# 更新

7月14日重新排版，更新了ajax在struts上的运用

2018年10月24日添加属性选择器和过滤器，更新基本选择器和层次

# jQuery介绍

前提

为了简化 JavaScript 的开发, 一些 JavsScript 库诞生了. JavaScript 库封装了很多预定义的对象和实用函数。能帮助使用者建立有高难度交互的 Web2.0 特性的富客户端页面, 并且兼容各大浏览器

当前流行的 JavaScript 库有: jQuery, MooTools, Prototype, Dojo, YUI, EXT\_JS DWR

[www.open-open.com](http://www.open-open.com)开源网页

jQuery是一个快速的，简洁的javaScript库，使用户能更方便地处理HTML documents、events、实现动画效果，并且方便地为网站提供AJAX交互。

什么是jQuery对象？

jQuery 对象就是通过jQuery包装DOM对象后产生的对象。

jQuery 对象是 jQuery 独有的. 如果一个对象是 jQuery 对象, 那么它就可以使用 jQuery 里的方法: $(“#test”).html();

虽然jQuery对象是包装DOM对象后产生的，但是jQuery无法使用DOM对象的任何方法，同理DOM对象也不能使用jQuery里的方法.乱使用会报错

# DOM和jQuery互转

DOM对象转成jQuery对象

对于已经是一个DOM对象，只需要用$()把DOM对象包装起来，就可以获得一个jQuery对象了。$(DOM对象)

<input type=”text” name=”a” id=”b” value=”温老师”/>

//dom方式获得id为b的input标签

var b=document.getElementById("b");

//转换为jquery对象

var a=$(b);

alert(a.val());

jQuery对象转成DOM对象 两种方式

(1) jQuery对象是一个数组对象，可以通过[index]的方法，来得到相应的DOM对象

//通过id选择器获得id为b的input标签

<input type=”text” name=”a” id=”b” value=”温老师”/>

var a=$("#b");

//转换成dom对象，使用dom方法value

var b=a[0];

alert(b.value);

(2) jQuery本身提供，通过.get(index)方法，得到相应的DOM对象

//通过id选择器获得id为b的input标签

var a=$("#b");

//转换成dom对象，使用dom方法value

var b=a.get(0);

alert(b.value);

dom中的value方法出错通过alert没有提示，jquery的val()方法alert有提示

# jQuery 选择器

常用

第一个：通过表单中的name获取指定的标签元素

<input name="userName" type="text" phone" >

$("input[name='userName']").click(function(){alert('点击我做什么'); })

## 基本选择器

jquery基本选择器和css样式的基本选择器是一样的

id选择器

#id 用法: $(“#myDiv”); 返回单个元素

相当于dom中的方法：document.getElementById("myDiv ");

标签选择器

$(“div”) 返回值 集合元素

$("input").click(function(){

alert("我被点击了");

});

类选择器

class 用法: $(”.myClass”) 返回值 集合元素

通配符选择器

\* 用法: $(”\*”) 返回值 集合元素

说明: 匹配所有元素,多用于结合上下文来搜索

## 层次选择器

通过基本选择器进行搭配使用

第一种 ：”空格” ：匹配第一个元素下的所有子孙元素

选择器1 选择器：2$ (“form input”) ; 返回值 集合元素

这个意思是在form标签下的所有input标签都选中

第二种：”>”:匹配第一个元素下的所有儿子元素

选择器1 >选择器：$(”form > input”) ; 返回值 集合元素

这个意思是在form标签下的的第一级（也就是儿子）input标签都选中

第三种：找出第一个元素紧挨着的一个弟弟

选择器1 +选择器：$(”label + input”) ; 返回值 一个元素

说明: 匹配所有紧接在 prev 元素后的 next 元素

第四种：找出第一个元素所有弟弟

1、选择器1 ~选择器： $(”form ~ input”) ; 返回值 集合元素

说明: 匹配 prev 元素之后的所有 siblings 元素.注意:是匹配之后的元素,不包含该元素在内,并且siblings匹配的是和prev同辈的元素,其后辈元素不被匹配.

2、$("#two").siblings("div").css();

选择所有兄弟使用siblings方法来获取

## 属性选择器

1、匹配包含给定属性的元素 [attribute]

描述:

查找所有含有 id 属性的 div 元素

HTML 代码:

<div>

<p>Hello!</p>

</div>

<div id="test2"></div>

jQuery 代码:

$("div[id]")

结果:

[ <div id="test2"></div> ]

2、匹配给定的属性是某个特定值的元素

①$("input[name='newsletter']") 值等于

②$("input[name!='newsletter']") 值非等于

## 过滤器

配合选择器或层级选择器找到的元素再进行过滤：过滤器写在最后面

第一种 选择器找到的所有元素：进行过滤掉找到第一个元素 :first

用法: $(“tr:first”) ; 返回值 单个元素的组成的集合

第二种 选择器找到的所有元素：进行过滤掉找到最后一个元素 :last

用法: $(”tr:last”) 返回值 集合元素

第三种 选择器找到的所有元素：进行过滤掉找到 未被选择器找到的元素 :not(selector)

一般用在单选框或复选框

html代码

<input name="apple" />

<input name="flower" checked="checked" />

js代码

$("input:not(:checked)")

结果匹配的元素:

<input name="apple" />   
用法: $(”input:not(:checked)”)返回值 集合元素

第四种 选择器找到的所有元素：进行过滤掉找到偶数的索引:even

选择器找到的所有元素：进行过滤掉找到奇数数的索引:odd

$("tr:even") 说明: 匹配所有索引值为偶数的元素，从 0 开始计数.js的数组都是从0开始计数的.例如要选择table中的行,因为是从0开始计数,所以table中的第一个tr就为偶数0.

查找表格的1、3、5...行（即索引值0、2、4...）

HTML 代码:

<table>

<tr><td> 1</td></tr>

<tr><td> 2</td></tr>

<tr><td> 3</td></tr>

</table>

jQuery 代码:

$("tr:even")

结果:

[ <tr><td> 1</td></tr>, <tr><td> 3</td></tr> ]

第五种 选择器找到的所有元素：匹配一个给定索引值的元素:eq(index)

匹配大于：:gt(index)、匹配小于：:lt(index)

描述:

查找第二行

HTML 代码:

<table>

<tr><td>Header 1</td></tr>

<tr><td>Value 1</td></tr>

<tr><td>Value 2</td></tr>

</table>

jQuery 代码:

$("tr:eq(1)")

结果:

[ <tr><td>Value 1</td></tr> ]

# 表格常用写法

html代码

<table border="1px" width="600px" id="tab">

<thead>

<tr >

<td>

选择全部<input type="checkbox" id="checkAll" />

</td>

<td>分类ID</td>

<td>分类名称</td>

<td>分类商品</td>

<td>分类描述</td>

<td>操作</td>

</tr>

</thead>

<tbody>

<tr>

<td>

<input type="checkbox" />

</td>

<td>1</td>

<td>手机数码</td>

<td>华为,小米,尼康</td>

<td>黑马数码产品质量最好</td>

<td>

<a href="#">修改</a>|<a href="#">删除</a>

</td>

</tr>

<tr>

<td>

<input type="checkbox" />

</td>

<td>2</td>

<td>成人用品</td>

<td>充气的</td>

<td>这里面的充气电动硅胶的</td>

<td><a href="#">修改</a>|<a href="#">删除</a></td>

</tr>

<tr>

<td>

<input type="checkbox" />

</td>

<td>3</td>

<td>电脑办公</td>

<td>联想,小米</td>

<td>笔记本特卖</td>

<td><a href="#">修改</a>|<a href="#">删除</a></td>

</tr>

</tbody>

</table>

表格的全选和全不选

js代码

第一种方式：使用属性选择器：选择器[属性名称='属性值']

$(function(){

//绑定点击事件

//this 代表的是当前函数的所有者

$("#checkAll").click(function(){

//第一步 获取当前选中状态

alert(this.checked); //值为打勾：为true，不打勾：为false

//第二步 获取所有分类项的checkbox很多种方式

//1、使用选择器[属性名称='属性值']

//这个意思是选中所有属性值为checkbox的标签，过滤掉第一个(点击全选)

$("input[type='checkbox']:gt(0)").prop("checked",this.checked);

//2、使用层级选择器来实现 tbody > tr > td > input

$("tbody > tr > td > input").prop("checked",this.checked);

//3、最简单的使用标签选择器

$("input").prop("checked",this.checked);

});

//4、使用jquery循环操作

$("input[name]").each(function(){

this.checked=ischecked;

})

});

隔行换色

$(function(){

//获得所有的索引为偶数的行 : 使用层次选择器

$("tbody > tr:even").css("background-color","#CCCCCC");

//修改基数行

$("tbody > tr:odd").css("background-color","#FFF38F");

// $("tbody > tr").css("background-color","#FFF38F");

});

# AJAX

Ajax本质上和普通的HTTP请求是一样的，只不过普通的HTTP请求是给人看的，而Ajax请求是给JS代码去用的

struts2获取值栈中的数据

在Struts 2中使用Ajax，Action一般就不会调用一个jsp视图来显示了（如果Ajax请求内容是页面的一部分也可能调用jsp视图），而是通过一些其他的方式。

第一步：导包

struts2-json-plugin-x.x.x.x.jar 在struts-2.3.24\lib可以找到

第二步：在核心配置文件package中extends应该改为json-default，result的type设为json

json-default是已经继承了strtus-default的

<package name="user" namespace="/" extends="json-default">

<action name="verific\_\*" class="verificAction" method="{1}">

<result name="userNameStatus" type="json">/register.jsp</result>

</action>

</package>

第三步 往值栈中放值：

public String verificUserName() {

ValueStack ac = ActionContext.getContext().getValueStack(); //创建值栈对象

List<Users> users=us.serviceUserNsme(userName);

if(users.size()>0) {

ac.set("nameStatus", users.get(0));//往值栈放值

}else {

ac.set("nameStatus", "no");

}

return "userNameStatus";//找第二步中配置的result

}

第四步 只能在js中的ajax才能取值

$.post(

"${pageContext.request.contextPath}/verific\_verificUserName", //url

{userName:n},

function(data){ //只能这里取值，（不使用ajax可以直接网页上标签取值）

alert(data.nameStatus.userName);//这里的值已经转换为json格式

if(data.nameStatus =='no'){

imgs.src="images/biao\_duigouh.jpg";

}else{

imgs.src="images/biao\_X\_h.jpg";

}

},"json"

);});

});

# JSON接收与发送

这次测试学习分离了前端的和服务端的来测试，害的出了很多问题

注意：必须要服务端的地址，不允许客户端的

url:' http://localhost:8080/springMVC/rrr '这样是客户端访问服务器，

url: "/springMVC/rrr ", //服务端内部发送

问题1：造成ajax设置不了头ContentType

问题2：无法获取服务端返回的数据 ：success:function (message)

## 接收服务端

js代码

$(function(){

$.ajax({

//第一步 设置请求类型

type:'post',

//第二步 服务器端的url地址

url:'/springMVC/rrr',

//第三步：返回的数据设置为json类型

dataType:'json',

//第四步：在此方法中为html设置值

success:function (message) {

//函数JSON.stringify(JSON json) 获取json的所有数据

alert("提交成功"+JSON.stringify(message));

//json串获取单个值操作 message.re

$("#ii").val(message.re);

//外边定义函数，这里来调用方法

update(message);

},

//第五步：提交失败(状态码404)执行此函数

error:function (message) {

alert("提交失败"+JSON.stringify(message));

}

});

服务端代码

protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {

String re = req.getParameter("re");

String age = req.getParameter("age");

System.out.println(re+" "+age);

String str="{\"re\":\"主题\",\"age\":\"35\"}";

//设置头ContentType信息：告诉浏览器是json串数据

resp.setContentType("application/json;charset=utf-8");

//发送给客户端

resp.getWriter().write(str);

}

## 发送服务端

js代码的操作

$(function(){

$.ajax({

//第一步 设置提交类型

type: "POST",

//第二步 服务器端的地址：注意：必须是服务器端发送(客户端的url不能设置请求头，contentType会失效)

//http://localhost:8080/springMVC/rrr这样的就是客户端

url: "/LoginServlet",//服务端内部发送

//第三步设置请求类型contentType的mime类型为application/json

contentType: "application/json;charset=utf-8",

//第四步 准备json数据

data:{"name":"yd","pwd":"123456"},

//第五步 设置服务器返回的数据类型

dataType: "json",

//第六步：成功提交返回数据

success:function (message) {

alert("提交成功"+JSON.stringify(message));

},

//第七步：提交失败(状态码404)执行此函数

error:function (message) {

alert("提交失败"+JSON.stringify(message));

}

});

});

java原始servlet的接收操作：

使用此流BufferedReader reader = req.getReader();req.getParameter("age");和有冲突

1、如果req.getParameter("age")先进行取出，req.getReader()就没有数据了：reader.readLine()就等于null

2、反之：先用req.getReader().readLine()取出所有数据req.getParameter("age")就为null；

protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {

//第一步 用req.getReader()方法获取缓冲字符流

BufferedReader reader = req.getReader();

//第二步 就是缓冲字符流中获取数据

String str="";

while((str=reader.readLine()) !=null) {

System.out.println(str);

}

//下面是测试：通过上面获取了数据这下面就获取不了数据了

String string = req.getParameter("age");

String string2 = req.getParameter("re");

System.out.println(string+" "+string2); null null

}