XML概述

XML

概念

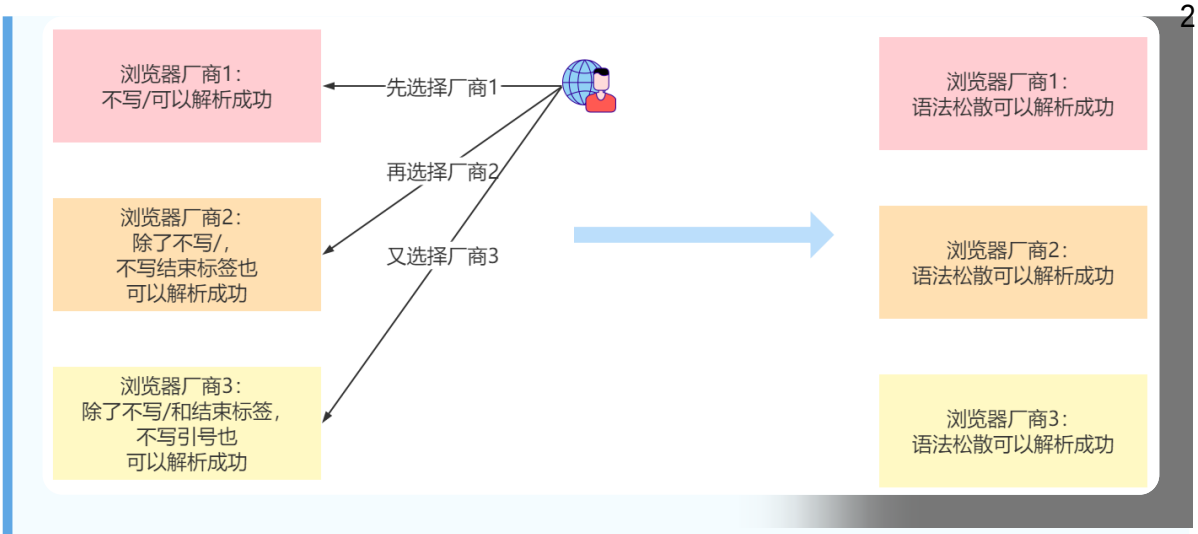
XML（Extensible Markup Language）：可扩展标记语言

可扩展：标签都是自定义的。

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。

> 是谁造成的HTML语法松散?
>
> 浏览器厂商。最开始W3C制定HTML的时候语法还是比较严格的。但浏览器厂商为了抢占市场，语法错误也可以解析成功HTML，最后“内卷”到HTML即使语法非常混乱也是可以被浏览器解析。



tips：归根到底是语法的**制定者**和**使用者**不一致造成了HTML语法混乱，JAVA语法严格就是因为java语言的运行工具java虚拟机也是sun公司（现在是oracle）出品的，语法不通过不让运行。

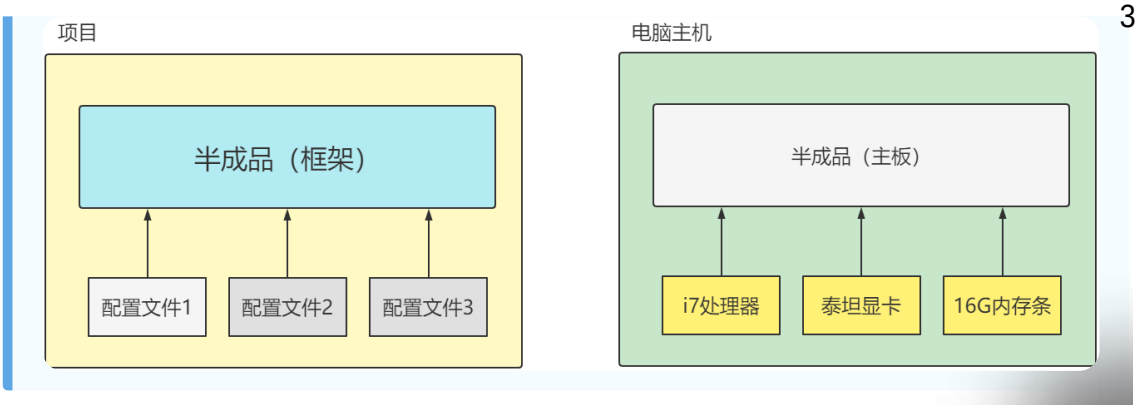
为什么程序员不使用XML写前端页面?

因为程序员松散惯了，不想写很严格的代码。同样挣一万块钱，谁会从每月上一天班的公司跳槽到996的公司呢?

XML的功能

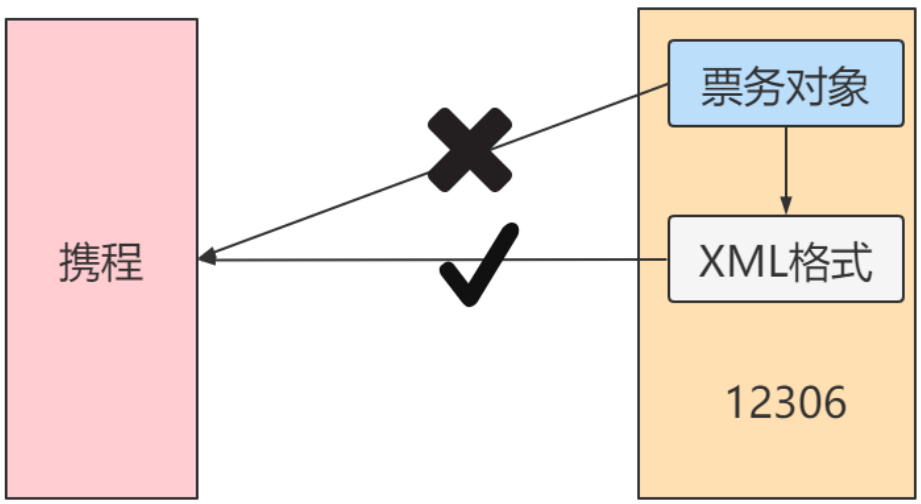
1. 配置文件：在今后的开发过程当中我们会频繁使用框架（框架：半成品软件），使用框架时，需要写配置文件配置相关的参数，让框架满足我们的开发需求。而我们写的配置文件中就有一种文件类型是XML。

   日后编写大型项目，不可能从头到尾都是原创代码，很多功能前人已经写好，我们只需要使用前人写好的半成品软件（框架），再加入一些符合我们需求的配置即可完成开发。比如我们组装一台电脑，不可能自己焊接电路板。而是先买入一块主板，这块主板就是半成品软件。根据自己的需求加入一些配置，比如要求流畅运行吃鸡，就需要配置i7处理器、泰坦显卡。



2. 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。

比如携程等旅游网站可以买火车票，但他们其实也是替12306卖票，此时他们就需要拿到12306的票务数据。JAVA对象不能在网络上传输，可以转为XML类型的字符串。



XML和HTML的区别

1. XML语法严格，HTML语法松散
2. XML标签自定义，HTML标签预定义

实时效果反馈

1. 关于XML，说法正确的是

A 标签都是自定义的

B 可用于编写配置文件

C 可用于网络数据传输

D 以上说法都正确

2. 关于XML和HTML的区别，说法正确的是

A XML标签预定义

B HTML标签预定义

C XML语法松散

D HTML语法严格

答案

1=>D　　2=>B

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
- 标签必须正确关闭
- 标签名区分大小写
- 属性值必须用引号（单双都可）引起来

实时效果反馈

1. 关于XML语法，说法正确的是

A 可以有多个根标签

B 第一行必须是根标签

C 第一行必须是文档声明

D 标签名不区分大小写

答案

1=>C

XML组成部分

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

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，可以以_开头

- 不能包含空格

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" ?>
```

属性

属性值必须用引号（单双都可）引起来

文本

如果想原样展示文本，需要设置CDATA区，格式为：

```
<![CDATA[文本]]>
```

实时效果反馈

1. XML文档组成部分不包括

A 文本

B 脚本

C 标签

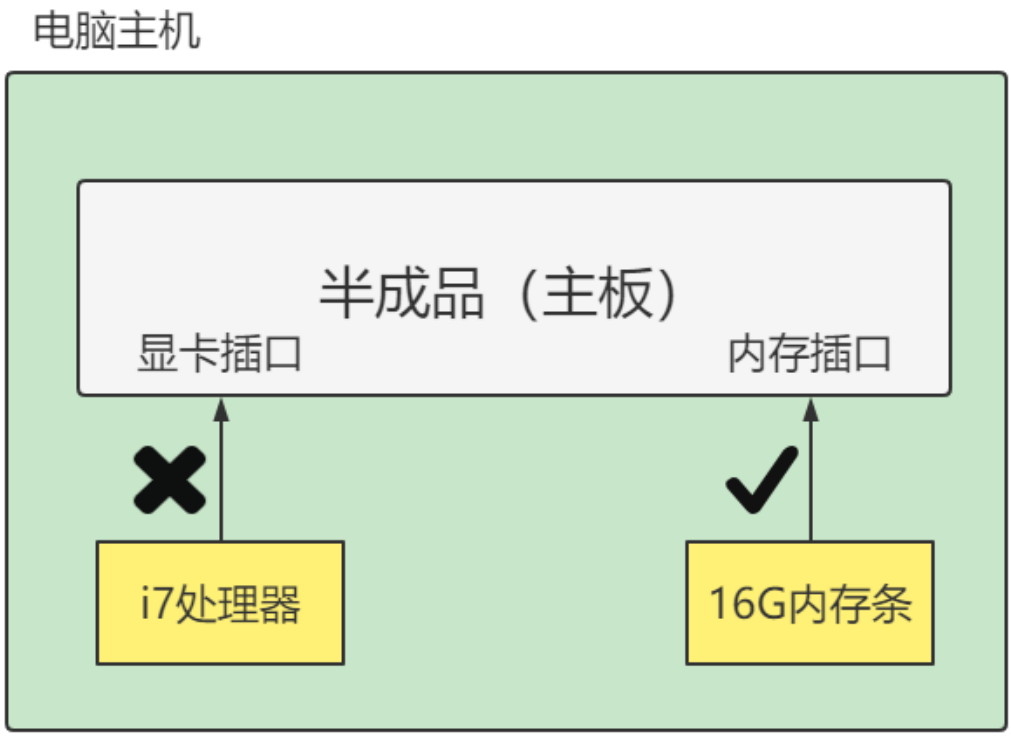
D 文档声明

答案

1=>B

约束_DTD约束

虽然XML标签是自定义的。但是作为配置文件时，也需要遵循一定的规则。就比如在主板上硬盘口只能插硬盘，不能插入其他硬件。约束就是定义XML书写规则的文件，约束我们按照框架的要求编写配置文件。



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。

DTD是一种较简单的约束技术，引入方式如下：

- 本地引入：

```
<!DOCTYPE 根标签名 SYSTEM "dtd文件的位置">
```

- 网络引入：

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

实时效果反馈

1. XML约束类型有

A DTD约束

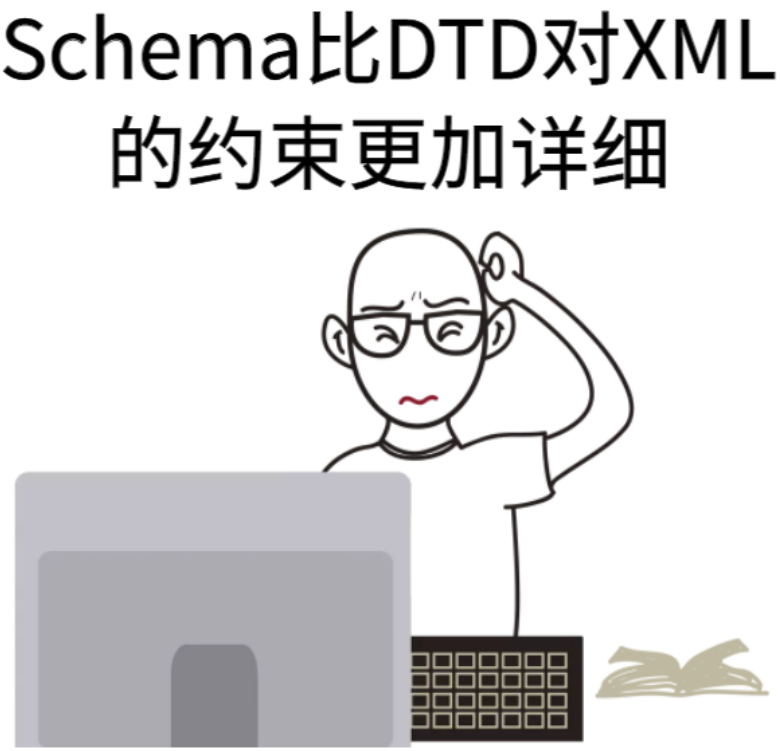
B Schema约束

C 以上都包括

答案

1=>C

约束_Schema约束



Schema比DTD对XML的约束更加详细，引入方式如下：

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">
```

实时效果反馈

1. 关于XML约束，说法正确的是

A DTD约束比Schema约束更详细

B Schema约束比DTD约束更详细

C Schema约束和DTD约束详细度差不多

D 以上说法都不正确

答案

1=>D

Jsoup解析器_XML解析思想

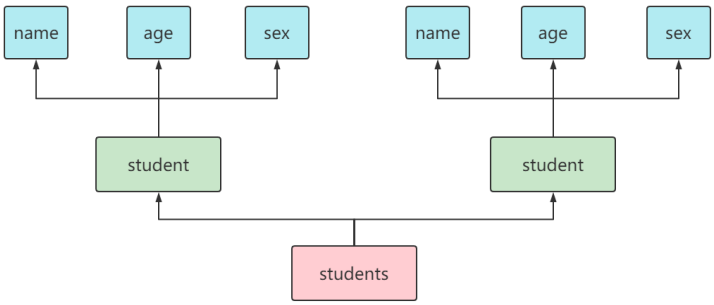
XML解析即读写XML文档中的数据。框架的开发者通过XML解析读取框架使用者配置的参数信息，开发者也可以通过XML解析读取网络传来的数据。XML有如下解析思想：

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

```
<?xml version="1.0"?>
<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">
   <student number="baizhan_0001">
      <name>xiyangyang</name>
      <age>10</age>
      <sex>male</sex>
   </student>
   <student number="baizhan_0002">
      <name>meiyangyang</name>
      <age>8</age>
      <sex>female</sex>
   </student>
</students>
```



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

```
<?xml version="1.0"?>
<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">
   <student number="baizhan_0001">
      <name>xiyangyang</name>
      <age>10</age>
      <sex>male</sex>
   </student>
   <student number="baizhan_0002">
      <name>meiyangyang</name>
      <age>8</age>
      <sex>female</sex>
   </student>
</students>
```

实时效果反馈

1. XML解析思想中，DOM思想的缺点是

A 操作复杂

B 占内存

C 不能增删改

D 以上说法都不正确

2. XML解析思想中，SAX思想的缺点是

A 操作复杂

B 占内存

C 不能增删改

D 以上说法都不正确

答案

1=>B　2=>C

Jsoup解析器_XML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，支持DOM思想。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

实时效果反馈

1. XML的解析器有

A JAXP

B DOM4J

C Jsoup

D 以上都正确

答案

1=>D

Jsoup解析器_Jsoup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

```
public class Demo1 {
    // 获取XML中所有学生的姓名
    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 获取类加载器
        ClassLoader classLoader = Demo1.class.getClassLoader();
        // 2.2使用类加载器，找到XML文档的路径
        String path = classLoader.getResource("com/itbaizhan/xsd/student.xml").getPath();
        // 2.3加载XML文档进内存，并转成Document 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);
        }
    }
}
```

Jsoup解析器_Jsoup

Jsoup：可以解析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

```
public class Demo2 {
    // Jsoup
    public static void main(String[] args) throws IOException {
        // 解析本地XML
        String path = Demo2.class.getClassLoader().getResource("com/itbaizhan/xsd/student.xml").getPath();
        Document document = Jsoup.parse(new File(path), "utf-8");
        System.out.println(document);
        System.out.println("----------------");

        // 解析字符串
        Document document1 = Jsoup.parse("<?xml version=\"1.0\" ?>\n" +
                "<students\n" +
                "
xmlns:xsi=\"http://www.w3.org/2001/XMLSchema-instance\"\n" +
                "
xsi:schemaLocation=\"http://www.itbaizhan.cn/xml student.xsd\"\n" +
                "
xmlns=\"http://www.itbaizhan.cn/xml\">\n" +
                "\n" +
                "    <student number=\"baizhan_0001\">\n" +
                "
<name>baizhan</name>\n" +
                "        <age>10</age>\n" +
```

```
                "        <sex>male</sex>\n" +
                "    </student>\n" +
                "    <student number=\"baizhan_0002\">\n" +
                "        <name>sxt</name>\n" +
                "        <age>11</age>\n" +
                "        <sex>female</sex>\n" +
                "    </student>\n" +
                "</students>");
        System.out.println(document1);
        System.out.println("-----------------------------------");

        // 解析网络资源
        Document document2 = Jsoup.parse(new URL("https://www.baidu.com"), 2000);
        System.out.println(document2);
    }
}
```

实时效果反馈

1. Jsoup解析器中Jsoup可以解析

A 本地文件

B html或xml字符串

C 网页源文件

D 以上都正确

答案

1=>D

Jsoup解析器_Document

Document可以获取元素对象

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。

```
1 public class Demo3 {
2     // Document
```

```
3      public static void main(String[] args) throws IOException {
4          String path = Demo3.class.getClassLoader().getResource("com/itbaizhan/jsoup/student.xml").getPath();
5          Document document = Jsoup.parse(new File(path), "utf-8");
6
7          // 根据id获取元素
8          Element baizhan_0001 = document.getElementById("baizhan_0001");
9          System.out.println(baizhan_0001);
10         System.out.println("----------------------");
11
12         // 根据标签获取元素
13         Elements age = document.getElementsByTag("age");
14         for (Element element : age) {
15             System.out.println(element);
16         }
17         System.out.println("----------------------");
18
19         // 根据属性获取元素
20         Elements english = document.getElementsByAttribute("english");
21         for (Element element : english) {
22             System.out.println(element);
23         }
24         System.out.println("----------------------");
```

```
25
26          // 根据属性名=属性值获取元素
27          Elements elementsByAttributeValue =
document.getElementsByAttributeValue("englis
h", "bz");
28          for (Element element :
elementsByAttributeValue) {
29              System.out.println(element);
30          }
31          System.out.println("----------------
------");
32
33          // 使用CSS选择器获取元素
34          Elements select =
document.select("#baizhan_0001");
35          System.out.println(select);
36          System.out.println("----------------
------");
37          Elements sex =
document.select("sex");
38          System.out.println(sex);
39          System.out.println("----------------
------");
40          Elements select1 =
document.select(".aa");
41          System.out.println(select1);
42      }
43  }
```

实时效果反馈

1. Jsoup解析器中，Document根据选择器选取元素的方法是

A getElementById()

B getElementsByTag()

C getElementsByAttribute()

D select()

2. Jsoup解析器中，Document根据Id选取元素的方法是

A getElementById()

B getElementsByTag()

C getElementsByAttribute()

D select()

答案

1=>D 2=>A

Jsoup解析器_Element

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。

```
1  public class Demo4 {
2      // Document
3      public static void main(String[] args)
   throws IOException {
4          String path =
   Demo4.class.getClassLoader().getResource("co
   m/itbaizhan/jsoup/student.xml").getPath();
5          Document document = Jsoup.parse(new
   File(path), "utf-8");
6
7          // 使用CSS选择器获取元素
8          Elements elements =
   document.select("#baizhan_0001");
9          Element element = elements.get(0);
10         System.out.println(element.text());
```

```
11          System.out.println("----------------
-------");
12          System.out.println(element.html());
13          System.out.println("----------------
-------");
14
     System.out.println(element.attr("id"));
15      }
16  }
```

实时效果反馈

1. Jsoup解析器中，Element获取元素包含的纯文本的方法是

A text()

B html()

C attr()

D 以上方法都不对

答案

1=>A

Jsoup解析器_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。

```
public class Demo5 {
    // Document
    public static void main(String[] args) throws IOException, XpathSyntaxErrorException {
        String path = Demo5.class.getClassLoader().getResource("com/itbaizhan/jsoup/student.xml").getPath();
        // 1. 获取`Document`对象
        Document document = Jsoup.parse(new File(path), "utf-8");
        //2. 将`Document`对象转为`JXDocument`对象
        JXDocument jxDocument = new JXDocument(document);
```

```
        //3. `JXDocument`调用`selN(String xpath)`，获取`List<JXNode>`对象。
//          List<JXNode> jxNodes = jxDocument.selN("//name");
        // 想拿到baizhan_0001的年龄
        List<JXNode> jxNodes = jxDocument.selN("//student[@id='baizhan_0001']/age");
        //4. 遍历`List<JXNode>`，调用`JXNode`的`getElement()`，转为`Element`对象。
        for (JXNode jxNode : jxNodes) {
            Element element = jxNode.getElement();
            //5. 处理`Element`对象。
            System.out.println(element);
        }
    }
}
```

实时效果反馈

1. XPath是

A JAVA路径语言

B HTML路径语言

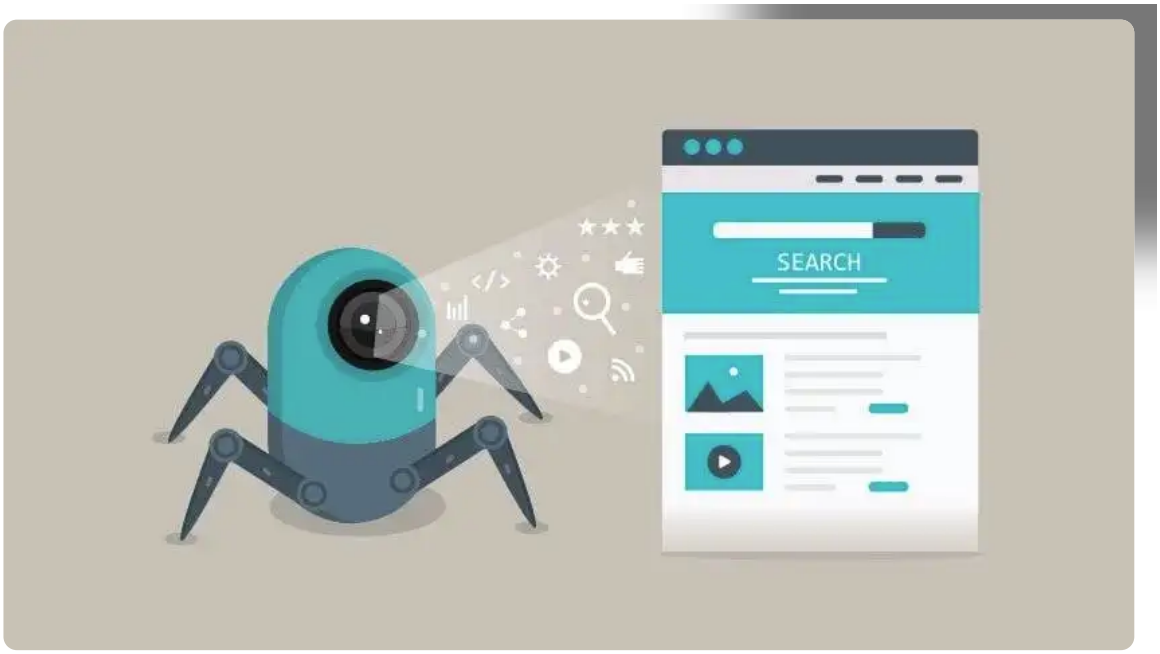
C CSS路径语言

D XML路径语言

答案

1=>D

XML案例_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。

比如我们要开发一个网站，在网页的右上角需要显示今日天气，如果人工查询天气预报维护非常繁琐，我们就可以使用爬虫程序自动爬取天气网站的程序，自动更新。

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。

爬虫步骤：

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

```
public class CrawlerDemo {
    public static void main(String[] args) {
        int min = 9734020;
        int max = 9734346;

         // 循环爬取数据
        for (int i = min; i <= max; i++) {
            try {
                //1. 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);
                //2. 通过Document对象，获取需要的Element对象
                Elements headerImgEle = document.getElementsByAttributeValue("alt", "头图");
                Elements titleEle = document.select(".DailyHeader-title");
                Elements authorEle = document.select(".author");
```

```
                Elements contentELe = document.select(".content");
                //3. 获取Element对象的数据。

                System.out.println(headerImgEle.get(0).attr("src"));

                System.out.println(titleEle.get(0).text());

                System.out.println(authorEle.get(0).text());

                System.out.println(contentELe.get(0).text());
            }catch (Exception e){}
        }
    }
}
```

XML案例_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。

1. 编写爬虫程序的XML配置文件 Crawler.xml

```
<?xml version="1.0"?>
<Crawler>
    <min>9734020</min>
    <max>9734346</max>
</Crawler>
```

编写爬虫程序

```
public class CrawlerDemo {
    public static void main(String[] args) throws IOException {
        int min = 0;
        int max = 0;
        // 解析XML配置文件
        String path = CrawlerDemo.class.getClassLoader().getResource("com/itbaizhan/crawler/Crawler.xml").getPath();
        Document document1 = Jsoup.parse(new File(path), "utf-8");
        Elements minELe = document1.getElementsByTag("min");
        Elements maxELe  = document1.getElementsByTag("max");
        min = Integer.parseInt(minELe.get(0).text());
        max = Integer.parseInt(maxELe.get(0).text());

        // 循环爬取数据
        for (int i = min; i <= max; i++) {
            try {
                //1. 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);
```

```
18                  //
System.out.println(document);
19                  //2. 通过Document对象，获取
需要的Element对象
20                  Elements headerImgEle =
document.getElementsByAttributeValue("alt"
, "头图");
21                  Elements titleEle =
document.select(".DailyHeader-title");
22                  Elements authorEle =
document.select(".author");
23                  Elements contentELe =
document.select(".content");
24                  //3. 获取Element对象的数据。
25
 System.out.println(headerImgEle.get(0).at
tr("src"));
26
 System.out.println(titleEle.get(0).text()
);
27
 System.out.println(authorEle.get(0).text(
));
28
 System.out.println(contentELe.get(0).text
());
29              }catch (Exception e){}
30          }
31      }
32  }
33
```