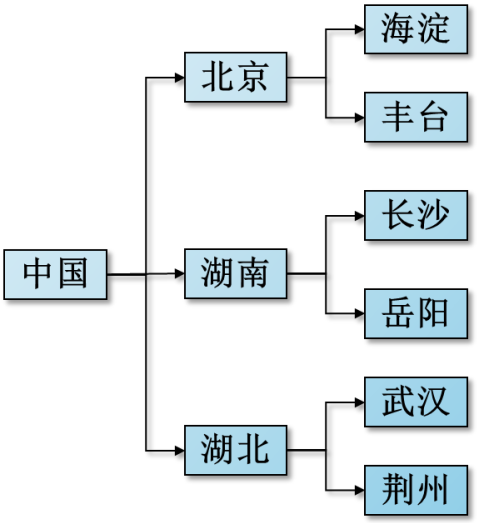
第一部分：XML 初步

1 XML 概述

1.1 什么是 XML

eXtensible Markup Language 可扩展标记语言——由 W3C 组织发布，目前推荐遵守的是 W3C 组织于 2000 年发布的 XML1.0 规范。

XML 的使命，就是以一个统一的格式，组织有关系的数据，为不同平台下的应用程序服务。



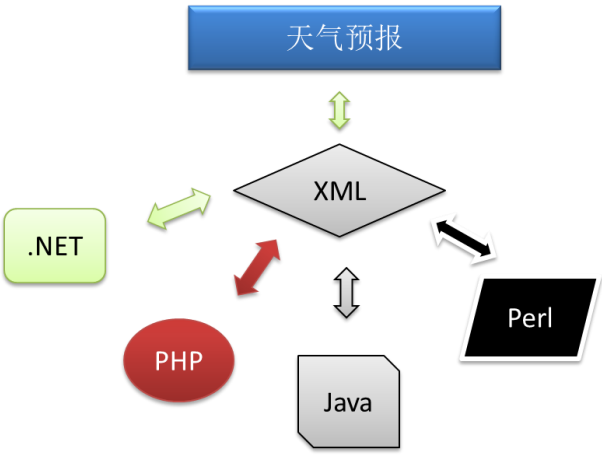

```
<?xml version="1.0" encoding="utf-8"?>
<中国>
    <北京>
        <海淀></海淀>
        <丰台></丰台>
    </北京>
    <湖南>
        <长沙></长沙>
        <岳阳></岳阳>
    </湖南>
    <湖北>
        <武汉></武汉>
        <荆州></荆州>
    </湖北>
</中国>
```

1.2 XML 的主要用途

配置文件
- JavaWeb
- 框架

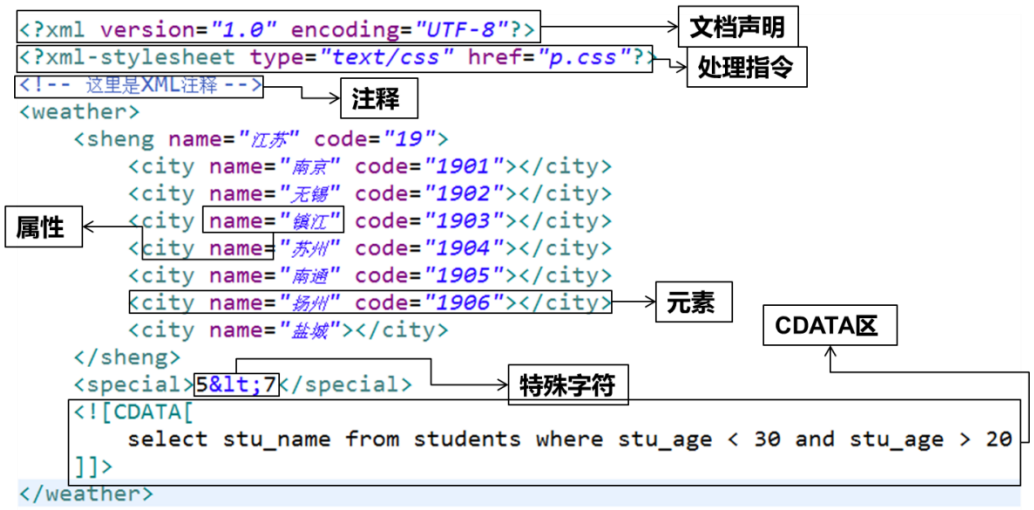
数据交换
- Ajax
- WebService

数据存储
- 保存关系型数据



2 XML 语法规范

2.1 XML 文档结构




```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/css" href="p.css"?>
<!-- 这里是XML注释 -->
<weather>
    <sheng name="江苏" code="19">
        <city name="南京" code="1901"></city>
        <city name="无锡" code="1902"></city>
        <city name="镇江" code="1903"></city>
        <city name="苏州" code="1904"></city>
        <city name="南通" code="1905"></city>
        <city name="扬州" code="1906"></city>
        <city name="盐城"></city>
    </sheng>
    <special>5&lt;7</special>
    <![CDATA[
        select stu_name from students where stu_age < 30 and stu_age > 20
    ]]>
</weather>
```

2.2 XML 文档声明

①在编写 XML 文档时，必须在文件的第一行书写文档声明。
最简单的声明语法：<?xml version="1.0" ?>
②用 encoding 属性说明读取文档所用的解码的字符集：
<?xml version="1.0" encoding="GB2312" ?>
这样就要求保存文件时，必须用 GB2312 编码保存。此时要求 XML 文档的作者确认当前编辑器保存文档的编码方式。
eclipse 会自动按照解码字符集进行编码保存
记事本需要另存为指定的字符集

2.3 语法规则

①第一行为 XML 声明，且必须顶格写
②只能有一个根标签
③标签必须正确结束
④标签不能交叉嵌套
⑤严格区分大小写
⑥属性必须有值，且必须加引号
⑦标签不能以数字开头

2.4 XML 转义字符

特殊字符	替代符号
<	<
>	>
&	&
“	"
‘	'

2.5 CDATA 区

①当 XML 文档中需要写一些程序代码、SQL 语句或其他不希望 XML 解析器进行解析的内容时，就可以写在 CDATA 区中
②XML 解析器会将 CDATA 区中的内容原封不动的输出
③CDATA 区的定义格式：<![CDATA[…]]>
例如：

```
<![CDATA[
    select * from employees
             where salary > 20 and salary < 100
]]>
```

2.6 注释

①Xml 文件中的注释采用：“<!--注释-->” 格式。
注意：
- ●XML 声明之前不能有注释
- ●注释不能嵌套

2.7 处理指令

①处理指令，简称 PI （processing instruction）。处理指令用来指挥解析引擎如何解析 XML 文档内容。

②处理指令必须以“<?”作为开头，以“?>”作为结尾，XML 声明语句就是最常见的一种处理指令。

例如，在 XML 文档中可以使用 xml-stylesheet 指令，通知 XML 解析引擎，应用 css 文件显示 xml 文档内容。
<?xml-stylesheet type="text/css" href="p.css"?>

3 XML 解析技术

3.1 XML 解析方式

dom：(Document Object Model，即文档对象模型) 是 W3C 组织推荐的处理 XML 的一种方式。

它下面有两个分支：jDom 与 dom4j

它们可都可以对 xml 文件进行增删改查的操作

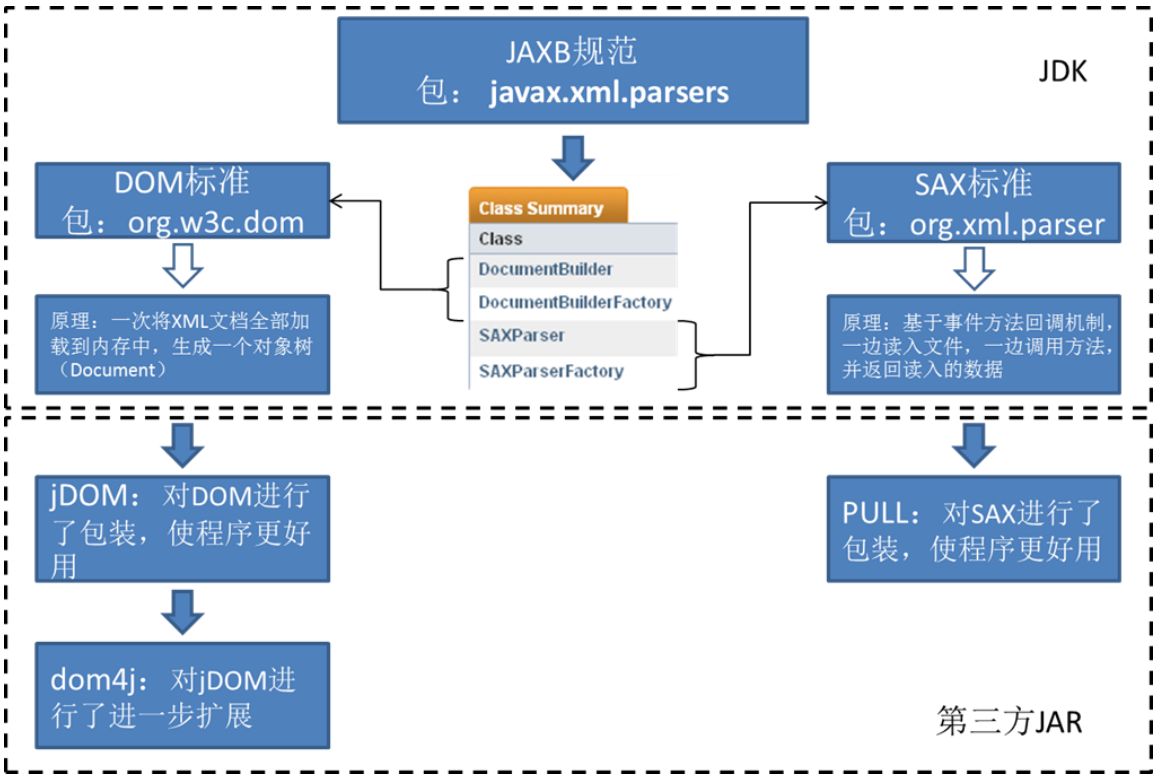
sax： (Simple API for XML) 不是官方标准，但它是 XML 社区事实上的标准，几乎所有的 XML 解析器都支持它。

只能进行解析（查询）

pull: Pull 解析和 Sax 解析很相似，都是轻量级的解析，它是一个第三方开源的 Java 项目，但在 Android 的内核中已经嵌入了 Pull 。

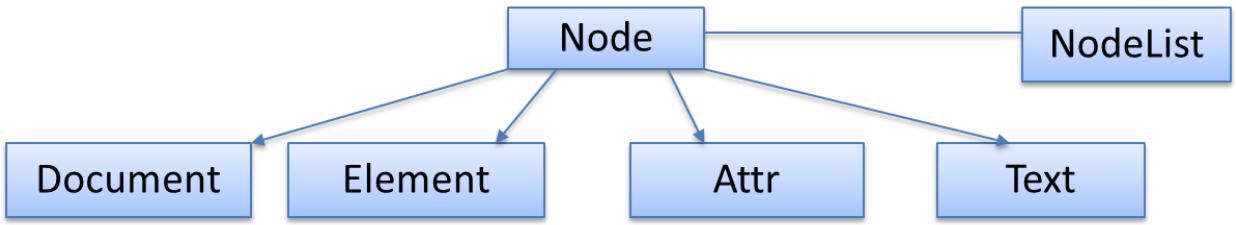
只能进行解析（查询）

3.2 XML 解析技术体系



第二部分：XML DOM 解析

1. DOM 中对象接口关系图



- Node：xml 文件所有对象的根接口　节点
- Document：代表 xml 文件的整个内容的对象的接口
- Element：代表某个元素或标签的对象的接口
- Attr：代表某个元素的某个属性对象的接口
- Text：代表标签体文本标签本对象的接口

- NodeList：代表包含多个 Node 接口对象的集合对象

2.主要方法概览

Node	appendChild(newChild)	将新的节点添加为最后一个子节点
	insertBefore(newChild,refChild)	在某个子节点前插入一个新子节点
	removeChild(oldChild)	删除指定的某个子节点
	replaceChild(newChild,oldChild)	将指定的子节点替换成新的子节点
	setTextContent(textContent)	设置文本内容（一般用在元素对象上）
	getNodeName()	得到节点名称（一般用在元素对象上）
	getParentNode()	得到父节点（一般用在元素对象上）
	getTextContent()	得到文本内容(一般用在元素对象上)
	getFirstChild()	得到第一个子节点
	getLastChild()	得到最后一个子节点
	getNextSibling()	得到下一个兄弟节点
	getPreviousSibling()	得到上一个兄弟节点

Document	createElement(name)	创建一个指定名称的标签对象返回
	getDocumentElement ()//*getRootElement()*	得到文档的根元素对象
	getElementById(id)	根据子元素的 id 属性找到对应的子元素
	getElementsByTagName(name)	根据标签名得到对应的子标签的集合
Element		
	setAttribute(name, value)	设置元素的属性名和属性值
	removeAttribute(attrName)	根据属性名删除对应的属性
	getAttribute(attrName)	根据属性名得到对应的属性
	getElementsByTagName(name)	根据标签名得到对应的子标签的集合
	getTagName()	得到标签名
Attr、Text	极少直接操作这两个接口的对象，一般通过 Element 对象来操作	
NodeList	getLength()得到包含的节点对象的个数　item(index)根据下标得到某个节点	

3.练习：读取 id 为 008 的学生的年龄

```
<?xml version="1.0" encoding="UTF-8"?>
<class>
	<student id="007">
		<name>王五</name>
		<age>18</age>
	</student>
	<student id="008">
		<name>张三</name>
		<age>23</age>
	</student>
</class>
```

4.提示：获取解析器对象

javax.xml.parsers 包中的 DocumentBuilder 类用于加载 xml 文件，并产生一个 Document 对象

```
//创建一个 xml 文档解析器工厂对象
DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();

//通过工厂创建一个 xml 文档解析器对象
DocumentBuilder builder = factory.newDocumentBuilder();

//通过解析器对象解析一个文件对象得到 Document 对象
Document document = builder.parse(new File("d:/users.xml"));
```

第三部分：XML dom4j

1.Dom4j 是一个简单、灵活的开放源代码的库。Dom4j 是由早期开发 JDOM 的人分离出来而后独立开发的。与 JDOM 不同的是，dom4j 使用接口和抽象基类，虽然 Dom4j 的 API 相对要复杂一些，但它提供了比 JDOM 更好的灵活性。

2.Dom4j 是一个非常优秀的 Java XML API，具有性能优异、功能强大和极易使用的特点。现在很多软件采用的 Dom4j，例如 Hibernate。使用 Dom4j 开发，需下载 dom4j 相应的 jar 文件。

3.获取 Document 对象

```
SAXReader reader = new SAXReader();
Document    document = reader.read(new File( "input.xml“ ));
```

4.元素操作

```
• //获取文档的元素.
  Element root = document.getRootElement();
• //获取某个元素的指定名称的第一个子节点
  Element element = element.element(“书名");
```

```
• //获取某个元素的指定名称的所有子元素的集合
  List list = element.elements(“书名”);
• //添加一个指定名称的子元素
  Element childEle = parentEle.addElement(“书名”);
• //删除某个元素指定的子元素
  parentEle.remove(childEle);
```

5.属性操作

```
• //获取某个元素的指定名称的属性对象
  Attribute attr = element.attribute(“id”);
• //获取某个元素的指定名称的属性值
  String id = element.attributeValue(“id”);
• //给元素添加属性或更新其值
  Attribute attr = element.addAttribute(“id”,”123”);
• //删除某个元素的指定属性
  element.remove(attribute);
```

6.文本操作

```
• //获取某个元素的文本内容
  String text = element.getText();
• //给某个元素添加或更新文本内容
  element.setText(“Tom”);
```

7.将文档写入 XML 文件，使更改生效

```
OutputFormat format = OutputFormat.createPrettyPrint();
XMLWriter writer = new XMLWriter( newFileOutput( "output.xml“ ),format);
writer.write(document);
writer.close();
```

8.练习：

①查找某一个节点: 查找 id 为 007 的学员的所有信息

②添加某一个节点: "姓名：王五，id:134,年龄:18

第四部分：XPath

1. XPath 是在 XML 文档中查找信息的语言

XPath 是通过元素和属性进行查找

XPath 简化了 Dom4j 查找节点的过程

使用 XPath 必须导入 jaxen-1.1-beta-6.jar

否则出现

NoClassDefFoundError: org/jaxen/JaxenException

2.XPath 语法示例

/students/student	从根元素开始逐层找，以”/”开头
//name	直接获取所有 name 元素对象, 以“//”开头
//student/*	获取所有 student 元素的所有子元素对象

//student[1]或 //student[last()]	获取所有 student 元素的第一个或最后一个
//student[@id]	获取所有带 id 属性的 student 元素对象
//student[@id=‘002’]	获取 id 等于 002 的 student 元素对象

3.查询节点

- 获取所有符合条件的节点
  - document.selectNodes(String xpathExpression) 返回 List 集合
- 获取符合条件的单个节点
  - document.selectSingleNode(String xpathExpression)
  - 返回一个 Node 对象。如果符合条件的节点有多个，那么返回第一个。

4.练习：

①查找 id 为 007 的学员姓名

②查找所有的学员姓名

第五部分：XML SAX 解析

1. 为什么会出现 SAX 解析？

在使用 DOM 解析 XML 文档时，需要读取整个 XML 文档，在内存中构架代表整个 DOM 树的 Doucment 对象，从而再对 XML 文档进行操作。此种情况下，如果 XML 文档特别大，就会消耗计算机的大量内存，并且容易导致内存溢出。

SAX 解析允许在读取文档的时候，即对文档进行处理，而不必等到整个文档装载完才会文档进行操作。

注意：sax 只能用于读取 xml 文件，无法作更新

2. SAX 采用事件处理的方式解析 XML 文件，利用 SAX 解析 XML 文档，涉及两个部分：解析器和事件处理器(对象)：

解析器可以使用 JAXP 的 API 创建，创建出 SAX 解析器后，就可以指定解析器去解析某个 XML 文档。

解析器采用 SAX 方式在解析某个 XML 文档时，它只要解析到 XML 文档的一个组成部分，都会去调用事件处理器的一个方法，解析器在调用事件处理器的方法时，会把当前解析到的 xml 文件内容作为方法的参数传递给事件处理器。

事件处理器由程序员编写，程序员通过事件处理器中方法的参数，就可以很轻松地得到 sax 解析器解析到的数据，从而可以决定如何对数据进行处理。

3.解析方式

- 使用 SAXParserFactory 创建 SAX 解析工厂
  SAXParserFactory factory = SAXParserFactory.newInstance();
- 通过 SAX 解析工厂得到解析器对象
  SAXParser sp = factory.newSAXParser();
- 通过解析器对象得到一个 XML 的读取器
  XMLReader xmlReader = sp.getXMLReader();
- 设置读取器的事件处理器

```
xmlReader.setContentHandler(new BookParserHandler());
• 解析 xml 文件
xmlReader.parse("book.xml");
```

4.练习：查找 id 为 007 的学生姓名

第六部分：XML Pull 解析

1. 为什么会出现 PULL 解析？

Pull 解析与 sax 解析类似都是基于事件方法回调机制来实现对 xml 文件解析。

Sax 解析不足：即使已经找到所要的数据，xml 数据还是会全部加载进来并产生方法调用。程序员无法停止这些无用的操作，而 pull 解析解决了此问题。

Pull 解析：加载每一部分数据产生方法调用，都必须通过程序员调用一个固定的方法才能进行下去，否则解析工作就停止了。Next()

注意：pull 解析也只能用于读取 xml 文件，无法作更新

2. Pull 解析依赖两个 jar 包：xmlpull_1_0_5.jar 和 kxml2-2.3.0.jar

3.常用接口或类

XmlPullParserFactory

XmlPullParser(既是解析器又是数据的存储器)

XmlPullParserException

4.练习：查找 id 为 007 的学员姓名