TP Nº 4: XPATH

Introducción

Desde hace un tiempo viene tomando fuerza en el mundo de Internet, la idea de derribar los "silos de información" que representa cada sitio Web, en favor de la portabilidad de los datos entre todos los sitios Web. El objetivo que se busca es estandarizar de algún modo la forma en que se transmite y comparte la información entre sitios Web, la forma en que se definen los intereses de un usuario, un evento público en cierto lugar, la autenticación y autorización entre sitios web, etc.

Si hablamos de buscar estandarizar la forma en que se comparte de forma portable la información entre sitios Web, pensamos entonces en XML como tipo de documento. Así es como surgió hace algunos años el proyecto http://dataportability.org/ que nuclea a diversas iniciativas tendientes a desarrollar estándares para cada uno de los puntos mencionados previamente.

Entre estos estándares, podemos mencionar:

- OpenID: intenta unificar la autenticación a un sitio web
- OAuth: protocolo para otorgar autorizaciones entre aplicaciones o sitios web
- RSS: ya visto en el TP 1 para el intercambio de noticias.
- APML (Attention Profiling Markup Language)
- OPML, Microformats, RDF, XMPP, etc

A lo largo de este TP trabajaremos con APML, un estándar basado en XML que intenta homogeneizar la forma en que un usuario de Internet puede describir sus intereses personales. En un escenario ideal, los sitios Web podrían aceptar la entrada de un APML por parte del usuario y entregar de esta forma contenido personalizado según los intereses de éste.

Para poder hacer las consultas pedidas en esta guía, utilizar alguno de los siguientes motores de XPath online (algunos de ellos vistos durante la clase):

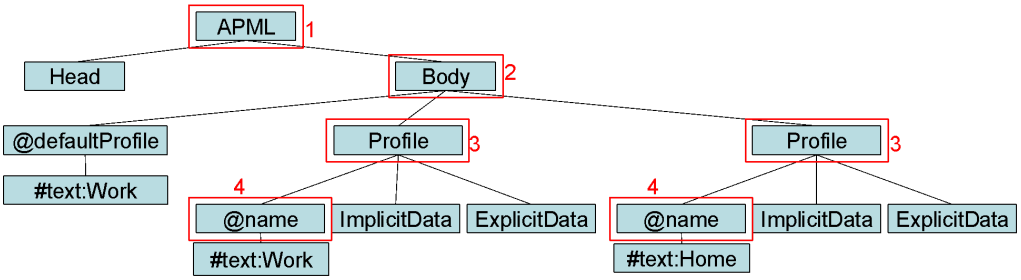
- XPathTester: http://www.xpathtester.com/xpath
- Free Formatter: https://www.freeformatter.com/xpath-tester.html
- CodeBeautify: https://codebeautify.org/Xpath-Tester
- XPather: http://xpather.com/a74jlWTw
- Cualquier otro XPath engine online disponible

Ejercicio 1

Para el archivo apml1.xml, escribir las expresiones Xpath para poder obtener:

1. Los nombres de todos los perfiles de atención del usuario. En este caso la respuesta esperada sería: "Work, Home".

   En el siguiente diagrama se representa la porción del XML que nos interesa. Y marcamos en rojo los nodos que deberemos recorrer para llegar a la respuesta:

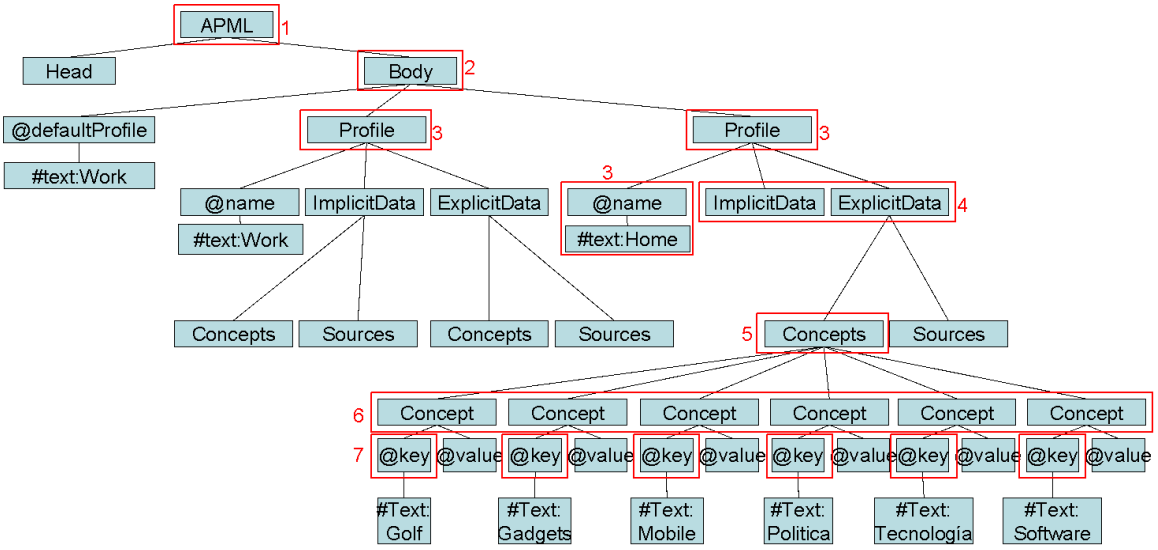


La expresión xpath que buscamos es entonces:
`/APML/Body/Profile/@name`
Esta expresión retornará el resultado esperado: Work, Home.

2. El listado de los nombres (key) de los conceptos en el perfil de atención de intereses "Home". En este caso la respuesta esperada sería: "Golf, Gadgets, Mobile, Politica, Tecnologia, Software".

   En esta oportunidad, al llegar al nodo "Profile" se nos pide que filtremos el perfil de atención de nombre "Home", lo que haremos mediante una condición adicional: @name="Home" aplicada al nodo Profile.

   El perfil de atención tiene tanto datos explícitos como implícitos, y queremos recuperar los conceptos que figuren en ambos. Lograremos este efecto navegando desde el nodo Profile hasta el nodo Concepts con un * que machee con ExplicitData y con ImplicitData.

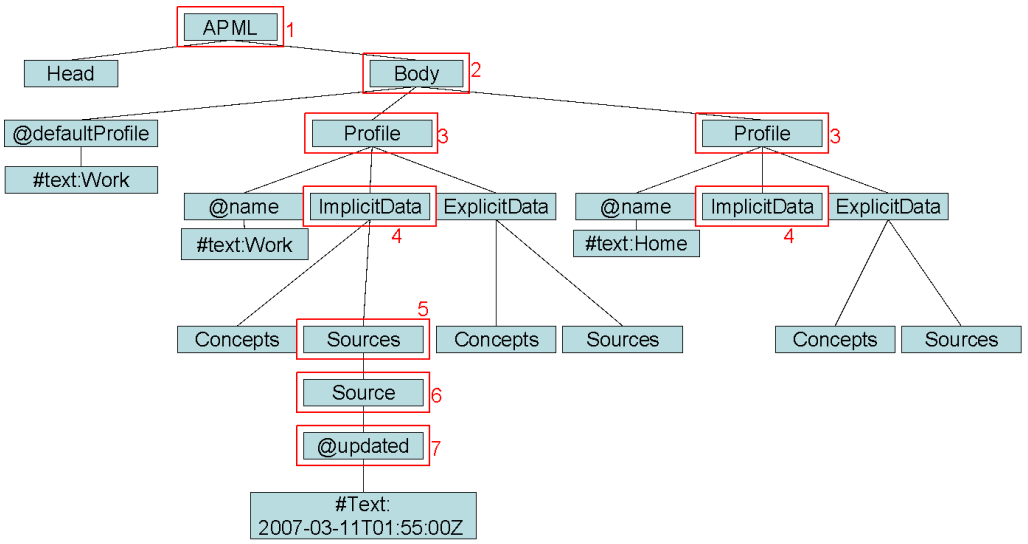


La expresión xpath que retorna el resultado esperado es entonces:
`/APML/Body/Profile[@name="Home"]/*/Concepts/Concept/@key`

3. El listado de los nombres (key) de los conceptos en todos los perfiles de atención . En este caso la respuesta esperada sería: "attention, content distribution, information, business, alerting, intelligent agents, development, services, direct attention, Golf, Gadgets, Mobile, Politica, Tecnologia, Software"

   Este caso es similar al del punto 2, solo que esta vez no se nos pide filtrar por un perfil de atención en particular. Por lo tanto, la expresión xpath buscada resulta:
   `/APML/Body/Profile/*/Concepts/Concept/@key`

4. La fecha de actualización de cada una de las fuentes (sources) de los datos implícitos. En este caso la respuesta esperada sería: “2007-03-11T01:55:00Z”.



En este caso, recorremos los nodos hasta llegar a la fecha de actualización de la fuente.
`/APML/Body/Profile/ImplicitData/Sources/Source/@updated`

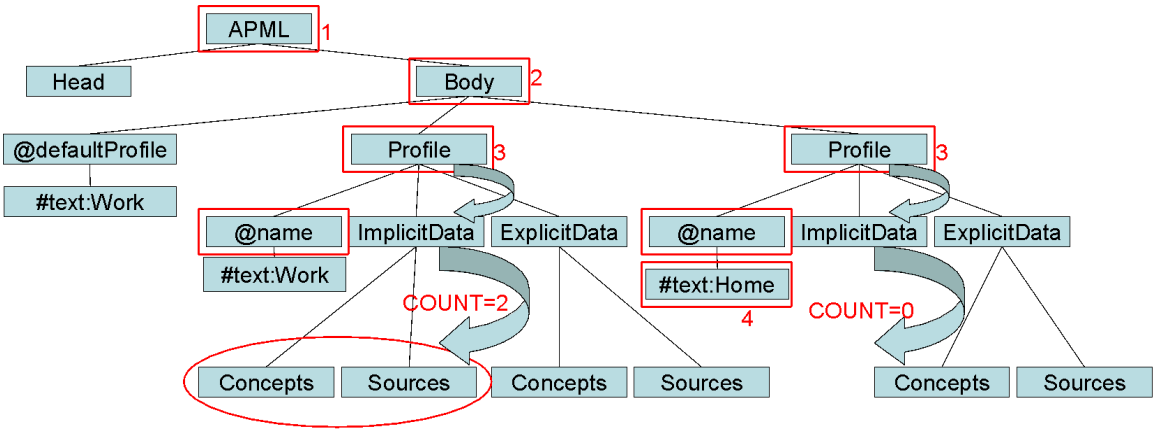
5. Los nombres de todos los perfiles de atención donde los datos implícitos asociados es un conjunto vacío. En este caso la respuesta sería: “Home”

   Para resolver esta consulta, vamos a partir de la consulta del punto 1.
   Sabemos que la expresión

   ```
   /APML/Body/Profile/@name
   ```

   retorna los nombres de todos los perfiles.
   Nos falta entonces agregar la condición de que los datos implícitos asociados al perfil sean un conjunto vacío.
   Una forma de obtener este resultado es pedir que la cantidad de elementos de ImplicitData sea cero. Esto lo hacemos con la función count de xpath:

   `/APML/Body/Profile[count(ImplicitData/*)=0]/@name`



También podríamos usar la función not de xpath. Preguntando por aquellos ImplicitData que no contengan elementos:

```
/APML/Body/Profile[not(ImplicitData/*)]/@name
```

Ejercicio 2

Para el archivo apml1.xml, escribir las expresiones Xpath para poder obtener:

1. Los conceptos (no importa en qué perfil de atención) que refieren al término "business" en alguna parte. En este caso la respuesta sería: "<Concept key="business" value="0.93" from="GatheringTool.com" updated="2009-03-11T01:55:00Z" />"

   Como se nos pide una lista de conceptos como resultados, lo primero que haremos será navegar hasta los conceptos. Esto lo hacemos con la expresión xpath:

   ```
   /APML/Body/Profile/*/Concepts/Concept
   ```

   Para completar el ejercicio, deberemos filtrar ahora aquellos conceptos que refieran al término "business" en alguna parte. Entonces, preguntaremos qué atributos incluyen la palabra "business" en su texto, utilizando la condición (evaluada sobre un concepto):

   ```
   contains(., "business")
   ```

   Entonces, la expresión resultante será:

   ```
   /APML/Body/Profile/*/Concepts/Concept/attribute::*[contains(.,"business")]/..
   ```

2. Los conceptos cuyo valor sea mayor o igual a 0.95 En este caso la respuesta sería: "<Concept key="attention" value="0.99" from="GatheringTool.com" updated="2007-03-11T01:55:00Z" />, <Concept key="content distribution" value="0.97" from="GatheringTool.com" updated="2008-03-11T01:55:00Z" />, <Concept key="information" value="0.95" from="GatheringTool.com" updated="2008-03-11T01:55:00Z" />, <Concept key="direct attention" value="0.99" />"

   En este caso también navegaremos hacia los conceptos.

Pero en este caso, se nos pide comparar el contenido del atributo “value” contra un valor fijo 0.95. La comparación utilizará el operador relacional “>=”.

```
/APML/Body/Profile/*/Concepts/Concept[@value>=0.95]
```

3. Los nombres de los perfiles de atención del usuario que tengan menos conceptos que el promedio de conceptos por perfil. En este caso la respuesta sería: “Home”

   En este caso, se nos pide como resultado final “los nombres de los perfiles de atención”. Para navegar hasta esos nodos, utilizamos la expresión:

   ```
   /APML/Body/Profile/@name
   ```

   Pero, deberemos filtrar los perfiles “que tengan menos conceptos que el promedio de conceptos por perfil”.

   Centrémonos en la primer parte “perfiles de atención que tengan menos conceptos que”. Supongamos por un instante que queremos aquellos perfiles con menos conceptos que un valor fijo (para el ejemplo usaremos 10).
   Para esa condición necesitamos un count de los conceptos del perfil actual, y luego comparar mediante un operador relacional el resultado.

   ```
   /APML/Body/Profile[
         count(*/Concepts/Concept) < 10
      ]/@name
   ```

   Ahora, faltaría cambiar el valor fijo 10 por el pedido original de la consigna: “el promedio de conceptos por perfil”.
   ¿Cómo se calcula el promedio de conceptos por perfil? Necesitamos conocer qué cantidad de conceptos hay en el xml completo y qué cantidad de perfiles hay y luego, efectuar la división de ambos valores.

   La expresión xpath que nos da la cantidad de conceptos en el xml (15) es:
   ```
   count(//APML/Body/Profile/*/Concepts/Concept)
   ```

   La expresión xpath que nos da la cantidad de perfiles de atención (2) es:
   ```
   count(//APML/Body/Profile)
   ```

   Y finalmente el promedio de conceptos por perfil (7,5) será:
   ```
   count(//APML/Body/Profile/*/Concepts/Concept) div
   count(//APML/Body/Profile)
   ```

   Juntando las expresiones anteriores, en una única expresión xpath que resuelva el ejercicio, llegamos a:
   ```
   /APML/Body/Profile[
         count(*/Concepts/Concept)
         <
         (
            count(//APML/Body/Profile/*/Concepts/Concept)
            div
   ```

```
        count(//APML/Body/Profile)
    )
]/@name
```

4. Los nombres (key) de los conceptos cuyo nombre (key) empiecen con una vocal. En este caso la respuesta sería: “attention, information, alerting, intelligent agents”

   En este caso, se nos pide retornar los nombres (key) de los conceptos. Como vimos en ejercicios anteriores, la expresión xpath para lograr esto es:
   `/APML/Body/Profile/*/Concepts/Concept/@key`

   Pero aun nos falta filtrar esos resultados, retornando solo aquellos keys cuyo texto comience con una vocal. Para saber si un texto comienza con un string determinado, utilizamos la función xpath “starts-with”. Y en este caso particular, preguntamos si comienza con cada una de las vocales:

```
/APML/Body/Profile/*/Concepts/Concept/@key[ starts-with(.,"a") ] |
/APML/Body/Profile/*/Concepts/Concept/@key[ starts-with(.,"e")] |
/APML/Body/Profile/*/Concepts/Concept/@key[ starts-with(.,"i")] |
/APML/Body/Profile/*/Concepts/Concept/@key[ starts-with(.,"o")] |
/APML/Body/Profile/*/Concepts/Concept/@key[ starts-with(.,"u")]
```

```
/APML/Body/Profile/*/Concepts/Concept/@key[ starts-with(.,"a") or
starts-with(.,"e") or starts-with(.,"i") or starts-with(.,"o") or
starts-with(.,"u")]
```

Ejercicio 3

Para el archivo apml1.xml, escribir las expresiones Xpath para poder obtener:

1. El peso promedio de los conceptos de su perfil de intereses personal, del trabajo y para todos los perfiles juntos.

Empecemos por el peso promedio de todos los perfiles juntos. Para calcular el peso promedio, tengo que saber la suma de todos los pesos y dividirla por la cantidad de pesos.

La suma de los pesos se expresa como:

```
sum(/APML/Body/Profile/*/Concepts/Concept/@value)
-> 9.25
```

La cantidad de pesos se expresa como:

```
count(/APML/Body/Profile/*/Concepts/Concept/@value)
-> 15
```

En base a las dos expresiones anteriores podemos calcular el promedio:

```
sum(/APML/Body/Profile/*/Concepts/Concept/@value) div
   count(/APML/Body/Profile/*/Concepts/Concept/@value)
-> 0.61
```

Para calcular el promedio por perfil, simplemente aplicamos el filtro del perfil requerido en la expresión anterior.

```
sum(/APML/Body/Profile[@name="Home"]/*/Concepts/Concept/@value) div
   count(/APML/Body/Profile[@name="Home"]/*/Concepts/Concept/@value)
-> 0.15
```

```
sum(/APML/Body/Profile[@name="Work"]/*/Concepts/Concept/@value) div
   count(/APML/Body/Profile[@name="Work"]/*/Concepts/Concept/@value)
-> 0.927777777777778
```

2. El peso promedio de los conceptos para todos los perfiles juntos, agrupados por las fechas de actualización.

   Para resolver este ejercicio necesitaríamos crear nodos nuevos que permitieran expresar el resultado de la agrupación:

   <resultado><fecha>2009-09-28</fecha><promedio>0.98</promedio></resultado>
   <resultado><fecha>2009-09-01</fecha><promedio>0.95</promedio></resultado>

   Para ello, necesitaríamos un lenguaje más potente como es XQuery.

3. La cantidad de fuentes que conforman todos sus perfiles de atención. En este caso la respuesta sería 5.

   Para obtener la cantidad de fuentes, simplemente navegamos hasta las fuentes y aplicamos la función xPath count:

```
count(/APML/Body/Profile/*/Sources/Source)
```

4. El promedio de la cantidad de fuentes de cada uno de los perfiles.

   Al igual que en el caso del ejercicio 2.3, no se pueden crear nuevos nodos, en donde se muestren el nombre del perfil y el promedio de fuentes de cada uno.

Ejercicio 4

Para el archivo apml1.xml, escribir las expresiones Xpath para poder obtener:

1. El nombre del primer perfil de atención. En este caso la respuesta sería “Work”.

   ```
   /APML/Body/Profile[1]/@name
   ```

2. El nombre (key) del primer concepto del último perfil de atención. En este caso la respuesta sería “Golf”.

   ```
   /APML/Body/Profile[last()]/*/Concepts/Concept[1]/@key
   ```

3. Los nombres (key) del tercer concepto (en caso de existir) en los datos implícitos de cada uno de los perfiles de atención. En este caso la respuesta sería: “information”

   ```
   /APML/Body/Profile/ImplicitData/Concepts/Concept[3]/@key
   ```

Ejercicio 5

Para el archivo apml1.xml, escribir las expresiones Xpath para poder obtener:

1. Todos los nodos ancestros de cada uno de los nodos del perfil de atención. En este caso la respuesta sería: "<Body defaultprofile="Work"> ... </Body>, <APML xmlns="http://www.apml.org/apml-0.6" version="0.6" > ... </APML>"

   `/APML/Body/Profile/ancestor::*`

2. Todos los nodos descendientes de los nodos fuentes (sources). En este caso la respuesta sería: "<Sources> <Source key="http://feeds.feedburner.com/apmlspec" name="APML.org" value="1.00" type="application/rss+xml" from="GatheringTool.com" updated="2007-03-11T01:55:00Z"> … </Source> </Sources>, <Sources> <Source key="http://feeds.feedburner.com/TechCrunch" name="Techcrunch" type="application/rss+xml" value="0.4"> … </Source> </Sources>, <Sources> <Source key="http://feeds.feedburner.com/TechCrunch" name="Techcrunch" type="application/atom+xml" value="0.4"> … </Source> <Source key="http://www.uberbin.net/feed" name="Uberbin" type="application/rss+xml" value="0.2"> … </Source> <Source key="http://lanacion.com.ar/herramientas/rss/index.asp" name="LaNacion.com" type="application/rss+xml" value="0.3"> … </Source> </Sources><Sources…>…</Source>…<Author>…</Author>"

```
/APML/Body/Profile/*/Sources/descendant-or-self::*
```

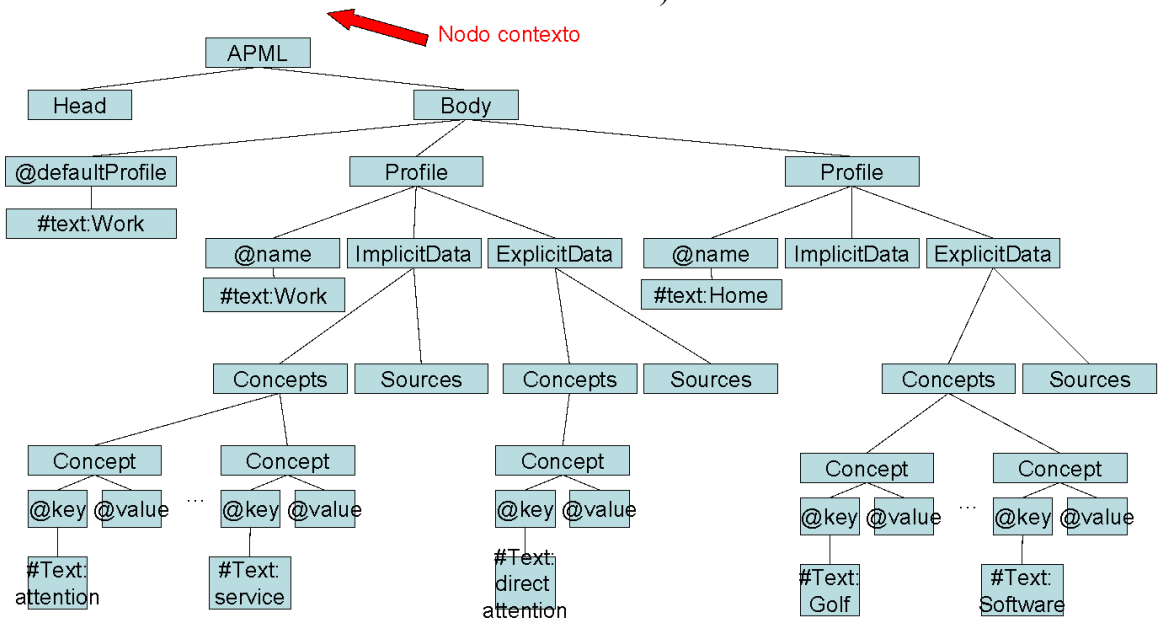
Ejercicio 6

Graficar el nodo contextual para cada uno de las sub-expresiones generadas en las consultas, para los ejercicios 2.1 y 4.3.

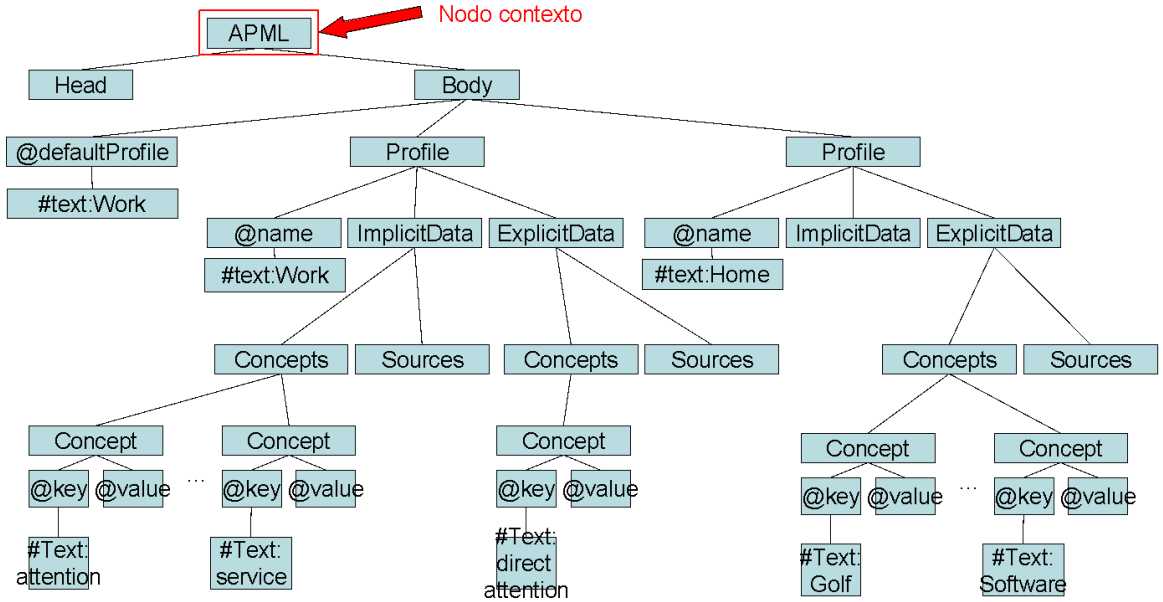
2.1) **`/APML/Body/Profile/*/Concepts/Concept[contains(attribute::*,"business")]`**

Para graficar el paso a paso vamos a tomar como gráfico base un subset del XML que muestra los nodos que se evalúan en la expresión.

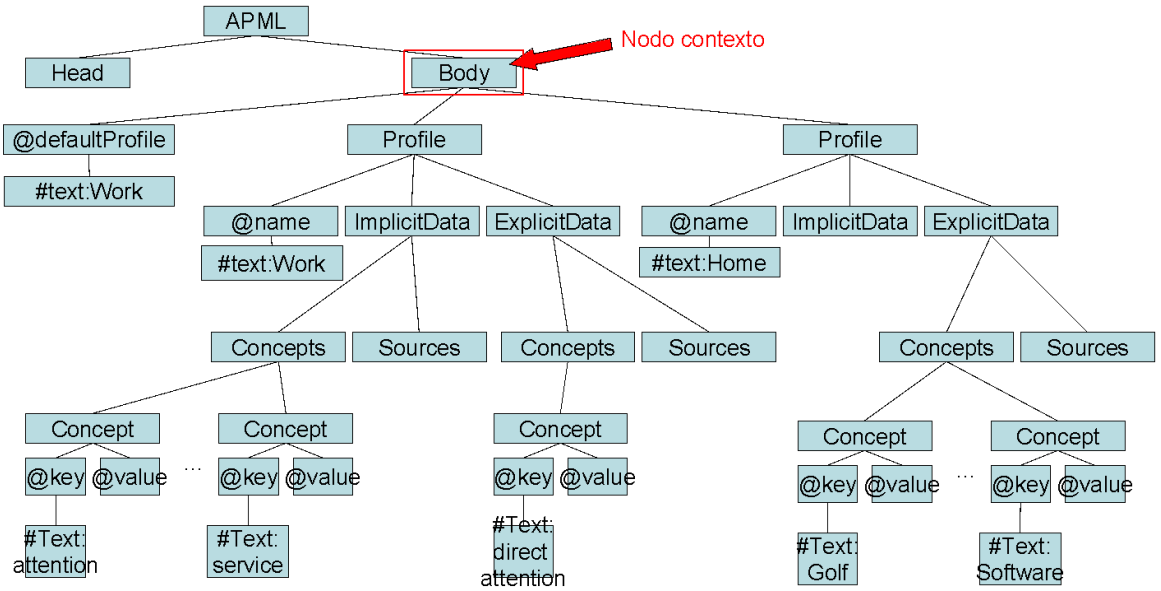
Antes de comenzar a evaluarse la expresión, el *nodo context*o actual está referenciando fuera del único nodo del árbol XML, normalmente se dice que apunta a la raíz (o sea, para XPath la raíz no es el nodo *APML* sino está fuera del árbol).



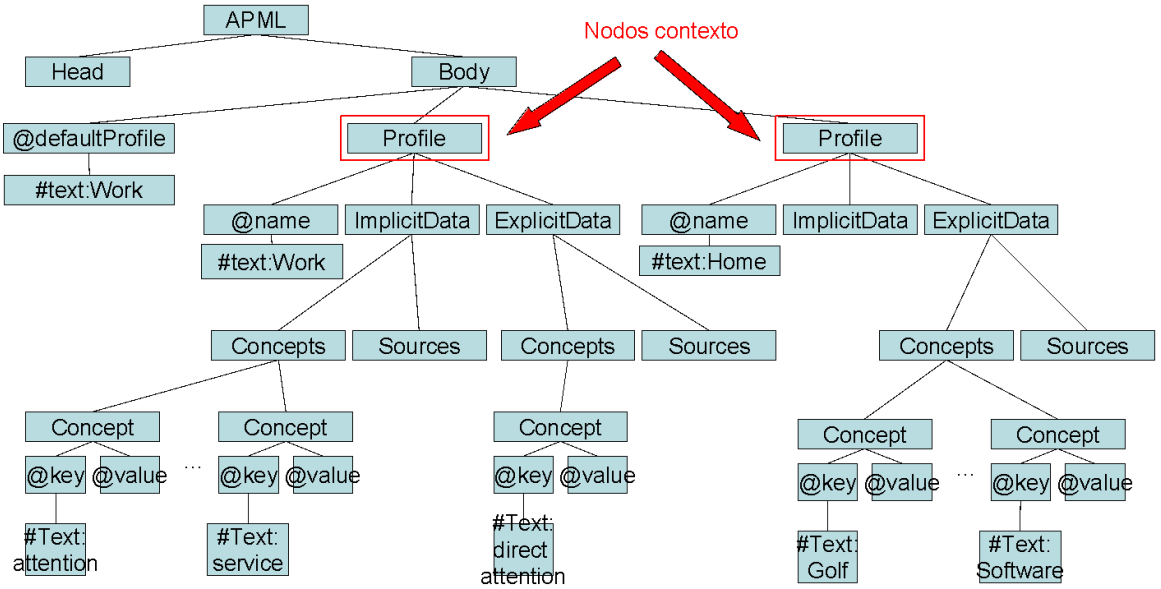
Al evaluar la primera subexpresión se selecciona como *nuevos nodos contextos* a los nodos hijos llamados **APML** (en este caso uno solo). Más precisamente, se usó como axis a **child::** ***y como nodo test a*** *APML*. O sea, el nuevo nodo contexto resulta ser***:***



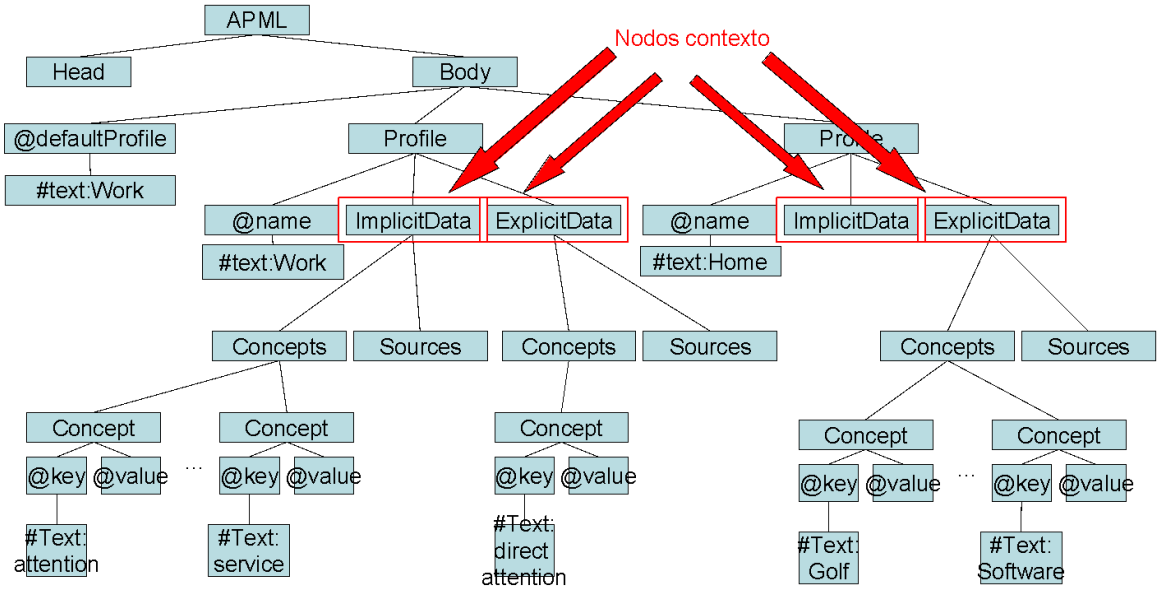
Al evaluar la segunda subexpresión se selecciona como *nuevos nodos contextos* a los nodos hijos (de los nodos contexto actuales) que se llamen **Body** (en este caso es 1). Más precisamente, se usó como axis a **child::** y como nodo test a **Body**. O sea, los nuevos nodos ***contextos resultan ser:***



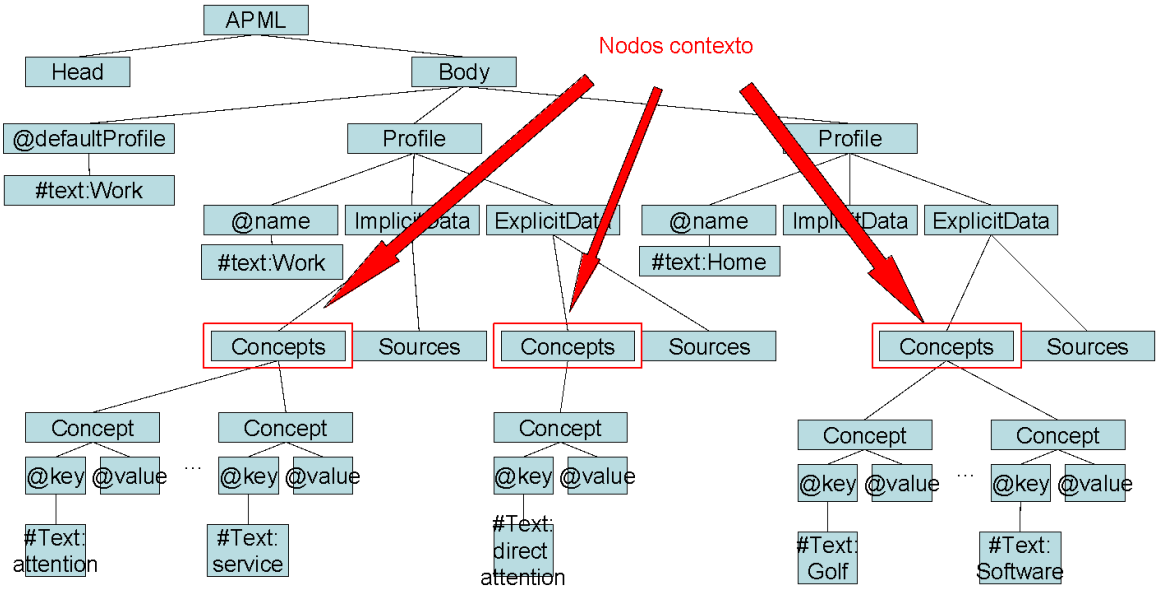
Al evaluar la tercera subexpresión se selecciona como *nuevos nodos contextos* a los nodos hijos (de los nodos contexto actuales) que se llamen **Profile** (en este caso son 2). Más precisamente, se usó como axis a **child::** y como nodo test a **Profile**. O sea, los nuevos nodos contextos resultan ser:



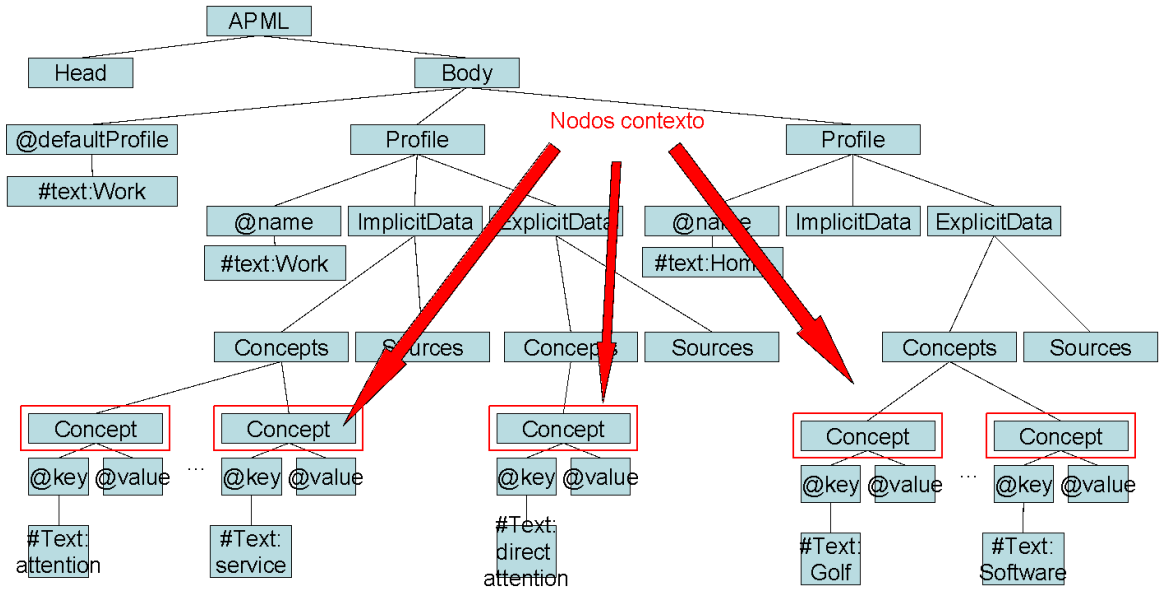
Al evaluar la cuarta subexpresión se selecciona como *nuevos nodos contextos* a TODOS los nodos hijos de los nodos contexto actuales (en este caso son 2). Más precisamente, se usó como axis a **child::** y como nodo test a *. O sea, los nuevos nodos contextos resultan ser:

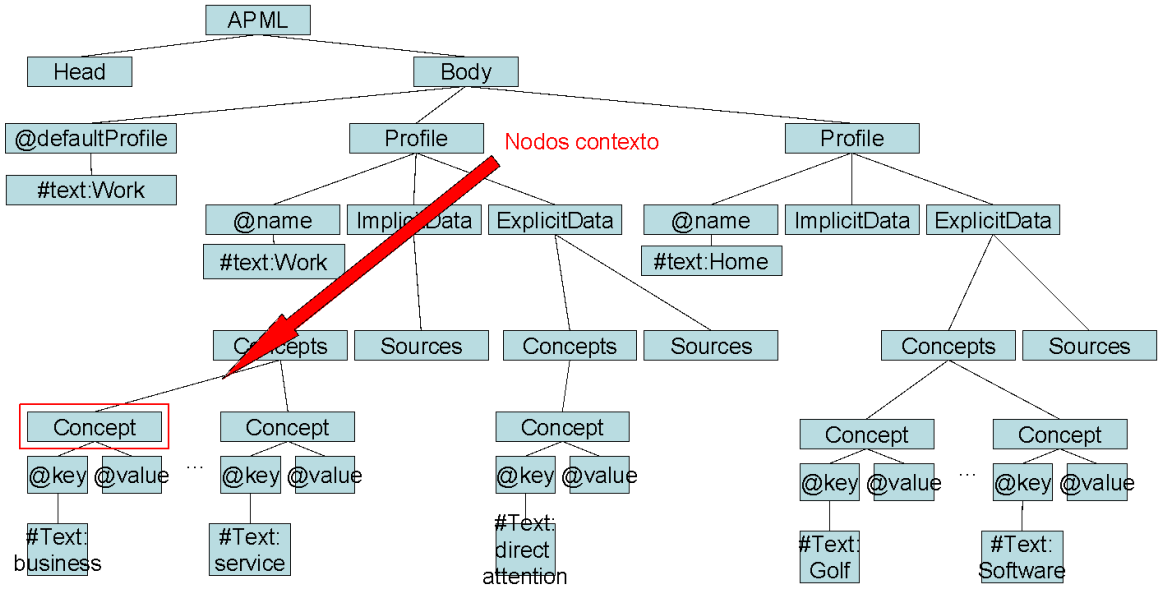


Al evaluar la quinta subexpresión se selecciona como *nuevos nodos contextos* a los nodos hijos (de los nodos contexto actuales) que se llamen **Concepts** (en este caso son 3). Más precisamente, se usó como axis a **child::** y como nodo test a **Concepts**. O sea, los nuevos nodos contextos resultan ser:



Se seleccionan los nodos contexto **Concept** y se evalúa por cada uno de ellos el predicado. Al terminar la primera evaluación, queda el únicos nodo contexto que tienen un atributo con **"business"** en su texto.

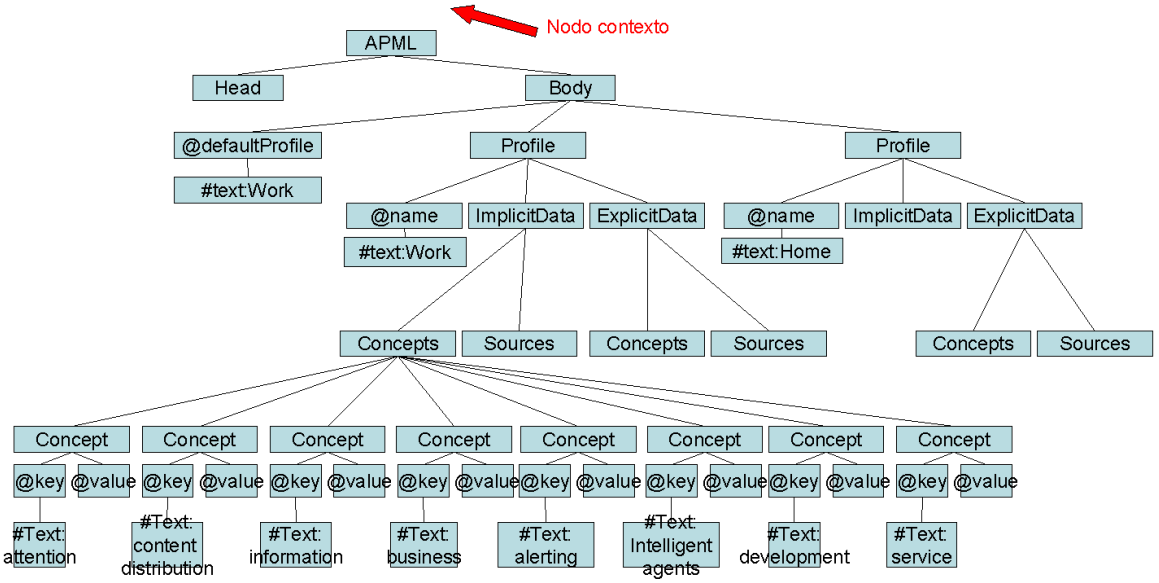

APML
Head
Body
@defaultProfile
#text:Work
Profile
Nodos contexto
Profile
@name
ImplicitData
ExplicitData
#text:Work
@name
ImplicitData
ExplicitData
#text:Home
Concepts
Sources
Concepts
Sources
Concepts
Sources
Concept
Concept
Concept
Concept
Concept
@key
@value
@key
@value
@key
@value
@key
@value
@key
@value
#Text: attention
#Text: service
#Text: direct attention
#Text: Golf
#Text: Software


APML
Head
Body
@defaultProfile
#text:Work
Profile
Nodos contexto
Profile
@name
ImplicitData
ExplicitData
#text:Work
@name
ImplicitData
ExplicitData
#text:Home
Concepts
Sources
Concepts
Sources
Concepts
Sources
Concept
Concept
Concept
Concept
Concept
@key
@value
@key
@value
@key
@value
@key
@value
@key
@value
#Text: business
#Text: service
#Text: direct attention
#Text: Golf
#Text: Software

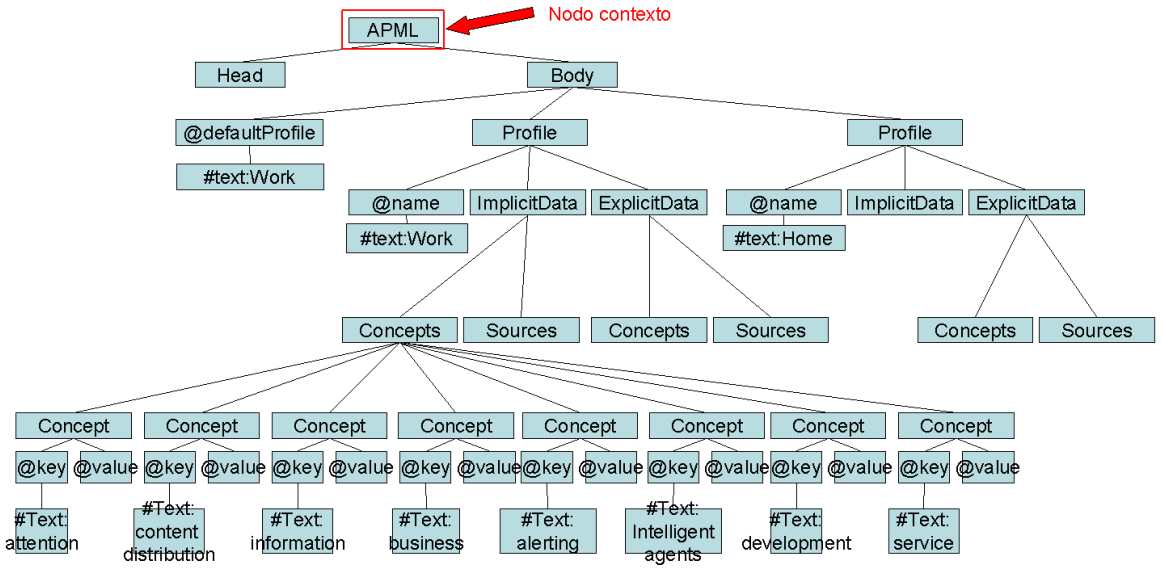
4.3) `/APML/Body/Profile/ImplicitData/Concepts/Concept[3]/@key`

Para graficar el paso a paso vamos a tomar como gráfico base un subset del XML que muestra los nodos que se evalúan en la expresión.

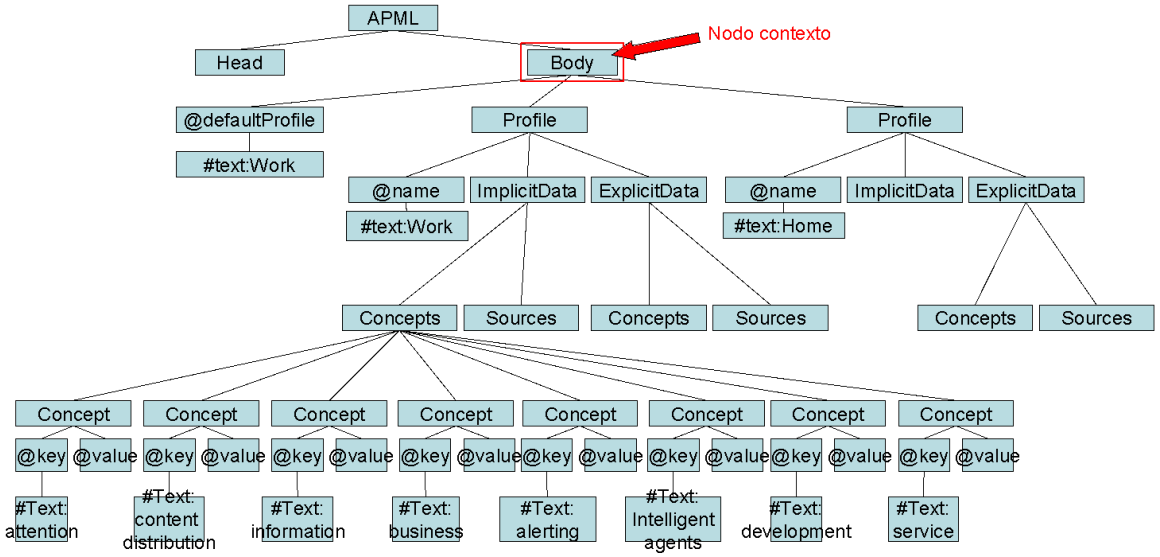
Antes de comenzar a evaluarse la expresión, el *nodo context*o actual está referenciando fuera del único nodo del árbol XML, normalmente se dice que apunta a la raíz (o sea, para XPath la raíz no es el nodo *APML* sino está fuera del árbol).



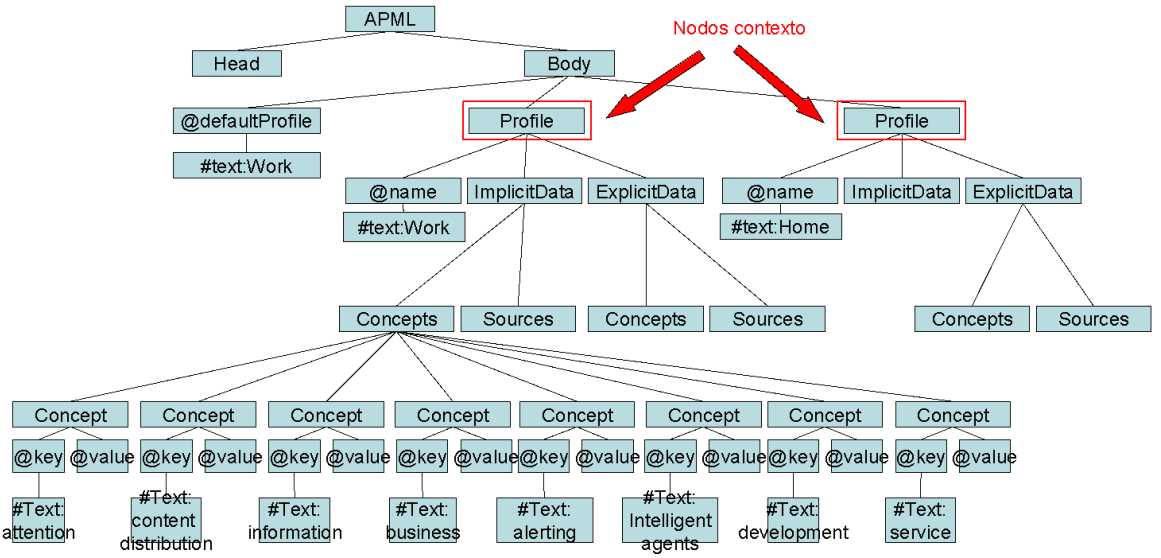
Al evaluar la primera subexpresión se selecciona como *nuevos nodos contextos* a los nodos hijos llamados **APML** (en este caso uno solo). Más precisamente, se usó como axis a **child::** ***y como nodo test a*** *APML*. ***O sea, el nuevo nodo contexto resulta ser:***



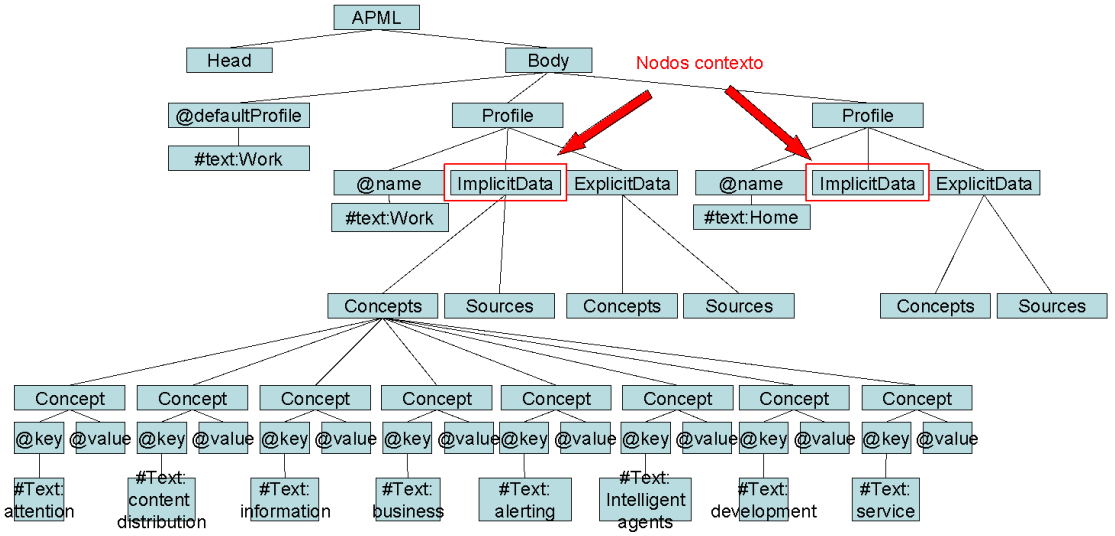
Al evaluar la segunda subexpresión se selecciona como *nuevos nodos contextos* a los nodos hijos (de los nodos contexto actuales) que se llamen **Body** (en este caso es 1). Más precisamente, se usó como axis a **child::** y como nodo test a **Body**. O sea, los nuevos nodos ***contextos resultan ser:***



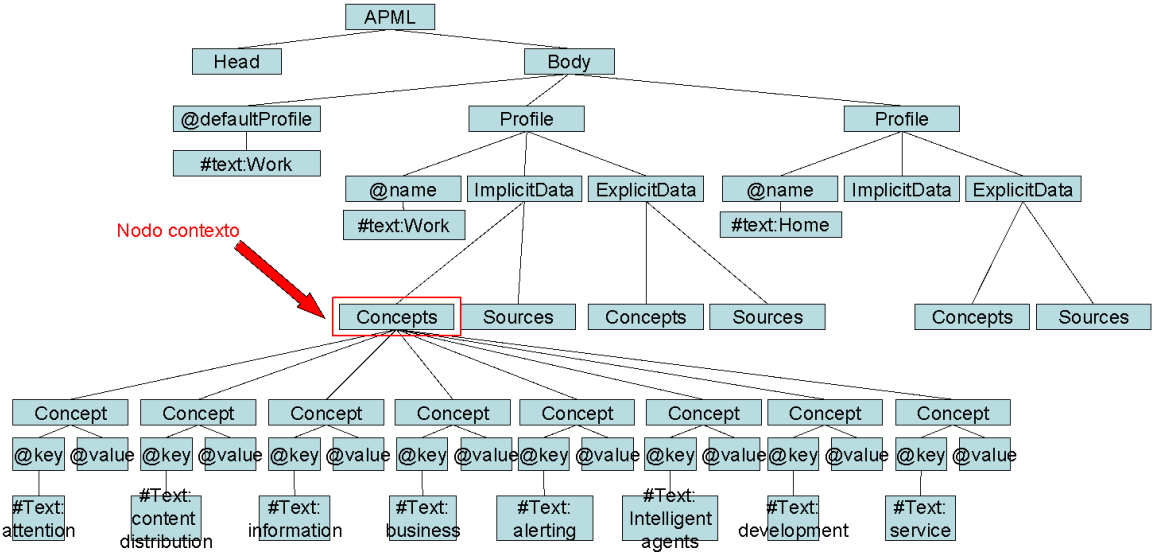
Al evaluar la tercera subexpresión se selecciona como *nuevos nodos contextos* a los nodos hijos (de los nodos contexto actuales) que se llamen **Profile** (en este caso son 2). Más precisamente, se usó como axis a **child::** y como nodo test a **Profile**. O sea, los nuevos nodos contextos resultan ser:



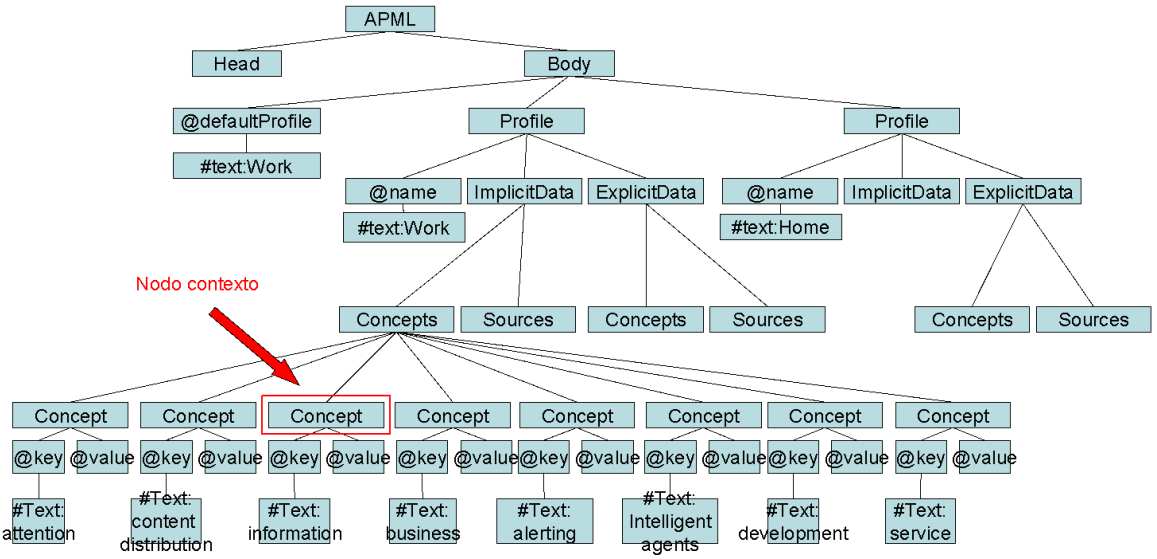
Al evaluar la cuarta subexpresión se selecciona como *nuevos nodos contextos* a los nodos hijos (de los nodos contexto actuales) que se llamen **ImplicitData** (en este caso son 2). Más precisamente, se usó como axis a **child::** y como nodo test a **ImplicitData**. O sea, los nuevos nodos contextos resultan ser:



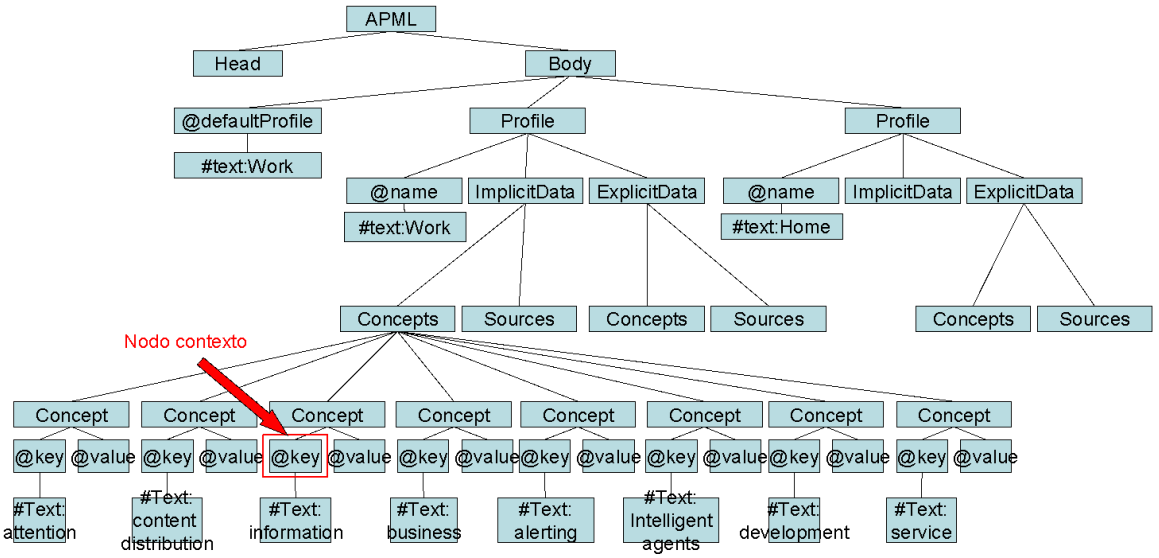
Al evaluar la quinta subexpresión se selecciona como *nuevos nodos contextos* a los nodos hijos (de los nodos contexto actuales) que se llamen **Concepts** (en este caso es 1). Más precisamente, se usó como axis a **child::** y como nodo test a **Concepts**. O sea, los nuevos nodos contextos resultan ser:



Al evaluar la sexta subexpresión se selecciona como *nuevos nodos contextos* a los nodos hijos (de los nodos contexto actuales) que se llamen **Concept y que ocupen la posición número 3** (en este caso es 1). Más precisamente, se usó como axis a **child::** como nodo test a **Concept,** y como condición **position()=3**. O sea, los nuevos nodos contextos resultan ser:



Al evaluar la séptima subexpresión se selecciona como *nuevos nodos contextos* a los nodos hijos (de los nodos contexto actuales) que se llamen **key** (en este caso es 1). Más precisamente, se usó como axis a **attribute::** y como nodo test a **key**. O sea, los nuevos nodos contextos resultan ser:



Ya no hay nada más para evaluar y el resultado de la expresión XPath es la siguiente lista de nodos (NO es un XML), o sea los *nodos contexto actuales* de la última expresión:

“information”

Ejercicio 7

Para el archivo apml1.xml, escribir las expresiones Xpath para poder obtener:

1. Los perfiles de atención distintos que tiene el usuario.

   Si quisiera retornar “los perfiles de atención del usuario”, podría hacerlo, tal como venimos haciendo en la práctica, con la expresión xpath siguiente:

   ```
   /APML/Body/Profile
   ```

   Pero aquí se nos pide buscar “los perfiles de atención DISTINTOS que tiene el usuario. Entonces, debemos eliminar los elementos duplicados de la respuesta.

   Suponemos que dos Profile son iguales si su atributo name es igual.

   Existe una función para eliminar duplicados que se llama distinct-values(), pero esta función recién aparece en XPath 2.0. En XPath 1.0 es necesario utilizar el axis “preceding” para detectar valores duplicados.

   Entonces, vamos a resolver la expresión de la siguiente manera: "Retornar todos los perfiles de atención, siempre que el nombre de ese elemento no sea igual al nombre de uno de los perfiles de atención previamente analizados”.

   ```
   /APML/Body/Profile[not(./@name = preceding::Profile/@name)]
   ```

   Si agregáramos al xml una copia del perfil “Work”, la primera expresión retornaría como resultado tres perfiles: Work, Work, Home. Mientras que la segunda retorna solo dos: Work y Home.

2. La cantidad de conceptos distintos que conforman el perfil de intereses personal (Home). En este caso la respuesta sería 6.

   ¿Podemos resolver este ejercicio de la misma manera que el anterior?

   ```
   /APML/Body/Profile[@name="Home"]/*/Concepts/
     Concept[not(./@key = preceding::*/@key)]
   ```

   No!!!
   Al utilizar preceding estaríamos comparando con TODOS los nodos anteriores, no solo los Concept asociados al perfil Home.
   Para probar esto se podría agregar un Concept de nombre “Software” en el perfil “Work”.

   Una solución es entonces, utilizar la función preceding-sibling:

   ```
   /APML/Body/Profile[@name="Home"]/*/Concepts/Concept
       [not(./@key = preceding-sibling::*/@key)]
   ```

   Y aún así todavía hay un problema: estamos asumiendo que hay un único perfil de nombre “Home” en el documento. Si hubiera dos perfiles del mismo nombre y compartieran nombres de Concept, los mismos aparecerían dos veces.

Ejercicio 8

Para el archivo apml1.xml, escribir al menos dos expresiones Xpath distintas para poder obtener:

1. Los nombres de todos los perfiles de atención del usuario.

   /APML/Body/Profile/@name
   /APML/Body/Profile/attribute::name

2. Los nombres de todos los perfiles de atención del usuario ordenados alfabéticamente.

   Xpath no permite ordenar. Para eso necesitamos xquery.

3. Los nodos predecesores de todos los perfiles de atención del usuario.

   /APML/Body/Profile/ancestor::*
   /APML/Body/ancestor-or-self::*

4. Los perfiles de atención del usuario que tengan más de tres conceptos como parte de los datos implícitos.

   /APML/Body/Profile[count(ImplicitData/Concepts/Concept)>3]/@name
   /APML/Body/Profile/@name[count(../ImplicitData/Concepts/Concept)>3]

Ejercicio 9

Para el archivo asistencia.xml, armar los pares posibles de expresiones XPath equivalentes (conceptualmente iguales), indicando qué resultado arroja cada una de dichos pares. Indicar también el resultado obtenido para aquellas expresiones Xpath que no armen pares (si hubiera).

1.
```
/work-resources/work-resource
[@name = //work-resource/@name[
count(../work-exceptions/absent) > 1]
]/@name
```

   Retorna el nombre de aquellos recursos que hayan tenido más de una ausencia.

2.
```
//work-resource[count(./work-exceptions/absent) > 2]
```

   Retorna aquellos recursos que hayan tenido más de dos ausencias.

3.
```
/work-resources/work-resource[count(./work-exceptions/absent)    >
2]
```

   Retorna aquellos recursos que hayan tenido más de dos ausencias.

4.
```
/work-resources/work-resource/@name[count(../work-exceptions/absent) > 1]
```

   Retorna el nombre de aquellos recursos que hayan tenido más de una ausencia.

Los pares de expresiones son: (1,4) (2,3)