

# 课程回顾

## 1 选择器

```
1  层级选择器：
2    子级选择器 父>子
3    后代选择器 祖先 后代
4    相邻选择器 前面节点+紧挨着后面的一个节点
5    同辈选择器 前面节点~后面所有的同辈节点
6
7  属性选择器：[属性名=,!=,^=,$=,*='值']
8  过滤选择器：:关键字
9    奇数 :odd
10   偶数 :even
11   :eq
12   :lt
13   :gt
14   :first
15   :last
16   :nth-child
17   .....
18  重点：
19  表单选择器
20  表单项的标签名input 选择器写法，统一语法：
21  :type的值
22
23  下拉列表：select
24  $("select")
25
26  多行文本：textarea
27  $("textarea").val()
28
29  表单属性过滤选择器
30  获取选中的列表项： :selected
31  $("select option:selected")
32
33  单选按钮、复选框获取选中项： :checked
34  $(":radio:checked")
35  $(":checkbox:checked")
```

## 2 DOM

```
1  查：表示获取功能  jquery中已经通过选择器可以实现
2  增：append() prepend appendTo prependTo after before insertAfter
    insertBefore
3  删：
4    删除一个或一些：remove()
5    清空选中父节点内部的子节点：父.empty()
6  改：
7    replaceWith()
8  创建节点：$(拼接好的标签字符串)
```

### 3 课后作业：三级联动

省份：

市：

区：

详细地址：

功能一：

网页加载完毕，就把所有的省份的数据放入到第一个下拉列表中

实现步骤：

1. 事件源 事件 注册事件监听器

事件源：document、

事件：ready

```
$(document).ready(function(){
```

```
})
```

现在通常将上面的代码简写成：\$(function(){})

2. 获取保存的省份的数组，通过DOM操作完成节点的添加

province数组循环，each()，父.append(新增子节点)

### 参考代码

```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <meta charset="UTF-8">
5     <title></title>
6     <script type="text/javascript" src="js/jquery-1.8.3.js" ></script>
7   </head>
8   <script type="text/javascript">
9     //定义数组保存所有的省
10    var province=["湖北省","河南省","湖南省","河北省"];
11    //使用省份名称为key，存入该省对应的市
12    province["湖北省"]=["武汉市","黄冈市","襄阳市","荆州市"];
13    province["河南省"]=["郑州市","信阳市","洛阳市","驻马店"];
14    province["湖南省"]=["长沙市","郴州市","岳阳市","衡阳市"];
15    province["河北省"]=["石家庄市","邯郸市","秦皇岛市"];
16    //一个格式，实现市和区的对应
17    province["武汉市"]=["洪山区","武昌区","汉口区"];
18    province["襄阳市"]=["襄州区","宜城市","谷城市"];
19    province["长沙市"]=["羊区","人区"];
20
21    $(function(){
22      //省份循环，并添加到省份下拉列表中
23      $(province).each(function(i,domele){
24        //将数组每一个元素添加到省份下拉列表中
25        $('#selProvince').append("<option>"+domele+"</option>");
26      });
27      //根据选择省份添加对应的城市
28      /**
29       * 1.selProvince的下拉列表的change
30       * 2.获取province[省份]对应的市
31       * 3.selCity添加数组每一个城市数据
32       */
33      $("#selProvince").change(function(){
34        //清空列表
35        /**
36         * option.empty()清空option的文本，理解empty()清空的选中项的后代
37         */
```

```

38         $("#selCity option:gt(0)").remove();
39         //1.清空县区
40     //     $("#selRegion option:gt(0)").remove();
41     //调用城市change事件
42     $("#selCity").change();//事件对应函数的执行
43     //1.用户选择的省份
44     var prov=$(this).val();//this.value;//$("#selProvince
option:selected").val();
45     //2.获取province[省份]对应的市
46     var arrCities=province[prov];
47     //3.遍历城市，并追加到selCity下拉列表中
48     $(arrCities).each(function(i,ele){
49         //1.创建一个新节点
50         var objNewOption=$("#<option></option>");
51         //1-1 设置选项中的数据
52         objNewOption.text(ele);
53         //2.新节点放到父节点中
54         $("#selCity").append(objNewOption);
55     });
56 });
57 //根据选择城市添加对应的县区
58 /**
59  * 1.selCity的下拉列表的change
60  * 2.获取province[城市名称]对应的县区数组
61  * 3.selRegion添加数组每一个城市数据
62  */
63     $("#selCity").change(function(){
64         //1.清空
65         $("#selRegion option:gt(0)").remove();
66         //2.获取数据并循环加载到selRegion的列表中
67         var arrRegions=province[$(this).val()];
68
69         $(arrRegions).each(function(i,ele){
70             $("#selRegion").append("<option>"+ele+"</option>");
71         });
72     });
73
74     $(".button").click(function(){
75         var address='';
76         address+=$("#selProvince option:selected").val();
77         address+=$("#selCity option:selected").val();
78         address+=$("#selRegion option:selected").val();
79         address+=$(".text").val();
80         alert(address);
81     });
82 });
83 </script>
84 <body>
85     省份:
86     <select id="selProvince" name="province">
87         <option>====请选择所在的省份====</option>
88     </select><br />
89     市: <select id="selCity" name="city">
90         <option>====请选择所在的市====</option>
91     </select><br />
92     区: <select id="selRegion" name="region">
93         <option>====请选择所在的县区====</option>
94     </select><br />

```

```
95      详细地址: <input type="text" name="detail" /><input type="button"  
      value="保存地址" />  
96      </body>  
97 </html>
```

# 课程目标

## 1 DOM补充 === 掌握

## 2 XML解析（侧重xml数据获取） === 理解

## 3 XML约束文档 == 了解

### DTD文档

### schema文档

# 课程实施

## 1 DOMの补充

### 1-1 css操作方法

```
1  css():设置或获取匹配项的指定属性的值  
2  举例:  
3  css("background-color","pink"):设置背景色功能  
4  css("background-color"):获取背景色  
5  支持类样式  
6  addClass():追加样式  
7  removeClass():移除样式  
8  toggleClass():交替样式  
9  hasClass():判断是否有指定的样式,有true 没有false
```

### 课堂练习

- 添加单个样式

```
1  <!DOCTYPE html>  
2  <html>  
3      <head>  
4          <meta charset="UTF-8">  
5          <title></title>  
6          <!--导入js文件-->  
7          <script type="text/javascript" src="js/jquery-1.8.3.js" ></script>  
8      </head>  
9      <script type="text/javascript">  
10         //给div添加边框样式: border粗细、border颜色、border外观  
11         //给div添加背景色,用green  
12         //给div添加字体样式: 字体粗体 倾斜 字号34px 颜色 红色  
13         $(function(){  
14             //给div1添加样式  
15             $("#div1").css("background-color","green").css("font-  
weight","bolder").css("color","red");
```

```

16     })
17     </script>
18     <body>
19         <div id="div1">
20             Hello JQuery
21         </div>
22     </body>
23 </html>

```

- css添加多个样式

```

1  <!DOCTYPE html>
2  <html>
3      <head>
4          <meta charset="UTF-8">
5          <title></title>
6          <!--导入js文件-->
7          <script type="text/javascript" src="js/jquery-1.8.3.js" ></script>
8      </head>
9      <script type="text/javascript">
10         //给div添加边框样式: border粗细、border颜色、border外观
11         //给div添加背景色, 用green
12         //给div添加字体样式: 字体粗体 倾斜 字号34px 颜色 红色
13         $(function(){
14             //给div1添加样式
15             //css({属性名:属性值,属性名:属性值,...,属性名:属性值}):设置多个样式
16             $("#div1").css({"background-color":"green","font-
17 weight":"bolder",
18                             "color":"red","font-size":"34px","font-
19 style":"italic",
20                             "border":"1px solid black"});
21         })
22     </script>
23     <body>
24         <div id="div1">
25             Hello JQuery
26         </div>
27     </body>
28 </html>

```

- addClass() removeClass() hasClass()

```

1  <!DOCTYPE html>
2  <html>
3      <head>
4          <meta charset="UTF-8">
5          <title></title>
6          <!--导入js文件-->
7          <script type="text/javascript" src="js/jquery-1.8.3.js" ></script>
8      </head>
9      <!--css-->
10     <style type="text/css">
11         .divaa{
12             background-color: pink;
13             font-style: italic;

```

```

14         font-weight: bolder;
15         border: dashed 2px green;
16         width:200px;
17         height: 200px;
18     }
19 </style>
20 <script type="text/javascript">
21     //给div添加边框样式: border粗细、border颜色、border外观
22     //给div添加背景色, 用green
23     //给div添加字体样式: 字体粗体 倾斜 字号34px 颜色 红色
24     $(function(){
25         $("#div1").click(function(){
26             //如果没有divaa的样式, 就加上这个样式
27             if(!$(this).hasClass("divaa")){
28                 $(this).addClass("divaa");
29             }else{
30                 //如果有divaa的样式, 移除该样式
31                 $(this).removeClass("divaa");
32             }
33         });
34     })
35 </script>
36 <body>
37     <div id="div1">
38         Hello JQuery
39     </div>
40 </body>
41 </html>
42

```

- toggleClass()

```

1 <!DOCTYPE html>
2 <html>
3     <head>
4         <meta charset="UTF-8">
5         <title></title>
6         <!--导入js文件-->
7         <script type="text/javascript" src="js/jquery-1.8.3.js" ></script>
8     </head>
9     <!--css-->
10    <style type="text/css">
11        .divaa{
12            background-color: pink;
13            font-style: italic;
14            font-weight: bolder;
15            border: dashed 2px green;
16            width:200px;
17            height: 200px;
18        }
19    </style>
20    <script type="text/javascript">
21        //给div添加边框样式: border粗细、border颜色、border外观
22        //给div添加背景色, 用green
23        //给div添加字体样式: 字体粗体 倾斜 字号34px 颜色 红色
24        $(function(){
25            $("#div1").click(function(){

```

```

26         //toggleClass(): div上有divaa这个样式，就移除，没有就添加
27         $(this).toggleClass("divaa");//toggler交替
28     });
29 }
30 </script>
31 <body>
32     <div id="div1">
33         Hello JQuery
34     </div>
35 </body>
36 </html>
37

```

## 1-2 jquery复合事件

```

1 click:单击
2 keyup:键盘弹起
3 mouseover:鼠标停留
4 mouseleave:鼠标离开
5 mouseout:鼠标离开

```

### 课堂案例

#### 表格高亮

编号	姓名	年龄
1	张三	24
2	jack	29
3	lily熊	14
3	lily熊	14

[首页](#)
[上一页](#)
[下一页](#)
[尾页](#)

事件:  
 事件源: tr  
 事件: mouseover mouseleave  
 功能: 当前鼠标停留的行添加背景色

```

1 <!DOCTYPE html>
2 <html>
3     <head>
4         <meta charset="UTF-8">
5         <title></title>
6         <script type="text/javascript" src="js/jquery-1.8.3.js" ></script>
7     </head>
8     <style type="text/css">
9         .high{
10             background-color: yellow;
11         }
12
13     </style>
14     <script type="text/javascript">
15         $(function(){

```

```

16 //鼠标停留在tbody的哪一个tr上，就是用high
17 $("tbody tr").mouseover(function(){
18     //添加样式
19     $(this).addClass("high");
20 }).mouseleave(function(){
21     //移除样式
22     $(this).removeClass("high");
23 });
24 })
25 </script>
26 <body>
27     <table border="1" cellspacing="0" cellpadding="4">
28         <thead>
29             <tr>
30                 <th>编号</th>
31                 <th>姓名</th>
32                 <th>年龄</th>
33             </tr>
34         </thead>
35         <tbody>
36             <tr>
37                 <td>1</td>
38                 <td>张三</td>
39                 <td>24</td>
40             </tr>
41             <tr>
42                 <td><p id="mini">2</p></td>
43                 <td>jack</td>
44                 <td>29</td>
45             </tr>
46             <tr>
47                 <td>3</td>
48                 <td>lily熊</td>
49                 <td>14</td>
50             </tr>
51             <tr>
52                 <td>3</td>
53                 <td>lily熊</td>
54                 <td>14</td>
55             </tr>
56         </tbody>
57         <tfoot>
58             <tr>
59                 <td colspan="3">
60                     <a href="#">首页</a>
61                     <a href="#">上一页</a>
62                     <a href="#">下一页</a>
63                     <a href="#">尾页</a>
64                 </td>
65             </tr>
66         </tfoot>
67     </table>
68 </body>
69 </html>

```



## 表格奇偶行+高亮

```
1  <!DOCTYPE html>
2  <html>
3      <head>
4          <meta charset="UTF-8">
5          <title></title>
6          <script type="text/javascript" src="js/jquery-1.8.3.js" ></script>
7      </head>
8      <style type="text/css">
9
10         .odd{
11             background-color: pink;
12         }
13         .even{
14             background-color: lightblue;
15         }
16         .high{
17             background-color: yellow;
18         }
19     </style>
20     <script type="text/javascript">
21     $(function(){
22         //文档加载完毕，表格奇偶行交替背景色
23         $("tbody tr:odd").addClass("odd");
24         $("tbody tr:even").addClass("even");
25         //鼠标停留在tbody的哪一个tr上，就是用high
26         $("tbody tr").mouseover(function(){
27             if($(this).hasClass("even")){
28                 //移除
29                 $(this).removeClass("even");
30             }else if($(this).hasClass("odd")){
31                 $(this).removeClass("odd");
32             }
33             //添加样式
34             //tr上有两个类样式: even-lightblue high-yellow 按照css里面定义顺序来
35             $(this).addClass("high");
36         }).mouseleave(function(){
37             //移除样式
38             $(this).removeClass("high");
39             //文档加载完毕，表格奇偶行交替背景色
40             $("tbody tr:odd").addClass("odd");
41             $("tbody tr:even").addClass("even");
42         });
43     })
44     </script>
45     <body>
46         <table border="1" cellspacing="0" cellpadding="4">
47             <thead>
48                 <tr>
49                     <th>编号</th>
50                     <th>姓名</th>
51                     <th>年龄</th>
52                 </tr>
53             </thead>
54             <tbody>
```

```

55         <tr>
56             <td>1</td>
57             <td>张三</td>
58             <td>24</td>
59         </tr>
60         <tr>
61             <td><p id="mini">2</p></td>
62             <td>jack</td>
63             <td>29</td>
64         </tr>
65         <tr>
66             <td>3</td>
67             <td>lily熊</td>
68             <td>14</td>
69         </tr>
70         <tr>
71             <td>3</td>
72             <td>lily熊</td>
73             <td>14</td>
74         </tr>
75     </tbody>
76     <tfoot>
77     <tr>
78         <td colspan="3">
79             <a href="#">首页</a>
80             <a href="#">上一页</a>
81             <a href="#">下一页</a>
82             <a href="#">尾页</a>
83         </td>
84     </tr>
85     </tfoot>
86 </table>
87 </body>
88 </html>

```

## hover

```

1  <!DOCTYPE html>
2  <html>
3      <head>
4          <meta charset="UTF-8">
5          <title></title>
6          <script type="text/javascript" src="js/jquery-1.8.3.js" ></script>
7      </head>
8      <style type="text/css">
9          #div1{
10              border: 1px green double;
11              width: 200px;
12              height: 200px;
13          }
14      </style>
15      <script type="text/javascript">
16          $(function(){
17              //模拟鼠标停留mouseover和鼠标离开mouseleave复合事件
18              //div在鼠标停留添加背景色 鼠标离开移除背景
19              //hover(mouseover(),mouseleave)
20              $("#div1").hover(function(){

```

```

21         $(this).css("background-color","pink");
22     },function(){
23         $(this).css("background-color","white");
24     });//模拟鼠标停留和离开
25 })
26 </script>
27 <body>
28     <div id="div1">Hello world</div>
29 </body>
30 </html>
31

```

## toggle

```

1  <!DOCTYPE html>
2  <html>
3      <head>
4          <meta charset="UTF-8">
5          <title></title>
6          <script type="text/javascript" src="js/jquery-1.8.3.js" ></script>
7      </head>
8      <style type="text/css">
9          #div1{
10             border: 1px green double;
11             width: 200px;
12             height: 200px;
13         }
14      </style>
15      <script type="text/javascript">
16          //div1点击一次，就在现在的字体大小+4
17          $(function(){
18              //toggle(fn1,fn2,...): 模拟鼠标点击在fn1 fn2交替执行
19              $("#div1").toggle(function(){
20                  $(this).css("font-size","18px");
21              },function(){
22                  $(this).css("font-size","24px");
23              },function(){
24                  $(this).css("font-size","36px");
25              });//模拟鼠标在div上面的多次单击
26          })
27      </script>
28      <body>
29          <div id="div1">Hello world</div>
30      </body>
31  </html>
32

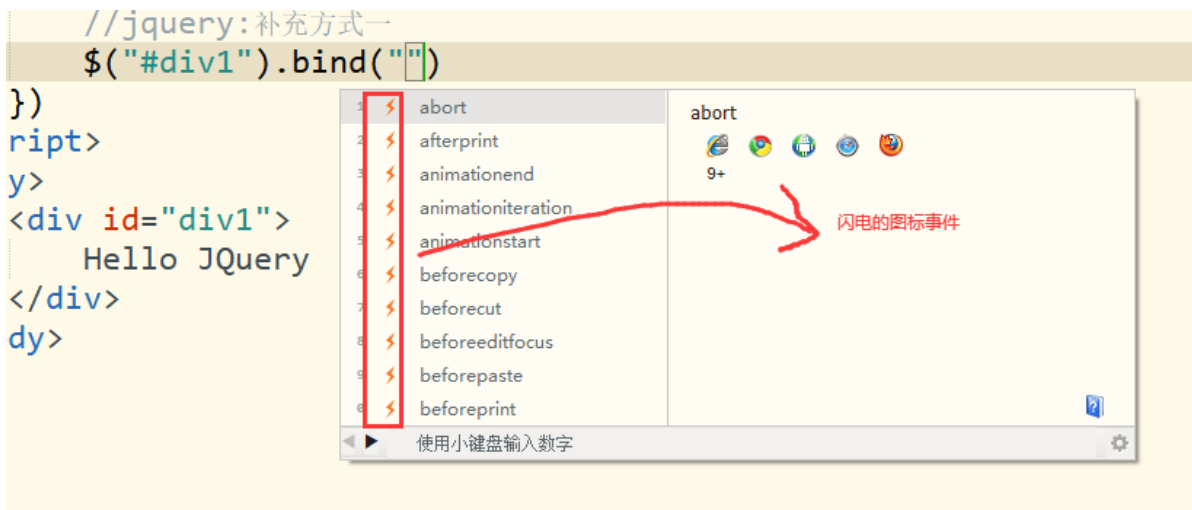
```

## 1-3 事件注册方式

```

1  bind("事件名",事件处理函数)
2  unbind("事件名")
3  one("事件名",事件处理函数)
4  delegate("委派节点选择器","事件名",事件处理函数)
5  undelegate("委派节点选择器","事件名")

```



## 课堂案例:

- ☐ 闭月
- ☐ 羞花
- ☐ 沉鱼
- ☐ 落雁
- ☐ aaaaaa

click()注册事件时, 只对html现有的li标签起作用

append()新增 aaaa这个li, 单击没有效果

### 添加节点的方法

## bind one unbind实现事件绑定

```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <meta charset="UTF-8">
5     <title></title>
6     <!--导入js文件-->
7     <script type="text/javascript" src="js/jquery-1.8.3.js" ></script>
8   </head>
9   <script type="text/javascript">
10     $(function(){
11       //div1添加单击事件
12       //$("#div1").click(function(){});
13       //jquery:补充方式一 bind(事件名, 事件处理函数)
14       /*$("#div1").bind("click",function(){
15         alert(this.innerText);//div1里面的文本
16         //移除click
17         $(this).unbind("click");//div的click事件移除
18       });*/
19
20       //jquery:补充方式二 one(事件名, 事件处理函数)事件绑定时, 事件有且只会执行
21       一次
22       $("#div1").one("click",function(){
23         alert(this.innerText);//div1里面的文本
24       });
25     })
26   </script>
27 </html>
```

```

25     </script>
26     <body>
27         <div id="div1">
28             Hello JQuery
29         </div>
30     </body>
31 </html>
32

```

## delegate实现事件绑定

```

1  <!DOCTYPE html>
2  <html>
3      <head>
4          <meta charset="UTF-8">
5          <title></title>
6          <script src="js/jquery-1.8.3.js" type="text/javascript"></script>
7          <script type="text/javascript">
8              $(function(){
9                  //给ul里面每一个li添加单击事件，单击任意一个li，alert(li里面显示文字)
10                 /*$("ul li").click(function(){
11                     alert($(this).text());
12                 })*/*
13
14                 /* bind click事件注册时，只对当前html现有li有效果，代码添加的li就没
有效果
15                 * $("ul li").bind("click",function(){
16                     alert($(this).text());
17                 })*/*
18
19                 //厉害：delegate不仅可以给html当前节点注册事件，还可以对未来代码创建
的节点添加事件
20                 $("body").delegate("li","click",function(){
21                     alert($(this).text());
22                 })
23
24                 //下标指定按钮
25                 //1.获取所有的按钮
26                 var arrButtons=$(":button");
27                 //2.获取单个按钮，做click事件注册
28                 // arrButtons.eq(0);//下标从0开始
29                 $(":button:eq(0)").click(function(){
30                     $("ul").append("<li><input type='checkbox'
/>aaaaaa</li>")
31                 });
32
33                 $(":button:eq(1)").click(function(){
34                     //after:同级节点.prependTo(ul)
35                     //获取沉鱼节点
36                     var objLi=$("ul li:nth-child(3)");
37                     //沉鱼后面添加bbbbbb
38                     objLi.after("<li><input type='checkbox' />bbbbbb</li>")
39                 });
40             });
41     </script>
42 </head>
43 <body>

```

```

44         <ul>
45             <li class="li1"><input type="checkbox" />闭月</li>
46             <li><input type="checkbox" />羞花</li>
47             <li class="li1"><input type="checkbox" />沉鱼</li>
48             <li><input type="checkbox" />落雁</li>
49         </ul>
50         <fieldset>
51             <legend>添加节点的方法</legend>
52             <input type="button" name="" id="" value="append" />
53             <input type="button" name="" id="" value="after" />
54         </fieldset>
55     </body>
56 </html>
57

```

## 3 XML

### 3-1 概念

xml: Extensible Markup Language **可扩展标记语言**

### 3-2 xml和html有什么区别？

```

1  xml:SSM框架、JavaEE配置servlet filter listener使用xml完成
2  xml:侧重数据存储，提供给ssm自行解析  程序员xml文档内容copy保存xml中
3
4  html:网页，用户看html展示和数据提交
5  html:侧重数据展示和提交后台使用  程序员总是要操作html

```

### 3-3 xml优势

```

1  xml优势:
2  相比较数据存储的文件格式:
3      U1学习IO, .txt  .doc .class
4  阅读性好。简单易操作
5
6  缺点: 标签名自定义，团队开发不便于管理。所以需要XML约束!!!
7

```

### 3-4 xml的书写要求

#### xml基本语法和格式

```

1  xml必须有关闭标签
2  区分大小写（在xml中html和HTML两个标签）
3  属性需要有引号
4  标签必须正确的嵌套
5  必须有根元素（xml文档必须有一个根节点，类似html中<html>）
6
7  xml是标签语言，标签可以自己随便写!!!
8  html的标签必须按照提供的单词写，自己不能发明创造的
9
10  格式
11      xml的文档声明

```

```
12  文档声明：    通常出现在xml的第一行第一列的位置!!!
13  写法：
14  <?xml    属性名="属性值"    属性名="属性值"    ...>
15  version:    版本号,必须的    默认值1.0
16  encoding:    编码字符集,是使用浏览器打开的时候采用的默认的字符集的编码。
17  standalone: 描述xml文档是否需要依赖其他的文件。
```

## xml标签名的书写规范

### XML 命名规则

XML 元素必须遵循以下命名规则：

- 名称可以含字母、数字以及其他的字符
- 名称不能以数字或者标点符号开始
- 名称不能以字符 “xml” ( 或者 XML、Xml ) 开始
- 名称不能包含空格

可使用任何名称，没有保留的字词。

## xml文档案例参考

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <!--数据存储和传输-->
3  <!--people理解成是xml的根节点，根节点有且只有一个，类似html的<html>
4  xml标签都是自定义的，遵循命名规则即可：
5  -->
6  <people>
7      <!--
8          xml优势：阅读性好。简单易操作
9          缺点：标签名自定义，团队开发不便于管理
10         XML约束!!!
11         相比较数据存储的文件格式：
12         U1学习IO, .txt .doc .class
13     -->
14     <person id="p001">
15         <name>张无忌</name>
16         <age>24<12</age>
17         <sex>男</sex>
18         <!--          . . . . -->
19     </person>
20     <person id="p002">
21         <name>周芷若</name>
22         <age>22</age>
23         <sex>女</sex>
24         <!--          . . . . -->
25     </person>
26     <person id="p003">
27         <name>金毛狮王</name>
28         <age>42</age>
29         <sex>男</sex>
30         <!--          . . . . -->
31     </person>
32     <person id="p004">
33         <name>杨逍</name>
34         <age>36</age>
35         <sex>男</sex>
```

```
36      <!--      。 。 。 。      -->
37      </person>
38 </people>
```

### 3-5 xml书写时常见错误

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- 数据存储和传输-->
<!-- people 理解成是xml的根节点，根节点有且只有一个，类似html的<html>-->
<people>

</people>
<people>
</people>
```

Multiple root tags      xml根节点不能重复出现

### 6.xml 的特殊字符和 CDATA 区

xml 的特殊字符      与HTML一致

&lt;	<	小于
&gt;	>	大于
&amp;	&	和号
&apos;	'	单引号
&quot;	"	引号

## 4 xml 解析

读取一个xml保存的所有数据

### 4-1 xml解析步骤

```
1 1. 准备一个xml文档（xml文档要求格式良好）
2 2. 引用第三方jar包 *****DOM4J
3 3. 写代码实现数据读取
```

### 4-2 XML解析技术



DOM: 一次性将文档加载到内存,形成树形结构进行解析

优点: 可以对xml进行增删改的操作

缺点: 如果文档特别大,容易导致内存溢出

SAX解析: 事件驱动的方式,逐行进行解析

优点: 如果文档特别大,不会导致内存溢出

缺点: 不能够对文档进行增删改的操作

## DOM解析

```
<people>
  <!--
    xml 优势: 阅读性好。简单易操作
    缺点: 标签名自定义, 团队开发不便于管理
    XML 约束!!!
    相比较数据存储的文件格式:
    U1 学习IO, .txt .doc .class
  -->
  <person id="p001">
    <name>张无忌</name>
    <age>24</age>
    <sex>男</sex>
  </person>
  <person id="p002">
```

DOM解析: 一次性需要加载整个XML文件到内存, 维护DOM模型

缺点: xml文档较大, 可能会出现内存溢出

优点: 入门方便、读取节点、实现节点添加、修改、删除

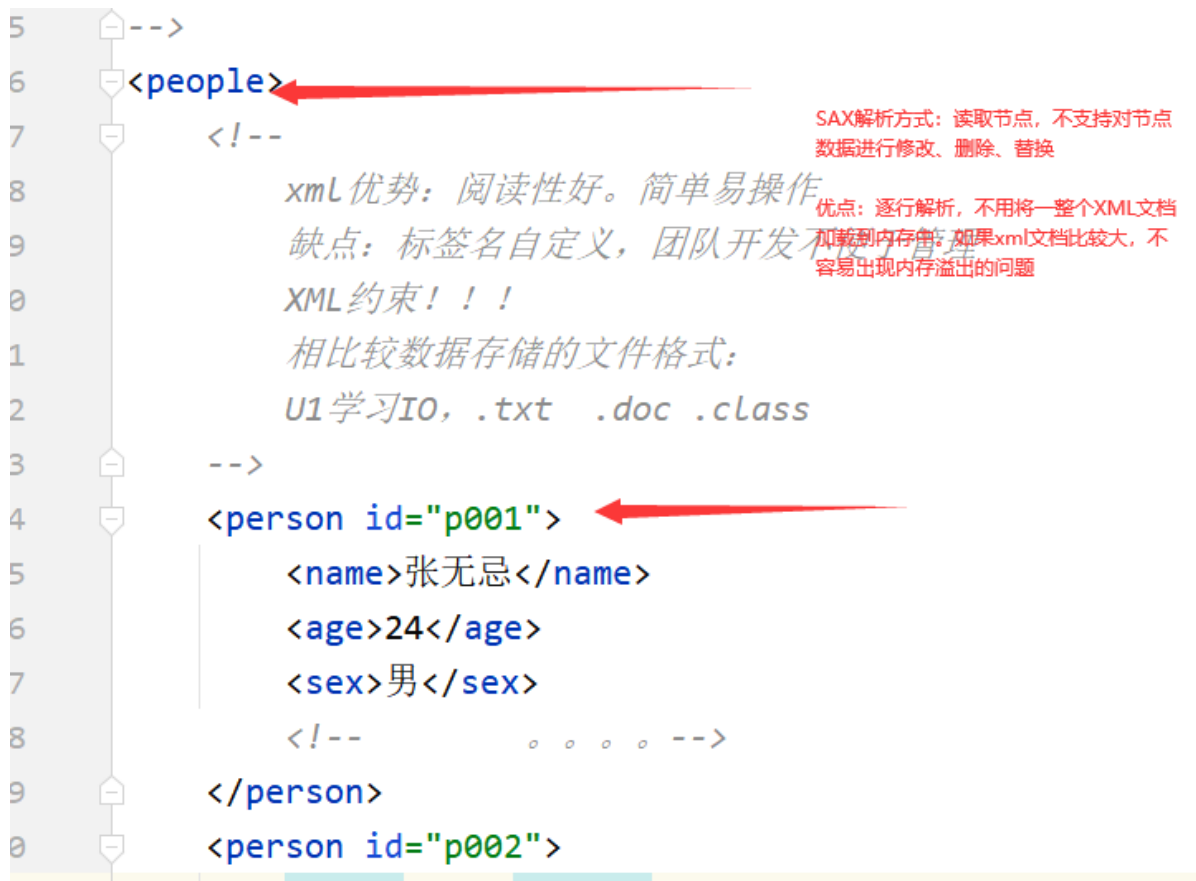
people

Person

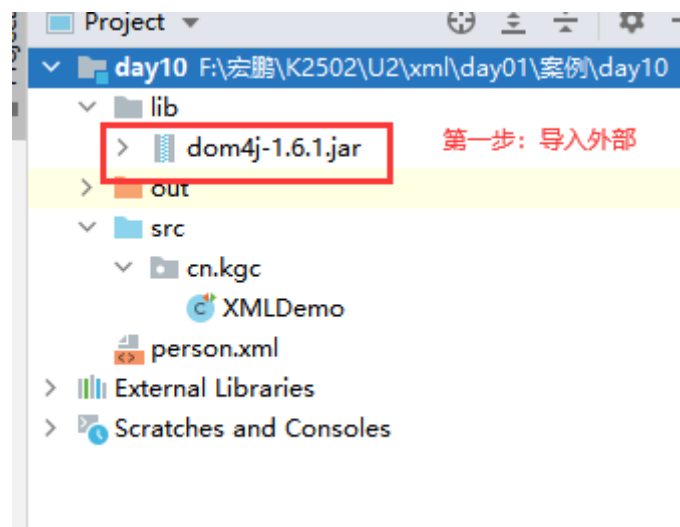
name age sex

people > person > name

## SAX解析



## 4-3 dom4j实现xml解析



### 参考代码

- 获取属性ID的值

```

1 package cn.kgc;
2
3 import org.dom4j.Document;
4 import org.dom4j.Element;
5 import org.dom4j.io.SAXReader;
6 import org.junit.Test;
7
8 import java.util.List;
9

```

```

10  /**
11   * @Author: lc
12   * @Date: 2022/4/27
13   * @Description: 实现xml文档的解析
14   * @Version: 1.0
15   */
16  public class XMLDemo {
17      @Test
18      public void getXmlInfo() throws Exception{
19          /**
20           * dom4j实现xml文档解析
21           * 实现思路:
22           * 1.SAXReader对象是DOM4J提供的将xml文件一次性加载到内存中,html的dom是一个
意思
23           * 返回一个document对象
24           * 2.使用document.获取方法()节点
25           */
26          //创建一个加载xml文档对象
27          SAXReader reader=new SAXReader();
28          //实现将person.xml转换为dom模型
29          Document document = reader.read("person.xml");
30          // 获取数据
31          /**
32           * document获取数据的步骤:
33           * 1.获取xml的根节点
34           * 2.根节点找子节点
35           * 3.子节点找孙节点
36           */
37          //获取根节点: people
38          Element root = document.getRootElement();
39          //找子节点: person 四个
40          List<Element> elements = root.elements();
41          System.out.println("子节点的数量: "+elements.size());
42
43          //获取第一个Person的数据
44          Element firstPerson = elements.get(0);
45          //获取id的属性值
46          //String id=firstPerson.attribute("id").getValue();
47          String id=firstPerson.attributeValue("id");
48          System.out.println("id="+id);
49      }
50  }

```

(m) <b>element</b> (String s)	获取指定标签名的那个标签对象	Element
(m) <b>element</b> (QName qName)		Element
(m) <b>elements</b> ()	获取父节点里面的子节点	List
(m) <b>elements</b> (String s)		List
(m) <b>elements</b> (QName qName)		List
(m) <b>elementIterator</b> ()		Iterator
(m) <b>elementIterator</b> (String s)		Iterator
(m) <b>elementIterator</b> (QName qName)		Iterator
(m) <b>elementText</b> (String s)	获取指定标签名存入的文字	String
(m) <b>elementText</b> (QName qName)		String
(m) <b>elementTextTrim</b> (String s)		String
(m) <b>elementTextTrim</b> (QName qName)		String

Ctrl+向下箭头 and Ctrl+向上箭头 will move caret down and up in the editor [Next Tip](#)

- 获取第一个person子标签name、age、sex的值

```

1  package cn.kgc;
2
3  import org.dom4j.Document;
4  import org.dom4j.Element;
5  import org.dom4j.io.SAXReader;
6  import org.junit.Test;
7
8  import java.util.List;
9
10 /**
11  * @Author: lc
12  * @Date: 2022/4/27
13  * @Description: 实现xml文档的解析
14  * @Version: 1.0
15  */
16 public class XMLDemo {
17     @Test
18     public void getXmlInfo() throws Exception{
19         /**
20          * dom4j实现xml文档解析
21          * 实现思路:
22          * 1. SAXReader对象是DOM4J提供的将xml文件一次性加载到内存中，html的dom是一个
意思
23          * 返回一个document对象
24          * 2. 使用document.获取方法()节点
25          */
26         //创建一个加载xml文档对象
27         SAXReader reader=new SAXReader();
28         //实现将person.xml转换为dom模型
29         Document document = reader.read("person.xml");
30         // 获取数据
31         /**
32          * document获取数据的步骤:
33          * 1. 获取xml的根节点
34          * 2. 根节点找子节点
35          * 3. 子节点找孙节点

```

```

36      */
37      //获取根节点: people
38      Element root = document.getRootElement();
39      //找子节点: person 四个
40      List<Element> elements = root.elements();
41      System.out.println("子节点的数量: "+elements.size());
42
43      //获取第一个Person的数据
44      Element firstPerson = elements.get(0);
45      //获取第一个Person里面的name标签值、age标签的值、sex标签的值
46      //elementText(name):获取name标签里面的文本,类似js的innerText
47      /*String name = firstPerson.elementText("name");
48      System.out.println(name);
49
50      //element(age):获取age标签对象
51      Element ageEle = firstPerson.element("age");
52      String age=ageEle.getText();
53      System.out.println(age);
54
55      //elementTextTrim(sex):获取sex标签内部的文本,并去掉文本首尾空格
56      String sex = firstPerson.elementTextTrim("sex");
57      System.out.println(sex);*/
58      //循环方式:
59      //1.获取person里面的子节点
60      List<Element> sonEles = firstPerson.elements();
61      for(Element e:sonEles){
62          String text = e.getText();
63          System.out.println(text);
64      }
65  }
66  }

```

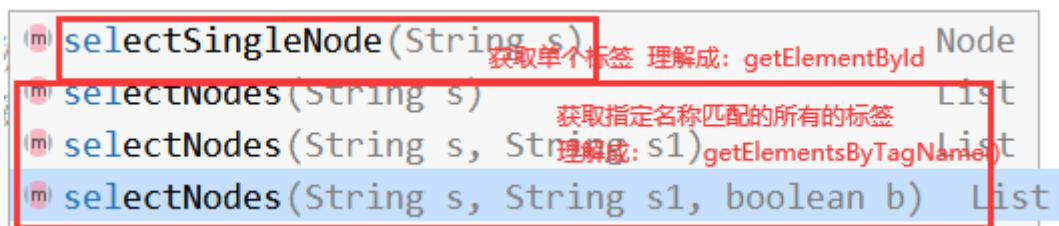
## 4-4 XPath结合DOM4J实现xml解析

传统的dom4j的方式解析xml文档,必须遵循根节点找子节点,再通过子节点找下一次子节点.....xml文件节点嵌套级别较多,解析一个xml代码就会非常冗繁。优化节点查找的方式,dom4j提供新的节点查找的方式: XPath

### 4-4-1 xpath优点

dom模型中,使用特殊的路径写法,更快做节点的定位。

### 4-4-2 xpath获取ID是p003的人的信息



```

1      @Test
2      public void getXmlInfo2() throws Exception{
3          /**
4              * 1.加载person.xml获取Document对象
5              * 2.使用document对象根据xpath指定的路径获取对应的节点
6              */

```

```

7      SAXReader reader=new SAXReader();
8      Document document = reader.read("person.xml");
9      //2.区别来了
10     //ID是p003的人的信息 只想找一个person标签?!
11     //Node是Element Attribute Text的父接口
12     //xpath //标签名[@属性名=属性值]
13     Node node = document.selectSingleNode("//person[@id='p003']");
14     //获取p003对应的name age sex
15     if(node!=null){//非空验证, 排除NullPointerException
16         if(node instanceof Element){//node父接口保存的变量向下转型实际类型
17             Element
18                 //向下转型
19                 Element p003Person=(Element)node;
20                 System.out.println("id="+p003Person.attributeValue("id"));
21                 System.out.println("name="+p003Person.elementText("name"));
22                 System.out.println("age="+p003Person.elementText("age"));
23                 System.out.println("sex="+p003Person.elementText("sex"));
24         }
25     }

```

## 5 XML约束

### 5-1 xml约束

- dtd约束
- schema约束

```

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3.org/TR/html4/loose dtd">
<html>
<head>
    <title>Title</title>
</head>
<body>
</body>
</html>

```

HTML书写的时候, 要求标签名字必须是指定的名字, 标签的嵌套必须按照规定格式嵌套, 都是源于html收到约束文档的约束。  
html使用约束文档: DTD格式

### 5-2 DTD约束书写

#### dtd文档定义

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <!--定义当前xml能书写标签以及属性、标签顺序-->
3 <!ELEMENT people (person*)>
4 <!ELEMENT person (name,age,sex)>
5 <!--          PCDATA表示文本 -->
6 <!ELEMENT name      (#PCDATA)>
7 <!ELEMENT age      (#PCDATA)>
8 <!ELEMENT sex       (#PCDATA)>
9
10 <!--person标签上得有一个id的属性，且属性值不能重复
11 ID:dtd的数据类型，表示id属性值不能重复
12 -->
13 <!ATTLIST person id ID #REQUIRED>

```

## xml引入dtd

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <!--导入本地外部的dtd文件-->
3 <!DOCTYPE people SYSTEM "person.dtd">
4 <!--数据存储和传输-->
5 <!--people理解成是xml的根节点，根节点有且只有一个，类似html的<html>
6   xml标签都是自定义的，遵循命名规则即可：
7   -->
8 <people>
9   <!--
10      xml优势：阅读性好。简单易操作
11      缺点：标签名自定义，团队开发不便于管理
12      XML约束!!!
13      相比较数据存储的文件格式：
14      U1学习IO, .txt .doc .class
15   -->
16   <person id="p001">
17     <name>张无忌</name>
18     <age>24</age>
19     <sex>男</sex>
20     <!--          。 。 。 。 -->
21   </person>
22   <person id="p002">
23     <name>周芷若</name>
24     <age>22</age>
25     <sex>女</sex>
26     <!--          。 。 。 。 -->
27   </person>
28   <person id="p003">
29     <name>金毛狮王</name>
30     <age>42</age>
31     <sex>男</sex>
32     <!--          。 。 。 。 -->
33   </person>
34   <person id="p004">
35     <name>杨逍</name>
36     <age>36</age>
37     <sex>男</sex>
38     <!--          。 。 。 。 -->
39   </person>
40 </people>

```

## 5-3 Schema约束书写

### xsd文档定义

```
1 <?xml version="1.0"?>
2 <!--
3 xmlns:xs:设置当前xsd文件遵循的约束文档
4 targetNamespace: 为当前文档定义引用url地址
5 targetNamespace="http://www.kgc.com/person"提供给xml引用使用
6 url地址自己编的, 随便写
7 elementFormDefault: 约束文档使用时约束
8 -->
9 <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
10           targetNamespace="http://www.kgc.com/person"
11           elementFormDefault="qualified">
12     <!-- 根标签的名称 -->
13     <xs:element name="people">
14       <!-- complexType: people标签是一个复合标签, 有子标签 -->
15       <xs:complexType>
16         <!-- sequence people子标签出现的顺序 -->
17         <xs:sequence>
18           <!-- minOccurs:设置person出现的最少次数    maxOccurs: 设置person出现的
19             最多次数-->
20             <xs:element name="person" minOccurs="1" maxOccurs="unbounded">
21               <xs:complexType>
22                 <xs:sequence>
23                   <xs:element name="name" type="xs:string"/>
24                   <xs:element name="age" type="xs:integer"/>
25                   <xs:element name="sex" type="xs:string"/>
26                 </xs:sequence>
27                 <!-- 设置person上面的属性-->
28                 <xs:attribute name="id" type="xs:ID" use="required"/>
29               </xs:complexType>
30             </xs:element>
31           </xs:sequence>
32         </xs:complexType>
33       </xs:element>
34     </xs:schema>
```

### xml引入schema文件

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <!--
3 xmlns="http://www.w3.org/2001/XMLSchema-instance" xml文件要引用schema
4 xmlns="http://www.kgc.com/person"设置了引用schema url地址
5 xs:schemaLocation="http://www.kgc.com/person person.xsd" url地址中具体schema
6 文件的名称
7 -->
8 <people xmlns:xs="http://www.w3.org/2001/XMLSchema-instance"
9         xmlns="http://www.kgc.com/person"
10         xs:schemaLocation="http://www.kgc.com/person person.xsd">
11   <person id="p001">
12     <name>张无忌</name>
13     <age>34</age>
14     <sex>男</sex>
15   </person>
```



```
15     <person id="p002">
16         <name>周芷若</name>
17         <age>22</age>
18         <sex>女</sex>
19     </person>
20     <person id="p003">
21         <name>金毛狮王</name>
22         <age>42</age>
23         <sex>男</sex>
24     </person>
25     <person id="p004">
26         <name>杨逍</name>
27         <age>36</age>
28         <sex>男</sex>
29     </person>
30 </people>
```

## 课程总结

---

1. 重点：掌握选择器+dom动态添加节点、删除节点
2. 理解：addClass removeClass()
3. 了解：bind unbind delegate one hover toggle
4. 掌握：xml书写格式
5. 理解：xml解析的方式
6. 了解：xml约束

## 预习安排

---

**\*mysql服务器安装**

mysql数据库