链滴

xml 和 xml 解析

作者：whoms

原文链接：https://ld246.com/article/1640251662462

来源网站：链滴

<h2 id="XML">XML</h2>
<h4 id="1--XML介绍">1. XML 介绍</h4>
<blockquote>
<code>eXtensible Markup Language</code> 可扩展标记语言，通常用于各种组或者框架的配置文件，文件内容由各种标签组成
</blockquote>
<h4 id="2--XML与HTML区别">2. XML 与 HTML 区别</h4>
<table>
<thead>
<tr>
<th>区别</th>
<th>HTML</th>
<th>XML</th>
</tr>
</thead>
<tbody>
<tr>
<td>功能</td>
<td>制作网页</td>
<td>用于配置文件</td>
</tr>
<tr>
<td>大小写</td>
<td>不区分大小写</td>
<td>区分大小写</td>
</tr>
<tr>
<td>语法严谨性</td>
<td>不严谨，若一个标签有开始，没有结束，浏览器也能解析</td>
<td>很严谨，标签的开始和结束必须严格配对</td>
</tr>
<tr>
<td>可扩展性</td>
<td>没有可扩展性，所有的标签的功能都已固定</td>
<td>可以扩展，因为所有的标签都是自定义的</td>
</tr>
</tbody>
</table>
<h4 id="3--XML组成">3. XML 组成</h4>
<blockquote>
XML 文件由 7 个组成元素构成：

文档声明

语法：以 <code><?xml</code> 开头，以 <cod>?></code> 结尾
位置：必须在 XML 文件的第一行
例子：<code><?xml version="1.0" encoding="UTF-8"?></code>
三个属性：

<table>
<thead>

```
<tr>
<th><strong>文档声明的三个属性</strong></th>
<th><strong>说明</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>version</strong></td>
<td><strong>用于指定 XML 使用哪个版本，固定的写法 1.0</strong></td>
</tr>
<tr>
<td><strong>enconding</strong></td>
<td><strong>指定当前 XML 编码</strong></td>
</tr>
<tr>
<td><strong>standalone</strong></td>
<td><strong>yes/no 默认是 yes，这个 XML 文件是否是一个单独的文档</strong></td>
</tr>
</tbody>
</table>
</li>
<li>
<p><strong>标签元素 Element</strong></p>
</li>
</ul>
<table>
<thead>
<tr>
<th><strong>语法：有主体标签 &lt; 标签 &gt; 和无主体标签：&lt; 标签/&gt;</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>主体部分：分为有主体和无主体标签，主体部分可以包含文本或者元素</strong></t
>
</tr>
<tr>
<td><strong>空元素：无主体标签也需要关闭</strong></td>
</tr>
<tr>
<td><strong>大小写：标签名区分大小写</strong></td>
</tr>
<tr>
<td><strong>命名不能有空格，不能有冒号</strong></td>
</tr>
<tr>
<td><strong>根元素：每个 XML 文档有且仅有一个根元素，</strong></td>
</tr>
</tbody>
</table>
<ul>
<li><strong>属性 Attribute</strong></li>
</ul>
<table>
```

```
<thead>
<tr>
<th><strong>属性位置：必须放在开始标签中</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>属性的值必须使用双引号或单引号</strong></td>
</tr>
<tr>
<td><strong>在同一个标签不能出现多个同名的属性</strong></td>
</tr>
<tr>
<td><strong>命名中不能出现空格和冒号</strong></td>
</tr>
</tbody>
</table>
<ul>
<li><strong>注释 Comment</strong>
<ul>
<li><strong>与 HTML 注释相同，不可嵌套</strong></li>
</ul>
</li>
<li><strong>转义字符（实体字符）</strong></li>
</ul>
<table>
<thead>
<tr>
<th><strong>说明</strong></th>
<th><strong>字符</strong></th>
<th><strong>转义字符</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>小于</strong></td>
<td><strong>&amp;lt;</strong></td>
<td><strong>lt;</strong></td>
</tr>
<tr>
<td><strong>大于</strong></td>
<td><strong>&amp;gt;</strong></td>
<td><strong>gt;</strong></td>
</tr>
<tr>
<td><strong>双引号</strong></td>
<td><strong>"</strong></td>
<td><strong>quot;</strong></td>
</tr>
<tr>
<td><strong>单引号</strong></td>
<td><strong>'</strong></td>
<td><strong>apos;</strong></td>
</tr>
```

<tr>
<td>与符号</td>
<td>&</td>
<td>amp;</td>
</tr>
</tbody>
</table>

字符数据区（用于显示大量特殊字符）

格式：<code><![CDATA[文本数据]]></code>
可以让某些字符不被 XML 解析，始终作为纯文本来操作

处理指令（不常用）

格式：<code><?xml-stylesheet 属性=""... ?></code>
作用：处理指令，简称 PI（Processing instruction）用来指挥解析引擎如何解析 XML 文档

</blockquote>
<h4 id="4--XML作用">4. XML 作用</h4>
<blockquote>
主要用于数据的处理和表达，因为 HTML 等其他标记语言无法准确表达其本质内容，主要是表现样式，导致带来诸多不便，所以 XML 就诞生了
</blockquote>
<blockquote>
在企业开发中的主要应用场景：

数据交换，用于不同的系统之间，或不同的数据库之间进行数据交换
用于各种框架的配置

</blockquote>
<h4 id="5--XML文件约束">5. XML 文件约束</h4>
<blockquote>
因为 XML 是可扩展标记语言，可以写任意标记，所以需要对 XML 进行一些约束，保 XML 文件数据的正确性和有效性
</blockquote>

<h5 id="DTD约束">DTD 约束</h5>

<blockquote>
DTD：Document Type Definition 文档类型定义
作用：用于约束 XML 文件，DTD 本身是个文本文件
</blockquote>
<table>
<thead>
<tr>
<th>导入 DTD 文件的两种格式</th>
<th>说明</th>

```
</tr>
</thead>
<tbody>
<tr>
<td>&lt;<strong>!DOCTYPE 根元素 SYSTEM "DTD 文件"</strong>&gt;</td>
<td><strong>系统的 DTD 文件，使用范围比较小，一般用于公司，不对外开放的 DTD 文件</stro
g></td>
</tr>
<tr>
<td>&lt;<strong>!DOCTYPE 根元素 PUBLIC "DTD 文件"</strong>&gt;</td>
<td><strong>公共的 DTD 文件，用于互联网上，使用广泛的用途</strong></td>
</tr>
</tbody>
</table>
<ul>
<li>
<h5 id="Schema约束"><strong>Schema 约束</strong></h5>
</li>
</ul>
<blockquote>
<ul>
<li><strong>XML Schema Definition 比 DTD 强大，是 DTD 的替代者</strong></li>
<li><strong>本身也是 XML 文档，扩展名为 xsd</strong></li>
<li><strong>功能更强大，数据约束类型更完善</strong></li>
</ul>
</blockquote>
<h2 id="XML解析"><strong>XML 解析</strong></h2>
<ul>
<li>
<h4 id="三种解析方式"><strong>三种解析方式</strong></h4>
<ul>
<li><strong>DOM：要求解析器把整个 XML 文档装载到内存并解析称为一个 Document 对象</st
ong>
<ul>
<li><strong>优点：元素与元素之间保留了结构关系，可以进行增删改查操作</strong></li>
<li><strong>缺点：若 XML 过大可能出现内存溢出</strong></li>
</ul>
</li>
<li><strong>SAX：是一种速度更快更有效的方法。逐一扫描文档，一边扫描一边解析。并以事件
动的方式进行具体解析，每执行一行，都触发对应的事件</strong>
<ul>
<li><strong>优点：处理速度快，可以处理大文件</strong></li>
<li><strong>缺点：只能读，逐行后将释放资源，解析操作繁琐</strong></li>
</ul>
</li>
<li><strong>PULL：Android 内置的 XML 解析方式，类似 SAX</strong></li>
</ul>
</li>
<li>
<h4 id="常见解析开发包"><strong>常见解析开发包</strong></h4>
<ul>
<li><strong>JAXP：Oracle 公司提供支持 DOM 和 SAX 开发包</strong></li>
<li><strong>Dem4j：比较简单的解析开发包，将整个 XML 加载到内存中作为一个 DOM 树</stro
g></li>
```

JDom：与 Dem4j 类似
Jsoup：功能强大的 DOM 方式的 XML 解析包，同时对 HTML 的解析也很方便

<h2 id="Jsoup">Jsoup</h2>
<blockquote>
将整个文档加载到内存，生成一个 DOM 树，并获得一个 Document 对象，通过 Document 对象操作 DOM 树
</blockquote>

<h5 id="获取Document对象的三种方式">获取 Document 对象的三种方式</h5>

<code>public static Document parse(String html)</code>：将一段 HTML 内容化成一个 Document 对象
<code>public static Document parse(File in,String charsetName)</code>：指定 XML 或者 HTML 文件和字符集，得到一个 Document 对象
<code>public static Document connect(String url)</code>：解析互联网上的某个 URL 地址，创建连接，通过连接获取一个 Document 对象

<h5 id="获取元素对象">获取元素对象</h5>

通过 GET 方式得到元素对象

<table>
<thead>
<tr>
<th>返回类型</th>
<th>Document 对象的方法</th>
<th>说明</th>
</tr>
</thead>
<tbody>
<tr>
<td>Element</td>
<td>getElementById(String id)</td>
<td>通过 id 得到唯一元素对象</td>
</tr>
<tr>
<td>Elements</td>
<td>getElementsByTag(String tagName)</td>
<td>通过标签名得到一组元素</td>
</tr>
<tr>
<td>Elements</td>
<td>getElementsByClass(String className)</td>
<td>通过类名得到一组元素对象</td>
</tr>
</tbody>

```
</table>
<ul>
<li><strong>通过 CSS 选择器得到元素</strong></li>
</ul>
<table>
<thead>
<tr>
<th><strong>返回类型</strong></th>
<th><strong>方法</strong></th>
<th><strong>说明</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Elements</strong></td>
<td><strong><strong>select(String cssQuery)</strong></strong></td>
<td><strong>作用：通过选择器得到多个元素</strong></td>
</tr>
<tr>
<td><strong>Element</strong></td>
<td><strong>selectFirst(String cssQuery)</strong></td>
<td><strong>作用：通过选择器得到第一个元素</strong></td>
</tr>
</tbody>
</table>
</li>
<li>
<h5 id="选择器"><strong>选择器</strong></h5>
<ul>
<li><strong>基本选择器</strong></li>
</ul>
<table>
<thead>
<tr>
<th><strong>选择器类型</strong></th>
<th><strong>描述</strong></th>
<th><strong>选择器语法</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ID 选择器</strong></td>
<td><strong><strong>根据 id 获取元素，id 前面使用</strong>#<strong>符号</strong></st
ong></td>
<td>#<strong>id</strong></td>
</tr>
<tr>
<td><strong>class 选择器</strong></td>
<td><strong><strong>根据元素 class 属性获取元素</strong></strong></td>
<td><strong>.类名</strong></td>
</tr>
<tr>
<td><strong>标签选择器</strong></td>
<td><strong><strong>根据元素名称获取元素</strong></strong></td>
```

<td>标签名</td>
</tr>
<tr>
<td>属性选择器</td>
<td>获取含有该属性名的所有元素</td>
<td>[属性名]</td>
</tr>
<tr>
<td>属性选择器</td>
<td>利用属性值来查找元素</td>
<td>[属性名=属性值]</td>
</tr>
</tbody>
</table>

组合选择器

<table>
<thead>
<tr>
<th>选择器代码</th>
<th>说明</th>
</tr>
</thead>
<tbody>
<tr>
<td>标签名[属性名]</td>
<td>属性选择器，包含指定属性的标签。 如：a[href]，即选中 a 标签包含 href 属性的元
</td>
</tr>
<tr>
<td>标签名.类名</td>
<td>交集选择器，同时指定标签名和类名的选择器 例如 div.one 即 <<stron
>div class="one"> 内容 </div></
d>
</tr>
<tr>
<td>父元素 > 子元素</td>
<td>子选择器，找某个元素下的所有子元素，选择器之间使用空格隔开</td>
</tr>
<tr>
<td>父元素 子元素</td>
<td>后代选择器，找到父元素下面的子元素（包括子元素下面的所有子元素、孙元素等）<
strong></td>
</tr>
<tr>
<td>兄弟 A+ 兄弟 B</td>
<td>兄弟选择器，查找在 A 元素后面第一个同级元素 B</td>
</tr>
</tbody>
</table>

<h2 id="JsoupXPath">JsoupXPath</h2>

<blockquote>
JsoupXPath 是一款纯 Java 开发的使用 xpath 解析 HTML 的解析器，JsoupXPath 是 Jsoup 的一部分，是在 Jsoup 基础上进行的扩展。XPath 使用路径表达式来选取 HTML 或 XML 档中的元素节点或属性节点
</blockquote>

<h5 id="核心API">核心 API</h5>

JXDocument

<code>public JXDocument JXDocument(Document doc)</code>：通过 Document 对象创建一个新的 JXDocument 对象
<code>public List<JXNode> selN(String xpath)</code>：通过 xpath 表达得到指定的节点对象集合
<code>public JXNode selOne(String xpath)</code>：通过 xpath 表达式得到符条件的节点对象

JXNode
<blockquote>
JXNode 表示一个节点对象，元素、文本都是节点对象
</blockquote>

<code>public Element getElement()</code>：通过节点得到元素
<code>public List<JXNode> sel(String xpath)</code>：用在相对路径上，当前节点开始向下查询其他子节点

<h5 id="四种XPath语法表达方式">四种 XPath 语法表达方式</h5>

相对路径表达式

以“/”开头，一级一级描述标签的层级路径，不可跨越层级（从 body 开始往下搜索
开头的“/”表示根元素

相对路径表达式

以“./”开始，一级一级描述标签的层级路径，不可跨越层级
“.”表示当前路径

全文搜索路径

“//”符号，不用逐层写路径，直接选取到对应的节点
<code>//li</code>：查询所有的 li 元素，无论在哪一级
<code>//a@href</code>：查询所有的 a 元素，找到它的 href 属性

```
</ul>
</li>
<li><strong>条件筛选</strong>
<ul>
<li><code>//元素[@属性名=值]</code><strong>：获取“属性=值”的所有元素</strong></li>

<li><code>//元素[@属性名&gt;值]@属性名</code><strong>：获取“属性 &gt; 值”的所有元的属性值</strong></li>
<li><code>//元素[@属性名=值]test()</code><strong>：获取符合条件元素的文本数据</strong></li>
<li><code>//元素[@属性名=值]html()</code><strong>：获取符合条件元素的 HTML 数据</strong></li>
</ul>
</li>
</ul>
</li>
</ul>
```