Professor Hans Noodles

41 уровень

17.05.2019 55354 8

Что такое XML

Статья из группы Java Developer

42757 участников

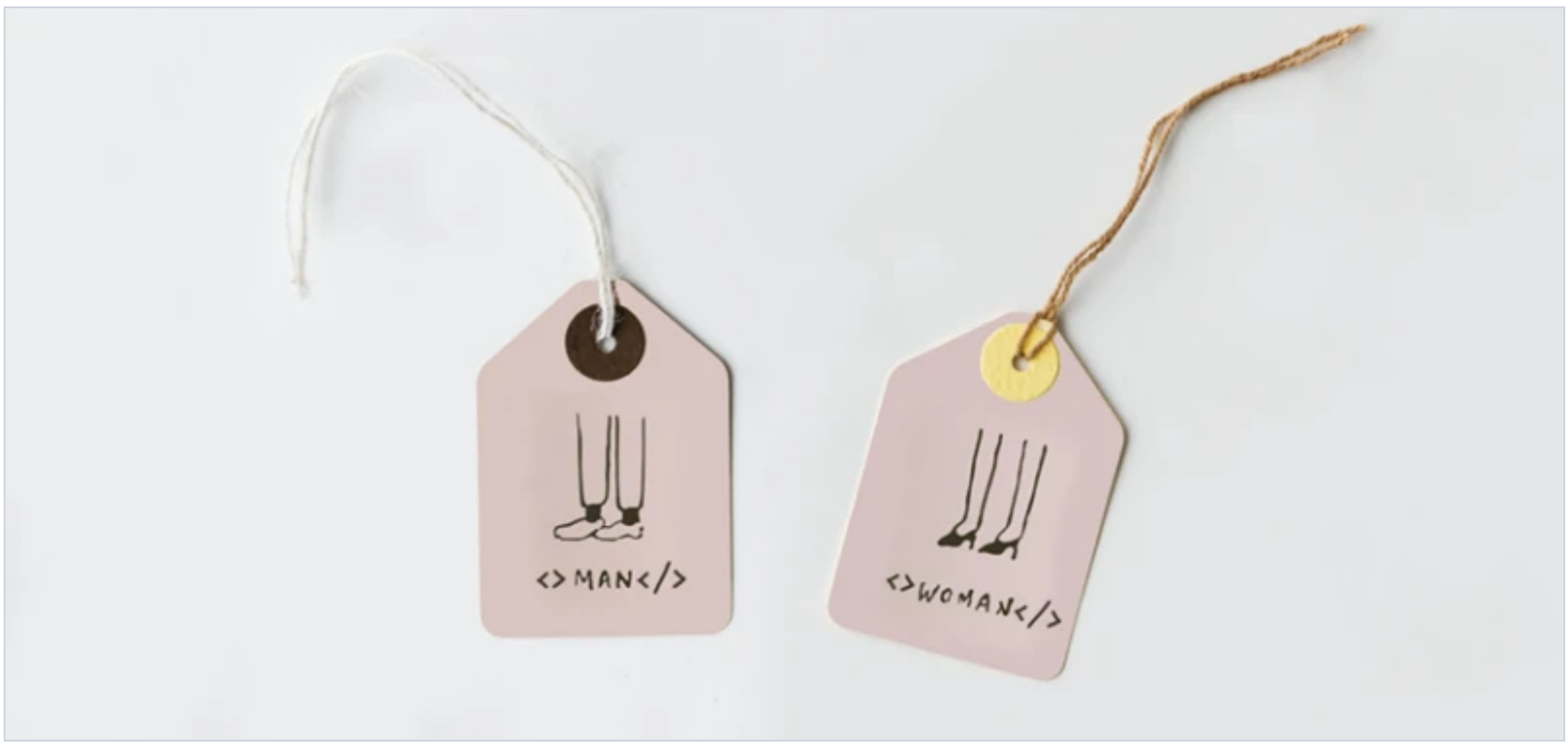
Вы в группе

Привет!

Сегодня мы познакомимся с еще одним форматом данных, который называется **XML**.

Это очень важная тема. В работе над настоящими Java-приложениями ты почти наверняка столкнешься с задачами, связанными с XML. Этот формат используется в Java-разработке практически повсеместно (зачем именно — узнаем ниже), поэтому рекомендую тебе читать лекцию не «по диагонали», а разобраться во всем досконально и заодно изучить дополнительную литературу/ссылки :) Это время точно не будет потрачено впустую.

Итак, начнем с простого — «что» и «зачем»!



Что такое XML?

XML расшифровывается как eXtensible Markup Language — «расширяемый язык разметки».
Один из языков разметки тебе, возможно, уже знаком: ты слышал об HTML, с помощью которого создаются веб-страницы :)
HTML и XML похожи даже внешне:

HTML 1	XML 1
`<h1>title</h1>`	`<headline>title</headline>`

HTML 2	XML 2

`<h1>title</h1>`
`<p>paragraph</p>`
`<p>paragraph</p>`
	`<chief>title</chief>`
	`<paragraph>paragraph</paragraph>`
	`<paragraph>paragraph</paragraph>`

Иными словами, XML — это язык для описания данных.

Зачем нужен XML?

XML изначально придумали для более удобного хранения и передачи данных, в том числе через Интернет.
У него есть ряд преимуществ, которые позволяют успешно справляться с этой задачей.

Во-первых, он легко читается и человеком, и компьютером.

Думаю, ты без труда поймешь, что описывает этот xml-файл:

```
<?xml version="1.0" encoding="UTF-8"?>
<book>
   <title>Harry Potter and the Philosopher’s Stone</title>
   <author>J. K. Rowling</author>
   <year>1997</year>
</book>
```

Компьютер тоже без труда понимает такой формат.

Во-вторых, поскольку данные хранятся в простом текстовом формате, при их передаче с одного компьютера на другой не возникнет никаких проблем с совместимостью.

Важно понимать, что **XML — это не исполняемый код, а язык описания данных**. После того, как ты описал данные с помощью XML, тебе нужно написать код (например, на Java), который сможет эти данные отправить/принять/обработать.

Как устроен XML?

Его главная составная часть — теги: вот такие штуки в угловых скобках:

```
<book>
</book>
```

Теги бывают открывающими и закрывающими. У закрывающего есть дополнительный символ — “`/`”, это видно на примере выше.

Каждому открывающему тегу должен соответствовать закрывающий.
Они показывают, где начинается и где заканчивается описание каждого элемента в файле.

Теги могут быть вложенными!

В нашем примере с книгой у тега **<book>** есть 3 вложенных тега — **<title>**, **<author>** и **<year>**.

Это не ограничивается одним уровнем: у вложенных тегов могут быть свои вложенные теги, и т. д. Такая конструкция называется деревом тегов.

НАЧАТЬ ОБУЧЕНИЕ

```xml
<?xml version="1.0" encoding="UTF-8"?>
<carstore>
   <car category="truck">
       <model lang="en">Scania R 770</model>
       <year>2005</year>
       <price currency="US dollar">200000.00</price>
   </car>
   <car category="sedan">
       <title lang="en">Ford Focus</title>
       <year>2012</year>
       <price currency="US dollar">20000.00</price>
   </car>
   <car category="sport">
       <title lang="en">Ferrari 360 Spider</title>
       <year>2018</year>
       <price currency="US dollar">150000.00</price>
   </car>
</carstore>
```

Здесь у нас есть тег верхнего уровня — <**carstore**>. Его еще называют «root» — корневой тег.

У <carstore> есть один дочерний тег — <car>. У <car>, в свою очередь, тоже есть 3 своих дочерних тега — <model>, <year> и <price>.

Каждый тег может иметь атрибуты — дополнительную важную информацию. В нашем примере у тега <model> есть атрибут «lang» — язык, на котором написано название модели:

```xml
<model lang="en">Scania R 770</model>
```

Так мы можем указать, что название написано на английском языке.

У нашего тега <price> (цена) есть атрибут «currency» — «валюта».

```xml
<price currency="US dollar">150000.00</price>
```

Так мы можем указать, что цена за машину указана в американских долларах.

Таким образом, **у XML есть «самоописывающий» синтакс**. Ты можешь добавить любую нужную тебе информацию для описания данных.

Также в начало файла можно добавить строку с указанием версии XML и кодировки, в которой записаны данные.

Она называется «**prolog**» и выглядит вот так:

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

Мы применяем XML версии 1.0 и кодировку UTF-8. Это не обязательно, но может пригодиться, если ты, например, используешь в своем файле текст на разных языках.

Мы упомянули о том, что XML переводится как «расширяемый язык разметки», но что значит «расширяемый»?

НАЧАТЬ ОБУЧЕНИЕ

К примеру, мы хотим, чтобы в нашем автосалоне начали продавать еще и мотоциклы!

При этом в программе нам нужно поддерживать обе версии <carstore> — и старую (без мотоциклов), и новую.

Вот наша старая версия:

```
<?xml version="1.0" encoding="UTF-8"?>
<carstore>
   <car category="truck">
       <model lang="en">Scania R 770</model>
       <year>2005</year>
       <price currency="US dollar">200000.00</price>
   </car>
   <car category="sedan">
       <title lang="en">Ford Focus</title>
       <year>2012</year>
       <price currency="US dollar">20000.00</price>
   </car>
   <car category="sport">
       <title lang="en">Ferrari 360 Spider</title>
       <year>2018</year>
       <price currency="US dollar">150000.00</price>
   </car>
</carstore>
```

А вот новая, расширенная:

```
<?xml version="1.0" encoding="UTF-8"?>
<carstore>
   <car category="truck">
       <model lang="en">Scania R 770</model>
       <year>2005</year>
       <price currency="US dollar">200000.00</price>
   </car>
   <car category="sedan">
       <title lang="en">Ford Focus</title>
       <year>2012</year>
       <price currency="US dollar">20000.00</price>
   </car>
   <car category="sport">
       <title lang="en">Ferrari 360 Spider</title>
       <year>2018</year>
       <price currency="US dollar">150000.00</price>
   </car>
   <motorcycle>
       <title lang="en">Yamaha YZF-R6</title>
       <year>2018</year>
       <price currency="Russian Ruble">1000000.00</price>
       <owner>Vasia</owner>
   </motorcycle>
   <motorcycle>
       <title lang="en">Harley Davidson Sportster 1200</title>
       <year>2011</year>
```

НАЧАТЬ ОБУЧЕНИЕ

```
    </motorcycle>
</carstore>
```

Вот так легко и просто мы добавили описание мотоциклов в наш файл :)

При этом нам совершенно не нужно задавать для мотоциклов те же дочерние теги, что и для машин. Обрати внимание, что у мотоциклов, в отличие от машин, есть элемент <owner> — владелец.

Это никак не помешает компьютеру (да и человеку тоже) прочитать данные.

Отличия XML от HTML

Мы уже сказали, что XML и HTML внешне очень похожи. Поэтому, очень важно знать, чем они отличаются.

Во-первых, они используются для разных целей.

HTML — для разметки веб-страниц. Например, если тебе нужно создать веб-сайт, с помощью HTML ты сможешь указать: «Меню должно быть в верхнем правом углу. В нем должны быть такие-то кнопки». Иными словами, задача HTML — отображение данных.

XML — для хранения и передачи информации в удобном для человека и компьютера виде. Этот формат не содержит никаких указаний на то, как эти данные нужно отображать: это зависит от кода самой программы.

Во-вторых, у них есть основное техническое отличие. Теги HTML являются заранее заданными («predefined»).

Иными словами, для создания заголовка (например, большой надписи в начале страницы) в HTML используются только теги <h1></h1> (для заголовков поменьше — <h2></h2>,<h3></h3>). Не получится создать заголовки в HTML, используя теги с другими названиями.

XML не использует заранее заданные теги. Ты можешь давать тегам любые названия, какие захочешь — <header>, <title>, <idontknow2121>.

Разрешение конфликтов

Свобода, которую предоставляет XML, может привести и к некоторым проблемам.

К примеру, одна и та же сущность (например, машина) может использоваться программой в разных целях.

К примеру, у нас есть XML-файл в котором описаны машины. Однако, наши программисты не договорились заранее между собой. И теперь, помимо данных реальных автомобилей, в наши xml попадают еще и данные игрушечных моделей! Более того, у них одинаковые атрибуты.

В нашу программу приходит вот такой XML-файл. Как же нам отличить настоящую машину от игрушечной модельки?

```
<?xml version="1.0" encoding="UTF-8"?>
<carstore>
   <car category="truck">
       <model lang="en">Scania R 770</model>
       <year>2005</year>
       <price currency="US dollar">200000.00</price>
   </car>
   <car category="sedan">
       <title lang="en">Ford Focus</title>
       <year>2012</year>
```

```
   </car>
</carstore>
```

Здесь нам помогут префиксы и пространства имен.

Чтобы отделять в нашей программе игрушечные машины от настоящих (да и вообще — любые игрушечные вещи от их реальных прототипов), мы вводим два префикса — «real» и «toy».

```
<real:car category="truck">
   <model lang="en">Scania R 770</model>
   <year>2005</year>
   <price currency="US dollar">200000.00</price>
</real:car>
<toy:car category="sedan">
   <title lang="en">Ford Focus</title>
   <year>2012</year>
   <price currency="US dollar">100.00</price>
</toy:car>
```

Теперь наша программа сможет различить сущности! Все, что имеет префикс toy, будет отнесено к игрушкам :)

Однако, мы пока не закончили. Чтобы использовать префиксы, нам надо зарегистрировать каждый из них в качестве пространства имен (namespace).

Ну, на самом деле, «зарегистрировать» — это громко сказано :) Достаточно просто придумать уникальное имя для каждого из них.

Это как с классами: у класса есть короткое имя (`Cat`) и полное имя с указанием всех пакетов (`zoo.animals.Cat`)

Для создания уникальных имен namespace обычно используют URI. Иногда сюда подставляют адрес в Сети, где подробно описаны функции предназначение этого пространства имен.

Но это не обязательно должен быть действующий интернет-адрес. Очень часто на проектах используют просто URI-подобные строки, которые помогают отследить иерархию пространств имен.

Вот пример:

```
<?xml version="1.0" encoding="UTF-8"?>
<carstore xmlns:real="http://testproject.developersgroup1.companyname/department2/namespaces/real"
          xmlns:toy="http://testproject.developersgroup1.companyname/department2/namespaces/toy">
<real:car category="truck">
   <model lang="en">Scania R 770</model>
   <year>2005</year>
   <price currency="US dollar">200000.00</price>
</real:car>
<toy:car category="sedan">
   <title lang="en">Ford Focus</title>
   <year>2012</year>
   <price currency="US dollar">100.00</price>
</toy:car>
</carstore>
```

НАЧАТЬ ОБУЧЕНИЕ

Но тут есть полезная информация: за создание пространства имен «real» отвечает группа разработчиков «developersgroup1» из отдела «department2». Если нужно будет внести новые имена, или обсудить с ними возможные конфликты, мы знаем куда обратиться.

Иногда в качестве уникального имени для namespace используют реальный адрес в Сети с описанием этого пространства имен. Например, если это большая компания, и ее проект будет использоваться миллионами людей по всему миру. Но это делается далеко не всегда: на Stackoverflow есть обсуждение этого вопроса.

В принципе, требование использовать URI в качестве имен для namespace не является строгим: можно и просто рандомные строки.

Такой вариант тоже будет работать:

```
xmlns:real="nvjneasiognipni4435t9i4gpojrmeg"
```

Но у использования URI есть ряд преимуществ. Подробнее об этом ты можешь почитать вот тут.

Основные стандарты XML

Стандарты XML — это набор расширений, которые придают xml-файлам дополнительные возможности.

XML имеет очень много стандартов, но мы лишь посмотрим на самые важные из них, и узнаем, что они позволяют делать

AJAX — один из самых известных стандартов XML. Он позволяет изменять содержимое веб-страницы без ее перезагрузки! Звучит круто? :) Можешь испробовать эту технологию лично вот тут.

XSLT — позволяет преобразовывать XML-текст в другие форматы. Например, используя XSLT, ты можешь преобразовать XML в HTML! Задача XML, как мы уже говорили, — описание данных, а не отображение. Но с использованием XSLT мы можем обойти это ограничение!

Вот здесь есть «песочница» с работающим примером, где ты можешь сам посмотреть как это работает :)

XML DOM — позволяет получать, изменять, добавлять или удалять отдельные элементы из XML-файла.

Вот небольшой пример как это работает. У нас есть файл books.xml:

```
<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>
</bookstore>
```

В нем две книги. У книг есть такой элемент как заголовок — <title>.

консоль:

```
<!DOCTYPE html>
<html>
<body>

<p id="demo"></p>

<script>
var xhttp = new XMLHttpRequest();
xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
  myFunction(this);
  }
};
xhttp.open("GET", "books.xml", true);
xhttp.send();

function myFunction(xml) {
    var xmlDoc = xml.responseXML;
  document.getElementById("demo").innerHTML =
  xmlDoc.getElementsByTagName("title")[0].childNodes[0].nodeValue;
}
</script>

</body>
</html>
```

Опять же, рекомендую посмотреть, как работает этот пример, используя песочницу:)

DTD («document type definition») — позволяет определить список разрешенных элементов для какой-то сущности в XML-файле.

К примеру, мы работаем над сайтом книжного магазина, и все команды разработчиков договорились, что для элемента book в XML-файлах должны быть указаны только атрибуты title, author и year.

Но как нам защитить себя от невнимательности?

Очень легко!

```
<?xml version="1.0"?>
<!DOCTYPE book [
       <!ELEMENT book (title,author,year)>
       <!ELEMENT title (#PCDATA)>
       <!ELEMENT author (#PCDATA)>
       <!ELEMENT year (#PCDATA)>
       ]>

<book>
   <title>The Lord of The Rings</title>
   <author>John R.R. Tolkien</author>
   <year>1954</year>
</book>
```

Здесь мы определили список допустимых атрибутов для <book>. Попробуй добавить туда новый элемент — и сразу получишь ошибку!

```
<book>
   <title>The Lord of The Rings</title>
   <author>John R.R. Tolkien</author>
   <year>1954</year>
   <mainhero>Frodo Baggins</mainhero>
</book>
```

Ошибка! “Element mainhero is not allowed here”

Есть и много других XML-стандартов. Ознакомиться с каждым из них и попробовать «поковыряться» в коде ты можешь на сайте WC3 (раздел «Important XML Standarts»).

Да и вообще, если тебе нужна информация по XML, там можно найти практически все :)

Ну а наша лекция на этом подошла к концу.

Настало время вернуться к задачам! :)

До встречи!

+151

Комментарии (8) популярные новые старые

JavaCoder

Введите текст комментария

Модератор Уровень 51, Молдова 9 декабря 2021, 21:49

Ответить 0

Павел Уровень 10, Москва, Россия 1 апреля 2020, 17:16

Всем привет
Почему в одном примере model, в другом title
<model lang="en">Scania R 770</model>
<year>2005</year>
<price currency="US dollar">200000.00</price>
</real:car>
<toy:car category="sedan">
<title lang="en">Ford Focus</title>
<year>2012</year>
<price currency="US dollar">100.00</price>

Ответить 0

xodavit Уровень 28, Москва, Россия 9 апреля 2020, 11:25

потому, что: один из них real:car, а другой toy:car и они могут быть разные, т.к. это разные описания и ты сам из задаешь.
поправьте, если не прав.

Ответить +1

НАЧАТЬ ОБУЧЕНИЕ

десериализации (если в данном случае описываются объекты). И не получилось бы такое провернуть с использованием DTD

Ответить

0

Артемий Уровень 8, Минск

27 марта 2020, 22:58

Спасибо автору очень круто все изложил по полочкам. Ни одного вопроса не возникло.!

Ответить

0

Andre Makkin Уровень 23, Rybinsk, Россия

17 мая 2019, 17:02

А какое приложение выведет ошибку "Ошибка! “Element mainhero is not allowed here”" ?

Ответить

+3

Джон Дориан Уровень 37, Россия

12 июня 2019, 22:39

просто создайте xml-файл в IDEa и скопируйте туда содержимое из лекции. Ошибка подсветится прямо в файле

Ответить

+4

Andre Makkin Уровень 23, Rybinsk, Россия

12 июня 2019, 23:20

открывал всегда xml в Notepad++, чет даже в голову не пришло в idea попробовать. +

Ответить

+2