XML

Intro

Problem

Как удобно хранить и передавать данные?

Solution

- **XML**
- **XML** был создан для того, что бы информацию можно было:
  - структурировать
  - хранить
  - передавать

XML

XML

- **XML** - аббревиатура от англ. **eXtensible Markup Language** (расширяемый язык разметки).
- **XML** - язык разметки, который напоминает **HTML**.
- **XML** предназначен для передачи данных, а не для их отображения (в отличии от **HTML**).
- Теги **XML** не предопределены. Вы должны сами определять нужные теги.
- **XML** описан таким образом, чтобы быть самоопределяемым.

Example

```
<?xml version="1.0" encoding="UTF-8"?>
<note>
    <to>Nick</to>
    <from>Mike</from>
    <heading>Напоминание</heading>
    <body>Не забудь обо мне в эти выходные!</body>
</note>
```

About tags

- В языке **XML** нет предопределенных тегов.
- **XML** позволяет автору определять свои языковые теги и свою структуру документа.

Как используют XML

- **XML** отделяет данные от HTML
- **XML** упрощает распределение данных
- **XML** упрощает передачу данных
- **XML** упрощает модификацию платформы
- **XML** делает ваши данные более доступными
- **XML** используется для создания новых интернет-языков

Tree structure

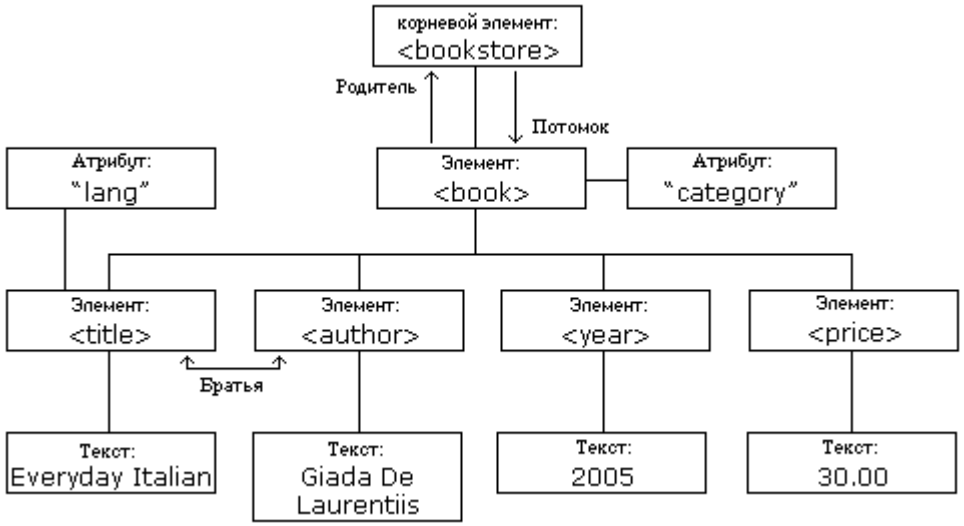
Tree structure

```
<корневой>
    <потомок>
        <подпотомок>...</подпотомок>
    </потомок>
</корневой>
```

Tree structure

```xml
<bookstore>
    <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>
```

Tree structure



Правила синтаксиса XML

Все XML элементы должны иметь закрывающий тег

```
<!-- Bad -->
<p>Это параграф.
<br>
```

```
<!-- Good -->
<p>Это параграф.</p>
<br>
```

Теги XML регистрозависимы

```
<!-- Bad -->
<Message>Это неправильно</message>
```

```
<!-- Good -->
<message>Это правильно</message>
```

XML элементы должны соблюдать корректную вложенность

```
<!-- Bad -->
<b><i>Это жирный и курсивный текст</b></i>
```

```
<!-- Good -->
<b><i>Это жирный и курсивный текст</i></b>
```

У XML документа должен быть корневой элемент

```
<!-- Bad -->
<корневой>
    <потомок>
    </потомок>
</корневой>
<корневой>
    <потомок>
    </потомок>
</корневой>
```

```
<!-- Good -->
<корневой>
    <потомок>
        <подпотомок>.....</подпотомок>
    </потомок>
</корневой>
```

XML пролог

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

Значения XML атрибутов должны заключаться в кавычки

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

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

Сущности

```
<!-- Bad -->
<message>если жалование < 1000</message>
```

```
<!-- Good -->
<message>если жалование &lt; 1000</message>
```

Сущности

Сущность	Символ	Значение
`<`	`<`	меньше, чем
`>`	`>`	больше, чем
`&`	`&`	амперсанд
`&apos;`	`'`	апостроф
`&quot;`	`"`	кавычки

Только символы `<` и `&` строго запрещены в XML. Символ `>` допустим, но лучше его всегда заменять на сущность.

Комментарии в XML

```
<!-- Это комментарий -->

<!-- Это - - комментарий -->
```

В XML пробелы сохраняются

В XML новая строка сохраняется как LF

- Windows новая строка хранится в следующем виде: символ перевода каретки и символ новой строки (CR+LF)
- Unix и Mac OSX используют LF
- Старые Mac системы используют CR
- XML сохраняет новую строку как LF

Валидация XML документов

Валидация XML документов

- XML документ с корректным синтаксисом называется **Well Formed XML**
- "Валидный" XML документ кроме всего прочего должен соответствовать *определенному типу* документов.

Определения документа

С XML можно использовать различные типы определений документа:

- **DTD** (**Document Type Definition**) - оригинальное определение типа документа
- **XML schema/XSD**(**XML Schema Definition**) - более новый тип определений, основанный на XML.

DTD

XML example

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE note SYSTEM "note.dtd">
<note>
    <to>Nick</to>
    <from>Mike</from>
    <heading>Напоминание</heading>
    <body>Не забудь обо мне в эти выходные!</body>
</note>
```

DTD example

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

Использование DTD для определения сущностей

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE note [
<!ENTITY nbsp " ">
<!ENTITY writer "Writer: Donald Duck.">
<!ENTITY copyright "Copyright: MSiter.RU.">
]>

<note>
    <to>Nick</to>
    <from>Mike</from>
    <heading>Напоминание</heading>
    <body>Не забудь обо мне в эти выходные!</body>
    <footer>&writer; &copyright;</footer>
</note>
```

Зачем нужно использовать DTD?

- С **DTD** ваш **XML** файл может нести собственный формат.
- С **DTD** различные, не связанные друг с другом группы людей могут приходить к соглашению о стандартах пересекающихся данных.
- С **DTD** вы можете быть уверены, что получаемые из внешних источников данные будут корректными.

Когда не стоит использовать DTD?

- Для работы **XML** не требуется **DTD**.
- Когда вы экспериментируете с **XML** или работаете с небольшими **XML** файлами, то создание **DTD** может оказаться излишней тратой времени.
- Если вы разрабатываете приложения, то стоит подождать, пока спецификации не станут стабильными, и только тогда добавлять определения. В обратном случае ваше приложение может перестать работать из-за ошибок валидации.

XML Schema

XML example

```xml
<?xml version="1.0" encoding="UTF-8"?>
<note xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:noNamespaceSchemaLocation="note.xsd">
    <to>Nick</to>
    <from>Mike</from>
    <heading>Напоминание</heading>
    <body>Не забудь обо мне в эти выходные!</body>
</note>
```

XML schema example

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <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>
</xsd:schema>
```

XML схема мощнее DTD

- **XML** схема мощнее **DTD**
- **XML** схема пишется на **XML**
- **XML** схема легко расширяется
- **XML** схема поддерживает типы данных
- **XML** схема поддерживает пространства имен

Зачем нужно использовать XML схему?

- С **XML** схемой ваш **XML** файл может нести собственный формат.
- С **XML** схемой различные, не связанные друг с другом группы людей могут приходить к соглашению о стандартах пересекающихся данных.
- С **XML** схемой вы можете проверять корректность данных.

XML схема поддерживает типы данных

Одним из мощнейших свойств **XML схемы** является поддержка типов данных:

- Упрощается описание содержимого документа
- Упрощается определение ограничений по данным
- Упрощается проверка корректности данных
- Упрощается преобразование данных из одного типа в другой

XML схема использует синтаксис XML

- Вам не нужно изучать новый язык программирования
- Вы можете использовать тот же **XML** редактор для создания файлов **XML** схем
- Вы можете использовать тот же **XML** парсер для разбора файлов **XML** схем
- Вы можете манипулировать **XML** схемами при помощи **XML DOM**
- Вы можете трансформировать **XML** схемы при помощи **XSLT**

XPath

XPath

- **XPath** stands for **XML Path Language**
- **XPath** предоставляет специальный синтаксис для поиска и выборки данных в XML документе.
- Используя **XPath** выражения, можно из любой части **XML**:
  - произвести выборку по условию
  - найти узлы
  - найти точное значение

Selecting Nodes

Expression	Description
`nodename`	Selects all nodes with the name "nodename"
`/`	Selects from the root node
`//`	Selects nodes in the document from the current node that match the selection no matter where they are
`.`	Selects the current node
`..`	Selects the parent of the current node
`@`	Selects attributes

Selecting Nodes: examples

Path Expression	Result
`bookstore`	Selects all nodes with the name "bookstore"
`/bookstore`	Selects the root element bookstore (Note: If the path starts with a slash (/) it always represents an absolute path to an element!)
`bookstore/book`	Selects all book elements that are children of bookstore
`//book`	Selects all book elements no matter where they are in the document
`bookstore//book`	Selects all book elements that are descendant of the bookstore

	element, no matter where they are under the bookstore element
`//@lang`	Selects all attributes that are named lang

Predicates

Path Expression	Result
`/bookstore/book[1]`	Selects the first book element that is the child of the bookstore element. (Note: In IE 5,6,7,8,9 first node is[0], but according to W3C, it is [1].
`/bookstore/book[last()]`	Selects the last book element that is the child of the bookstore element
`/bookstore/book[last()-1]`	Selects the last but one book element that is the child of the bookstore element
`/bookstore/book[position()<3]`	Selects the first two book elements that are children of the bookstore element

Predicates

`//title[@lang]`	Selects all the title elements that have an attribute named lang
`//title[@lang='en']`	Selects all the title elements that have a "lang" attribute with a value of "en"
`/bookstore/book[price>35.00]`	Selects all the book elements of the bookstore element that have a price element with a value greater than 35.00
`/bookstore/book[price>35.00]/title`	Selects all the title elements of the

	book elements of the bookstore element that have a price element with a value greater than 35.00

Selecting Unknown Nodes

Wildcard	Description
`*`	Matches any element node
`@*`	Matches any attribute node
`node()`	Matches any node of any kind

Selecting Unknown Nodes: examples

Path Expression	Result
`/bookstore/*`	Selects all the child element nodes of the bookstore element
`//*`	Selects all elements in the document
`//title[@*]`	Selects all title elements which have at least one attribute of any kind

Selecting Several Paths

Path Expression	Result
`//book/title | //book/price`	Selects all the title AND price elements of all book elements
`//title | //price`	Selects all the title AND price elements in the document
`/bookstore/book/title | //price`	Selects all the title elements of the book element of the bookstore element AND all the price elements in the document