今日内容

```
1. XML
    1. 概念
    2. 语法
    3. 解析
```

XML：

```
1. 概念：Extensible Markup Language 可扩展标记语言
    * 可扩展：标签都是自定义的。 <user>  <student>

    * 功能
        * 存储数据
            1. 配置文件
            2. 在网络中传输
    * xml与html的区别
        1. xml标签都是自定义的，html标签是预定义。
        2. xml的语法严格，html语法松散
        3. xml是存储数据的，html是展示数据

    * w3c:万维网联盟

2. 语法：
    * 基本语法：
        1. xml文档的后缀名 .xml
        2. xml第一行必须定义为文档声明
        3. xml文档中有且仅有一个根标签
        4. 属性值必须使用引号(单双都可)引起来
        5. 标签必须正确关闭
        6. xml标签名称区分大小写
    * 快速入门：
        <?xml version='1.0' ?>
        <users>
            <user id='1'>
                <name>zhangsan</name>
                <age>23</age>
                <gender>male</gender>
                <br/>
            </user>

            <user id='2'>
                <name>lisi</name>
                <age>24</age>
                <gender>female</gender>
            </user>
        </users>
```

* 组成部分：
    1. 文档声明
        1. 格式：<?xml 属性列表 ?>
        2. 属性列表：
            * version：版本号，必须的属性
            * encoding：编码方式。告知解析引擎当前文档使用的字符集，默认值：ISO-8859-1
            * standalone：是否独立
                * 取值：
                    * yes：不依赖其他文件
                    * no：依赖其他文件
    2. 指令(了解)：结合css的
        * <?xml-stylesheet type="text/css" href="a.css" ?>
    3. 标签：标签名称自定义的
        * 规则：
            * 名称可以包含字母、数字以及其他的字符
            * 名称不能以数字或者标点符号开始
            * 名称不能以字母 xml（或者 XML、Xml 等等）开始
            * 名称不能包含空格

    4. 属性：
        id属性值唯一
    5. 文本：
        * CDATA区：在该区域中的数据会被原样展示
            * 格式： <![CDATA[数据]]>

* 约束：规定xml文档的书写规则
    * 作为框架的使用者(程序员)：
        1. 能够在xml中引入约束文档
        2. 能够简单的读懂约束文档

    * 分类：
        1. DTD:一种简单的约束技术
        2. Schema:一种复杂的约束技术

    * DTD：
        * 引入dtd文档到xml文档中
            * 内部dtd：将约束规则定义在xml文档中
            * 外部dtd：将约束的规则定义在外部的dtd文件中
                * 本地：<!DOCTYPE 根标签名 SYSTEM "dtd文件的位置">
                * 网络：<!DOCTYPE 根标签名 PUBLIC "dtd文件名字" "dtd文件的位置URL">

```
        * Schema:
            * 引入：
                1.填写xml文档的根元素
                2.引入xsi前缀.  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                3.引入xsd文件命名空间.  xsi:schemaLocation="http://www.itcast.cn/xml
student.xsd"
                4.为每一个xsd约束声明一个前缀,作为标识  xmlns="http://www.itcast.cn/xml"

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

```
3. 解析：操作xml文档，将文档中的数据读取到内存中
    * 操作xml文档
        1. 解析(读取)：将文档中的数据读取到内存中
        2. 写入：将内存中的数据保存到xml文档中。持久化的存储

    * 解析xml的方式：
        1. 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
        2. SAX：逐行读取，基于事件驱动的。
            * 优点：不占内存。
            * 缺点：只能读取，不能增删改
```

```
    * xml常见的解析器：
        1. JAXP：sun公司提供的解析器，支持dom和sax两种思想
        2. DOM4J：一款非常优秀的解析器
        3. 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。
        4. PULL：Android操作系统内置的解析器，sax方式的。
```

```
    * 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。
        * 快速入门：
            * 步骤：
                1. 导入jar包
                2. 获取Document对象
                3. 获取对应的标签Element对象
                4. 获取数据

        * 代码：
             //2.1获取student.xml的path
            String path =
JsoupDemo1.class.getClassLoader().getResource("student.xml").getPath();
            //2.2解析xml文档，加载文档进内存，获取dom树--->Document
            Document document = Jsoup.parse(new File(path), "utf-8");
            //3.获取元素对象 Element
            Elements elements = document.getElementsByTag("name");


            System.out.println(elements.size());
```

```
            //3.1获取第一个name的Element对象
            Element element = elements.get(0);
            //3.2获取数据
            String name = element.text();
            System.out.println(name);
```

* 对象的使用：
    1. Jsoup：工具类，可以解析html或xml文档，返回Document
        * parse：解析html或xml文档，返回Document
            * parse•(File in, String charsetName)：解析xml或html文件的。
            * parse•(String html)：解析xml或html字符串
            * parse•(URL url, int timeoutMillis)：通过网络路径获取指定的html或xml的文档对象
    2. Document：文档对象。代表内存中的dom树
        * 获取Element对象
            * getElementById•(String id)：根据id属性值获取唯一的element对象
            * getElementsByTag•(String tagName)：根据标签名称获取元素对象集合
            * getElementsByAttribute•(String key)：根据属性名称获取元素对象集合
            * getElementsByAttributeValue•(String key, String value)：根据对应的属性名和属性值获取元素对象集合
    3. Elements：元素Element对象的集合。可以当做 ArrayList<Element>来使用
    4. Element：元素对象
        1. 获取子元素对象
            * getElementById•(String id)：根据id属性值获取唯一的element对象
            * getElementsByTag•(String tagName)：根据标签名称获取元素对象集合
            * getElementsByAttribute•(String key)：根据属性名称获取元素对象集合
            * getElementsByAttributeValue•(String key, String value)：根据对应的属性名和属性值获取元素对象集合

        2. 获取属性值
            * String attr(String key)：根据属性名称获取属性值
        3. 获取文本内容
            * String text():获取文本内容
            * String html():获取标签体的所有内容(包括字标签的字符串内容)
    5. Node：节点对象
        * 是Document和Element的父类

* 快捷查询方式：
    1. selector:选择器
        * 使用的方法：Elements   select•(String cssQuery)
            * 语法：参考Selector类中定义的语法
    2. XPath：XPath即为XML路径语言，它是一种用来确定XML（标准通用标记语言的子集）文档中某部分位置的语言
        * 使用Jsoup的Xpath需要额外导入jar包。
        * 查询w3cshool参考手册，使用xpath的语法完成查询
        * 代码：

```
            //1.获取student.xml的path
            String path = JsoupDemo6.class.getClassLoader().getResource("student.xml").getPath();
            //2.获取Document对象
            Document document = Jsoup.parse(new File(path), "utf-8");

            //3.根据document对象，创建JXDocument对象
            JXDocument jxDocument = new JXDocument(document);
```

```
//4.结合xpath语法查询
//4.1查询所有student标签
List<JXNode> jxNodes = jxDocument.selN("//student");
for (JXNode jxNode : jxNodes) {
    System.out.println(jxNode);
}

System.out.println("--------------------");

//4.2查询所有student标签下的name标签
List<JXNode> jxNodes2 = jxDocument.selN("//student/name");
for (JXNode jxNode : jxNodes2) {
    System.out.println(jxNode);
}

System.out.println("--------------------");

//4.3查询student标签下带有id属性的name标签
List<JXNode> jxNodes3 = jxDocument.selN("//student/name[@id]");
for (JXNode jxNode : jxNodes3) {
    System.out.println(jxNode);
}
System.out.println("--------------------");
//4.4查询student标签下带有id属性的name标签 并且id属性值为itcast

List<JXNode> jxNodes4 = jxDocument.selN("//student/name[@id='itcast']");
for (JXNode jxNode : jxNodes4) {
    System.out.println(jxNode);
}
```