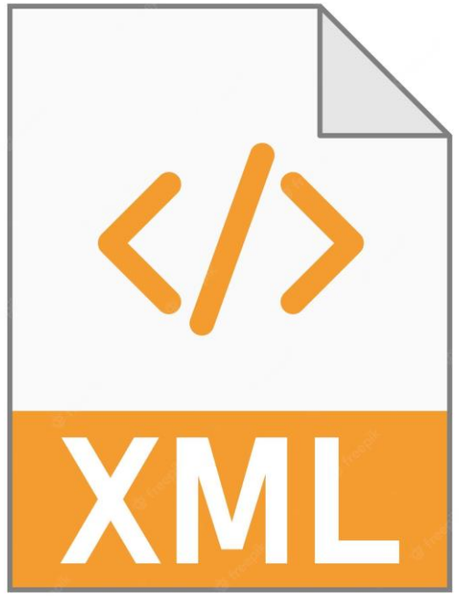



# UD4 – XML Y XHTML

# UD 4 – XML Y XHTML
# Índice de Contenidos

# UD 4 - XML Y XHTML
# 1 - XML

- **1 – XML**
  - **1.1 – Definición y características**
    - XML son las siglas de Extensible Markup Language (lenguaje de marcado extensible)
    - Es un LM semántico, fácil de aprender y que permite expresar y transmitir cualquier información.
    - Las etiquetas XML no están predefinidas, por eso es extensible.
    - Es un metalenguaje, es decir un lenguaje que permite crear otros lenguajes.
    - Es la base de muchos lenguajes de la web.

# UD 4 - XML Y XHTML
# 1 - XML

## 1.2 – Herramientas para trabajar con XML

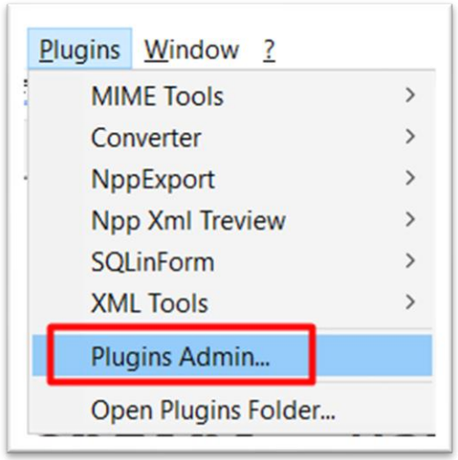
- Durante el curso vamos a utilizar:

- **Editores XML**: son editores de texto plano pero con funcionalidades extra para ayudarnos a escribir documentos XML
- **Analizadores XML**: son programas que verifican si un documento es correcto o no.
- **Bases de datos XML**: son BBDD que almacenan la información en XML y permiten la consulta de los datos almacenados

# UD 4 - XML Y XHTML
# 1 - XML

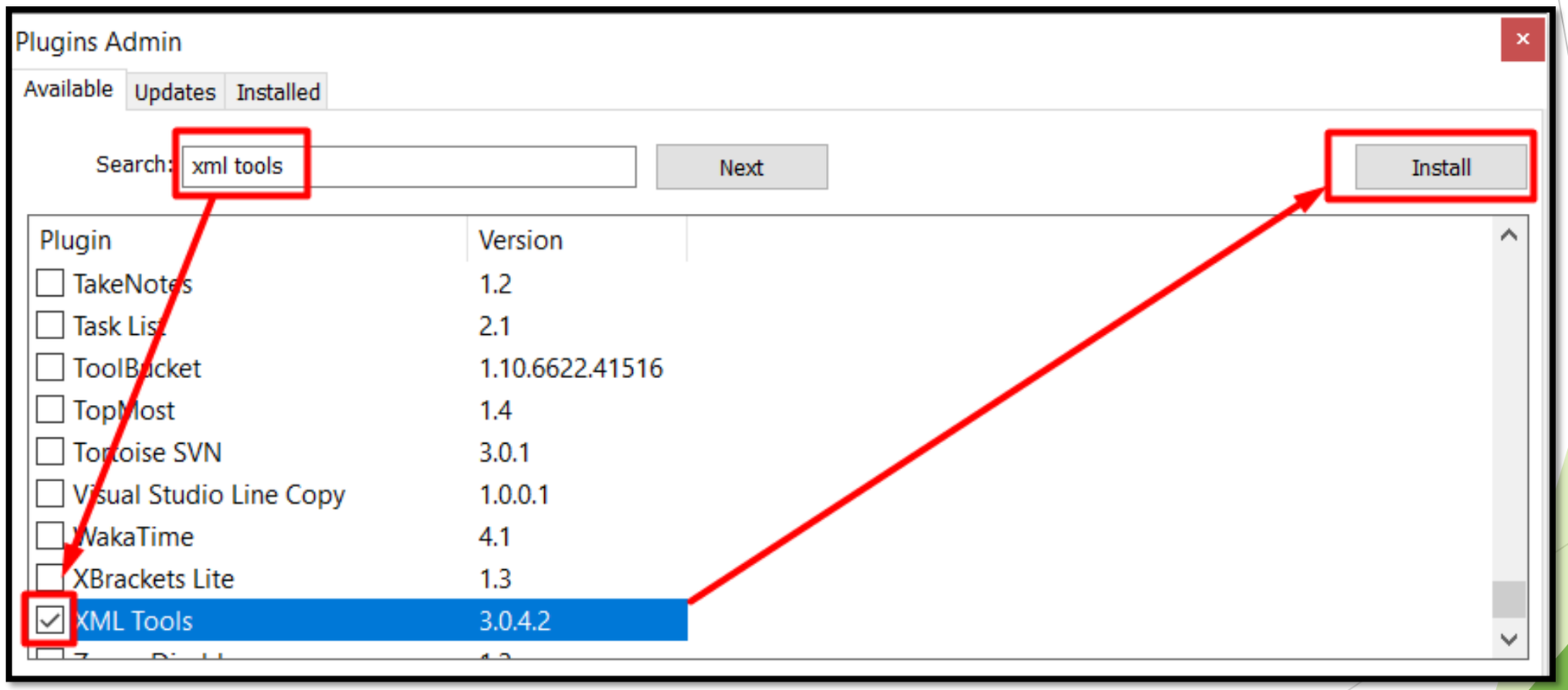
- En esta unidad usaremos:
- **Notepad++:** es un editor de texto plano multipropósito de software libre y que puede ser ampliado mediante plugins.
  - https://notepad-plus-plus.org/downloads/

- **Plugin XML Tools para Notepad++:** nos ayudará en la escritura de documentos XML y nos permitirá analizar los documentos escritos.
  - Para instalarlo hacemos desde el Notepad++:



# UD 4 - XML Y XHTML
# 1 - XML

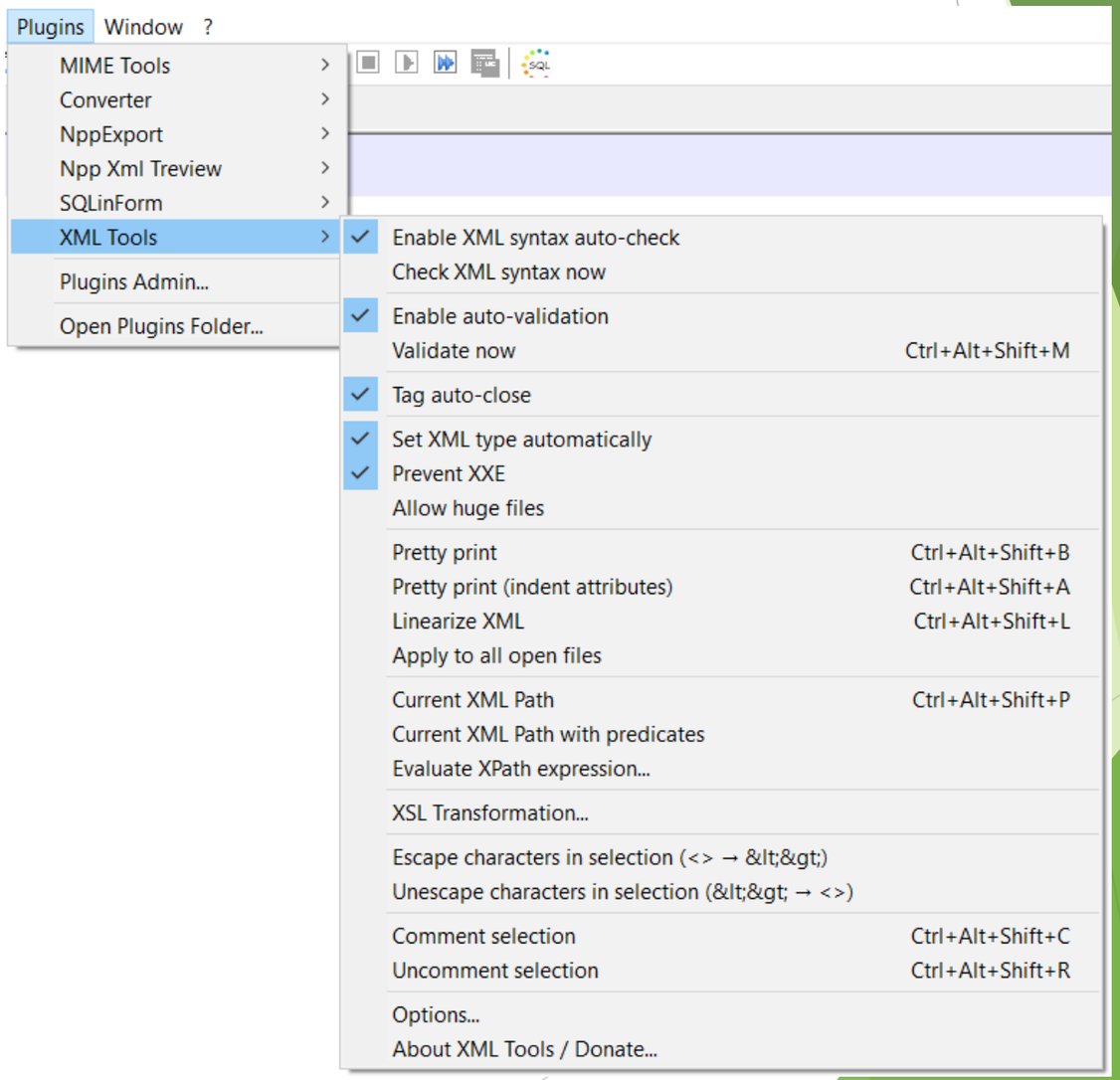
- A continuación, lo buscamos y lo instalamos:



# UD 4 - XML Y XHTML
# 1 - XML

- Una vez instalado podemos observar las distintas opciones desde el menú: Plugins -> XML Tools

- A medida que nos vayan haciendo falta iremos explicando las distintas opciones.



# UD 4 - XML Y XHTML
# 1 - XML

- También usaremos:
- **Cualquier navegador como analizador XML**: aunque el plugin anterior es capaz de analizar documentos XML también utilizaremos cualquier navegador del sistema para ello.

# UD 4 - XML Y XHTML
# 1 - XML

## 1.3 – Documentos bien formados y/o validados

Un **documento XML bien formado** es aquel que cumple todas las reglas sintácticas del lenguaje.

*En este tema estudiaremos estas reglas sintácticas y aprenderemos a escribir este tipo de documentos.*

Cuando un documento XML bien formado, además, cumple con un conjunto de reglas gramaticales entonces se dice que es un **documento XML válido o validado**.

*Este tipo de documentos los estudiaremos en la siguiente unidad didáctica "Validación de XML"*

# UD 4 - XML Y XHTML
# 1 - XML

## 1.4 – Sintaxis de XML

- Para crear documentos XML bien formados tenemos que conocer unas cuantas **reglas de sintaxis**:
- Los documentos se componen de:
  - Una **cabecera** o prólogo **opcional**, que debe aparece en primer lugar.
  - Un **cuerpo** que tiene al menos un elemento.

Cabecera

Cuerpo

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<nota>
        <alumno>Cazurro</alumno>
        <asignatura>Diseño de Bases de datos</asignatura>
        <calificación>0</calificación>
</nota>
```

# UD 4 - XML Y XHTML
# 1 - XML

- La cabecera no sigue las reglas de etiquetado de los elementos XML precisamente para poder distinguirla. Su sintaxis es:

```
<?xml version="1.0" encoding="UTF-8"?>
```

  - En este ejemplo se indica que se trata de un documento XML
  - Que la versión de XML es la 1.0
  - Y que la codificación de caracteres utilizada es la UTF-8. Para saber más sobre la codificación de caracteres: https://www.mclibre.org/consultar/htmlcss/html/html-utf8.html
  - Puede llevar más información pero que se estudiará en temas posteriores.

# UD 4 - XML Y XHTML
# 1 - XML

- El cuerpo es obligatorio y debe contener 1 solo elemento que llamaremos nodo o elemento raíz (root).

  Este elemento puede contener otros elemento formando una estructura en árbol:

```
<root>
  <child>
    <subchild>.....</subchild>
  </child>
</root>
```

Así que podemos hablar de nodos padres, hijos, hermanos…

# UD 4 - XML Y XHTML
# 1 - XML

- Un elemento siempre tiene una etiqueta de apertura, un contenido y una etiqueta de cierre:

```
<etiqueta>contenido</etiqueta>
```

En el caso de que el contenido sea prescindible o esté vacío se permiten las siguientes notaciones:

```
<apellido2></apellido2>
```
es equivalente a
```
<apellido2 />
```

- Las etiquetas siempre deben abrirse y cerrarse en el orden correcto. Esto provocaría un error sintáctico:

```
<alumno>Aitor Tilla<edad>25</alumno></edad>
```

# UD 4 - XML Y XHTML
# 1 - XML

- El nombre de la etiqueta debe cumplir las siguientes reglas:
  - Los nombres de los elementos distinguen entre mayúsculas y minúsculas. Esto daría un error sintáctico: `<Animal>Gato</animal>`
  - Los nombres de los elementos deben comenzar con una letra o un guión bajo
  - Los nombres de los elementos no pueden comenzar con las letras xml (o XML, o Xml, etc.)
  - Los nombres de los elementos pueden contener letras, letras con tilde, dígitos, guiones, guiones bajos y puntos.
  - Los nombres de los elementos no pueden contener espacios

Busca los nombres de etiquetas inválidos:

<jOsE>, <Álvaro>, <_Lucía 32>, <.Lucía>, <#Juanito>, <xmlpower>, <_xmlpower>, <3Barras>, <JohnieWalker>

# UD 4 - XML Y XHTML
# 1 - XML

- **Buenas prácticas con los nombres de etiquetas:**
- Usa nombres descriptivos: <persona>, <empresa>, <precio>… no queremos nombres como: <_32>, <a>, <mietiqueta>, <cosa>…
- Usa nombres cortos y simples: <libro_titulo> y no: <el_titulo_del_libro>.
- Evita el uso de "-“, “.” y “:” en la etiqueta. Son caracteres válidos pero hay lenguajes de programación que usan esos símbolos como operadores (resta, acceso a propiedades…) y pueden provocar errores en el procesado del XML
- Las letras que no estén en el alfabeto inglés (áéíóúü…) son válidas pero puede que el software que procese el XML no esté preparado para soportarlas.
- Utiliza un criterio de nombrado consistente en todo el documento. Por ejemplo:
  - Todo en mayúscula/minúscula y el “_” como separador: <libro_titulo>
  - Todo en minúscula excepto el comienzo de cada palabra: <LibroTitulo>

# UD 4 - XML Y XHTML
# 1 - XML

- Un contenido puede ser: un *texto* o un *número* o también *uno o más elementos*:

```
<alumno>Aitor Tilla</alumno>
<edad>25</edad>
<asignaturas>
  <asignatura>Programación</asignatura>
  <asignatura>Entornos de programación</asignatura>
  <asignatura>LLMM y SGI</asignatura>
</asignaturas>
```

# UD 4 - XML Y XHTML
# 1 - XML

- Para que un texto sea válido como contenido de un documento XML no puede utilizar los símbolos <> porque están reservados para las etiquetas y podrían hacer que el analizador falle… Esto daría un error sintáctico:

```
<MenorEdad>edad < 18</MenorEdad>
```

**¿Cómo lo se soluciona entonces?** Pues XML define unos elementos especiales llamados **entidades de caracteres** y que representan un carácter concreto pero sin escribirlo directamente.

El caso anterior se arreglaría así:

```
<MenorEdad>edad &lt; 18</MenorEdad>
```

# UD 4 – XML Y XHTML
# 1 – XML

- Esta solución también la siguen otros lenguajes como HTML y existe un conjunto grande de entidades de caracteres.
- Algunos ejemplos:

***Sintaxis:***
*&NombreEntidad;*

| Nombre de la Entidad | Carácter representado | Proviene de |
|---|---|---|
| lt | < | less than (menor que) |
| gt | > | greater than (mayor que) |
| quot | " | quote (cita) |
| amp | & | ampersand (idem) |
| apos | ' | apostrophe (apóstrofe) |
| … | … | … |

Cuando el analizador del documento XML se encuentra con una entidad de carácter sabe que tiene que cambiarla por el carácter que representa.

# UD 4 - XML Y XHTML
# 1 - XML

```
<?xml version="1.0"?>
<raiz>
    <texto>
        Si tu edad es &lt; 18 entoces pa&apos;casa
    </texto>
    <texto>
        Y dice el refrán &quot;Más vale pájaro
        en mano que ciento volando&quot;
    </texto>
</raiz>
```

ANALIZADOR

```
<?xml version="1.0"?>
- <raiz>
    <texto> Si tu edad es < 18 entoces pa'casa </texto>
    <texto> Y dice el refrán "Más vale pájaro en mano que ciento volando" </texto>
  </raiz>
```

# UD 4 - XML Y XHTML
# 1 - XML

- Si tenemos un texto muy largo esto de poner las entidades de caracteres puede ser una **pesadilla**. Para estos casos se usan las secciones CDATA (Character Data).
- Una sección CDATA es una porción del documento en la que el analizador no va a buscar etiquetas, es decir, es una sección que el analizador la va mostrar tal y como está escrita.
- Tiene una sintaxis un poco rara:

```
<![CDATA[

Aquí puedo poner lo que quiera <etiquetas> y todo. Se va a mostrar tal cual está aquí

]]>
```

# UD 4 - XML Y XHTML
# 1 - XML

- ¿Cómo crees que se muestra en un navegador el siguiente documento XML?

```xml
<?xml version="1.0"?>
<raiz>
    <TextoConCDATA>
        <![CDATA[
            <mensaje>Si tu edad es &lt; 18 entonces pa&apos;casa</mensaje>
        ]]>
    </TextoConCDATA>
    <TextoSinCDATA>
        <mensaje>Si tu edad es &lt; 18 entonces pa&apos;casa</mensaje>
    </TextoSinCDATA>
</raiz>
```

HAGAN SUS APUESTAS

# UD 4 – XML Y XHTML
# 1 – XML

```
<?xml version="1.0"?>
<raiz>
  - <TextoConCDATA>
      - <![CDATA[

                                  <mensaje>Si tu edad es &lt; 18 entonces pa&apos;casa</mensaje>


        ]]>
    </TextoConCDATA>
  - <TextoSinCDATA>
        <mensaje>Si tu edad es < 18 entonces pa'casa</mensaje>
    </TextoSinCDATA>
</raiz>
```

REALIZAR LOS EJERCICIOS DEL 1 AL 4 DEL BOLETÍN DE PROBLEMAS

# UD 4 - XML Y XHTML
# 1 - XML

- Dijimos anteriormente que las etiquetas de apertura opcionalmente pueden llevar uno o más **atributos**.

  Siguen la siguiente sintaxis:

  ```
  <libro id="23442">El Señor de los Anillos</libro>
  <alumno dni="28748415T" altura="178">Juan Sin Miedo</alumno>
  ```

  Los atributos son pares **nombre="valor"** que sirven para aportar una información de detalle extra a la etiqueta que lo contiene. *Es lo más parecido a un post-it sobre un documento.*

  Los atributos solo pueden contener **un valor**. No se permite lo siguiente:

  ```
  <alumno telefonos="651234234, 954342323">Juan Sin Miedo</alumno>
  ```

# UD 4 - XML Y XHTML
## 1 - XML

## ELEMENTOS VS ATRIBUTOS

Compara:

```
<person gender="female">
  <firstname>Anna</firstname>
  <lastname>Smith</lastname>
</person>
```

```
<person>
  <gender>female</gender>
  <firstname>Anna</firstname>
  <lastname>Smith</lastname>
</person>
```

*¿Qué opción pensáis que es mejor para organizar la información?*

# UD 4 - XML Y XHTML
## 1 - XML

- No hay reglas de cuándo utilizar atributos o cuándo utilizar elementos. Sin embargo, podemos decir que:
  - Los atributos son más fáciles de leer para los humanos que los elementos.
  - Los atributos están más limitados que los elementos porque no admiten la estructura en árbol o tener más de un valor.
  - Los elementos se expanden más fácilmente que los atributos.

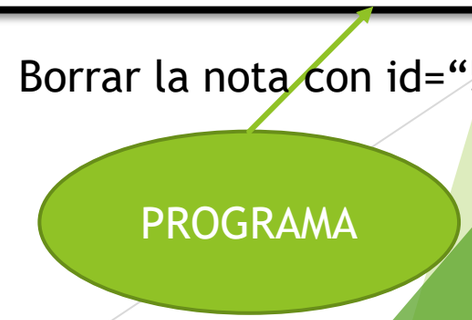
**ELEMENTOS**

# UD 4 - XML Y XHTML
# 1 - XML

- Sin embargo, en HTML se usan muchos atributos por dos razones:
  - Son más legibles y más compactos (ocupan menos)
  - Se utilizan para "marca" las etiquetas, es decir se usan como metadatos (piensa en el uso que le da a los atributos **id** o **class**).

  De este modo, un programa que maneje el documento XML puede editar o eliminar elementos identificándolos por su atributo.

```
<messages>
  <note id="501">
    <to>Tove</to>
    <from>Jani</from>
    <heading>Reminder</heading>
    <body>Don't forget me this weekend!</body>
  </note>
  <note id="502">
    <to>Jani</to>
    <from>Tove</from>
    <heading>Re: Reminder</heading>
    <body>I will not</body>
  </note>
</messages>
```



# UD 4 – XML Y XHTML
# 1 – XML

- Los **comentarios** son anotaciones que hace la persona que escribe el documento XML. Su sintaxis es la siguiente:

```
<!--Esto un comentario en XML-->
```

Son elementos ignorados por el analizador (parser) y siguen las siguientes reglas:

- No pueden aparecer antes del prólogo.
- No se pueden anidar un comentario dentro de otro.
- Pueden aparecer en cualquier parte de un documento excepto dentro de los valores de los atributos.

REALIZAR LOS EJERCICIOS 5-7 DEL BOLETÍN DE PROBLEMAS

## 1.5 – Espacio de nombres en XML

- Supongamos que tenemos una empresa internacional (www.interinvest.com) que ofrece servicios de inversión en bolsa en las capitales de todos los países del mundo.
- Una inversión en bolsa viene especificada por los siguientes datos:

```
<inversion>
    <producto>IBEX 35</producto>
    <capital>2000€</capital>
</inversion>
```

namespace

# UD 4 - XML Y XHTML
# 1 - XML

- Además, nuestra empresa toma información del INE (Instituto Nacional de Estadística) para conocer las distintas capitales de los países, recibiendo documentos como este:

```
<pais>
    <nombre>España</nombre>
    <capital>Madrid</capital>
</pais>
```

- ¿Cómo podríamos ampliar el elemento *inversión* para añadir la capital en la que se invierte? ¿Qué ocurriría con las etiquetas *capital*? ¿Cómo sabemos si nos referimos al dinero o a la ciudad?

# UD 4 - XML Y XHTML
## 1 - XML

- Es ambiguo. Las máquinas y las cosas ambiguas se llevan mal…

```
<?xml version="1.0"?>
<inversion>
    <producto>IBEX 35</producto>
    <capital>2000€</capital>
    <capital>Madrid</capital>
</inversion>
```

- Necesitamos algo que rompa la ambigüedad…
- ¡Un prefijo nos iría bien!. Por ejemplo:
  - **i:capital** se refiere al dinero (i=inversión)
  - **p:capital** se refiere a la ciudad (p=país)

- … lo escribimos y cruzamos los dedos a ver si funciona…

# UD 4 - XML Y XHTML
# 1 - XML

```
<?xml version="1.0"?>
<inversion>
    <producto>IBEX 35</producto>
    <i:capital>2000€</i:capital>
                XML Validation error - line 4, pos 13:
                Referencia a un prefijo de espacio de nombres no declarado: 'i'.
    <p:capital>Madrid</p:capital>
</inversion>
```

- Nos pide una cosa que se llama “espacio de nombres”
- Este concepto se usa muchísimo en programación y en administración de sistemas y cuanto antes lo tengáis claro mejor…

# UD 4 – XML Y XHTML
# 1 – XML

- **Namespace o espacio de nombres**: es un contenedor de nombres que cumple las siguientes reglas:

- El namespace tiene un identificador único en el mundo. No puede haber 2 que se llamen igual.
- Un namespace puede contener muchos nombres pero estos no pueden repetirse.

| NS 1 | NS 2 | NS 3 |
|---|---|---|
| • Inversión<br>• Capital<br>• Persona | • País<br>• Capital<br>• Habitante | • Hogar<br>• Persona<br>• País |

# UD 4 – XML Y XHTML
# 1 – XML

- De esta forma no hay ambigüedades porque:
  - NS1:Capital ≠ NS2:Capital
  - NS1:Persona ≠ NS3.Persona
  - NS2:País ≠ NS3:País

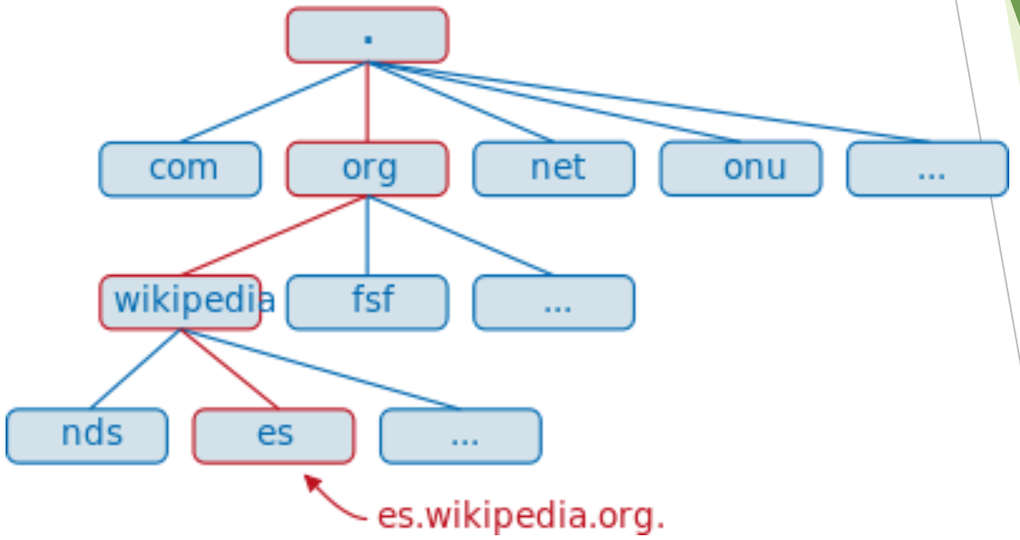
| NS 1 | NS 2 | NS 3 |
|---|---|---|
| • Inversión<br>• Capital<br>• Persona | • País<br>• Capital<br>• Habitante | • Hogar<br>• Persona<br>• País |


¿Pero quién se encarga de organizar esos NS1, NS2, NS3… para que no se repitan? ¿Quién se responsabiliza de que todos los namespaces del mundo tengan un identificador único?

*¿Y si alguien ya lo está haciendo y podemos aprovechar su trabajo?*

# UD 4 - XML Y XHTML
# 1 - XML

- ICANN (Internet Corporation for Assigned Names and Numbers) es una organización que controla que los dominios de Internet no se repitan (entre otras cosas).




Simplemente aprovechando que los nombres de dominio son únicos en el mundo tendríamos nuestro problema resuelto. Veamos cómo queda:

# UD 4 - XML Y XHTML
# 1 - XML

- Nuestro documento XML entonces quedaría así y sería correcto sintácticamente:

```
<?xml version="1.0"?>
<inversion>
    <producto>IBEX 35</producto>
    <i:capital xmlns:i="https://www.interinvest.com">
        2000€
    </i:capital>
    <p:capital xmlns:p="https://www.ine.es">
        Madrid
    </p:capital>
</inversion>
```

# UD 4 – XML Y XHTML
# 1 – XML

- Los analizadores XML no buscan información ni nada en el nombre de dominio que se indica.
- Sin embargo, un documento XML serio apuntaría a un fichero en el que se describa la etiqueta en cuestión.

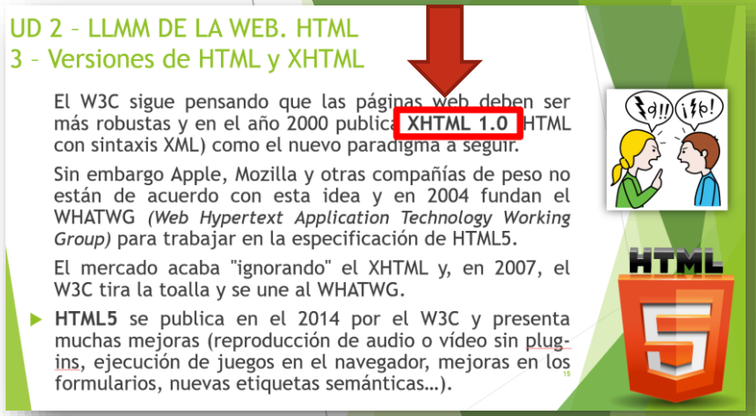
REALIZAR EL EJERCICIO 8 DEL BOLETÍN DE PROBLEMAS

*Lo más importante es que te quedes con el concepto claro porque te lo vas a encontrar a menudo.*

*¿Sabrías explicar qué es un espacio de nombres en XML?*

# UD 4 – XML Y XHTML
# 2 – XHTML

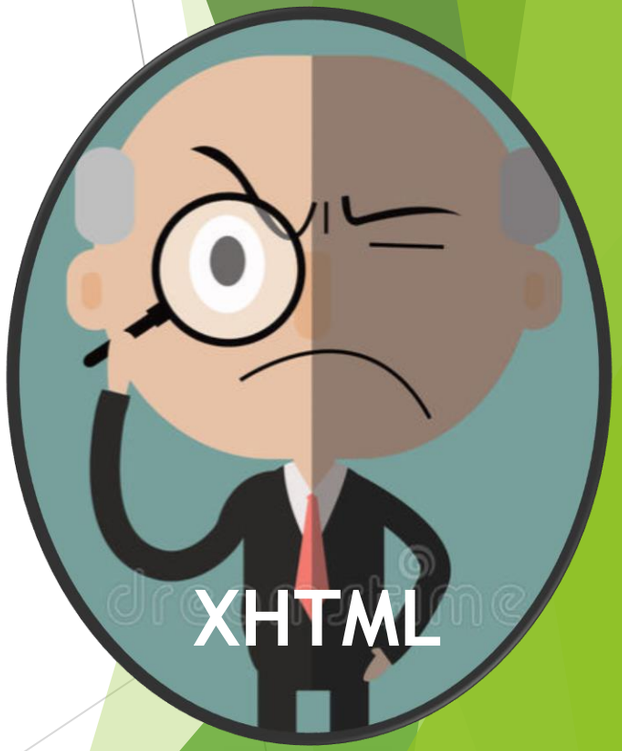
- **2 – XHTML**
  - Cuando estudiamos las versiones por las que pasó HTML dijimos que poco después del HTML 4.01 apareció el **XHTML** como un nuevo estándar aunque finalmente no fue adoptado en el mercado.
  - Como ahora conocemos XML vamos a estudiar mínimamente esta tecnología.
  - XHTML 1.0 se publicó en el 2000 con el gran objetivo de *"convertirlo en el lenguaje de transporte de información universal"*. Al escribir HTML con sintaxis XML se conseguía un HTML:
    - **Extensible:** pudiendo definir nuevas etiquetas, pudiendo crear así nuevos lenguajes de marcado.
    - **Más fiable:** ya que podemos analizarlo sintáctica y semánticamente, detectando cualquier tipo de error.




# UD 4 - XML Y XHTML
# 2 - XHTML

- Para cumplir este gran objetivo tenemos que **imponer nuevas restricciones** a nuestro HTML5. Las principales restricciones sintácticas a cumplir son:
  - ✓ La cabecera <! DOCTYPE> es obligatoria
  - ✓ Las etiquetas <html>, <head>, <title> y <body> son obligatorias
  - ✓ El atributo *xmlns* de la etiqueta <html> es obligatorio
  - ✓ Los elementos siempre deben estar correctamente anidados
  - ✓ Los elementos siempre deben estar cerrados
  - ✓ Los elementos siempre deben nombrarse en minúsculas
  - ✓ Los nombres de los atributos siempre deben estar en minúsculas
  - ✓ Los valores de atributo siempre deben estar entrecomillados
  - ✓ La minimización de atributos está prohibida

```
<ol reversed="reversed"> VS <ol reversed>
```



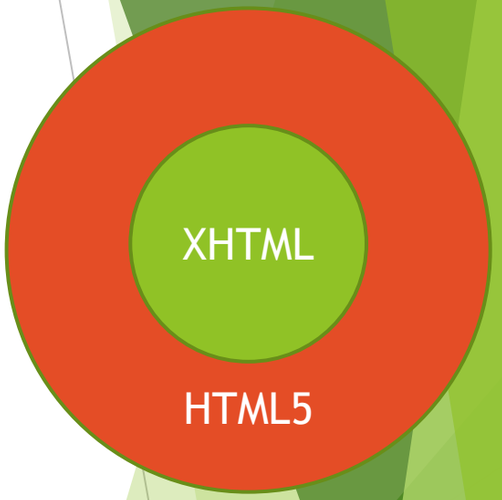
# UD 4 – XML Y XHTML
# 2 – XHTML

- Con el XHTML, el W3C quería ir mucho más allá de las páginas web, su propuesta era que **cualquier intercambio de información entre dos máquinas pudiera realizarse con XHTML…** solo teníamos que cumplir con las nuevas restricciones.
- Sin embargo, **lo "cómodo" venció a lo "correcto"**. El mercado no adoptó XHTML por ser demasiado exigente, prefirió quedarse con **el viejo HTML que era más "compasivo" con los errores** cometidos por los desarrolladores web.
- Cuando el W3C vio que el mercado no adoptaba XHTML abandonó la idea y se unió al grupo de trabajo que creaba HTML5… pero consiguió dejar su sello…

# UD 4 – XML Y XHTML
# 3 – XHTML vs HTML5

## 3 –XHTML vs HTML5

- La exigencia del W3C al participar en la creación de HTML5 fue la siguiente: *"HTML5 debe ser compatible con XHTML"*
- Esto quiere decir que si escribimos un documento en XHTML estamos cumpliendo la especificación de HTML5. Pero no todos los documentos escritos en HTML5 cumplirán con las especificación de XHTML.
- Por tanto, HTML5 presenta una sintaxis más flexible y relajada que XHTML, aunque puede ser escrito añadiendo todas las restricciones necesarias para que el resultado sea compatible con XHTML.
- Podríamos decir entonces que las reglas sintácticas de XHTML están incluidas en el conjunto de reglas para escribir HTML5.




El castellano puro (XHTML) se entiende en Andalucía, pero el andaluz (HTML5) puede no ser entendido por un castellano

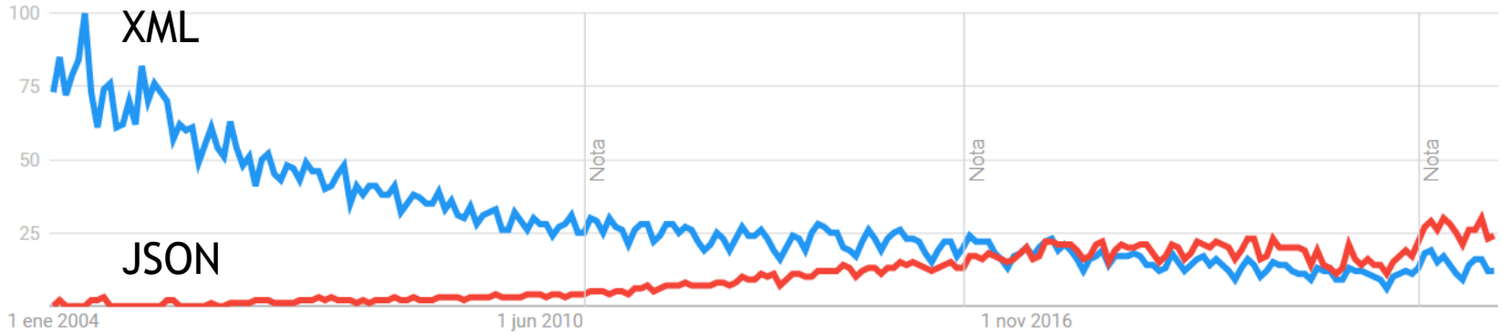
# UD 4 – XML Y XHTML
# 3 – XHTML vs HTML5

- Para terminar esta comparativa vamos a observar las semejanzas y diferencias entre ambos lenguajes:

| SEMEJANZAS | DIFERENCIAS |
|---|---|
| • Usan un lenguaje de marcas para etiquetar la información<br>• Usan etiquetas semánticas (<body>, <meta>…) y de presentación (<b>, <u>…)<br>• Los documentos puede ser representados con una estructura de árbol.<br>• Los documentos pueden ser leídos y comprendidos tanto por máquinas como por humanos. | • XHTML exige que el documento esté “bien formado”, sintácticamente correcto, para que se muestre en el navegador.<br>• HTML no exige la corrección sintáctica del documento para ser mostrado por un navegador.<br>• XHTML también se usa para transmitir información entre máquinas distintas a una página web |

# Cierre de la unidad

- XML se sigue usando para el transporte de datos de una máquina a otra, aunque cada vez menos, ya que JSON le está ganando la partida.




- XML se sigue usando habitualmente en ficheros de configuración de aplicaciones (pom.xml, web.xml…)
- XHMTL no se usa, quedó como un intento fallido de hacer algo mejor…
- En esta unidad hemos estudiado las reglas sintácticas que permiten escribir un documento XML bien formado. En la siguiente unidad tomaremos un documento XML bien formado y estudiaremos cómo saber si presenta fallos semánticos o no, de esta forma conseguimos darlo por válido o desecharlo.


THE
END