XML

HuSsAm Klhasan

hu_ss_am_89@hotmail.com

sep 2011

مقدمة : XML هي اختصار ل eXtensible Markup Language ,وهي مصممة لتخزين وتبادل البيانات ,تتميز بسهولة تعلمها كما أنها هي ليست بديل عن ال HTML

الفرق بين ال XML وال HTML هو:

XML مصممة لتخزين وتبادل البيانات , بينما HTML مصممة لعرض البيانات الـ XML هي ذاتية التو صيف وليس لها وسوم معرفة مسبقاكما في HTML

مثال:

```
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
```

نلاحظ ذاتية التوصيف في المثال فقد قمنا بكتابةٍالبيانات. وحددنا لها sender و receiver و heading و body هذا المستند لا يقوم بأي عمل هو فقط يخزن البيانات .

استخدامات الـ XML :

- **فصل البيانات عن الـ HTML :** إذاكان لدينا بيانات قابلة للتغيير ونريد عرضها باستخدام الـ HTML سيكون من الصعب تغيير ملف الـ HTML في كل مرة تتغير فيها البيانات .باستخدام الـ XML يمكن تخزين البيانات في ملف XML منفصل واستخدام الـ HTML للعرض فقط وبالتالي يمكن تغيير البيانات دون التأثير على كود الـ HTML .

باستخدام كود Java Script بسيط يمكن قراءة ملف XML وتعديل البيانات في صفحة الويب.

- **تبسيط عملية تشارك البيانات:** من أهم الأمور التي تستوقف المطورين هي تبادل البيانات بين أنظمة غير متوافقة . تبادل البيانات كـ XML يقلل من التعقيد في عملية تبادل البيانات لأن هذه البيانات ستكون قابلة للقراءة من قبل تطبيقات مختلفة وغير متوافقة.
- **يبسط عملية تغيير منصة العمل:** تطوير النظام (hard ware أو software) هو عملية مكلفة زمنيا وتتطلب تحويل البيانات وقد تؤدي إلى ضياع البيانات غير المتوافقة مع النظام الجديد.ملفات الـ XML تخزن بصيغة نصية وهذه يسهل عملية التطوير إلى نظام جديد أو مستعرض جديد أو تطبيق جديد بدون ضياع أي بيانات.
- **تستخدم لإنشاء لغات انترنت جديدة:** العديد من لغات الويب أنشأت اعتمادا على الـ XML , من الأمثلة : XHTML , RDF, RSS, WAP, WSDL SMIL, OWL,

شجرة الـ XML (XML tree) :

مستندات الـ XML تشكل بنية شجر ية تبدأ بالجذر وتتفرع إلى الأوراق أي مستند XML يجب أن يتضمن عنصر يمثل الجذر وهو الأب لكل العناصر في هذا المستند أي عنصر يمكن أن يكون له عناصر فرعية هي أبناؤه

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

ونستخدم المصطلحات أب, ابن, أخ لتوصيف العلاقة بين العناصر المختلفة بحيث العنصر الأب هو عنصر له أبناء والأبناء في نفس المستوى تكون أخوة

XML Syntax :

قواعد الـ XML بسيطة ومنطقية كما أنها سهلة التعلم والاستخدام

عندكتابة مستند XML يجب أن نراعي القواعد التالية :

- **كل عنصر يجب أن يكون له وسم إغلاق :**

في HTML ليس بالـ ضرورة أن يكون للعنصر وسم إغلاق مثلا:

```
<p>This is a paragraph
```

أما في الـ XML فيجب أن يكون لكل عنصر وسم إغلاق ,نفس المثال السابق في الـ XML سيكون بالشكل التالي :

```
<p>This is a paragraph</p>
<p>This is another paragraph</p>
```

- **الوسوم حساسة لحالة الأحرف :** مثلا الوسم <Letter> مختلف عن الوسم <letter>

ويجب أن نراعي أن وسم الفتح والإغلاق يجب أن تكون متطابقة في حالة الأحرف.

- **الوسوم يجب أن تكون متداخلة بشكل صحيح:**

في الـ HTML يمكن اعتبار ما يلي صحيحا

```
<b><i>This text is bold and italic</b></i>
```

أما في الـ XML فالصيغة السابقة خاطئة ويجب أن تكتب بالشكل التالي:

```
<b><i>This text is bold and italic</i></b>
```

التداخل بشكل صحيح نعني به أنه طالما الوسم <i> تم فتحه ضمن الوسم <b> فيجب أن يغلق ضمنه أيضا

- **أي مستند XML يجب أن يتضمن عنصر جذر.**
- **قيم الخصائص يجب أن توضع بين إشارات تنصيص :** عناصر الـ XML يمكن أن تأخذ خصائص مثل name , value تماماكما في الـ HTML ولكن في الـ XML يجب أن توضع قيم هذه الخصائص بين إشارات تنصيص

فيما يلي مستندين الأول يمثل صيغة خاطئة بينما يمثل الثاني صيغة صحيحة

```
<note date=12/11/2007>
 <to>Tove</to>
 <from>Jani</from>
</note>
```

false

```
<note date="12/11/2007">
 <to>Tove</to>
 <from>Jani</from>
</note>
```

true

الخطأ في المستند الأول هو عدم وضع قيمة التاريخ بين إشارات التنصيص

- **المحارف الخاصة :** لو وضعنا المحرف "> " في مستند الـ XML ستكون الصيغة خاطئة مثلا

```
<message>if salary < 1000 then</message>
```

لتجنب هذه الأخطاء نستبدل المحرف "> " بما يسمى entity reference أي:

```
<message>if salary &lt; 1000 then</message>
```

حيث لدينا خمسة entity reference معرفة بشكل مسبق وهي :

<	<	less than
>	>	greater than
&	&	Ampersand
'	'	Apostrophe
"	"	quotation mark

ملاحظة : فقط المحرفين "> " و "& " ممنوع استخدامها في XML أما المحرف "<" فلا مانع من استخدامه ولكن من الأفضل استبداله بال entity reference .

- **التعليقات في XML :** نعبر عنهاكما يلي

```
<!-- This is a comment -->
```

- **الفراغات :** في الـ HTML عدة فراغات متتالية يتم اسـ تبدالها بفراغ واحد في العرض أما في الـ XML فتبقى كما هي.

عناصر الـ XML :

العنصر (element) هوكل شيء موجود بين و سم فتح العنصر ووسم إغلاق العنصر

العنصر يمكن أن يتضمن ما يلي :

- عناصر أخرى
- نص
- خصائص

- مزيج من كل ما سبق

قواعد التسمية في الـ XML :

الأسماء يجب أن تخضع للقواعد التالية :

- الأسماء يمكن أن تحتوي أحرف وأرقام و المحارف الأخرى
- الاسم لا يمكن أن يبدأ برقم
- الاسم لا يمكن أن يبدأ بأي من xml أو XML أو Xml ...
- الاسم لا يمكن أن يحوي فراغات

الخصائص في الـ XML :

من خلال المثال التالي نوضح الفرق بين الخاصية والعنصر في الـ XML :

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

1

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

2

في المستند الأول sex هي خاصية (attribute) أما في المستند الثاني فـ sex هي عنصر (element) ولكن كلا المستندين يحملان نفس المعلومات

لا توجد قواعد تحدد متى نستخدم الخصائص ومتى نستخدم العناصر

مشاكل استخدام الخصائص :

- لا يمكن أن تحوي قيم متعددة (العناصر يمكن)
- لا يمكن أن تشكل بنية شجرية (العناصر يمكن)
- لا يمكن توسيعها بسهولة (للتغييرات المستقبلية)

الخصائص صعبة في القراءة والإدارة ,عادة تستخدم العناصر للبيانات والخصائص للمعلومات غير المتعلقة بهذه البيانات

كما تستخدم الخصائص للتعبير عن الـ Metadata مثل ID لعناصر الـ XML

XML Validation :

الـ XML المكتوب بصيغة صحيحة يسمى "Well Formed"

الـ XML الذي يحقق الـ DTD يسمى "Valid"

Well Formed XML Documents

هو مستند XML يحقق القواعد التي ذكرناها سابقا

Valid XML Documents

هو Well Formed XML Document ويحقق قواعد DTD (Document Type Definition)

مثال :

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE note SYSTEM "Note.dtd">
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
```

التصريح DOCTYPE يشير إلى ملف DTD خارجي

: XML DTD

الهدف من الـ DTD هو تعريف بنية مستند الـ XML

مثال:

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

: XML Schema

W3C تدعم بديل للـ DTD وهو XML Schema

مثال:

```
<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>
```

عرض الـ XML باستخدام XSLT :

XSLT هي اختصار لـ eXtensible Stylesheet Language Transformations

يمكن أن تستخدم في تحويل الـ XML إلى HTML قبل أن يعرض على المستعرض

التحويل يتم من قبل المستعرض في حال قام المستعرض بقراءة ملف الـ XML وبالتالي باختلاف المستعرض نحصل على نتائج مختلفة ولتجنب هذه المشكلة يمكن أن يتم التحويل على الـserver.

XML Parser :

معظم مستعرضات الويب الحديثة تملك XML Parser ضمني

حيث الـ XML Parser يقوم بتحويل مستند الـ XML إلى XML DOM وبالتالي يمكن معالجته بـ JavaScript

XML DOM :

DOM هي اختصار لـ Document Object Model وتعرف طريقة قياسية لمعالجة ملفات الـ XML , تظهر مستند الـ XML كبنية شجرية . من خلال الـ DOM tree يمكن الوصول إلى كل العناصر وتعديل محتوياتها أو حتى حذفها كما يمكن إنشاء عناصر جديدة حيث يتم التعامل مع العناصر والخصائص كعقد (node) .

في المثال التالي نبين عملية الـ parsing لمستند note.xml الذي مر معنا في الأمثلة السابقة :

```
<html>
<body>
<h1>W3Schools Internal Note</h1>
<div>
<b>To:</b> <span id="to"></span><br />
<b>From:</b> <span id="from"></span><br />
<b>Message:</b> <span id="message"></span>
</div>

<script type="text/javascript">
if (window.XMLHttpRequest)
 {// code for IE7+, Firefox, Chrome, Opera, Safari
 xmlhttp=new XMLHttpRequest();
 }
```

```
else
  {// code for IE6, IE5
  xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
  }
xmlhttp.open("GET","note.xml",false);
xmlhttp.send();
xmlDoc=xmlhttp.responseXML;

document.getElementById("to").innerHTML=
xmlDoc.getElementsByTagName("to")[0].childNodes[0].nodeValue;
document.getElementById("from").innerHTML=
xmlDoc.getElementsByTagName("from")[0].childNodes[0].nodeValue;
document.getElementById("message").innerHTML=
xmlDoc.getElementsByTagName("body")[0].childNodes[0].nodeValue;
</script>
</body>
</html>
```

بعض دوال الـ DOM :

- الدالة getElementsByTagName() تستخدم للوصول إلى العناصر وهي تعيد مصفوفة لذا لابد من استخدام الـ index حتى لو كان لدي عنصر واحد بهذا الاسم
- للوصول إلى قيمة خاصية نستخدم

```
txt=xmlDoc.getElementsByTagName("title")[0].getAttribute("lang
");
```

حيث هذه التعليمة ترجع قيمة الخاصية "lang" من العنصر " title"

- لتغيير قيمة عنصر نستخدم ما يلي

```
x=xmlDoc.getElementsByTagName("title")[0].childNodes[0
];

    x.nodeValue="Easy Cooking";
```

- الدالة setAttribute () تستخدم لإنشاء خاصية جديدة أو لتغيير قيمة خاصية موجودة مسبقا

المثال التالي يضيف خاصية جديدة (edition="first") لكل عنصر <book> :

```
x=xmlDoc.getElementsByTagName("book");


for(i=0;i<x.length;i++)
{
x[i].setAttribute("edition","first");
}
```

- لإنشاء عنصر جديد لدينا مجموعة من الدوال:
  - createElement() تنشأ عنصر جديد
  - الدالة createTextNode() تنشأ عقدة نصية
  - الدالة appendChild() تضيف عقدة ابن إلى عقدة ما

المثال التالي يشمل الدوال السابقة :

```
newel=xmlDoc.createElement("edition");
newtext=xmlDoc.createTextNode("First");
newel.appendChild(newtext);
```

- لحذف عنصر نستخدم ما يلي:

```
x=xmlDoc.getElementsByTagName("book")[0];
x.removeChild(x.childNodes[0]);
```

: XML namespaces

توفر طريقة لمنع التضارب في أسماء العناصر بما أن العناصر تسمى من قبل المستخدم فإنه يمكن عند دمج عدة مستندات XML مع بعضها أن نجد عناصر ذات معنى مختلف لها نفس الاسم لتجنب هذه المشكلة نستخدم الـ namespaces نعرف الـ namespace باستخدام الخاصية xmlns في وسم البداية للعنصر

مثال:

```
<root>
<h:table xmlns:h="http://www.w3.org/TR/html4/">
 <h:tr>
  <h:td>Apples</h:td>
  <h:td>Bananas</h:td>
 </h:tr>
</h:table>
<f:table xmlns:f="http://www.w3schools.com/furniture">
 <f:name>African Coffee Table</f:name>
```

```
  <f:width>80</f:width>
  <f:length>120</f:length>
</f:table>
</root>
```

كما يمكن أن نعرف الـ namespace في العنصر الجذر كما يلي:

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

....
```

: XML CDATA

في عملية الـ parsing يتم الأخذ بعين الاعتبار كل محتويات ملف الـ XML (النصية) فيما عدا النص المحتوى في مقطع الـ CDATA فيتم تجاهله

:(Parsed Character Data) PCDATA

بشكل افتراضي نعمل parsing لكل النص في مستند الـ XML مثلا :

```
<message>This text is also parsed</message>
```

عندما يتم parsing للعنصر message فالنص المحتوى بين وسمي الفتح والإغلاق لهذا العنصر أيضا يتم Parsing له

PDATA مصطلح يستخدم للدلالة على النص الذي سيتم Parsing له

CDATA - (Unparsed) Character Data

المصطلح CDATA يستخدم للدلالة على النص الذي لا يتم Parsing له

عرفنا سا بقا أنه من الخطأ استخدام المحرفين " < "و "& " ولتجنب هذا الخطأ يمكن أن نضع هذه المحارف في مقطع CDATA ليتم تجاهلها من قبل الـ Parser

مثال:

```
<script>
<![CDATA[
function matchwo(a,b){
if (a < b && a < 0) then  {
  return 1;  }
else  {
  return 0;  }}
]]>
</script>
```

ملاحظات :

- لا يمكن أن يحتوي مقطع الـ CDATA على "]]>"
- لا يمكن استخدام مقاطع CDATA متداخلة
- "]]>" التي تمثل نهاية المقطع لا يمكن أن تحوي فراغات