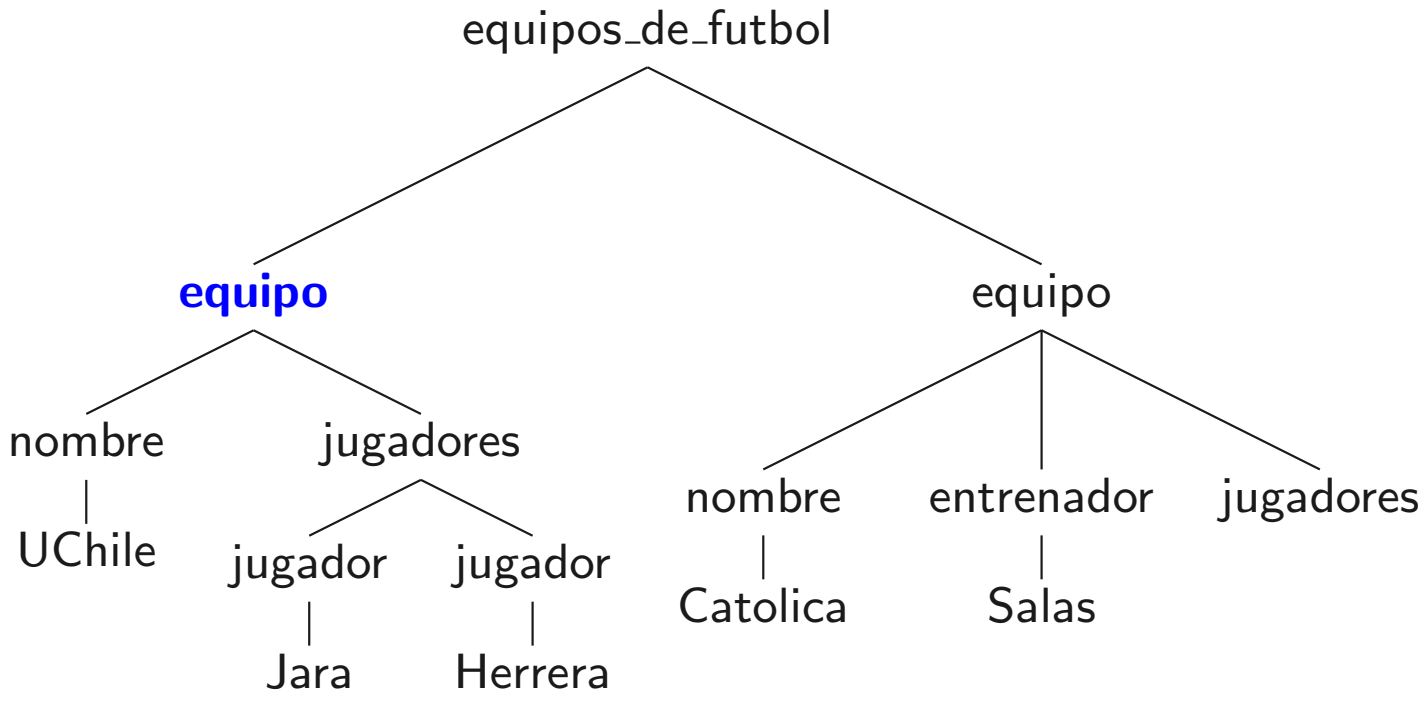
XML

Extraer información de XML

- Deseamos obtener información de forma declarativa.
- Con un lenguaje estándar.
- Que sea:
  - Lo más simple posible.
  - Lo más eficiente posible.
  - Lo más expresivo posible.

Notar que los últimos puntos se contraponen.

- Dado un documento y un nodo:



¿Qué tipo de consultas deseamos hacer?

XPath: XML Path Language

- Es EL lenguaje para extraer nodos en XML.
- Usado también para computar datos y valores.
- Es un estandár de la W3C.
  - Versión 1.0 - 2.0
- Es una base para tecnologìas como:
  - XML Schema.
  - XSLT.
  - XQuery.

Cómo funciona XPath

- Basado en el documento XML visto como árbol.
- XPath nos permite:
  - navegar el árbol XML.
  - filtrar los nodos.
  - extraer valores.

Primero: extracción de nodos

Formas de navegar el árbol

- hacia abajo (hijos)
- hacia el lado (hermano)
- hacia arriba (padre)
- Recursivamente (ancestro, descendiente, hermanos)

Expresiones de camino

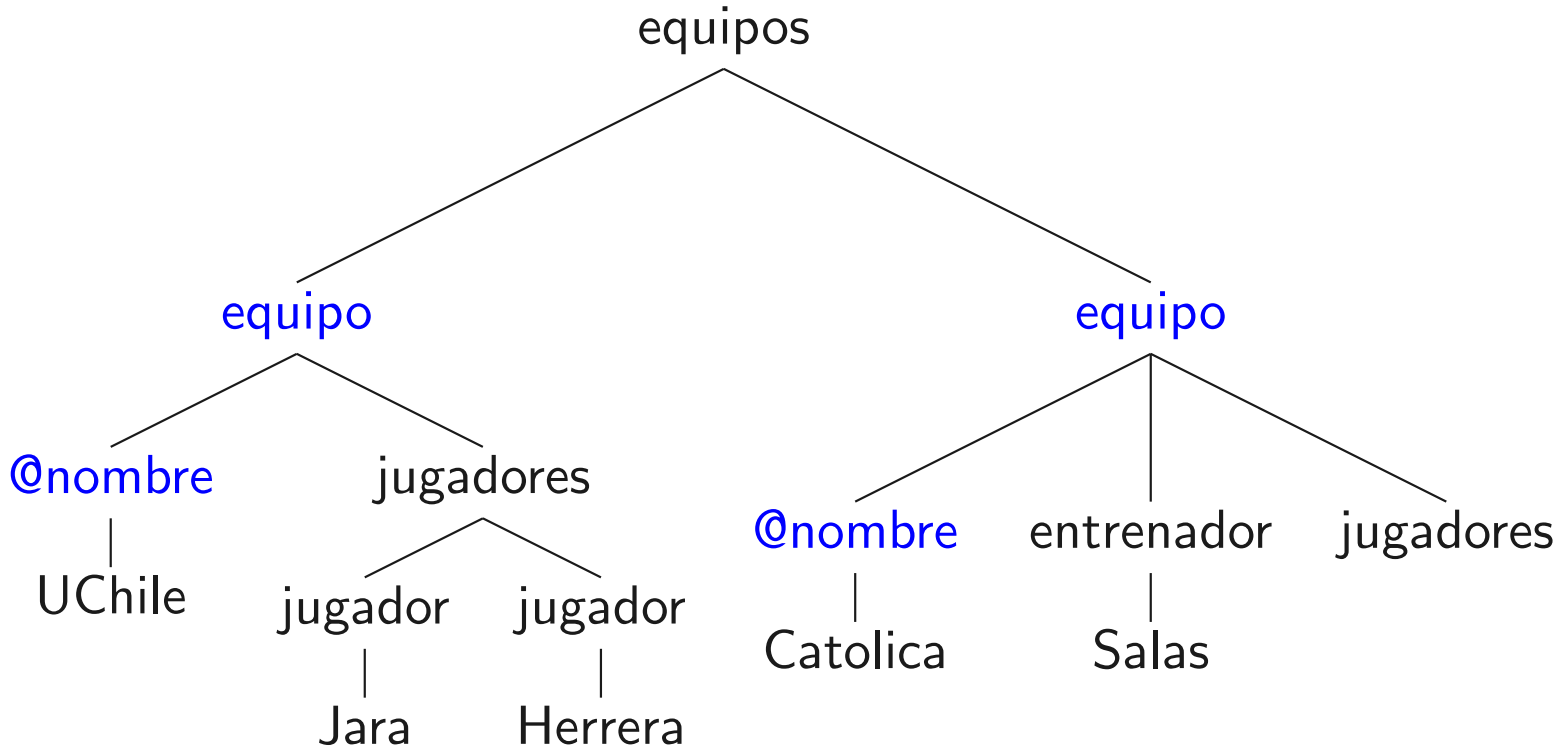
Una expresión de camino típicamente comienza en la raíz, y consiste en una secuencia

$$/T_1/T_2/\cdots/T_n$$

de tags y slashes.

La idea es ir evaluando los tags uno a uno. Cuando se busca el valor de un atributo, se antepone @.

Ejemplos



- /equipos/equipo
- /equipos/equipo/@*nombre*

Ejes en XPath

- Un eje es la consulta básica de XPath.

En realidad, /equipos/equipo/@nombre es una abreviación para:

/child::equipos/child::equipo/attribute::nombre

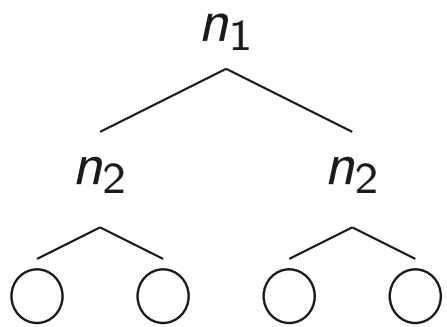
Ejes en XPath

- Un eje es la consulta básica de XPath.
- Se definen 13 ejes distintos.
- Sea D un documento XML.
- Sea N el conjunto de todos los nodos en D.

Un eje es una relación $A \subseteq N \times N$.

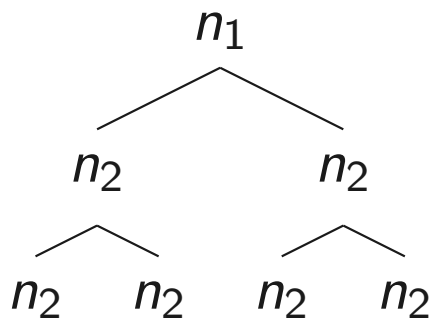
Ejes en XPath

- `self` $= \{(n, n) \mid n$ es un nodo en $D\}$
- `child` $= \{(n_1, n_2) \mid n_2$ es hijo de n_1 en $D\}$



Ejes en XPath

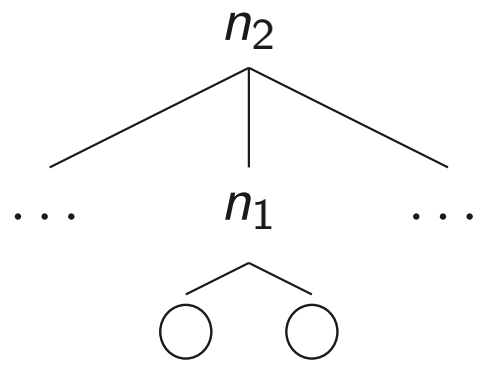
- $\texttt{descendant} = \{(n_1, n_2) \mid n_2 \text{ es descendiente de } n_1 \text{ en } D\}$



- $\texttt{descendant-or-self} = \texttt{descendant} \cup \texttt{self}$

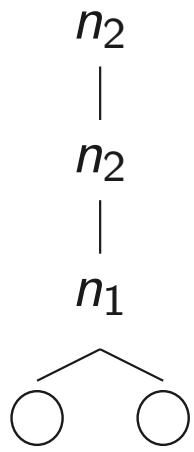
Ejes en XPath

- $\texttt{parent} = \{(n_1, n_2) \mid n_2 \text{ es padre de } n_1 \text{ en } D\}$



Ejes en XPath

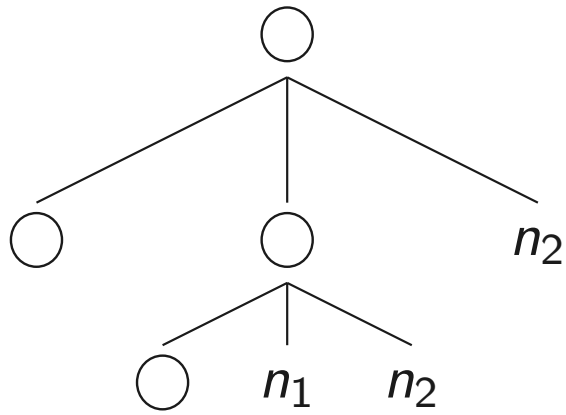
- $\texttt{ancestor} = \{(n_1, n_2) \mid n_2 \text{ es ancestro de } n_1 \text{ en } D\}$



- $\texttt{ancestor-or-self} = \texttt{ancestor} \cup \texttt{self}$

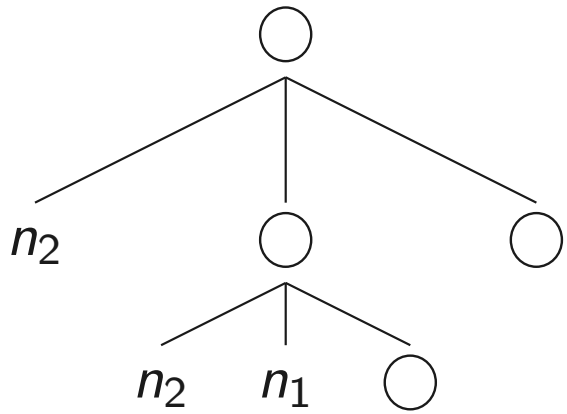
Ejes en XPath

- `following` $= \{(n_1, n_2) \mid n_2$ sucede a n_1 en $D\}$



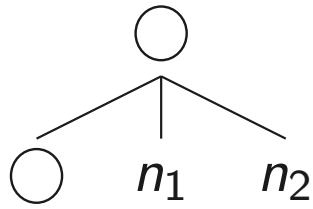
Ejes en XPath

- `preceding` $= \{(n_1, n_2) \mid n_2$ precede a n_1 en $D\}$

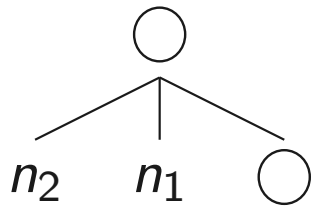


Ejes en XPath

- `following-sibling`
  $= \{(n_1, n_2) \mid n_2$ es hermano derecho de n_1 en $D\}$



- `preceding-sibling`
  $= \{(n_1, n_2) \mid n_2$ es hermano izquierdo de n_1 en $D\}$



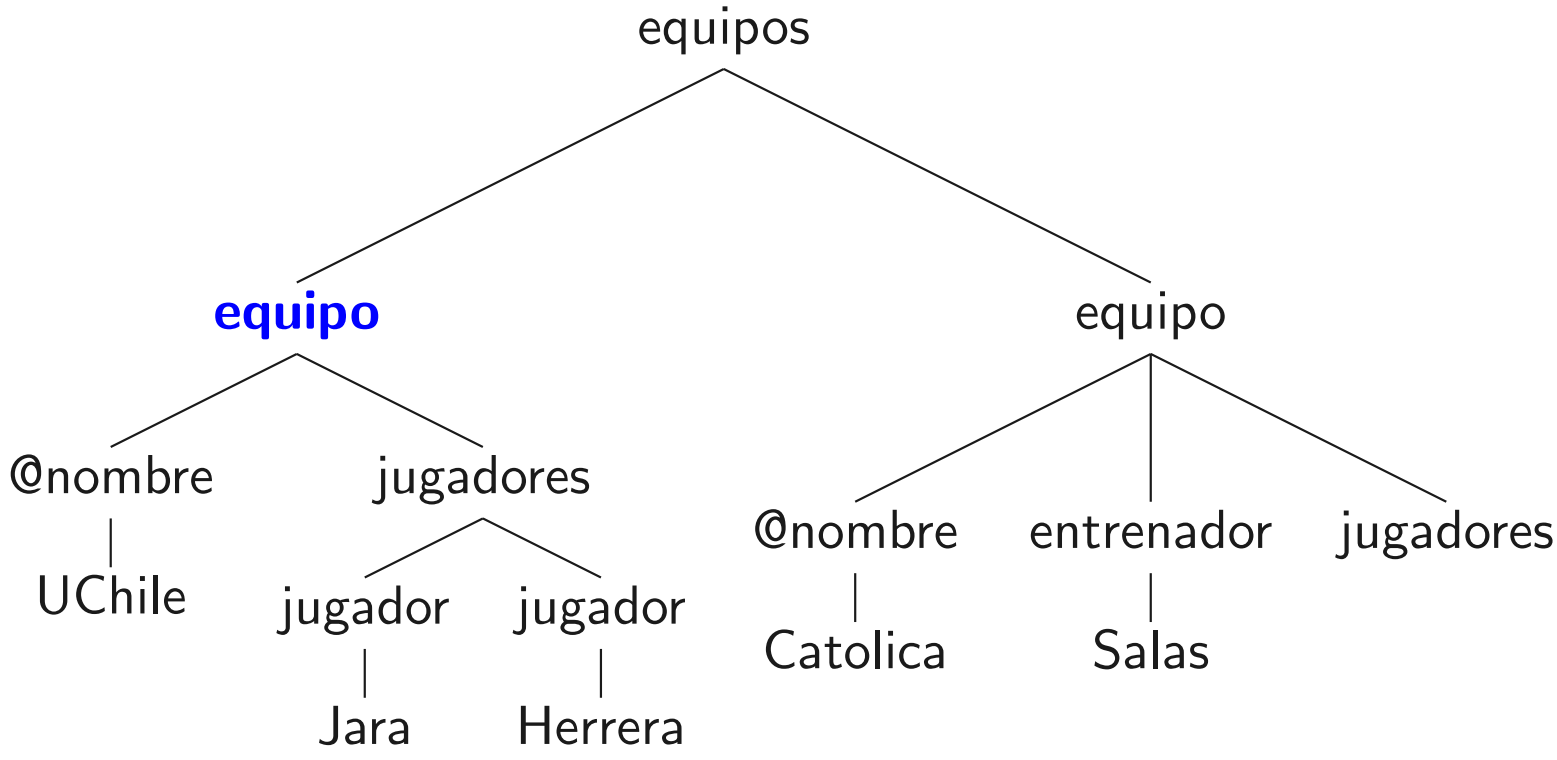
Para qué nos sirven estos ejes

- Dado un nodo, con ellos podemos hacer consultas simples.
- Sintaxis:

```
paso = eje::tag
```

- Ejemplos:
  - `child::equipo`
  - `descendant::jugador`
- El operador `*` selecciona cualquier tag.
- Ejemplos:
  - `parents::*`
  - `following::*`

Ejemplos


equipos
equipo
equipo
@nombre
jugadores
@nombre
entrenador
jugadores
UChile
jugador
jugador
Catolica
Salas
Jara
Herrera

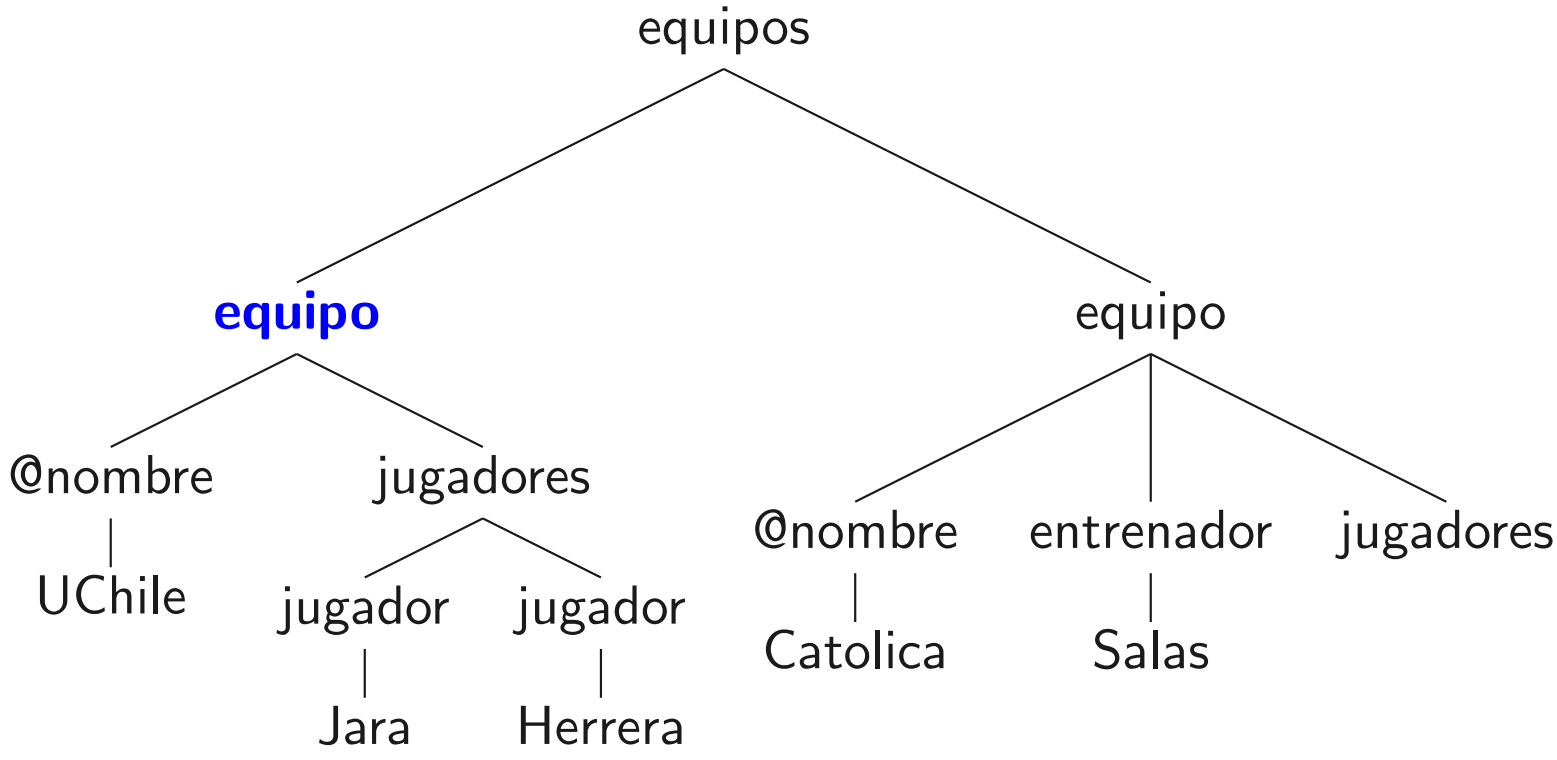
Consultas más complejas

- Navegando a través del árbol XML.
- Dando pasos con los ejes.
- Sintaxis:

```
camino = paso / paso
```

- Ejemplo:
  - `child::jugadores / child::jugador`
  - `parent::* / descendant::jugador`

Ejemplos de clases



Consultas en XPath

- Consultas absolutas comienzan desde el nodo raíz.
- Se antepone un / al comienzo de la consulta.
- Ejemplo:
  - `/child::equipos`
  - `/descendant::jugador`

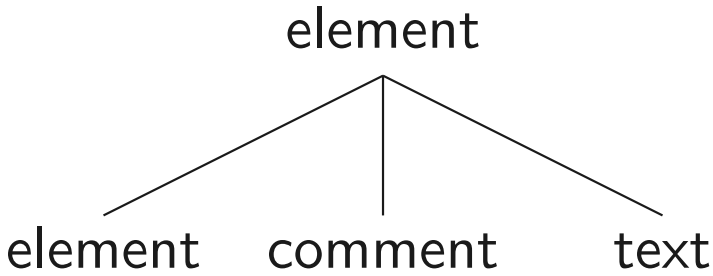
Modelo de árbol XPath

- XPath considera un modelo especial de árbol.
- En este modelo todos sus elementos son nodos.
- El modelo comprende distintos tipos de nodos.

Modelo de árbol XPath

Nodo element (el típico):

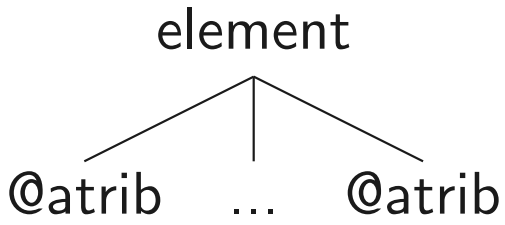
- Cada tag es representado con un nodo element.
- Los siguientes tipos de nodo heredan de element:
  - Element
  - Comment
  - Text



Modelo de árbol XPath

Nodo attribute:

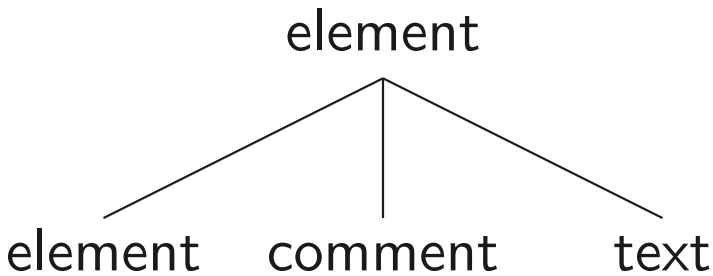
- Por cada atributo hay un nodo attribute.
- Un element es padre del nodo attribute.
- El acceso al nodo attribute es especial.
- El orden entre attributes no es importante.



Modelo de árbol XPath

Nodo comment y nodo text:

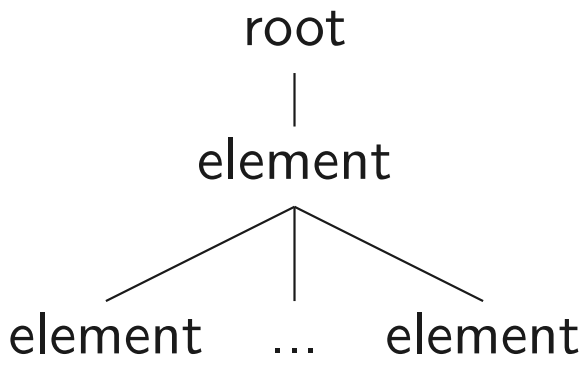
- Cada comentario es representado por un nodo comment.
- Cada PCDATA es representado por un nodo text.
- Ambos nodos pueden ser hijos de un nodo element.
- Aparecen según el orden del documento.



Modelo de árbol XPath

Nodo root:

- Ancestro de todos los nodos.
- No es parte del documento.
- La raíz del documento es hijo de este nodo.



Tag o nodos generales

- Sintaxis:

```
paso = eje::tag | eje::tipo()
```

- Existen test generales para nodos:
  - `text()` (toma el hijo de este nodo que corresponde a texto)
  - `comment()` (comentarios dentro del scope de este nodo)
  - `node()` (toma el nodo)

- Ejemplo:
  - `child::entrenador/child::text()`
  - `/child::comment`

Abreviaciones para operadores

- Existen abreviaciones para algunos operadores:

	$\rightarrow$	`child::`
`@`	$\rightarrow$	`attribute::`
`.`	$\rightarrow$	`self::node()`
`..`	$\rightarrow$	`parent::node()`
`//`	$\rightarrow$	`descendant-or-self::node()`

- Ejemplos:
  - `/equipos//jugador`
  - `..//equipo/@nombre`

Condiciones

A medida que avanza la consulta, quizá nos queremos enfocar sólo en algunos de los nodos.

- Sintaxis:

```
paso = eje::tag[filtro]
```

- El filtro puede ser cualquier operacion que retorne verdadero o falso...
  - Comparaciones de valores: $<$, $<=$, $=$, etc...
  - Operaciones *or* y *and*: [precio $=$ 10 or precio $=$ 20]

Condiciones

A medida que avanza la consulta, quizá nos queremos enfocar sólo en algunos de los nodos.

- Sintaxis:

```
paso = eje::tag[filtro]
```

- El filtro puede ser una misma consulta XPath.
- Ejemplos:
  - `//equipo[//jugador]`
  - `//jugadores[ancestor::equipo/@nombre = "UC"]`

Para filtrar es necesario usar funciones

Funciones (hay mucho más)

- Posición de un nodo:
  - `//jugador[1]`
  - Selecciona el primer hijo del nodo jugador.
- Contiene subelemento estadio:
  - `//equipo[estadio]`
- Strings en general:
  - `//jugador[contains(text(), "Jara")]`

Selección de dos nodos a la vez

- Ejemplo:
  - `//jugador/text() | //entrenador/text()`

Expresiones

- Toda consulta en XPath es una expresión.
- Es posible hasta calcular dentro en una consulta.
- Ejemplo:
  - `2*3 - 5`
- Funciones para calcular valores de nodo.
- Ejemplo:
  - `count(//equipo[entrenador])`