

XML - 63319 - Juan Ignacio García Vautrin Raggio

Uniones - xpath (cc = context)

Nodes: ②

- 2.1 • node(): lista de nodos cc. $\neq 1$ $\neq 1$
 - 2.2 • Count(node) → ej: 3.3, 3.4.
 - 2.3 • Position(): Position of cc node / Use it for requirements.
 - 2.4 • last() → ej: 2.3
 - 2.5 • Combinación de 2+ Node-sets
- //*[contains(name(), 'w')] //attribute::#[contains(name(), 'ID')]

Bool: ④

- 4.1 • not(boolean) → ej: 2.1
- 4.2 • =, !=, <, >, <=, >=

Selectors ⑤ A lo largo de todo un modo test, sirve para filtrar entre el cc.

- 5.1 • self::, child::, ancestor::, ancestor-or-self::
- 5.2 • descendant::, descendant-or-self::
- 5.3 • attribute::
- 5.4 • parent::

- 5.5 • preceding:: → Padre/nº inmediato (anterior)
 - 5.6 • preceding-sibling:: → Antes que este ancla del cc. (TODO)
- //Profile[not(.//name = preceding::Profile/@name)]

Si no hay un anterior con igual nombre, lo selecciona.

Para selección de elementos distintos

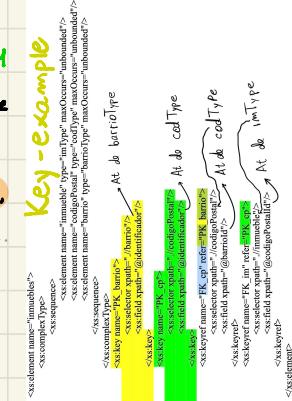
5.7 • preceding-sibling:: → Selección los que este arriba del cc y es "hermano".

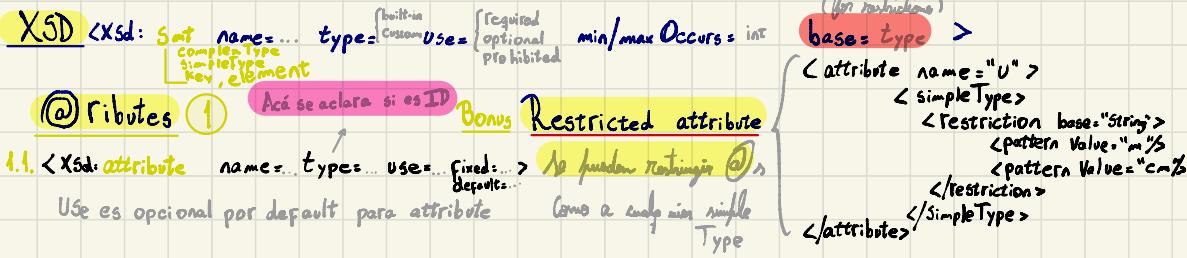
Si queremos seleccionar los que no se refieren dentro de un mismo "SCOPE", pero no nos importa si están o no en el resto del DOC, es decir si se refiere en otro lado, elegir igual

Anomalías • Redundancia de datos.

- de Actualización: 2 ramas distintas elementos con el mismo ID, si cambia algo en la rama 1, lo 2 ~~(x)~~
- de Pobrada: Un objeto contiene elementos con ID, si borro ese objeto, pierde el elemento.

Field: N-Campo @... ; Selector: Path a campo/atributo





Restrictions ② - base = SimpleType / complexType

2.1 Regex

`<xsd:pattern value="..."/>`

`2.1.1 ? x==0 || x==1`

`2.1.5 Id = [0-9]`

`2.1.2 * 0<x`

`2.1.6 \n Newline`

`2.1.3 + 1<x`

`2.1.7 \r Carriage Return`

`2.1.4 . Wildcard`

`2.1.8 \t tab`

2.2 Limits (for char limit use regex)

`2.2.1 <min/max Inclusive Value = "Limit">`

`2.2.2 <totalDigits Value = "Limit">` FD included.

`2.2.3 <fractionDigits Value = "Limit">`

`<restriction base="decimal">`
`<minInclusive value="100"/>` Guarda 2 de los 5 dígitos para los
`<max value="250"/>` dígitos 000100.2000 ✓, 250.1 x
`<totalDigits value="5"/>`
`<fractionDigits value="2"/>` se ignoran los 0 o extra.

`</restriction>`

2.2.4. Enumeration value = "valor fijo"/>

2.3 Union

Valido si el elem cumple con uno de los Type i

2.4 List

Uno tipo de enumeración lista de valores posibles.

`<list itemType="SimpleType"/>`

Valido si se tiene una lista (separada por espacio c/o) y todos cumplen

Si es float, lo siguiente valido:

`<Vary> 1 2 0.3 2.5 </Vary>`

③ ComplexType: Sequence / choice / All

3.1 Sequence : secuencia de elementos que DEBEN respetar el orden

Elements siempre dentro de alguno de ellos

Límite

3.2 Choice : Pool de elementos de los que se puede elegir 1 (si aparece `<choice maxOccurs="x">`)

3.3 All : Pool de elementos DEBEN estar todos pero en cualquier orden.

4. Extension (es la misma que restriccion pero el nombre, tambien se usa `base = "baseType"`)

5 DTD

DocType Descriptor

4.1 De complexType a complexType

5.1 Def elementos: <!ELEMENT nombre Content> (5.2)

```
<complexType name="tipoDefinido">
  <complexContent>
    <extension base="ComplexTypeBase">
      <sequence>
        <element ... />
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

5.2 Content: 5.2.1 EMPTY; 5.2.2 (Lista de subelementos) (separados: , ; |)

5.2.3 (#PCDATA) TXT sin subelementos o φ (<...>); 5.2.4 (#PCDATA1...) Mixto.

5.3 Def attributes <!ATTLIST elementoNombre atributo tipo valorDef>

5.3.1 elementoNombre: Elemento al que pertenece el atributo

5.3.2 atr:Tipo: 5.3.2.1 CDATA . TXT; 5.3.2.3 ID: Si el valor es unico en el Doc;

5.3.2.2 (list): separados por | ; 5.3.2.4 IDREFS: ID/frente/s en otros elemento/s;

#REQUIRED: Obligatorio

#FIXED "Valor": Siempre == Valor

#IMPLIED: Opcional

"Valor": Valor es default (puede Sobreescribir)