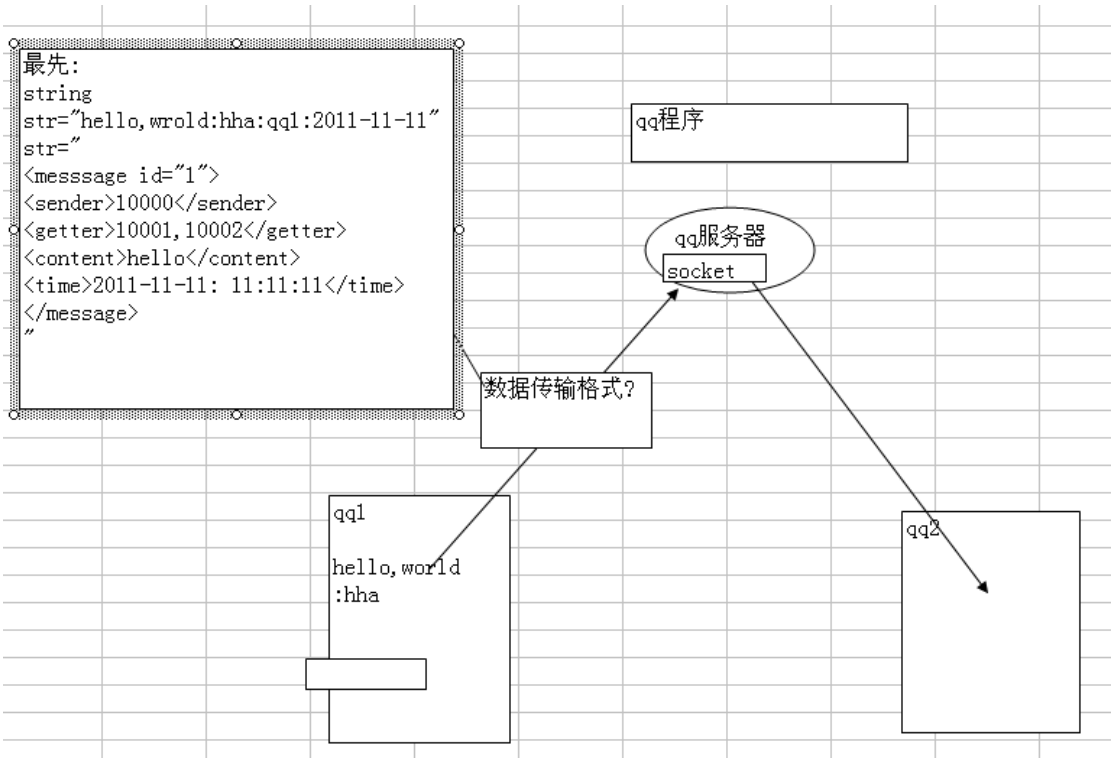
◆ Xml 简单的历史介绍

1969 gml(通用标记语言) [主要的目的是要在不同的机器进行通信的数据规范]
1985 sgml(标准通用标记语言)
1993 html (www 网)
Html 语言本身是有一些缺陷的
(1) 标记不能自定义
<html>
<table>
<hsp></hsp>
</table>
</html>
(2) html 本身缺少一些含义
<h1>水浒英雄</h1>
<table>
<tr><td>宋江</td><td>及时雨</td></tr>
</table>
(3) html 本身没有真正的国际化

html->xhtml->xml

1998 xml
Xml : extensiable markup language 可扩展标记语言

◆ 为什么要学习 xml
(1) 需求



(2) 做配置文件

(3) xml 文件还可以描述很复杂的数据关系

比如 家谱…

- Xml 的常见应用

(1) 数据传送通用格式

(2) 配置文件

(3) 充当小型数据库

- **Xml 语法**

入门案例: 用 xml 来记录一个班级信息

```
<?xml version="1.0" encoding="gb2312"?>
<class>
<stu id="a001">
<name>杨过</name>
<sex>男</sex>
<age>30</age>
</stu>
<stu id="a002">
<name>李莫愁</name>
<sex>女</sex>
<age>20</age>
</stu>
</class>
```

☞ 编码问题:

ansi 编码 是 american national standard insititu 美国国家标准协会 ,
ansi 编码在不同的国家不一样的 ansi ->gb2312 anis-gbk big5
日本 ansi->日文操作系统默认的编码.

◆ xml 的语法
(1) 文档声明
<?xml version=”1.0” encoding=”编码方式” standalone=”yes|no”?>
(2) 一个 xml 文档中，有且只有一个根元素

元素==标签==节点
(3) 在 xml 中
<name>xiaoming</name>
不等价与==
<name>
xiaoming
</name>
(4) 属性值用双引号（"）或单引号（'）分隔（如果属性值中有'，用"分隔；有"，用'分隔）
特别说明: 如果属性值有单引号，有双引号，则需要使用实体: html-> ©

&lt;	<
&gt;	>
&amp;	&
&quot;	"
&apos;	'

<stu id="a"0'0'1">
<name>杨过</name>
<sex>男</sex>
<age>30</age>
</stu>

(4) CDATA 节
有时我们希望传递一些特殊字符， <>@!#$%^&*(可以使用 CDATA 节包括
基本用法:
<intro><![CDATA[这个是好$$128qw8o8<Lk;>;akdf0sa98u329408><<K>>>学生]]></intro>

面试题:
问; 如何适用 xml 去传递小图片
答： 可以把文件读取成一个 byte[] ,然后放到 CDATA 节，再传递.

(5) 处理指令

看一个案例:

```
<?xml version="1.0" encoding="utf-8"?>
<?xml-stylesheet href="my.css" type="text/css"?>
<class>
<!--学生信息-->
<stu id="a&amp;quot;0&amp;apos;0&amp;apos;1&amp;lt;" >
<name>杨过</name>
<sex>男</sex>
<age>30</age>
</stu>
<stu id="a002">
<name>李莫愁</name>
<sex>女</sex>
<age>20</age>
</stu>
</class>
```

my.css

```
name{
  font-size:100px;
  font-weight:bold;
  color:red;
}
sex{
  font-size:50px;
  font-weight:bold;
  color:blue;
}
age{
        font-size:20px;
  font-weight:bold;
  color:green;
}
```

◆ **xml 语法小结:**

XML 声明语句

```
<?xml version="1.0" encoding="gb2312"?>
```

- 必须有且仅有一个根元素
- 标记大小写敏感
- 属性值用引号
- 标记成对
- 空标记关闭

- 元素正确嵌套
- 名称中可以包含字母、数字或者其它字符
- 名称中不能含空格 测
- 名称中不能含冒号(注：冒号留给命名空间使用) 测

◆ **dtd**

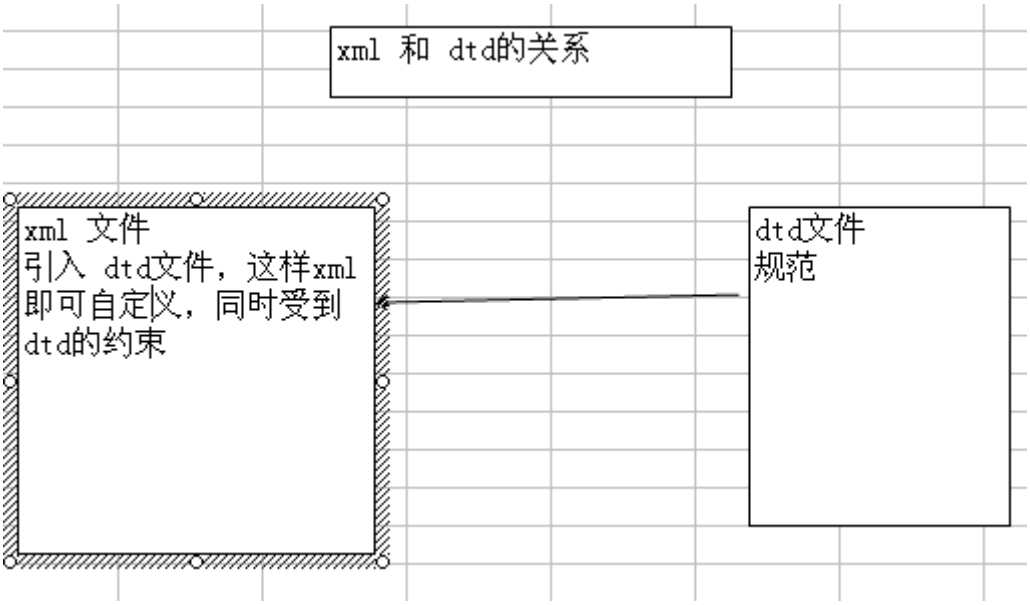
基本概念: dtd (document type definition 文档类型定义),该文件一般和 xml 文件配合使用,主要的用处是约束 xml, 除了 dtd 技术外， 还有一个 schema 的技术也可以用于约束 xml 文件的书写规范.

现在请看一个问题:

<stu id="a"0'0'1<" >
<name>杨过</name>
<sex>男</sex>
<age>30</age>
<介绍>我是好人</介绍>
<面积>100 平</面积>
</stu>

怎么解决 xml 过于自由的问题:->dtd

xml 和 dtd 关系



快速入门案例:

基本语法是:

<!ELEMENT 元素名 类型>

xml:

```
<?xml version="1.0" encoding="utf-8"?>
<!--引入 dtd 去约束该 xml 文件-->
<!DOCTYPE 班级 SYSTEM "myClass2.dtd">
<班级>
    <学生>
        <名字>周星驰</名字>
        <年龄>23</年龄>
        <介绍>学习刻苦</介绍>
    </学生>
    <学生>
        <名字>林青霞</名字>
         <年龄>32</年龄>
        <介绍>是一个好学生</介绍>
    </学生>
</班级>
```

myClass2.dtd

```
<!ELEMENT 班级 (学生+)>
<!ELEMENT 学生 (名字,年龄,介绍)>
<!ELEMENT 名字 (#PCDATA)>
<!ELEMENT 年龄 (#PCDATA)>
<!ELEMENT 介绍 (#PCDATA)>
```

完成校验的 html

```
<html>
<head>
<!--自己编写一个简单的解析工具，去解析 xml dtd 是否配套-->
<script language="javascript">
<!--
    var xmldoc = new ActiveXObject("Microsoft.XMLDOM");
    xmldoc.validateOnParse = "true";//开启校验
    xmldoc.load("myClass2.xml");//指定校验哪个 xml 文件
    document.writeln("错误信息是:"+xmldoc.parseError.reason+"<br/>");
    document.writeln("错误的行是:"+xmldoc.parseError.line);

//-->
</script>
</head>
<body>
</body>
</html>
```

- dtd 的细节

(1) dtd 的分类

内部 dtd

外部 dtd

内部 DTD 文档

<!DOCTYPE *根元素* [*定义内容*]>

外部 DTD 文档

<!DOCTYPE *根元素* SYSTEM "*DTD 文件路径*">

(2) 在 xml 中引入 dtd 有两种方法

1. 引入本地 dtd

<!DOCTYPE 根元素 SYSTEM ‘地址’>

2. 引入公共的 dtd

<!DOCTYPE 根元素 PUBLIC ‘地址’>

(2)

<!ELEMENT 元素名 类型>

类型:

EMPTY, ANY , #PCDATA

(3) dtd 的修饰符

修饰符

符号	用途	示例	示例说明
( )	用来给元素分组	（古龙|金庸|梁羽生），（王朔|余杰），毛毛	分成三组
|	在列出的对象中选择一个	（男人|女人）	表示男人或者女人必须出现，两者至少选一
+	该对象最少出现一次，可以出现多次(1或多次)	（成员+）	表示成员必须出现，而且可以出现多个成员
*	该对象允许出现零次到任意多次（0到多次）	（爱好*）	爱好可以出现零次到多次
?	该对象可以出现，但只能出现一次(0到1次)	（菜鸟?）	菜鸟可以出现，也可以不出现，如果出现的话，最多只能出现一次
,	对象必须按指定的顺序出现	（西瓜,苹果,香蕉）	表示西瓜、苹果、香蕉必须出现，并且按这个顺序出现

(4) 属性的细节

基本语法

<!ATTLIST 元素名

属性名 类型 特点

.....
>

- 类型有 五种:

CDATA 表示可以放入文本
ID 表示属性的值，不能重复,同时不要用数字开头.
IDREF/IDREFS 当一个元素的属性值，需要去引用另外一个 ID ,则使用 IDREF,如果希望引用多个，则使用 IDREFS,请用空格隔开.
Enumerated 表示属性的值，只能是例举出了 比如

```
<!ATTLIST 学生
    地址 CDATA #FIXED "北京"
    学号 ID #REQUIRED
    大哥 IDREFS #REQUIRED
    性别 (男|女) #REQUIRED
>
```

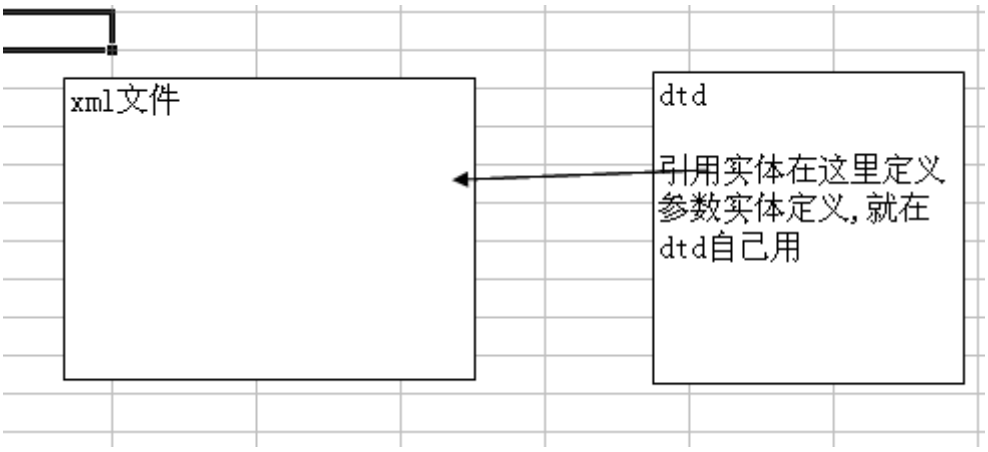
ENTITY

- 属性的特点有四种

#REQUIRED 表示必须有
#IMPLIED 表示可以有
#FIXED “值” 表示如果有，则必须是什么
Default “值” 表示如果不指定，则默认.

◆ 实体(ENTITY)

就是实体用于为一段内容创建一个别名，以后在 XML 文档中就可以使用别名引用这段内容了
java :
String str=”你好”;
定义 str,在别的地方，我们使用 str 就可以访问到 ‘你好’



(1) 分类
引用实体
案例
在 dtd 中定义:
<!ENTITY mycopy "我的公司版权">
说明：最好把定义放在 dtd 的最后
在 xml 中使用
&mycopy;

参数实体

基本语法
<!ENTITY % 实体名字 ”实体内容”>
引用
%实体名字;
举例:

```
<!ELEMENT 班级 (学生*)>
<!ENTITY % myname "名字">
<!ELEMENT 学生 (%myname;,介绍,年龄)>
<!ATTLIST 学生
    地址 CDATA #FIXED "北京"
    学号 ID #REQUIRED
    大哥 IDREFS #REQUIRED
    性别 (男|女) #REQUIRED
>
<!ELEMENT %myname; (#PCDATA)>
<!ELEMENT 年龄 (#PCDATA)>
<!ELEMENT 介绍 (#PCDATA)>
<!ENTITY mycopy "我的公司版权">
```

学习 dtd 的目标：一般公司很少让程序员自己写 dtd,要求程序员看的懂 dtd,同时可以根据给出的 dtd,写出对应的 xml

一个产品目录

```
<!ENTITY AUTHOR "John Doe">
<!ENTITY COMPANY "JD Power Tools, Inc.">
<!ENTITY EMAIL "jd@jd-tools.com">

<!ELEMENT CATALOG (PRODUCT+)>

<!ELEMENT PRODUCT
(SPECIFICATIONS+,OPTIONS?,PRICE+,NOTES?)>
```

```
<!ATTLIST PRODUCT
NAME CDATA #IMPLIED
CATEGORY (HandTool|Table|Shop-Professional) "HandTool"
PARTNUM CDATA #IMPLIED
PLANT (Pittsburgh|Milwaukee|Chicago) "Chicago"
INVENTORY (InStock|Backordered|Discontinued) "InStock">

<!ELEMENT SPECIFICATIONS (#PCDATA)>
<!ATTLIST SPECIFICATIONS
WEIGHT CDATA #IMPLIED
POWER CDATA #IMPLIED>

<!ELEMENT OPTIONS (#PCDATA)>
<!ATTLIST OPTIONS
FINISH (Metal|Polished|Matte) "Matte"
ADAPTER (Included|Optional|NotApplicable) "Included"
CASE (HardShell|Soft|NotApplicable) "HardShell">

<!ELEMENT PRICE (#PCDATA)>
<!ATTLIST PRICE
MSRP CDATA #IMPLIED
WHOLESALE CDATA #IMPLIED
STREET CDATA #IMPLIED
SHIPPING CDATA #IMPLIED>

<!ELEMENT NOTES (#PCDATA)>
```

xml...

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE CATALOG SYSTEM 'product.dtd'>
<CATALOG>
<PRODUCT NAME="康师傅矿泉水" CATEGORY="HandTool" PARTNUM="abc"
PLANT="Milwaukee"      INVENTORY="Backordered">
<SPECIFICATIONS WEIGHT="800" POWER="600" >这里是细节</SPECIFICATIONS>
<PRICE>110</PRICE>
</PRODUCT>
</CATALOG>
```

◆ xml 编程

为什么要学习 xml 编程(就是对 xml 文件进程 crud 操作)

1.xml 作为数据传递，需要解析

2.xml 作为配置文件，需要读取.

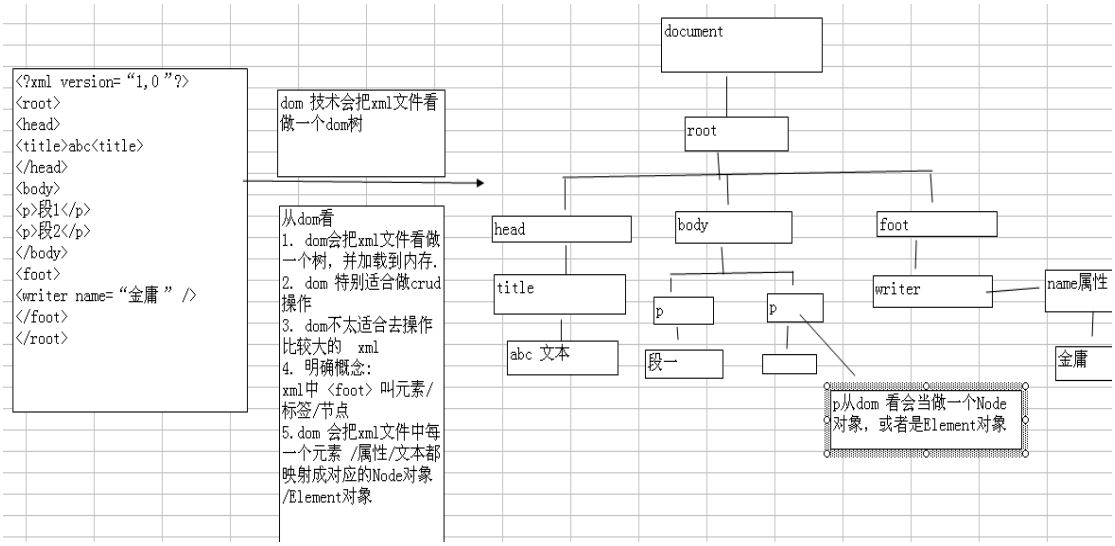
3.xml 作为小型数据库.crud

在 j2ee 技术中，主要是学习 java 对 xml 操作 ，和 js 对 xml 操作

目前有两种模式
dom 是 w3c 推出的标准
sax 是社区的标准

我们在授课中，主要讲三套 api
dom sax dom4j

java 解析 xml 【dom 技术】 看原理:



我们讲一个快速入门案例:

```
<?xml version="1.0" ?>
<班级>
    <学生>
        <名字>周星驰</名字>
        <年龄>23</年龄>
        <介绍>学习刻苦</介绍>
    </学生>
    <学生>
        <名字>林青霞</名字>
         <年龄>32</年龄>
        <介绍>是一个好学生</介绍>
    </学生>
</班级>
```

代码:（使用 dom 去遍历 xml 文件和指定获取某个节点）

```
    //具体的查询某个学生的信息(显示第一个学生的所有信息)
    //请考虑如何获取某个元素的属性值,(取出)
    public static void read(Document doc){
```

```
        NodeList nl=doc.getElementsByTagName("学生");
        //取出第一个学生
        Element stu=(Element) nl.item(0);
        System.out.println("学生的性别是"+stu.getAttribute("sex"));
        Element name=(Element) stu.getElementsByTagName("年龄
").item(0);
        System.out.println(name.getTextContent());
        //System.out.println("发现"+nl.getLength());

    }

    //遍历该xml文件
    public static void list(Node node){
        if(node.getNodeType()==node.ELEMENT_NODE){
            System.out.println("名字"+node.getNodeName());
        }
        //取出node的子节点
        NodeList nodeList=node.getChildNodes();
        for(int i=0;i<nodeList.getLength();i++){
            //再去显示
            Node n=nodeList.item(i);
            list(n);
        }

    }
```

下面的是使用 dom 取添加一个新的元素：

```
//添加一个学生到xml文件中
    public static void add(Document doc) throws Exception{

        //创建一个新的学生节点
        Element newStu=doc.createElement("学生");
        //添加一个属性值
        newStu.setAttribute("sex", "男");
        Element newStu_name=doc.createElement("名字");
        newStu_name.setTextContent("小明2");
        Element newStu_age=doc.createElement("年龄");
        newStu_age.setTextContent("34");
        Element newStu_intro=doc.createElement("介绍");
        newStu_intro.setTextContent("这是一个好孩子");
        newStu.appendChild(newStu_name);
        newStu.appendChild(newStu_age);
        newStu.appendChild(newStu_intro);
```

```
        //把新的学生节点添加到根元素
        doc.getDocumentElement().appendChild(newStu);

        //得到TransformerFactory
        TransformerFactory tff=TransformerFactory.newInstance();
        //通过TransformerFactory 得到一个转换器
        Transformer tf=tff.newTransformer();
        tf.transform(new DOMSource(doc), new
StreamResult("src/classes.xml"));


    }
```

◆ 删除某个元素或者是某个属性

```
//删除一个元素(删除小明2学生)
    public static void del(Document doc) throws Exception{

        //首先要找到这个学生
        //Node node= doc.getElementsByTagName("学生").item(0);
        //node.getParentNode().removeChild(node);
        //删除学生的sex属性
        Element node= (Element) doc.getElementsByTagName("学生
").item(0);
        node.removeAttribute("sex");

        //更新xml
        //得到TransformerFactory
        TransformerFactory tff=TransformerFactory.newInstance();
        //通过TransformerFactory 得到一个转换器
        Transformer tf=tff.newTransformer();
        tf.transform(new DOMSource(doc), new
StreamResult("src/classes.xml"));
    }
//更新操作
//更新元素(把第一个学生的名改成 宋江)
    public static void upd(Document doc) throws Exception{

            //找到
            Element node=(Element) doc.getElementsByTagName("学生
").item(0);
            Element node_name=(Element) node.getElementsByTagName("
名字").item(0);
            node_name.setTextContent("宋江");
```

```
            //node_name.setAttribute("sex", arg1)
            //更新xml
            //得到TransformerFactory
            TransformerFactory tff=TransformerFactory.newInstance();
            //通过TransformerFactory 得到一个转换器
            Transformer tf=tff.newTransformer();
            tf.transform(new DOMSource(doc), new
StreamResult("src/classes.xml"));
    }
```

- **sax 技术**

单说面试题: 说请下下面的代码会出现什么问题?

byte bytes[]=new byte[1024*1024*1000];

bytes[0]=0;

System.out.println(bytes[0]);

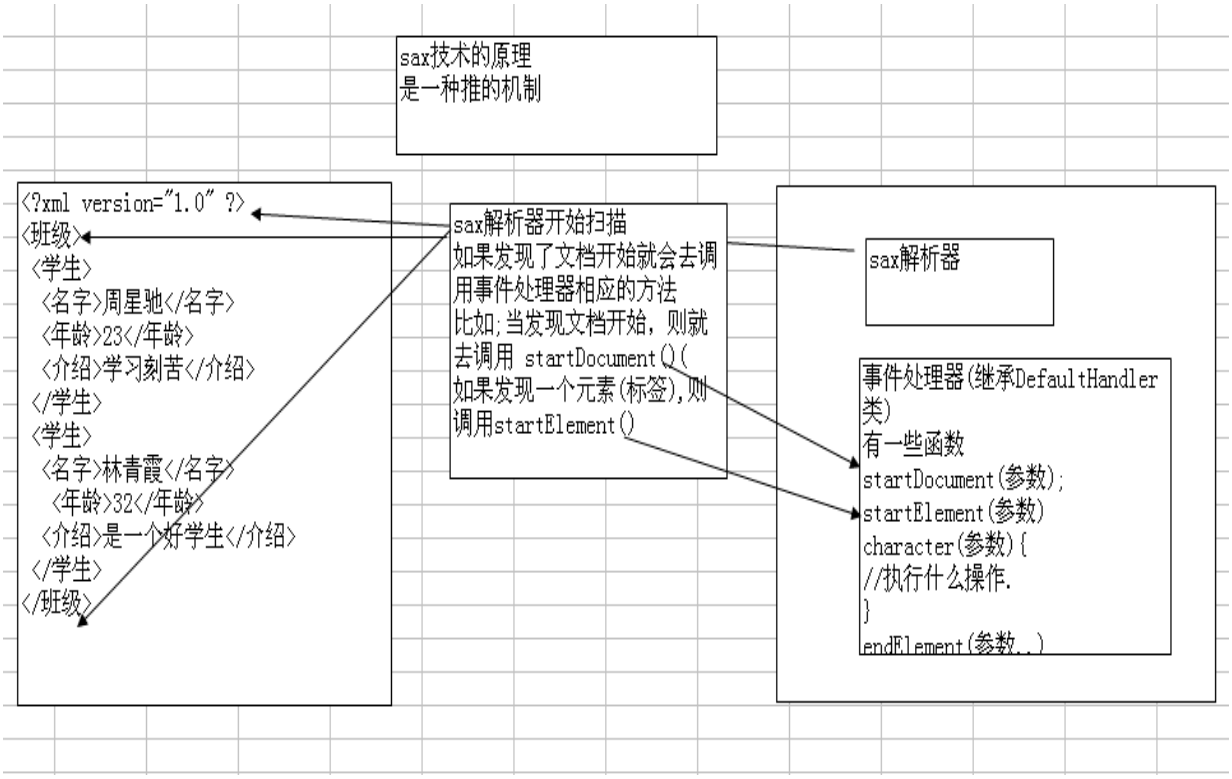
实际考察你会不会 指定 jvm 启动的 内存大小:

答：jvm 机启动时有一个默认大小 jdk5.0 64m, 如果我们希望改变 jvm 机启动的内存大小可以通过修改 –Xmx?m 来处理 ?可以自己指定

1.为什么会出现 sax 技术

因为 dom 技术，会把整个 xml 文件加载到内存中，这样如果 xml 过大，则可能会出现内存溢出.

3. sax 技术可以在不加载全部 xml 文件时，就可以解析 xml 文档,看一个原理图:



sax 技术的案例:

```
package com.sax.test;
import javax.xml.parsers.*;
import org.xml.sax.Attributes;
import org.xml.sax.SAXException;
import org.xml.sax.helpers.DefaultHandler;
public class Sax1 {
    //使用sax技术去解析xml文件.myclasses2.xml
    public static void main(String[] args) throws Exception,
SAXException {
        // TODO Auto-generated method stub
        //1.创建SaxParserFactory
        SAXParserFactory spf=SAXParserFactory.newInstance();
        //2.创建SaxParser 解析器
        SAXParser saxParser=spf.newSAXParser();
        //3 把xml文件和事件处理对象关联
        saxParser.parse("src/myclasses2.xml",new
MyDefaultHandler2() );
    }
}
//请思考，如何只显示学生的名字和年龄
class MyDefaultHandler2 extends DefaultHandler{
    private boolean isName=false;
    private boolean isAge=false;
    @Override
    public void characters(char[] ch, int start, int length)
            throws SAXException {
        // TODO Auto-generated method stub
        String con=new String(ch,start,length);
        if(!con.trim().equals("")&&(isName||isAge)){
            System.out.println(con);
        }
        isName=false;
        isAge=false;
        //super.characters(ch, start, length);
    }
    @Override
    public void endDocument() throws SAXException {
        // TODO Auto-generated method stub
        super.endDocument();
    }
    @Override
    public void endElement(String uri, String localName, String name)
```

```
            throws SAXException {
        // TODO Auto-generated method stub
        super.endElement(uri, localName, name);
    }
    @Override
    public void startDocument() throws SAXException {
        // TODO Auto-generated method stub
        super.startDocument();
    }
    @Override
    public void startElement(String uri, String localName, String
name,
            Attributes attributes) throws SAXException {
        // TODO Auto-generated method stub
        if(name.equals("名字")){
            this.isName=true;
        }else if(name.equals("年龄")){
            this.isAge=true;
        }
    }
}
//定义事件处理类
class MyDefaultHandler1 extends DefaultHandler{
    //发现文档开始
    @Override
    public void startDocument() throws SAXException {
        // TODO Auto-generated method stub
        System.out.println("startDocument()");
        super.startDocument();
    }
    //发现xml文件中的一个元素
    @Override
    public void startElement(String uri, String localName, String
name,
            Attributes attributes) throws SAXException {
        // TODO Auto-generated method stub
        System.out.println("元素名称="+name);
    }
    //发现xml文件中的文本
    @Override
    public void characters(char[] ch, int start, int length)
            throws SAXException {
        String con=new String(ch,start,length);
        //显示文本内容:
```

```
            if(!con.trim().equals("")){
                System.out.println(new String(ch,start,length));
            }
        }
        //发现xml文件中一个元素介绍</xx>
        @Override
        public void endElement(String uri, String localName, String name)
                throws SAXException {
            // TODO Auto-generated method stub
            super.endElement(uri, localName, name);
        }
        //发现文档结束
        @Override
        public void endDocument() throws SAXException {
            // TODO Auto-generated method stub
            System.out.println("endDocument()");
            super.endDocument();
        }
}
```

对 sax 说明:
1. sax 主要用于对 xml 文件解析(读取),不能去修改，删除，添加元素
2. sax 是推机制，把发现的内容告诉程序员(函数),程序员可以自己决定如何处理

- **dom4j（jdom）**

1. 为什么有 dom4j

dom 缺点 ： 比较耗费内存
sax 缺点: 只能对 xml 进行读取，但是不能去 修改，添加，删除.
dom4j :既可以提高效率，同时也可以进行 crud
特别说明: 因为 dom4j 不是 sun 公司的产品，所以我们开发 dom4j 需要引入 jar 包.

(1) 快速入门 如何适用 dom4j 技术对 xml 文件进程(crud)操作

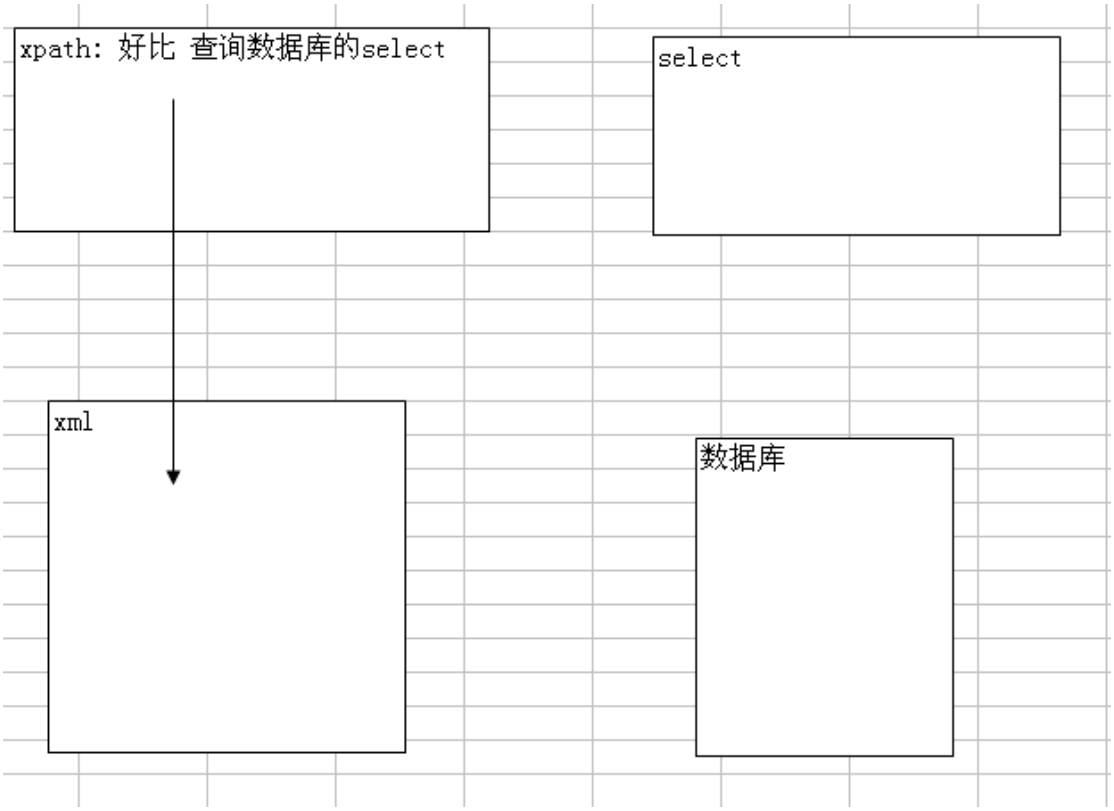
- xpath 的必要性

为了我们根据方便的访问的某个节点，我们可以使用 xpath 技术,当使用 xpath 后，就可以非常方便的读取到指定节点,xpath 往往是结合 dom4j 一并使用.

说明：如果要使用 xpath 则需要引入一个新的包:


jaxen-1.1-beta-6.jar

原理图:



特别说明:

//@id
选择所有的 id 属性
<AAA> <BBB id = "b1"/> <BBB id = "b2"/> <BBB name = "bbb"/> <BBB/> </AAA>

如果我们通过 //@id 取回的节点类型是 Attribue,不是 Element
用法:

```
//3.可以使用xpath随心读取
        List e=document.selectNodes("//@id");//返回多个元素
        System.out.println(((Attribute)e.get(1)).getText());
```

注意:xpath 是可以任意组合:
比如:

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<AAA>
        <BBB id = "b1">
        <CCC>
        <KKK>k2</KKK>
        </CCC>
        <CCC>
        <KKK>k1</KKK>
        </CCC>
        </BBB>
        <BBB id = "b2"/>
        <BBB name = "bbb"/>
        <BBB/>
    </AAA>
```

要找到 **<KKK>k2</KKK>**

xpath 应该这样写 ： /AAA/BBB[1]/CCC[1]/KKK

案例：

```
package com.dom4jxpath.test;

import java.util.List;

import org.dom4j.*;
import org.dom4j.io.*;

public class Test1 {

    //dom4j 配合 xpath案例
    public static void main(String[] args) throws Exception {
        // TODO Auto-generated method stub
        //1.得到SAXReader 解析器
        SAXReader saxReader =new SAXReader();
        //2.指定去解析哪个文件
        Document document =
saxReader.read("src/com/dom4jxpath/test/test.xml");

        //3.可以使用xpath随心读取
        List e=document.selectNodes("/AAA/BBB[1]/CCC[1]/KKK");//返回
多个元素 document.selectSingleNode
        System.out.println(((Element)e.get(0)).getText());
        //System.out.println(((Attribute)e.get(1)).getText());
```

```
        //如果我们确定只有一个Node,元素则可以使用selectSingleNode
        //Element e2=(Element)
document.selectSingleNode("/AAA/BBB[last()]");
        //System.out.println(e2.getText());
    }

}
```

作用：用dom4j+xpath 完成学生课程维护系统.