemas de Gestión de Información. Garceta.

- Entradas de Wikipedia (EN)

Fin