Desarrollo de Aplicaciones Multiplataforma
0486 Acceso a datos

Tema 1. Gestión de ficheros XML

XML

- XML: EXtensible Markup Language.

- XML se utiliza para transportar datos en un formato que pueda ser entendido con facilidad por personas y por máquinas.
- XML es un lenguaje de marcas (como lo es HTML) y es autodescriptivo.
- A diferencia de HTML (destinado a mostrar información) XML no tiene otro propósito predefinido aparte de transportar datos.

XML

XML tiene ventajas claves:

- Simplifica el transporte de datos entre plataformas.
- Es extensible, las etiquetas no están predefinidas.
- Es perfecto para compartir datos en sistemas actuales y futuros.

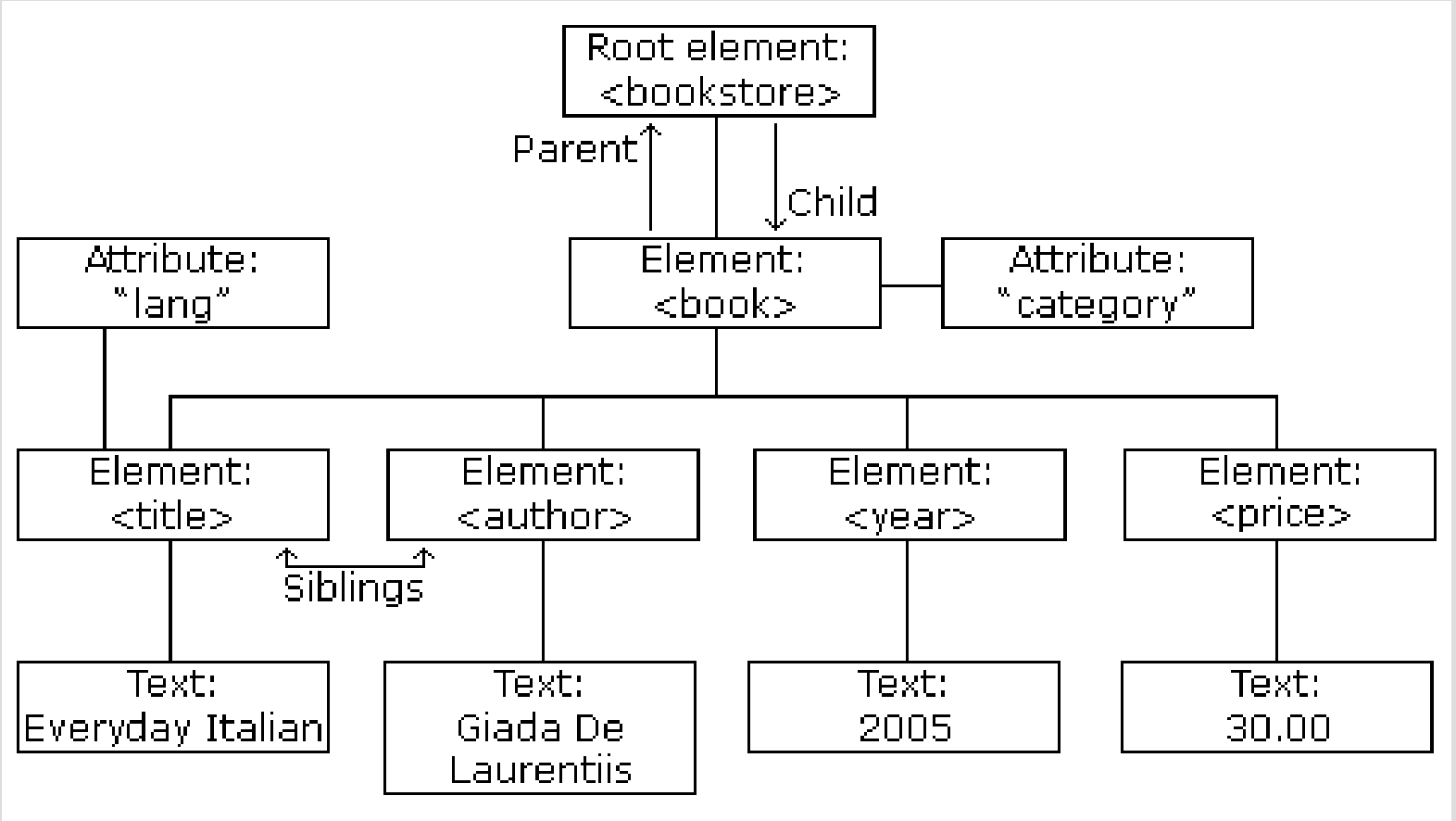
XML

Ejemplo

```
<?xml version="1.0" encoding="UTF-8"?>
<bookstore>
  <book category="cooking">
    <title lang="en">Everyday Italian</title>
    <author>Giada De Laurentiis</author>
    <year>2005</year>
    <price>30.00</price>
  </book>
  <book category="children">
    <title lang="en">Harry Potter</title>
    <author>J K. Rowling</author>
    <year>2005</year>
    <price>29.99</price>
  </book>
  <book category="web">
    <title lang="en">Learning XML</title>
    <author>Erik T. Ray</author>
    <year>2003</year>
    <price>39.95</price>
  </book>
</bookstore>
```

XML. Estructura

- Estructura de árbol



XML. Estructura

- Estructura de árbol. Una raíz (elemento root) de la que parten hijos (elementos child).
- Cada hijo tiene una referencia a un padre (elemento parent) o a la raíz.
- Cada hijo puede tener elementos al mismo nivel (elemento sibling).
- Todos los elementos pueden tener un texto asociado (ej: Everyday Italian) y uno o varios atributos ej: (lang=”en”).

XML. Sintaxis

- El archivo XML se describe a si mismo con esta línea:

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

No es imprescindible, pero sí muy recomendable.

XML. Sintaxis

- Deben tener siempre un elemento raíz:

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

XML. Sintaxis

- Deben tener siempre un elemento raíz:

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

- Cada etiqueta abierta debe ser cerrada

XML. Sintaxis

- XML es case sensitive. Ojo con los nombres de las etiquetas.
- Consistencia con el sistema de nombres. Los más extendidos son todo en minúsculas (y quizás lowerCamelCase).
- Evitar signos especiales como . o : o -
- Se pueden utilizar tildes o espacios, pero es poco recomendable para evitar problemas con determinados intérpretes de XML.

XML. Sintaxis

- El texto en las etiquetas no se escribe entre comillas.
- El texto de los atributos sí se escribe entre comillas.
- Ejemplo:

```
<state date=”22/09/2016”>
   <desc>Bastante ocupado</desc>
</state>
```

XML

- Atributo o elemento: No hay un modelo fijo, aunque en general los atributos aportan menos ventajas (no son tan fácilmente extensibles y no siguen la estructura de árbol). Normalmente los emplearemos para metadatos (identificadores, etc.)

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


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

XML. Otras posibilidades

- Namespaces para evitar conflictos semánticos mediante prefijos. Se definen en el elemento raiz. Las URI no se interpretan.

```
<root xmlns:h="http://www.w3.org/TR/html4/" xmlns:f="http://www.w3schools.com/furniture">

<h:table>
  <h:tr>
    <h:td>Apples</h:td>
    <h:td>Bananas</h:td>
  </h:tr>
</h:table>
<f:table>
  <f:name>African Coffee Table</f:name>
  <f:width>80</f:width>
  <f:length>120</f:length>
</f:table>

</root>
```

Validación

Un XML debe estar bien formado:

http://codebeautify.org/xmlvalidator

y seguir una normas definidas en un

- DTD: Document Type Definition
- XML Schema: alternativa al DTD basada en xml

¿Por qué?

DTD. Ejemplo de uso

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE note SYSTEM "Note.dtd">
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
```

DTD. Implementación

```
<!DOCTYPE note
[
<!ELEMENT note (to,from,heading,body)>
<!ELEMENT to (#PCDATA)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT heading (#PCDATA)>
<!ELEMENT body (#PCDATA)>
]>
```

Xml Schema. Implementación

```
<xs:element name="note">

<xs:complexType>
  <xs:sequence>
    <xs:element name="to" type="xs:string"/>
    <xs:element name="from" type="xs:string"/>
    <xs:element name="heading" type="xs:string"/>
    <xs:element name="body" type="xs:string"/>
  </xs:sequence>
</xs:complexType>

</xs:element>
(Ej: <xs:attribute name="id" type="xs:positiveInteger"/>)
```

Desarrollo de Aplicaciones Multiplataforma
0486 Acceso a datos

Tema 1. Gestión de ficheros XML
Bibliografía y ejemplos: http://www.w3schools.com

IES Castelar
Antonio Paniagua Navarro