XML 技术

主要内容

XML 的发展历程

XML 基本语法

XML 文件格式

XML 约束

XML 解析思想

XML 解析器

Jsoup

XPath

XML 案例

学习目标

知识点	要求
XML 的发展历程	了解
XML 基本语法	掌握
XML 文件格式	掌握
XML 约束	了解
XML 解析思想	了解
XML 解析器	了解
Jsoup	掌握
XPath	掌握
XML 案例	掌握

一、 XML 概述

1 概念

XML（Extensible Markup Language）：可扩展标记语言

可扩展：标签都是自定义的。

2 发展历程

HTML 和 XML 都是 W3C（万维网联盟）制定的标准，最开始 HTML 的语法过于松散，于是 W3C 制定了更严格的 XML 语法标准，希望能取代 HTML。但是程序员和浏览器厂商并不喜欢使用 XML，于是现在的 XML 更多的用于配置文件及传输数据等功能。

配置文件：在今后的开发过程当中我们会频繁使用框架（框架：半成品软件），使用框架时，需要写配置文件配置相关的参数，让框架满足我们的开发需求。而我们写的配置文件中就有一种文件类型是 XML。

传输数据：在网络中传输数据时并不能传输 java 对象，所以我们需要将 JAVA 对象转成字符串传输，其中一种方式就是将对象转为 XML 类型的字符串。

3 xml 和 html 的区别：

- xml 语法严格，html 语法松散
- xml 标签自定义，html 标签预定义

4 XML 基本语法

- 文件后缀名是.xml
- 第一行必须是文档声明
- 有且仅有一个根标签
- 属性值必须用引号（单双都可）引起来
- 标签必须正确关闭
- 标签名区分大小写

5 XML 组成部分

5.1 文档声明

文档声明必须放在第一行，格式为：<?xml 属性列表 ?>

属性列表：

version：版本号（必须）

encoding：编码方式

5.2 标签：

XML 中标签名是自定义的，标签名有以下要求：

- 包含数字、字母、其他字符
- 不能以数字和标点符号开头
- 不能包含空格

5.3 指令(了解)

指令是结合 css 使用的，但现在 XML 一般不结合 CSS，语法为：

<?xml-stylesheet type="text/css" href="a.css" ?>

5.4 属性

属性值必须用引号（单双都可）引起来

5.5 文本

如果想原样展示文本，需要设置 CDATA 区，格式为：<![CDATA[文本]]>

二、 约束

1 什么是约束

约束是一个文件，可以规定 xml 文档的书写规则。我们作为框架的使用者，不需要会写约束文件，只要能够在 xml 中引入约束文档，简单的读懂约束文档即可。XML 有两种约束文件类型：DTD 和 Schema。

2 DTD 约束

DTD 是一种较简单的约束技术

引入：

本地：<!DOCTYPE 根标签名 SYSTEM "dtd 文件的位置">

网络：<!DOCTYPE 根标签名 PUBLIC "dtd 文件的位置" "dtd 文件路径">

3 Schema 约束

引入：

（1）写 xml 文档的根标签

（2）引入 xsi 前缀：确定 Schema 文件的版本。

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

（3）引入 Schema 文件

xsi:schemaLocation="Schema 文件定义的命名空间 Schema 文件的具体路径"

（4）为 Schema 约束的标签声明前缀

xmlns:前缀="Schema 文件定义的命名空间"

例如：

```
<students
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://www.itbaizhan.cn/xml
student.xsd"
        xmlns="http://www.itbaizhan.cn/xml">
```

三、 XML 解析

XML 解析即读写 XML 文档中的数据。框架的开发者通过 XML 解析读取框架使用者配置的参数信息，开发者也可以通过 XML 解析读取网络传来的数据。

1 XML 解析思想

DOM：将标记语言文档一次性加载进内存，在内存中形成一颗 dom 树

优点：操作方便，可以对文档进行 CRUD 的所有操作

缺点：占内存

SAX：逐行读取，基于事件驱动的。

优点：不占内存，一般用于手机 APP 开发中读取 XML

缺点：只能读取，不能增删改

2 常见解析器

JAXP：SUN 公司提供的解析器，支持 DOM 和 SAX 两种思想

DOM4J：一款非常优秀的解析器

Jsoup：Jsoup 是一款 Java 的 HTML 解析器，可直接解析某个 URL 地址、HTML 文本内容。它提供了一套非常省力的 API，可通过 DOM，CSS 以及类似于 jQuery 的操作方法来取出和操作数据

PULL：Android 操作系统内置的解析器，支持 SAX 思想

四、 Jsoup 解析器

1 快速入门

步骤：

(1) 导入 jar 包

(2) 加载 XML 文档进内存，获取 DOM 树对象 Document

(3) 获取对应的标签 Element 对象

(4) 获取数据

```
// 获取 XML 中所有的学生姓名
public static void main(String[] args) throws IOException {
    // (2)加载 XML 文档进内存，获取 DOM 树对象 Document
    // 2.1 找到 XML 文档的绝对路径
    // 利用类加载器，通过项目中文件的相对路径找到硬盘中的绝对路径
    ClassLoader classLoader = Demo1.class.getClassLoader();
    String path =
classLoader.getResource("com/baizhan/xml/xsd/student.xml").getPath();
    // 2.2 根据 XML 文档的路径，把 XML 文档加载进内存，并解析成 Dom 树对象
    Document document = Jsoup.parse(new File(path), "utf-8");
    // (3)获取对应的标签 Element 对象
    Elements name = document.getElementsByTag("name");
    // (4)获取数据
    for (Element element : name) {
```

```
        String text = element.text();
        System.out.println(text);
    }
}
```

2 常用对象

- **Jsoup：**解析 xml 或 html，形成 dom 树对象。

  常用方法：

  static Document parse(File in, String charsetName)：解析本地文件

  static Document parse(String html)：解析 html 或 xml 字符串

  static Document parse(URL url, int timeoutMillis)：解析网页源文件

- **Document：xml 的 dom 树对象**

  常用方法：

  Element getElementById(String id)：根据 id 获取元素

  Elements getElementsByTag(String tagName)：根据标签名获取元素

  Elements getElementsByAttribute(String key)：根据属性获取元素

  Elements getElementsByAttributeValue(String key,String value)：根据属性名=属性值获取元素。

  Elements select(Sting cssQuery)：根据选择器选取元素。

- **Element: 元素对象**

常用方法：

String text()：获取元素包含的纯文本。

String html()：获取元素包含的带标签的文本。

String attr(String attributeKey)：获取元素的属性值。

3 XPath 解析

XPath 即为 XML 路径语言 ,它是一种用来确定标记语言文档中某部分位置的语言。

使用方法：

1. 导入 Xpath 的 jar 包
2. 获取 Document 对象
3. 将 Document 对象转为 JXDocument 对象
4. JXDocument 调用 selN(String xpath)，获取 List<JXNode>对象。
5. 遍历 List<JXNode>，调用 JXNode 的 getElement()，转为 Element 对象。
6. 处理 Element 对象。

五、 XML 案例

1 使用 Jsoup 完成网页爬虫

网络爬虫（web crawler）：自动抓取互联网信息的程序。

Jsoup 可以通过 URL 获取网页的 HTML 源文件，源文件中包含着网站数据，我们可以解析 HTML 源文件的数据来获取我们需要的信息。

步骤：

1. 引入 jar 包。
2. 使用 Jsoup 获取网页 HTML 源文件，转为 Document 对象
3. 通过 Document 对象，获取需要的 Element 对象
4. 获取 Element 对象的数据。
5. 设置循环自动爬取

2 使用 XML 配置爬虫程序的参数

爬虫程序有一些参数需要配置，如果直接将参数写在 JAVA 程序中，则修改参数非常不方便，所以此时我们将参数写在 XML 配置文件中，通过解析 XML 文件获取参数的配置信息。

步骤：

1. 写爬虫程序的 XML 配置文件
2. 解析 XML 配置文件：加载 XML 文件到内存，转为 Document 对象。
3. 获取 XML 配置文件的配置信息
4. 将配置信息应用于爬虫程序中

```
        int min = 0;
        int max = 0;
        try {
            // 2.解析 XML 配置文件：加载 XML 文件到内存，转为 Document 对象。
            String path =
CrawlerDemo.class.getClassLoader().getResource("com/baizhan/xml/crawler/Cr
```

```
awler.xml").getPath();
            Document document = Jsoup.parse(new File(path), "utf-8");
            // 3.获取 XML 配置文件的配置信息
            JXDocument jxDocument = new JXDocument(document);
            List<JXNode> minNodes = jxDocument.selN("//min");
            List<JXNode> maxNodes = jxDocument.selN("//max");

            Element minElement = minNodes.get(0).getElement();
            Element maxElement = maxNodes.get(0).getElement();
            String maxStr = maxElement.text();
            String minStr = minElement.text();

            // 4.将配置信息应用于爬虫程序中
            min = Integer.parseInt(minStr);
            max = Integer.parseInt(maxStr);
        } catch (IOException e) {
            e.printStackTrace();
        } catch (XpathSyntaxErrorException e) {
            e.printStackTrace();
        }
        // 设置循环自动爬取
        for (int i = min; i <= max; i++) {
```

```
            try {
                // 2.使用 Jsoup 获取网页 HTML 源文件，转为 Document 对象
                Document document = Jsoup.parse(new
URL("http://daily.zhihu.com/story/"+i), 3000);
//                System.out.println(document);
                // 3.通过 Document 对象，获取需要的 Element 对象
                // 获取头图、标题、作者、正文
                Elements headerImgEle =
document.getElementsByAttributeValue("alt", "头图");
                Elements headerEle = document.select(".DailyHeader-title");
                Elements authorEle = document.select(".author");
                Elements contentEle = document.select(".content");

                // 4.获取 Element 对象的数据。
                String headerImg = headerImgEle.get(0).attr("src");
                String header = headerEle.text();
                String author = authorEle.text();
                String content = contentEle.text();
                System.out.println(headerImg);
                System.out.println(header);
                System.out.println(author);
                System.out.println(content);
```

```
            } catch (IOException e) { }

        }
```