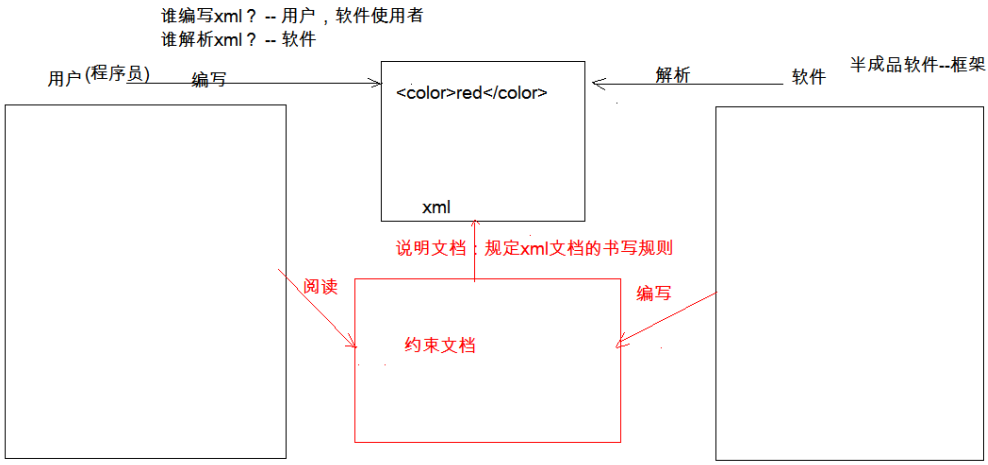
今日内容

1. XML
    1. 概念
    2. 语法
    3. 解析



注：这个软件使用者就是指程序员；这个说明文档就是约束文档；“编写”的意思在这里指“阅读”

XML：

1. 概念：Extensible Markup Language 可扩展标记语言
    * 可扩展：即标签构成的语言，标签全都是自定义，例如 <user>， <student>
    * 功能
        * 存储数据

1. 配置文件（properties用于存就简单的配置文件，但是xml用于存复杂的配置文件）

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

4. HTML是超文本标记语言；XML是可扩展标记语言

* w3c:xml与html俩是亲兄弟，xml是html的弟弟，都有一个共同的爹叫w3c，即万维网联盟（World Wide Web Consortium）

2. 语法：

* 基本语法：

1. xml文档的后缀名 .xml（xml的简单创建：右键创建本文文档，把“新建文本文档.txt”命名为“1.xml”）

2. xml第一行必须定义为文档声明(即<?xml version='1.0' ?>，它上面不能有空行，否则会报错)

3. xml文档中有且仅有一个根标签（即<users>）

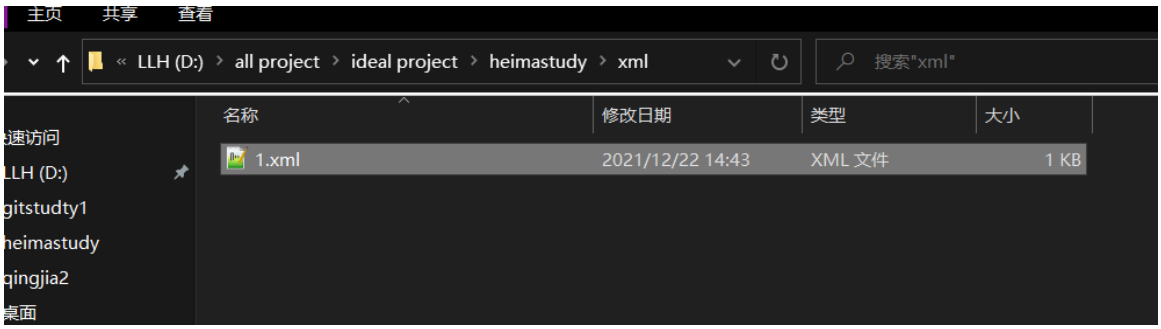
4. 属性值必须使用引号引起来，单双都可（图片中的id的属性值）

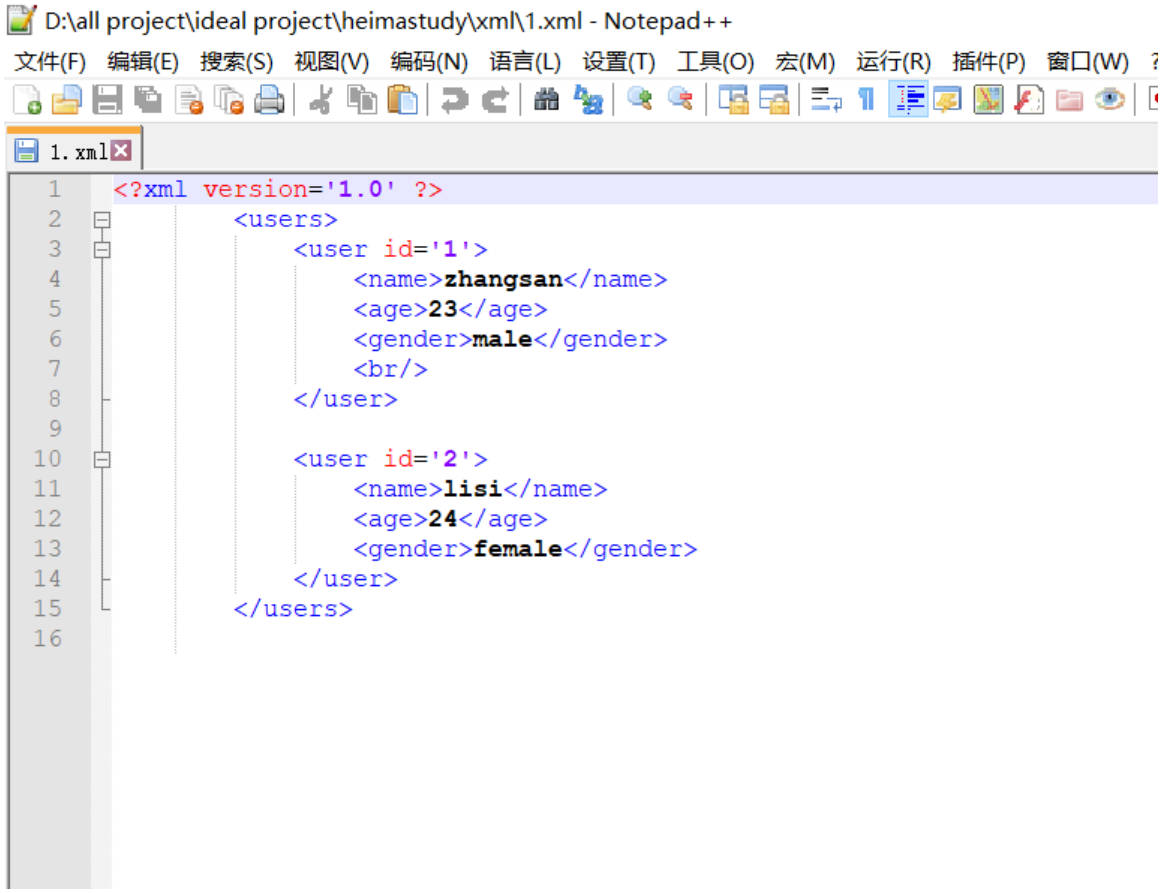
5. 标签必须正确关闭

6. xml标签名称区分大小写

7.属性是不一定非得写成id，即id不是关键字，属性值都是可以自定义的

- xml文件可以被任何的浏览器进行解析；
  现在手动写了一个1.xml文件，如果用浏览器打开后没有报错显示如下，则书写成功；
  且还可以点击“-”来进行收缩；
  再次修改<?xml version='1.0' ?>，使它上面有一空行；
  可见发生报错

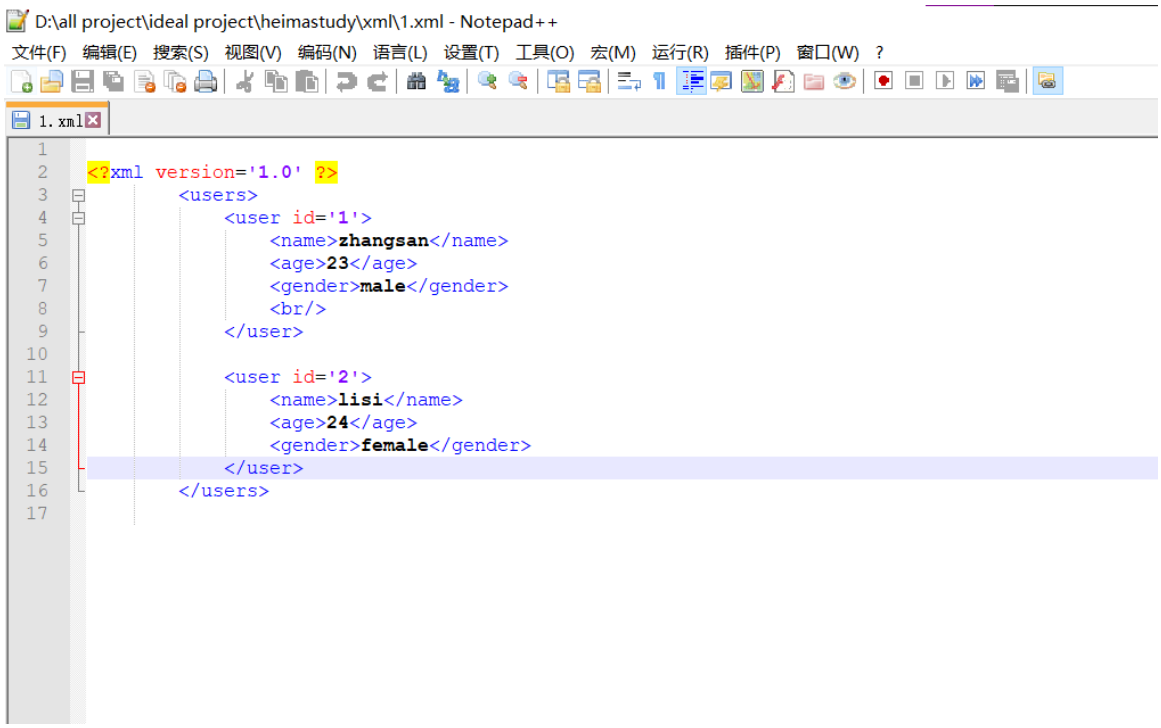


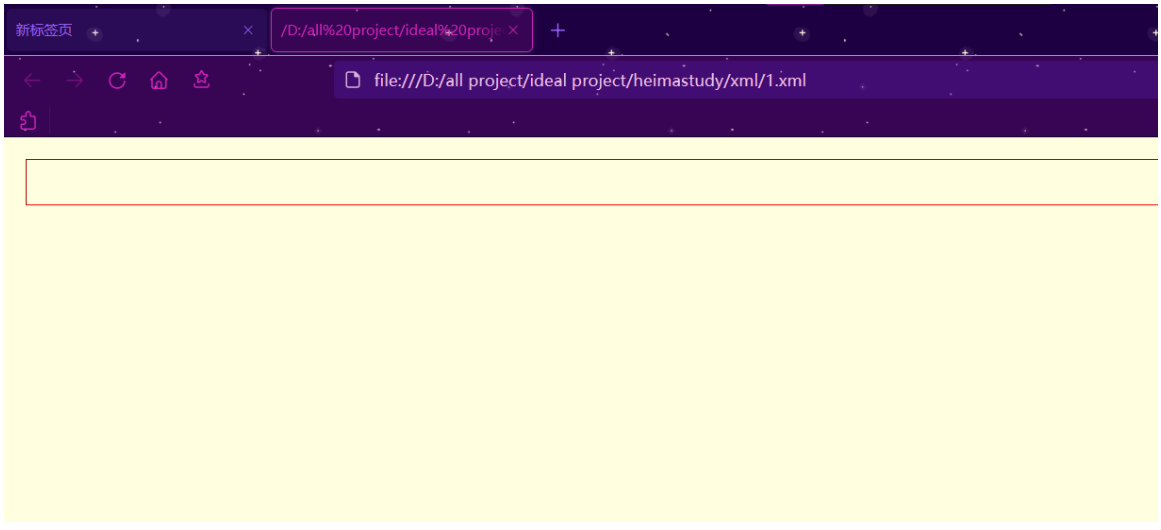


该 XML 文件并未包含任何关联的样式信息。文档树显示如下。

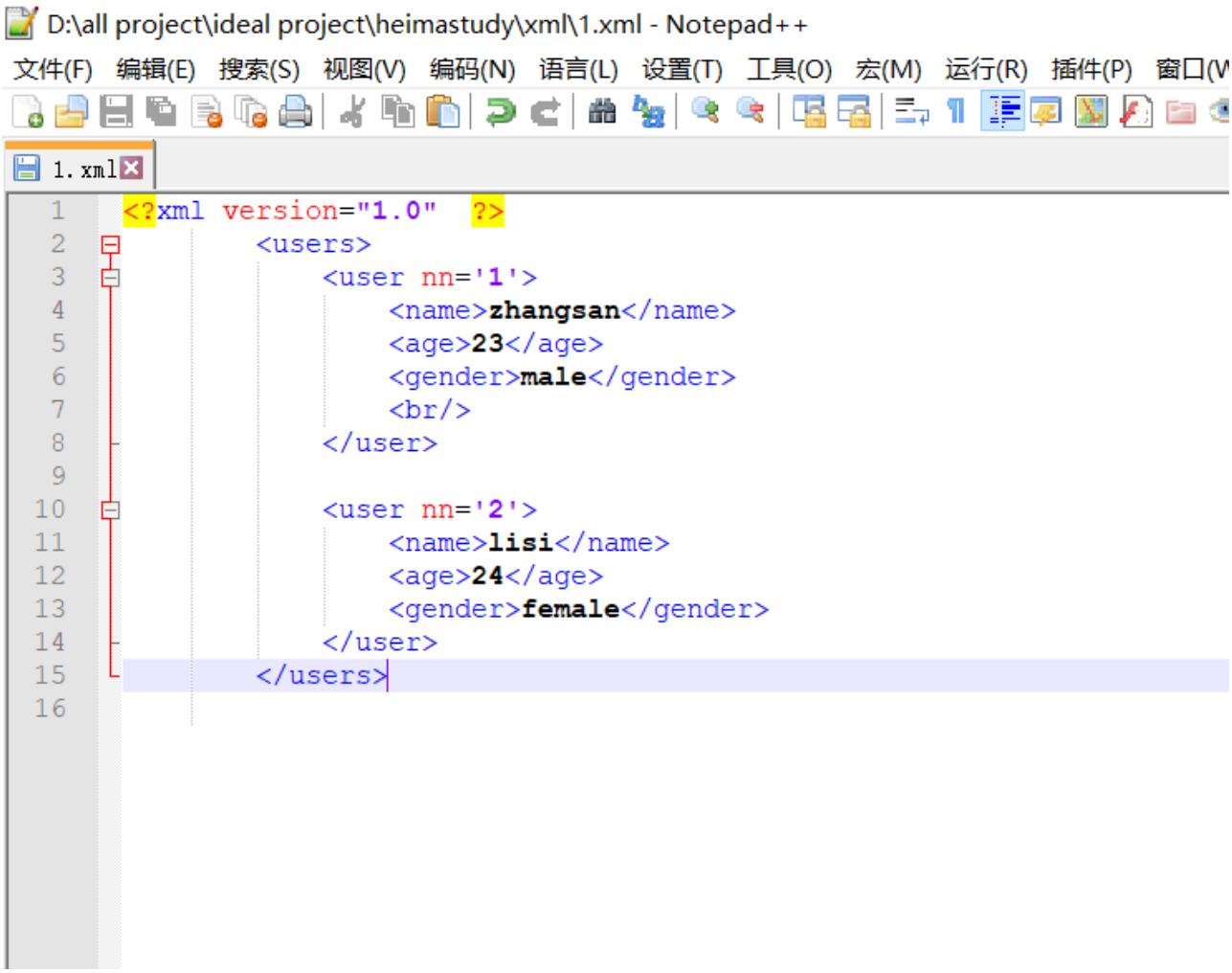

```
-<users>
  -<user id="1">
     <name>zhangsan</name>
     <age>23</age>
     <gender>male</gender>
     <br/>
   </user>
  -<user id="2">
     <name>lisi</name>
     <age>24</age>
     <gender>female</gender>
   </user>
 </users>
```


```
−<users>
  +<user id="1"></user>
  +<user id="2"></user>
 </users>
```

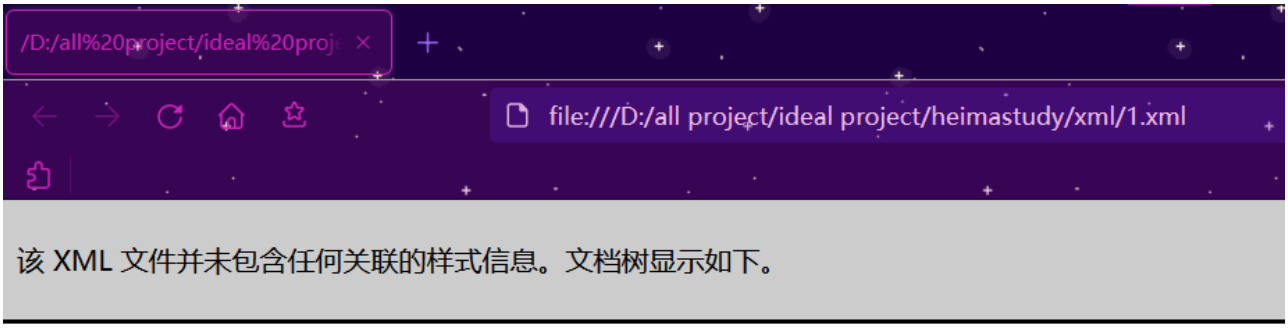





- 修改1.xml为如下；


D:\all project\ideal project\heimastudy\xml\1.xml - Notepad++
文件(F) 编辑(E) 搜索(S) 视图(V) 编码(N) 语言(L) 设置(T) 工具(O) 宏(M) 运行(R) 插件(P) 窗口(W
1.xml
<?xml version="1.0" ?>
<users>
<user nn='1'>
<name>zhangsan</name>
<age>23</age>
<gender>male</gender>

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




```
-<users>
  -<user nn="1">
     <name>zhangsan</name>
     <age>23</age>
     <gender>male</gender>
     <br/>
   </user>
  -<user nn="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 ?>　　（问号与xml之间是不能有空格的）
        2. 属性列表：
            * version：版本号，这个属性必须得写（现在的主流版本是1.0）（<?xml version='1.0' ?>）
            * encoding：编码方式,告知解析器以哪种模式来解码；这个属性不写的话默认值：ISO-8859-1
                （ <?xml version="1.0" encoding="gbk" ?> ）
                （注：虽然utf-8可以存中文，但是用utf-8不可以解析GBK，除非用utf来进行写xml）
                （IDEA很高级：比如如果xml里写了encoding="gbk"后，idea会自动根据encoding设置当　　前代码编码方式为utf-8，不再需要我们手动区设置）
            * standalone：是否独立(现在它已经被淘汰了，大多数不用设置)
                （ <?xml version="1.0" encoding="gbk" standalone='yes' ?> ）
                * 取值：
                    * yes：不依赖其他约束文件
                    * no：依赖其他约束文件
    2. 指令：结合css的

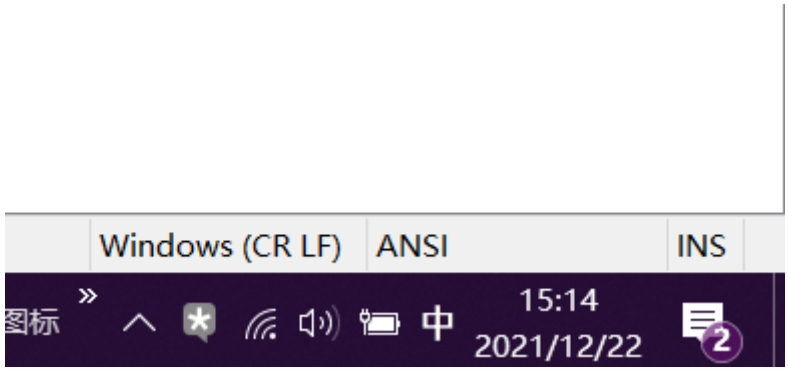
```
<?xml-stylesheet type="text/css" href="a.css" ?>
```

* 写在根标签外，与文档声明代码同等地位

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：
   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]]>

```
<![CDATA[
    if(a < b && a > c) {}
]]]>
```

- 修改1.xml为如下；
  浏览器打开它后发现乱码而且报错（原因是我们是用GBK进行书写的xml，但是浏览器却用ISO-8859-1进行解析;ANSI是本地编码的意思，中文问都是系统的本地编码都是GBK）；
  修改为： <?xml version="1.0" encoding="gbk"?> ；
  打开浏览器后发现成功



D:\all project\ideal project\heimastudy\xml\1.xml - Notepad++

文件(F) 编辑(E) 搜索(S) 视图(V) 编码(N) 语言(L) 设置(T) 工具(O) 宏(M) 运行(R) 插件(P) 窗口(W

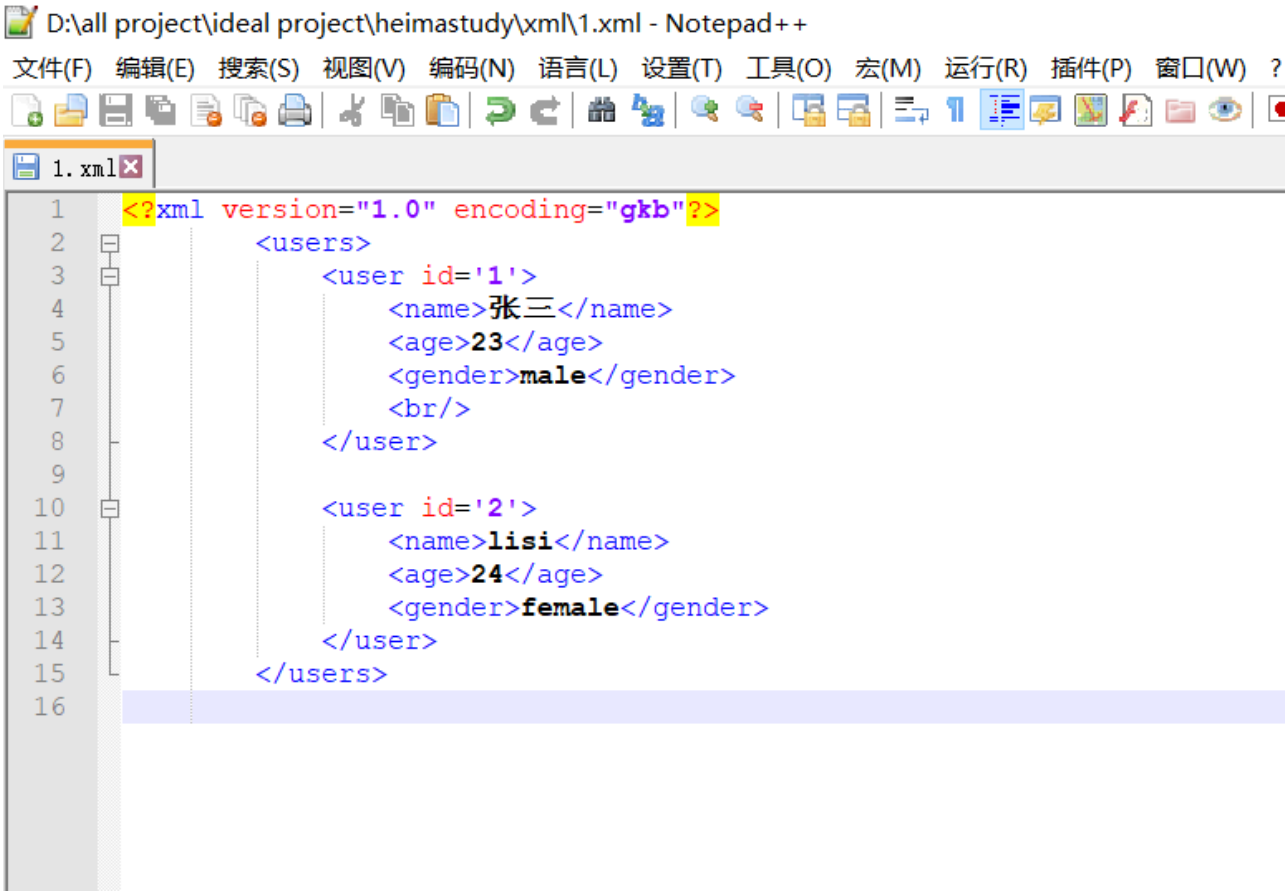
1.xml

```xml
<?xml version='1.0' ?>
    <users>
        <user id='1'>
            <name>张三</name>
            <age>23</age>
            <gender>male</gender>
            <br/>
        </user>

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

XML 解析错误：格式不佳
位置：file:///D:/all%20project/ideal%20project/heimastudy/xml/1.xml
行 4，列 11：

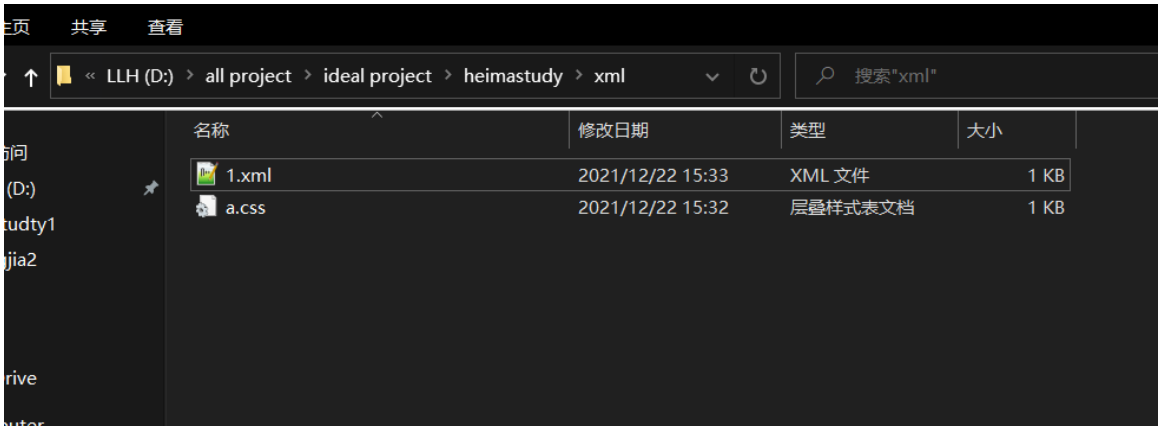
<name>

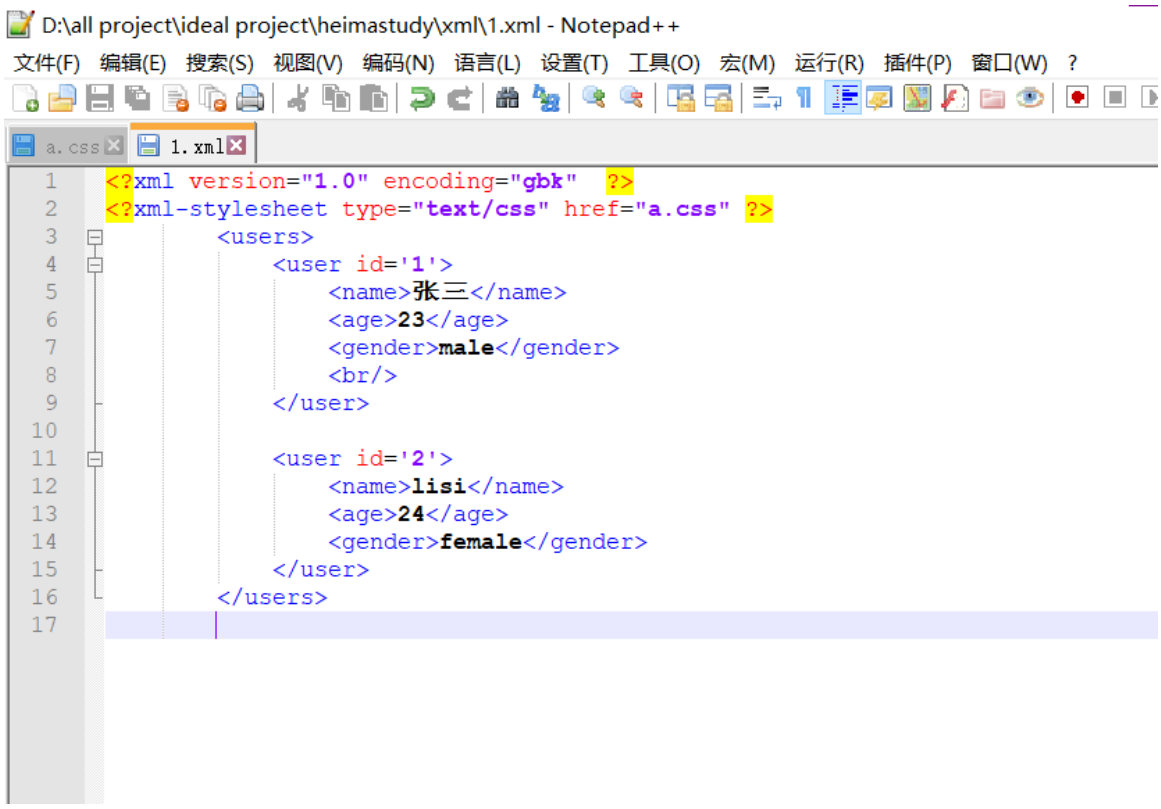


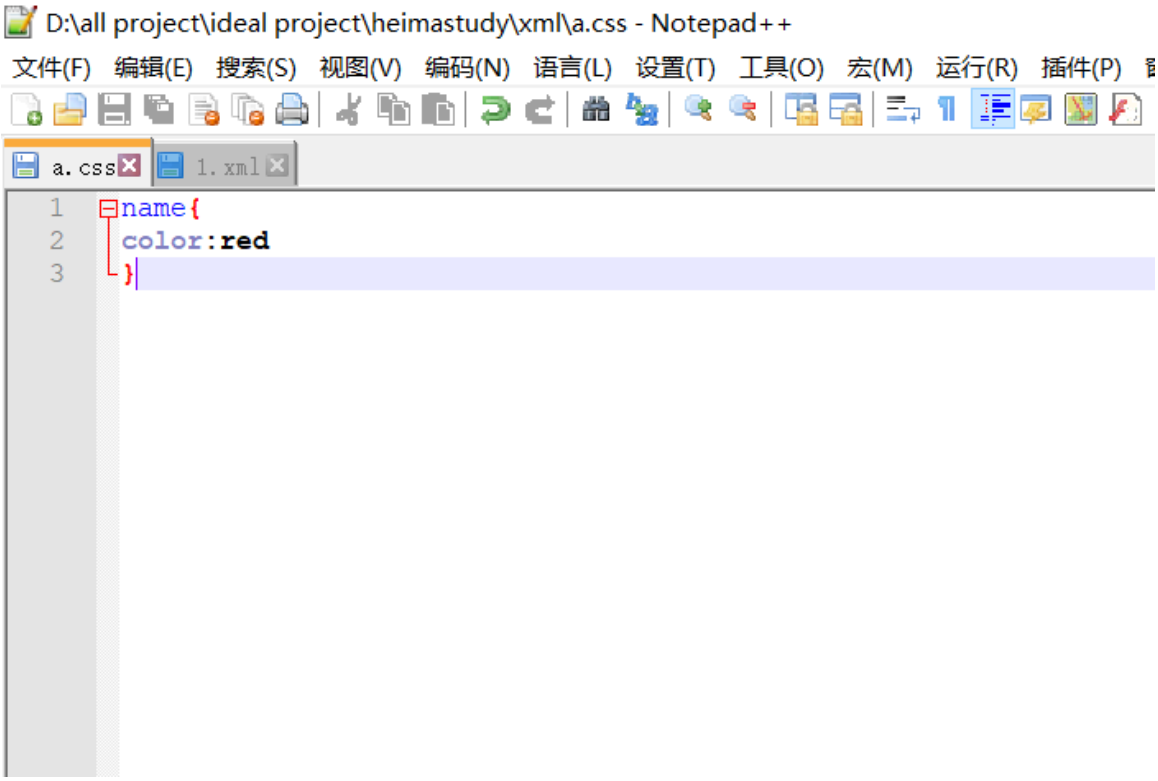

```
-<users>
  -<user id="1">
      <name>张三</name>
      <age>23</age>
      <gender>male</gender>
      <br/>
    </user>
  -<user id="2">
      <name>lisi</name>
      <age>24</age>
      <gender>female</gender>
    </user>
  </users>
```

- 指令的应用：

  修改1.xml为如下；
  新建a.css文件；
  浏览器打开；

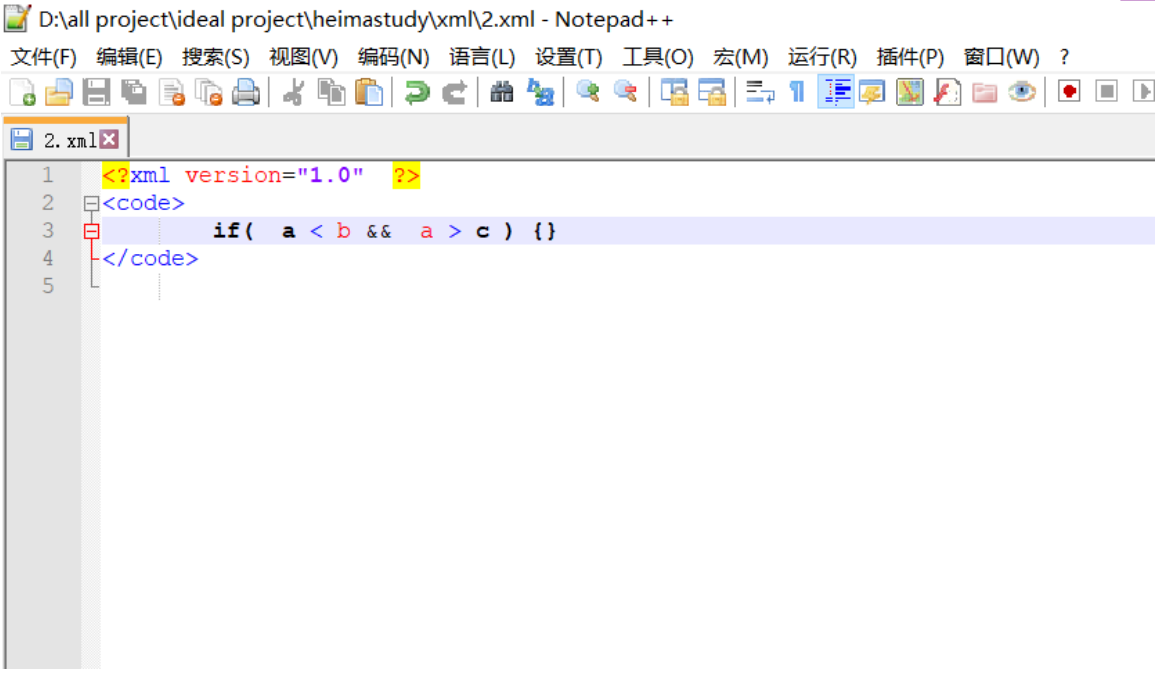


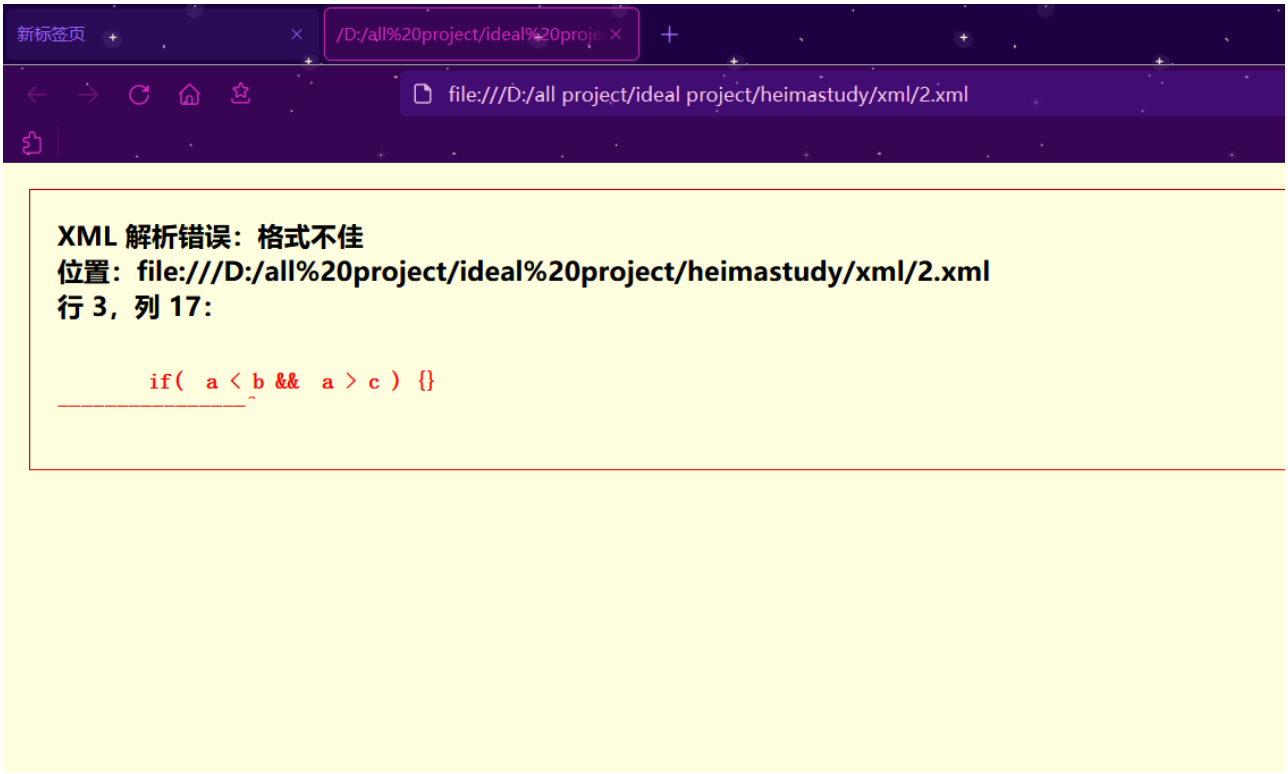


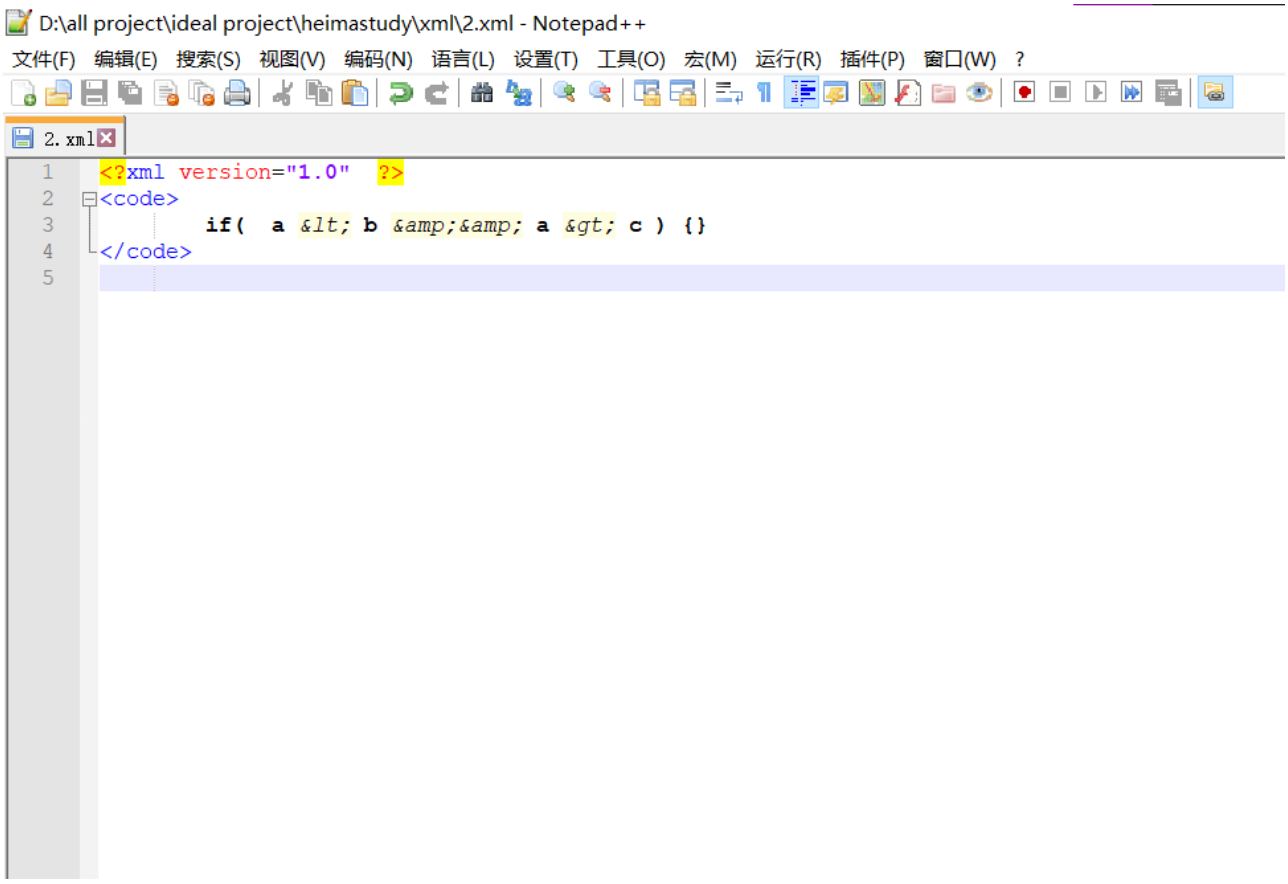


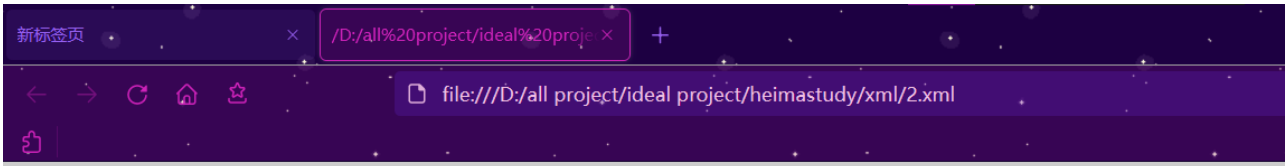
- CDATA区：

  新建2.xml，并写入以下内容；
  打开浏览器解析错误（原因是：<、&&、>是特殊字符，xml中也需要像HTML中的转义字符转义,&&是且的意思）；
  再修改为如下（ < 对应 < ）（&&amp 对应 && ）（ >对应 > ）；
  打开浏览器后解析正确；
  新建3.xml，并写入以下内容；
  打开浏览器后解析正确；



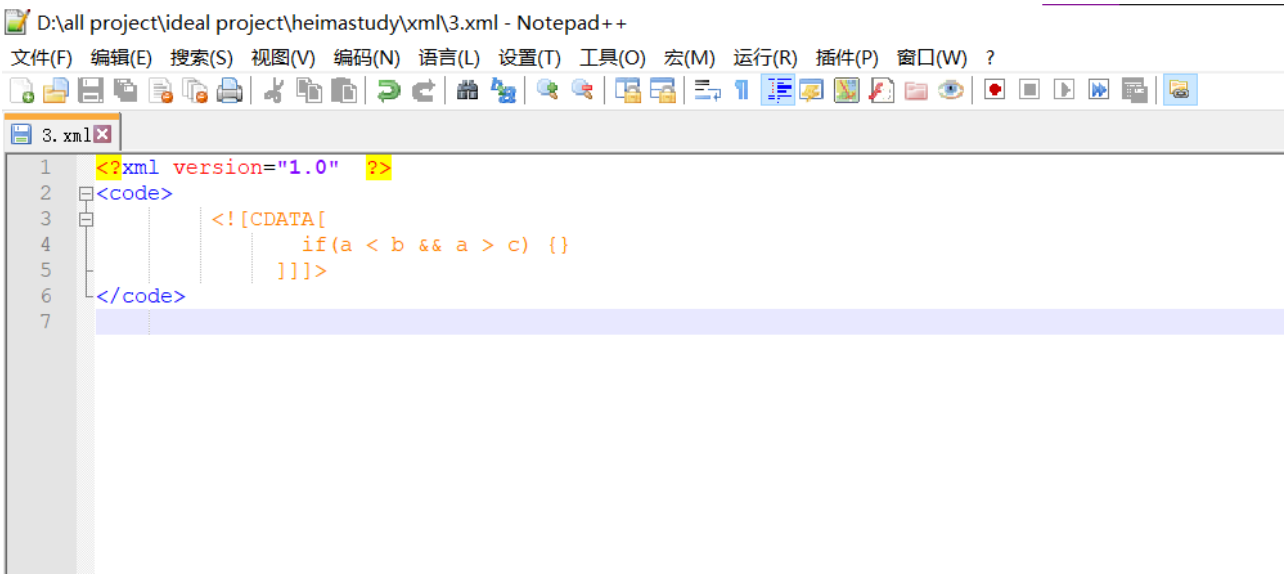

新标签页
/D:/all%20project/ideal%20proje
file:///D:/all project/ideal project/heimastudy/xml/2.xml
XML 解析错误：格式不佳
位置：file:///D:/all%20project/ideal%20project/heimastudy/xml/2.xml
行 3，列 17：
if(a < b && a > c) {}

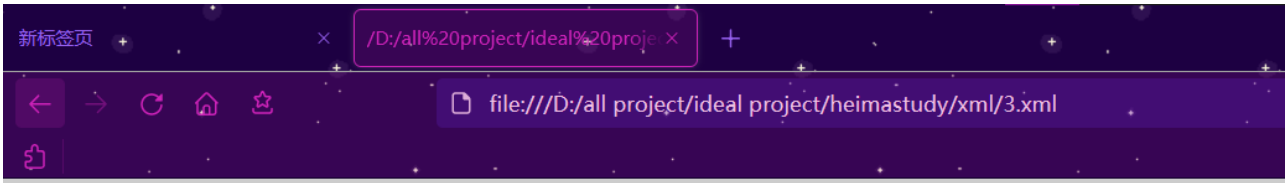

D:\all project\ideal project\heimastudy\xml\2.xml - Notepad++
文件(F) 编辑(E) 搜索(S) 视图(V) 编码(N) 语言(L) 设置(T) 工具(O) 宏(M) 运行(R) 插件(P) 窗口(W) ?
2.xml
<?xml version="1.0" ?>
<code>
if(a < b && a > c) {}
</code>



该 XML 文件并未包含任何关联的样式信息。文档树显示如下。

<code> if(a < b && a > c) {} </code>





该 XML 文件并未包含任何关联的样式信息。文档树显示如下。

-<code>
if(a < b && a > c) {}]
</code>

3. 约束：规定xml文档的书写规则（根据dtd文件来写xml文件,在idea里不按照dtd里的约束来编辑xml的话是会报错的）

* 对作为框架的使用者(程序员)夫的要求：

1. 能够在xml中引入约束文档

2. 能够简单的读懂约束文档，不需要会编写

*约束文件的分类：

1. DTD文件:一种简单的约束（文件后缀为dtd，编写简单）（有很大缺陷，不常用）

2. Schemaw文件:一种复杂的约束（文件后缀为xsd,编写复杂）（Schema其实就是一个xml文档）

*约束的附加好出：在写xml文件时IDEA会根据约束来进行快速提示代码

* 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（有很大缺陷，不常用）

* 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">（再xml文件里写的）

<!DOCTYPE students SYSTEM "student.dtd">（指同级目录)

* 网络：<!DOCTYPE 根标签名 PUBLIC "dtd文件名字" "dtd文件的位置URL">

注：这个"dtd文件名字"是自己随便起的，自己随便写的student.dtd例如：

<!ELEMENT students (student+) > 表示students根标签里的student标签可以出现1次或多次，必须得出现；

<!ELEMENT students (student*) > 表示students根标签里的student标签可以出现0次或多次）；

<!ELEMENT name (#PCDATA)> 要求name的标签体必须是字符串

<!ELEMENT student (name,age,sex)> 表示student标签里的namae、age、sex标签必须得按照此顺序来排列，且三个必须得有,不能没有，而且这三个都只能出现一次；namae、age、sex标签加上id属性后可以大于1个)； <!ATTLIST student number ID #REQUIRED> 声明student标签有属性，属性的名字是number，而且number这个属性值必须得写(注:这个ID是一个关键字，不是指属性值的命名）

student.xml例如：

```
<!DOCTYPE students SYSTEM "student.dtd">
```

* Schema:

* xml引入（能看动就行，不需要会写，能做简单的修改即可）：

1.填写xml文档的根元素（即根标签，元素就是标签的意思）

2.引入xsi前缀

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

（ideal里自动提示有很多可以选的，不止这一个，我们这一次就选择这个，并且以后也尽量都用这 个，这是固定的）

3.为每一个xsd约束声明一个前缀,作为标识

xmlns="http://www.itcast.cn/xml"

4.引入xsd文件命名空间

xsi:schemaLocation="http://www.itcast.cn/xml student.xsd"

（通过xsi来引对应文件的地址,schema文档的路径是: student.xsd)

（为student.xsd对应的schema自定义起一个名字叫做 http://www.itcast.cn/xml ，这是 该schema的“文件命名空间”）

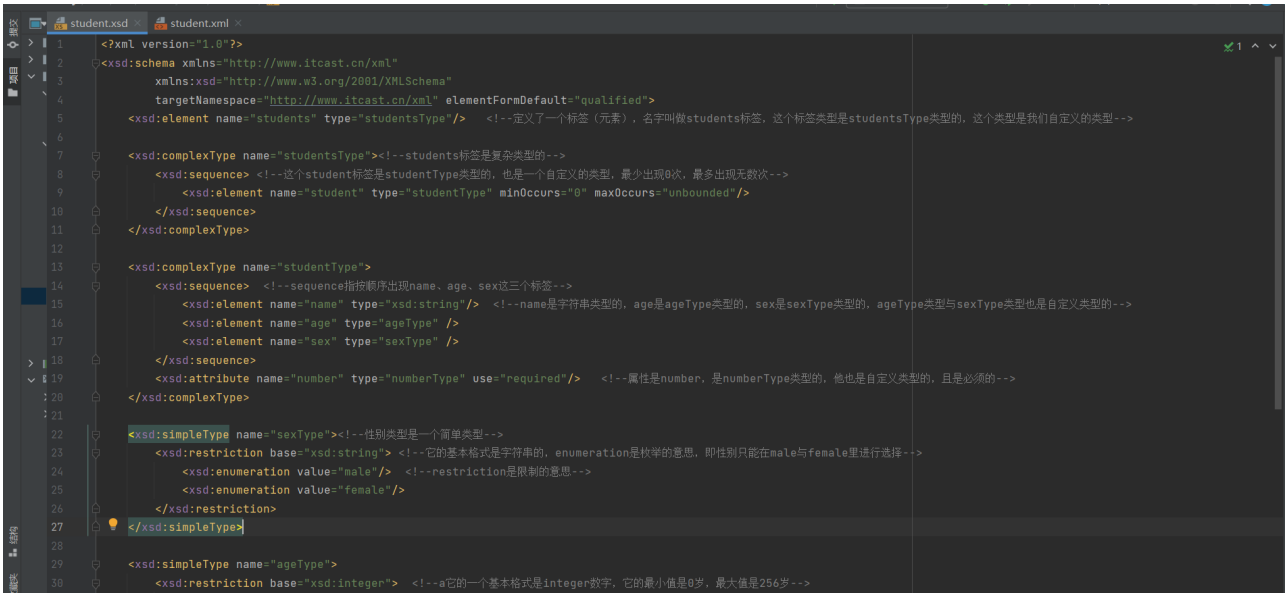
（如果我们要用student.xsd里的约束元素的话,必须得引用“文件命名空间", 即4.里的应用）

4.为每一个xsd约束声明一个前缀,作为标识

xmlns="http://www.itcast.cn/xml"

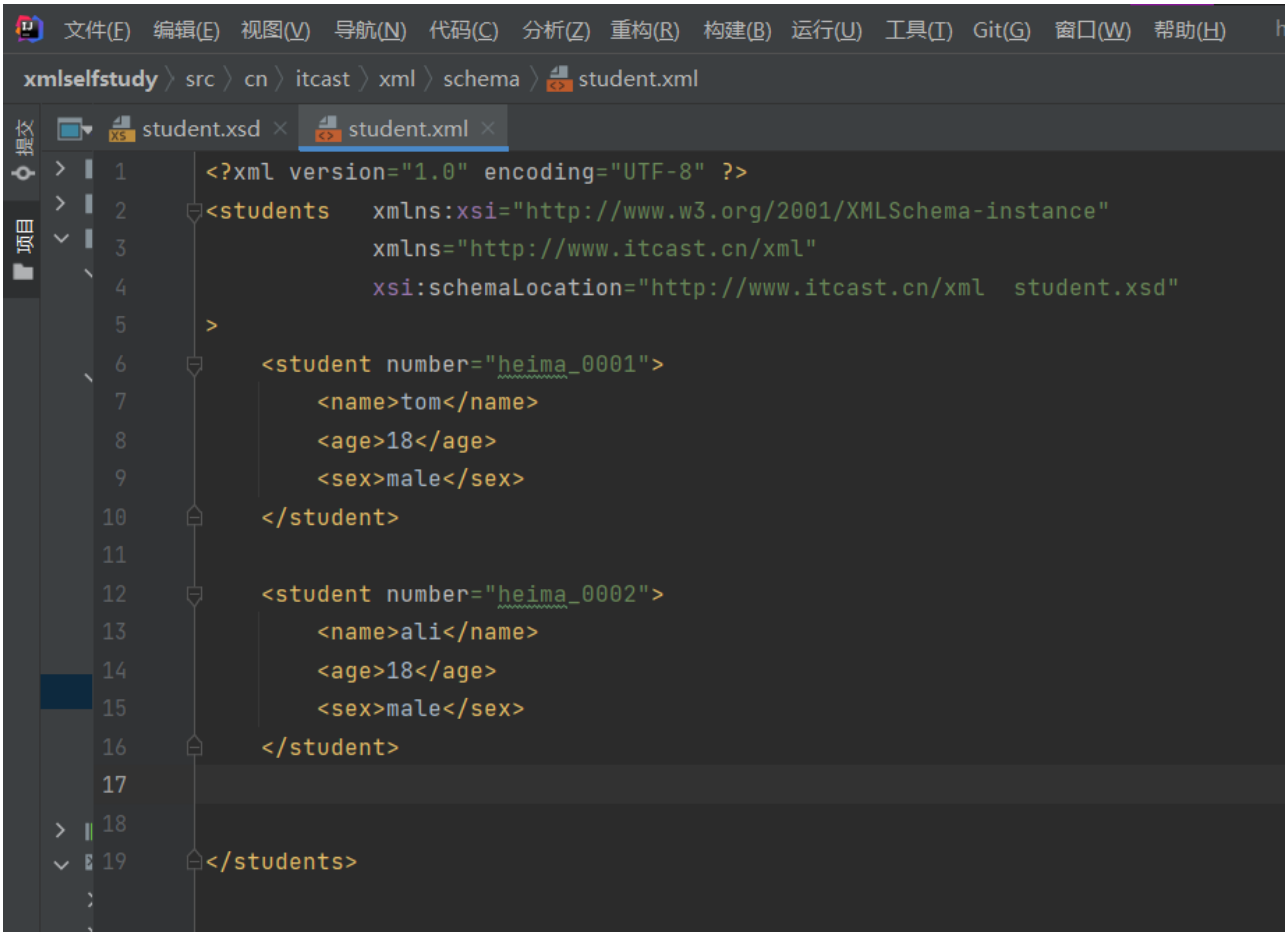
student.xml例如:

```
<?xml version="1.0"?>
<students   xmlns:xsi="http://wsww.w3.org/2001/XMLSchema-instance"
xmlns="http://www.itcast.cn/xml"
xsi:schemaLocation="http://www.itcast.cn/xml  student.xsd">
            ......................
</students>
```



```
    <xsd:simpleType name="ageType">
        <xsd:restriction base="xsd:integer">  <!--a它的一个基本格式是integer数字，它的最小值是0岁，最大值是256岁-->
            <xsd:minInclusive value="0"/>
            <xsd:maxInclusive value="256"/>
        </xsd:restriction>
    </xsd:simpleType>

    <xsd:simpleType name="numberType">  <!--基本格式是字符串，并且规定了它的pattern即它的组成格式必须叫"heima_四位的数字"-->
        <xsd:restriction base="xsd:string">
            <xsd:pattern value="heima_\d{4}"/>
        </xsd:restriction>
    </xsd:simpleType>
</xsd:schema>
```



文件(F) 编辑(E) 视图(V) 导航(N) 代码(C) 分析(Z) 重构(R) 构建(B) 运行(U) 工具(T) Git(G) 窗口(W) 帮助(H)

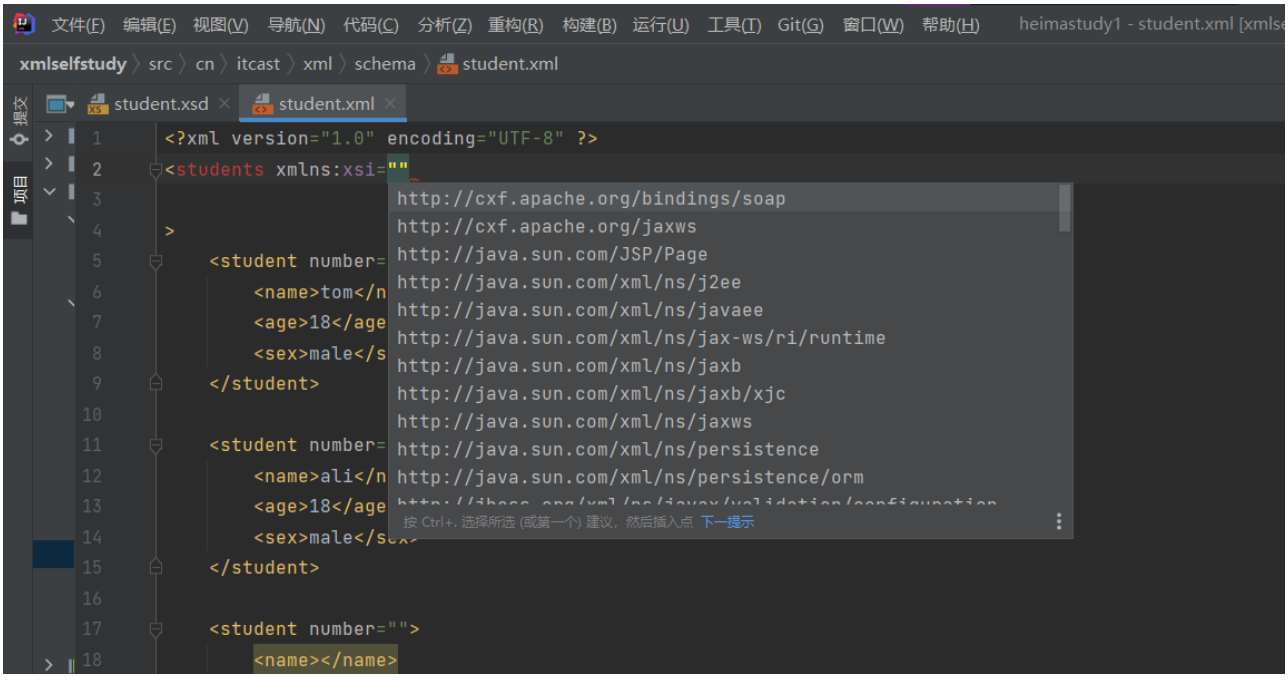
xmlselfstudy 〉 src 〉 cn 〉 itcast 〉 xml 〉 schema 〉 student.xml

student.xsd × student.xml ×

```
<?xml version="1.0" encoding="UTF-8" ?>
<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"
>
    <student number="heima_0001">
        <name>tom</name>
        <age>18</age>
        <sex>male</sex>
    </student>

    <student number="heima_0002">
        <name>ali</name>
        <age>18</age>
        <sex>male</sex>
    </student>


</students>
```



文件(F) 编辑(E) 视图(V) 导航(N) 代码(C) 分析(Z) 重构(R) 构建(B) 运行(U) 工具(T) Git(G) 窗口(W) 帮助(H) heimastudy1 - student.xml [xmlse

xmlselfstudy 〉 src 〉 cn 〉 itcast 〉 xml 〉 schema 〉 student.xml

student.xsd × student.xml ×

```
<?xml version="1.0" encoding="UTF-8" ?>
<students xmlns:xsi=""

>
    <student number=
        <name>tom</n
        <age>18</age
        <sex>male</s
    </student>

    <student number=
        <name>ali</n
        <age>18</age
        <sex>male</s
    </student>

    <student number="">
        <name></name>
```

http://cxf.apache.org/bindings/soap
http://cxf.apache.org/jaxws
http://java.sun.com/JSP/Page
http://java.sun.com/xml/ns/j2ee
http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/jax-ws/ri/runtime
http://java.sun.com/xml/ns/jaxb
http://java.sun.com/xml/ns/jaxb/xjc
http://java.sun.com/xml/ns/jaxws
http://java.sun.com/xml/ns/persistence
http://java.sun.com/xml/ns/persistence/orm

按 Ctrl+. 选择所选 (或第一个) 建议，然后插入点 下一提示

4. * 操作xml文档有两种方式

1. 解析(也叫读取)：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它的所有东西进内存，在内存中形成一颗dom树（服务器端用此,即javaweb用此）

* 优点：操作方便，可以对文档进行CRUD（增删改查）的所有操作

* 缺点：占内存

2. SAX：将标记语言文档里的内容逐行读取，读取就释放，然后再读取下一行，也就是说内存里永远只有一行内 容，基于事件驱动的。（移动，安卓方面，小型方面用此）

* 优点：不占内存。

* 缺点：只能读取，不能增删改

5.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（性能低，基本不用这个）

2. DOM4J：一款非常优秀的解析器（常用的DOM解析）

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。（可解析）（开源免费）

4. PULL：Android操作系统内置的解析器，sax方式的。

* Jsoup

* 快速入门：

* 步骤：

1. 导入jar包（jsoup-1.11.2.jar），然后右键所在文件夹lib进行添加为库（模块库，只有进行 添加库的话jar包才能被打开即jar包的二级文件夹才能被看见）

2. 新建一个类，获取Document对象

3. 获取对应的标签Element对象

4. 获取数据

* 对象的使用：(右击jsoup-1.11.2-javadoc.jar点击“解压到jsoup-1.11.2-javadoc”进行解压，进去后点击index.html进行API查找)

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（即解析xml或html字符串的原文本内容）

* 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，返回Element对象

* getElementsByTag(String tagName)：根据标签名称获取元素对象集合，返回Elements对象

* getElementsByAttribute(String key)：根据属性名称获取元素对象集合，返回Elements对象

* getElementsByAttributeValue(String key, String value)：根据属性名和属性值找到所对应的元素，来获取元素对象集合，返回Elements对象

3. Elements：元素Element对象的集合。可以当做 ArrayList<Element> 来使用

*size（）：返回数字

get（）：返回Element对象

4. Element：元素对象(元素是标签的意思，根标签是指根元素)

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,返回Element对象

* getElementsByTag(String tagName)：根据标签名称获取元素对象集合,返回Elements对象

* getElementsByAttribute(String key)：根据属性名称获取元素对象集合返回Elements对象

* 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,返回Elements对象

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（网上搜）

*select(String cssQuery) 是document的方法

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。（JsoupXpath-0.3.2.jar）

```
*  查询w3cshool参考手册，使用xpath的语法完成查询
```