Школа

DTD

Добро пожаловать в школу DTD

Школа DTD

В школе DTD вы узнаете что такое DTD. Вы также узнаете как применять DTD для того, чтобы задавать допустимые элементы XML-документа. Изучайте DTD!

Содержание

Введение в DTD[1] (open link)

Введение в DTD - для чего он применяется.

DTD - строительные блоки XML[2] (open link)

Строительные блоки XML можно задать в DTD.

Элементы DTD[3] (open link)

Как используя DTD задать допустимые в данном XML-документе элементы.

Атрибуты DTD[4] (open link)

Как используя DTD задать допустимые атрибуты элементов в данном XML-документе.

Сущности (entities) DTD[5] (open link)

Как задавать сущности XML, используя DTD

Проверка DTD[6] (open link)

Как загружая XML-документ проверить его на ошибки DTD

Примеры DTD[7] (open link)

Несколько реальных примеров DTD

1: http://xml.nsu.ru/dtd/dtd_intro.xml
2: http://xml.nsu.ru/dtd/dtd_building.xml
3: http://xml.nsu.ru/dtd/dtd_elements.xml
4: http://xml.nsu.ru/dtd/dtd_attributes.xml
5: http://xml.nsu.ru/dtd/dtd_entities.xml
6: http://xml.nsu.ru/dtd/dtd_validation.xml
7: http://xml.nsu.ru/dtd/dtd_examples.xml

Введение в DTD

Задача определения типа документа (DTD - Document Type Definition) - определить допустимые строительные блоки XML-документа. В нем задается структура документа и список допустимых элементов
DTD может быть объявлена внутри XML-документа или ссылкой на внешний файл DTD

Внутренняя декларация DOCTYPE

Если DTD включено в ваш исходный XML-документ, оно может быть заключено в декларацию DOCTYPE с помощью следующего синтаксиса:

```
<!DOCTYPE root-element [element-declarations]>
```

Вот пример XML-документа со встроенным DTD:

```
<?xml version="1.0"?>
<!DOCTYPE note [
  <!ELEMENT note (to,from,heading,body)>
  <!ELEMENT to      (#PCDATA)>
  <!ELEMENT from    (#PCDATA)>
  <!ELEMENT heading (#PCDATA)>
  <!ELEMENT body    (#PCDATA)>
]>
<note>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend</body>
</note>
```

Вы можете открыть этот XML-документ в Internet Explorer: [8] (open xml)

Это DTD интерпретируется следующим образом:

!DOCTYPE note (строка 2) определяет, что этот документ имеет тип note.

!ELEMENT note (строка 3) определяет, что элемент note содержит четыре элемента: "to,from,heading,body".

!ELEMENT to (строка 4) определяет, что элемент to должен иметь тип "#PCDATA".

!ELEMENT from (строка 5) определяет, что элемент from должен иметь тип "#PCDATA" и т.д.

Внешняя декларация DOCTYPE

Если декларация DTD находится вне вашего XML-документа, она должна включаться в определение DOCTYPE следующим образом:

```
<!DOCTYPE root-element SYSTEM "filename">
```

Это такой же документ, что был приведен выше, но со внешней DTD:

```
<?xml version="1.0"?>
<!DOCTYPE note SYSTEM "note.dtd">
<note>
```

8: http://xml.nsu.ru/dtd/note_in_dtd.xml

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

Вы можете открыть этот XML-документ в Internet Explorer: [9] (open xml)

Вот как выглядит файл "note.dtd", который содержит DTD:

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

Зачем используется DTD?

С помощью DTD каждый ваш XML-файл может заключать в себе описание своего собственного формата.

При помощи DTD независимые группы людей могут достичь согласия относительно использования общего DTD при обмене данными.

Ваша программа может использовать стандартную DTD для проверки правильности данных, которые вы получаете извне.

Вы можете использовать DTD для проверки своих собственных данных.

DTD - строительные блоки XML

Главными строительными элементами XML и HTML-документов являются элементы, заключенные в тэги, например, <body> .... </body>

Строительные блоки XML-документов

С точки зрения DTD, XML (и HTML) - документ состоит из следующих простых строительных блоков:

- Элементы (Elements)
- Тэги (Tags)
- Атрибуты (Attributes)
- Сущности (Entities)
- PCDATA
- CDATA

Далее мы познакомимся с кратким описанием каждого строительного блока

Элементы

Элементы - главные строительные блоки XML и HTML-документов.

Примерами HTML-элементов являются "body" and "table". Примерами XML-элементов могут

9: http://xml.nsu.ru/dtd/note_ex_dtd.xml

быть "note" and "message". Элементы могут содержать текст, другие элементы или быть пустыми. Пример пустых HTML- элементов: "hr", "br" и "img".

Тэги

Тэги применяются для разметки элементов.

Начальный тэг, например, <element_name> отмечает начало элемента, а конечный тэг, </element_name> отмечает его конец.

Примеры:

```
Элемент body размечен тэгами body:
<body>body text in between</body>.

Элемент message размечен тэгами message:
<message>some message in between</message>
```

Атрибуты

Атрибуты дают дополнительную информацию об элементах.

Атрибуты всегда располагаются внутри начального тэга элемента. Атрибуты применяются в виде пары имя атрибута/его значение. В этом примере элемент "img" содержит дополнительную информацию о файле-источнике:

```
<img src="computer.gif" />
```

Имя элемента - "img". Имя атрибута - "src". Значение атрибута равно "computer.gif". Поскольку сам элемент пустой, он сразу закрывается: " /".

Сущности

Сущности - это переменные, которые применяются для определения кусков текста, который используется в нескольких местах. Ссылки на сущности вызывают эти куски текста.

Большинство из вас знают ссылку на сущность, которая используется в HTML: " ". Это сущность "no-breaking-space". В HTML она используется для введения в документ дополнительных пробелов. Когда документ анализируется парсером, ссылкм на сущности заменяются кусками соответствующего текста.

Вот список сущностей, изначально заданных в XML:

Ссылка на сущность	Символ
<	< (знак меньше)
>	> (знак больше)
&	& (амперсанд)
"	" (двойная кавычка)
'	' (апостроф)

PCDATA

PCDATA расшифровывается как парсируемые символьные данные (parsed character data).

Символьные данные - это текст, расположенный между начальным и конечным тэгом XML-элемента.

PCDATA - это текст, который будет анализироваться парсером. Тэги в тексте будут трактоваться как разметка, а сущности будут заменяться соответствующими кусками текста.

CDATA

CDATA также расшифровывается как символьные данные.

CDATA - это текст, который НЕ будет анализироваться парсером. Тэги в тексте НЕ будут трактоваться как разметка, а сущности не будут заменяться соответствующими кусками текста.

Элементы DTD

В DTD элементы XML объявляются с помощью DTD-декларации ELEMENT

Декларация ELEMENT

В DTD элементы XML объявляются с помощью DTD-декларации element. При этом используется следующий синтакс:

```
<!ELEMENT element-name category>
```

Или:

```
<!ELEMENT element-name (element-content)>
```

Пустые элементы

Пустые элементы объявляются с помощью ключевого категориального слова EMPTY:

```
<!ELEMENT element-name EMPTY>
```

Пример DTD:

```
<!ELEMENT br EMPTY>
```

Пример XML:

```
<br />
```

Элементы, содержащие только символьные данные

Элементы, содержащие только символьные данные объявляются с помощью #PCDATA в круглых скобках:

```
<!ELEMENT element-name (#PCDATA)>
```

Пример:

```
<!ELEMENT from (#PCDATA)>
```

Элемент с произвольным содержимым

Элементы, объявленные с помощью ключевого категориального слова ANY могут содержать любой набор парсируемых данных:

```
<!ELEMENT element-name ANY>
```

Пример:

```
<!ELEMENT note ANY>
```

Элементы, содержащие один (или несколько) дочерних элементов

Элементы, содержащие один или несколько дочерних элементов определяются перечислением имен дочерних элементов в круглых скобках:

```
<!ELEMENT element-name
 (child-element-name)>
```

Или:

```
<!ELEMENT element-name
 (child-element-name,child-element-name,.....)>
```

Пример:

```
<!ELEMENT note (to,from,heading,body)>
```

Когда элементы объявляются с помощью последовательности, разделенной запятыми, дочерние элементы должны появляться в документе в той же последовательности. В полной DTD-декларации дочерние элементы должны также быть объявлены, при этом сами дочерние элементы также могут содержать свои дочерние элементы. Полная декларация документа note будет выглядеть так:

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

Объявление только одного вхождения элемента в родительский элемент

```
<!ELEMENT element-name (child-name)>
```

Пример:

```
<!ELEMENT note (message)>
```

В этом примере декларации объявляется, что дочерний элемент message может появляться внутри элемента note только один раз.

Объявление минимум одного вхождения элемента в родительский элемент

```
<!ELEMENT element-name (child-name+)>
```

Пример:

```
<!ELEMENT note (message+)>
```

Знак "+" этом примере декларации означает, что дочерний элемент message должен появляться внутри элемента note как минимум один раз.

Объявление произвольного числа вхождений элемента в родительский элемент

```
<!ELEMENT element-name (child-name*)>
```

Пример:

```
<!ELEMENT note (message*)>
```

Знак "*" этом примере декларации означает, что дочерний элемент message должен появляться внутри элемента note 0 или несколько раз.

Объявление нулевого или единичного числа вхождений элемента в родительский элемент

```
<!ELEMENT element-name (child-name?)>
```

Пример:

```
<!ELEMENT note (message?)>
```

Знак "?" этом примере декларации означает, что дочерний элемент message должен появляться внутри элемента note 0 или один раз.

Объявление элементов с условиями вхождения "либо-либо" или с условием "или/и"

Пример:

```
<!ELEMENT note (to,from,header,(message|body))>
```

В этом этом примере декларации объявляется, что элемент note должен содержать элемент to, элемент from, элемент header, и один из двух элементов: message или body.

Объявление смешанного содержимого

Пример:

```
<!ELEMENT note (#PCDATA|to|from|header|message)*>
```

В этом этом примере декларации объявляется, что элемент note должен содержать парсируемые символьные данные или/и любое число элементов to, from, header и message.

Атрибуты DTD

В DTD атрибуты объявляются с помощью DTD-декларации ATTLIST

Объявление атрибутов

При объявлении атрибутов используется следующий синтаксис:

```
<!ATTLIST element-name attribute-name
 attribute-type default-value>
```

Пример DTD:

```
<!ATTLIST payment type CDATA "check">
```

Пример XML:

```
<payment type="check" />
```

Типы значений атрибутов могут быть следующие:

Тип значения	Пояснение
CDATA	Значение - символьные данные
(en1|en2|..)	Список допустимых значений атрибута
ID	Значение атрибута - уникальный id
IDREF	Значение атрибута - id другого элемента
IDREFS	Значением атрибута является список других id
NMTOKEN	Значением атрибута является допустимое XML-имя
ENTITY	Значением атрибута является сущность
ENTITIES	Значением атрибута является список сущностей
NOTATION	Значением атрибута является запись (notation)
xml:	Значением атрибута является изначально заданное в XML значение

Описание применения атрибута может иметь следующие значения:

Значения	Пояснение
value	Атрибут имеет значение по умолчанию
#DEFAULT значение	Атрибут имеет значение по умолчанию
#REQUIRED	Атрибута обязательно должен присутствовать в элементе
#IMPLIED	Атрибута не обязательно должен присутствовать в элементе
#FIXED значение	Значение атрибута фиксировано

Пример объявления атрибута

Пример DTD:

```
<!ELEMENT square EMPTY>
<!ATTLIST square width CDATA "0">
```

Пример XML:

```
<square width="100"></square>
```

В этом примере объявляется, что элемент square - пустой и имеет атрибут width типа CDATA. Если атрибута width в элементе нет, значение атрибута width приравнивается 0.

Значение атрибута по умолчанию

Синтаксис:

```
<!ATTLIST element-name attribute-name
attribute-type "default-value">
```

Пример DTD:

```
<!ATTLIST payment type CDATA "check">
```

Пример XML:

```
<payment type="check" />
```

Задание значения атрибута по умолчанию гарантирует, что атрибут будет иметь определенное значение, даже если автор XML-документа его не задал.

Необязательный атрибут

Синтаксис:

```
<!ATTLIST element-name attribute-name
attribute-type #IMPLIED>
```

Пример DTD:

```
<!ATTLIST contact fax CDATA #IMPLIED>
```

Пример XML:

```
<contact fax="555-667788" />
```

Используйте атрибут IMPLIED, если вы не хотите заставлять автора XML-документа применять данный атрибут. При этом вы не можете задавать значение по умолчанию.

Обязательный атрибут

Синтаксис:

```
<!ATTLIST element-name attribute_name
 attribute-type #REQUIRED>
```

Пример DTD:

```
<!ATTLIST person number CDATA #REQUIRED>
```

Пример XML:

```
<person number="5677" />
```

Используйте атрибут REQUIRED, если вы не можете применять значение атрибута по умлочанию и при этом хотите заставить автора применять данный атрибут.

Фиксированное значение атрибута

Синтаксис:

```
<!ATTLIST element-name attribute-name
 attribute-type #FIXED "value">
```

Пример DTD:

```
<!ATTLIST sender company CDATA #FIXED "Microsoft">
```

Пример XML:

```
<sender company="Microsoft" />
```

Используйте атрибут FIXED, если атрибут должен иметь фиксированное значение без возможности его изменения. Если автор использует другое значение, парсер выдаст сообщение об ошибке.

Список возможных значений атрибута

Синтаксис:

```
<!ATTLIST element-name
 attribute-name (en1|en2|..) default-value>
```

Пример DTD:

```
<!ATTLIST payment type (check|cash) "cash">
```

Пример XML:

```
<payment type="check" />
или
<payment type="cash" />
```

Используйте список допустимых значений атрибута, если вы хотите, чтобы атрибут имел одно из нескольких допустимых значений.

Сущности (entities) DTD

Сущности - это переменные, используемые для создания кратких ссылок на различные куски текста
Ссылки на сущности (entity references) - это указатели на сущности, являющиеся кусками какого-либо текста. Объявляемые сущности могут находиться как внутри, так и вовне документа

Объявление внутренней сущности

Синтаксис:

```
<!ENTITY entity-name "entity-value">
```

Пример DTD:

```
<!ENTITY writer "Donald Duck.">
<!ENTITY copyright "Copyright W3Schools.">
```

Пример XML:

```
<author>&writer;&copyright;</author>
```

Объявление внешней сущности

Синтаксис:

```
<!ENTITY entity-name SYSTEM "URI/URL">
```

Пример DTD:

```
<!ENTITY writer
 SYSTEM "http://www.w3schools.com/entities/entities.xml">
<!ENTITY copyright
 SYSTEM "http://www.w3schools.com/entities/entities.dtd">
```

Пример XML:

```
<author>&writer;&copyright;</author>
```

Проверка DTD

Internet Explorer 5.0 может проверить правильность вашего XML-документа на основе DTD

Проверка XML-парсером

Если вы попробуете открыть XML-документ, XML-парсер может выдать сообщение об ошибке. Получив доступ к объекту parseError, извлекая код ошибки, текст, содержащий ошибку и даже номер строки, содержащей ошибку, можно найти и устранить ее:

```
var xmlDoc = new ActiveXObject("Microsoft.XMLDOM")
xmlDoc.async="false"
xmlDoc.validateOnParse="true"
xmlDoc.load("note_dtd_error.xml")

document.write("<br>Error Code: ")
document.write(xmlDoc.parseError.errorCode)
document.write("<br>Error Reason: ")
document.write(xmlDoc.parseError.reason)
document.write("<br>Error Line: ")
document.write(xmlDoc.parseError.line)
```

Попробуйте в действии:[10] (open editor)

Или просто взгляните на XML-файл:[11] (open xml)

Отключение проверки

Проверка может быть отключена установкой XML-парсера validateOnParse="false":

```
var xmlDoc = new ActiveXObject("Microsoft.XMLDOM")
xmlDoc.async="false"
xmlDoc.validateOnParse="false"
```

10: http://www.w3schools.com/dtd/tryit.asp?filename=note_error_three">
11: http://xml.nsu.ru/dtd/note_dtd_error.xml

```
xmlDoc.load("note_dtd_error.xml")

document.write("<br>Error Code: ")
document.write(xmlDoc.parseError.errorCode)
document.write("<br>Error Reason: ")
document.write(xmlDoc.parseError.reason)
document.write("<br>Error Line: ")
document.write(xmlDoc.parseError.line)
```

Попробуйте в действии:[12] (open editor)

Общий XML-валидатор

Вы можете проверить любой XML-файл с помощью созданного нами специального валидатора:[13] (open link)

Объект parseError Object

Вы можете прочитать больше об объекте parseError на нашей школе XML DOM School[14] (open link)

Примеры DTD в Интернете

DTD телевизионной программы

Автор: David Moisan. Пример взят с его сайта: http://www.shore.net/~dmoisan/[1]

```
<!DOCTYPE TVSCHEDULE [

<!ELEMENT TVSCHEDULE (CHANNEL+)>
<!ELEMENT CHANNEL (BANNER, DAY+)>
<!ELEMENT BANNER (#PCDATA)>
<!ELEMENT DAY ((DATE, HOLIDAY) | (DATE, PROGRAMSLOT+))+>
<!ELEMENT HOLIDAY (#PCDATA)>
<!ELEMENT DATE (#PCDATA)>
<!ELEMENT PROGRAMSLOT (TIME, TITLE, DESCRIPTION?)>
<!ELEMENT TIME (#PCDATA)>
<!ELEMENT TITLE (#PCDATA)>
<!ELEMENT DESCRIPTION (#PCDATA)>

<!ATTLIST TVSCHEDULE NAME CDATA #REQUIRED>
<!ATTLIST CHANNEL CHAN CDATA #REQUIRED>
<!ATTLIST PROGRAMSLOT VTR CDATA #IMPLIED>
<!ATTLIST TITLE RATING CDATA #IMPLIED>
<!ATTLIST TITLE LANGUAGE CDATA #IMPLIED>
```

12: http://www.w3schools.com/dtd/tryit.asp?filename=note_error_four
13: http://www.w3schools.com/dom/validate_xml.asp">
14: http://xml.nsu.ru/dom/dom_home.xml
1: http://www.shore.net/~dmoisan/

```
]>
```

DTD газетной статьи

Пример взят с http://www.vervet.com/[2]

```
<!DOCTYPE NEWSPAPER [

<!ELEMENT NEWSPAPER (ARTICLE+)>
<!ELEMENT ARTICLE (HEADLINE, BYLINE, LEAD, BODY, NOTES)>
<!ELEMENT HEADLINE (#PCDATA)>
<!ELEMENT BYLINE (#PCDATA)>
<!ELEMENT LEAD (#PCDATA)>
<!ELEMENT BODY (#PCDATA)>
<!ELEMENT NOTES (#PCDATA)>

<!ATTLIST ARTICLE AUTHOR CDATA #REQUIRED>
<!ATTLIST ARTICLE EDITOR CDATA #IMPLIED>
<!ATTLIST ARTICLE DATE CDATA #IMPLIED>
<!ATTLIST ARTICLE EDITION CDATA #IMPLIED>

<!ENTITY NEWSPAPER "Vervet Logic Times">
<!ENTITY PUBLISHER "Vervet Logic Press">
<!ENTITY COPYRIGHT "Copyright 1998 Vervet Logic Press">

]>
```

DTD каталога продукции

Пример взят с http://www.vervet.com/[3]

```
<!DOCTYPE CATALOG [

<!ENTITY AUTHOR "John Doe">
<!ENTITY COMPANY "JD Power Tools, Inc.">
<!ENTITY EMAIL "jd@jd-tools.com">

<!ELEMENT CATALOG (PRODUCT+)>

<!ELEMENT PRODUCT
(SPECIFICATIONS+,OPTIONS?,PRICE+,NOTES?)>
<!ATTLIST PRODUCT
NAME CDATA #IMPLIED
CATEGORY (HandTool|Table|Shop-Professional) "HandTool"
PARTNUM CDATA #IMPLIED
PLANT (Pittsburgh|Milwaukee|Chicago) "Chicago"
INVENTORY (InStock|Backordered|Discontinued) "InStock">

<!ELEMENT SPECIFICATIONS (#PCDATA)>
<!ATTLIST SPECIFICATIONS
WEIGHT CDATA #IMPLIED
```

2: http://www.vervet.com/
3: http://www.vervet.com/

```
POWER CDATA #IMPLIED>

<!ELEMENT OPTIONS (#PCDATA)>
<!ATTLIST OPTIONS
FINISH (Metal|Polished|Matte) "Matte"
ADAPTER (Included|Optional|NotApplicable) "Included"
CASE (HardShell|Soft|NotApplicable) "HardShell">

<!ELEMENT PRICE (#PCDATA)>
<!ATTLIST PRICE
MSRP CDATA #IMPLIED
WHOLESALE CDATA #IMPLIED
STREET CDATA #IMPLIED
SHIPPING CDATA #IMPLIED>

<!ELEMENT NOTES (#PCDATA)>

]>
```

Developed by Metaphor (c) 2002