~XML~

~Extensible Markup Language~

~可扩展的标记语言~

1:历史

1998年由w3c推出，专门用来负责数据的封装和传输，和html不同（html主要用来负责数据的

展示，而xml就像一个仓库一样仅仅负责数据封装），xml没有任何容错性，严格区分大小写，xml中标签在dtd的规定下书写，如果没有dtd规定，则可以随意书写，支持中文

2:XML的格式

1)头信息

<?xml version="1.0" encoding="UTF-8"?>

以上为xml的头信息，版本至今是1.0版

encoding:表示本文的编码,如果使用中文则必须使用utf-8或者gbk，头信息必须顶格书写

2)注释

<!---->

3)元素

<标签名>封装的内容</标签名>

注意这里标签名完全依靠dtd的规范，如果未书写dtd则可以随便书写标签名，支持中文

4)属性

<标签名 属性名="属性值"></标签名>

5)特殊字符

&gt; &lt; &nbsp; &quot; &apos;

6)CDATA格式

直接在标签中书写文本，换行、空格、特殊字符都不能被正确的

解析，如果封装数据较为精确，例如sql语句等等，一般使用CDATA格式保存文本数据，在CDATA格式中，空格换行特殊字符都可以被浏览器正确识别

3:DTD (Document Type Definition 文档类型定义)

XML的语法规范

1)内部DTD

直接将dtd文件书写在xml文件中

（html文件也使用的是内部dtd）

A:定义元素结构

内部dtd来约束页面的元素名

如果违反dtd的约束，则xml文件解析错误

<!DOCTYPE 根元素[

<!ELEMENT 根元素 (一级子元素\*)>

<!ELEMENT 一级子元素 (二级子元素)>

<!ELEMENT 二级子元素 (元素类型)>

]>

\*:可以出现任意多次

,:用来隔离元素，注意元素出现的顺序，必须符合

逗号的隔离顺序，如果元素只有元素名，则此元素

只出现一次

(元素1|元素2):枚举类型，元素1和2只能出现一个

+:至少出现1次上不封顶

?:出现0次或者1次

元素类型：

(#PCDATA):表示任意字符串

ANY:可以是任意字符串，也可以再次书写子元素

也可以同时出现

EMPTY:表示元素内部只能为空

B:定义元素属性

dtd来约束元素的属性

<!ATTLIST 依附元素 属性名 属性值类型 默认值>

属性值类型:

ID:全文唯一，不能以数字开头

CDATA:属性值是任意字符串

(10k|20k|30k):任选其一

默认值:

#REQUIRED:必须书写不能不写

#IMPLIED:爱写不写

#FIXED:固定值，一般后面紧跟固定的值

具体数值:默认值

2)外部DTD

A:本地方式

<!DOCTYPE 根元素 SYSTEM "独立dtd文件路径">

B:网络资源方式

<!DOCTYPE 根元素 PUBLIC "别名" "引入的网络dtd路经地址">

==========================================================================================

~Html~

~Hyper Text Markup Language~

~超文本标记语言~

文本：仅仅指代页面上的普通字符串

超文本：不仅仅囊括了文本，还包括页面上所有的要素，例如标签、音频、视频等。

一切页面的基础~

1993年，html超文本标记语言由W3C推出了1.0版本，奠定了现在网页

最初的结构和样式规范，最初的1.0版本较为混乱，经过之后不断的更

新，现在已经发展为html5.1（2016年10月发布），目前页面应用

最多的为html4.01和html5版本

超文本标记语言使用标签(tag)来渲染页面的结构和样式(注意样式已经被CSS技术取代)，页面上一切要素(超文本)都由标签来渲染

标签的种类：

<标签名>

1:开闭合标签

由一个开标签和一个闭合标签组成，中间嵌套了要被处理的超文本

<b>要被加粗</b>

<h2>二级标题</h2>

<i>倾斜</i>

2:整合标签

仅仅使用一个标签来决定特定的结构或者样式

<br />

标签会渲染页面上的所有要素，通过浏览器解析html页面从而渲染出特定的结构和样式，html页面不区分大小写，以.html 或者 .htm为后缀，如果我们遵循了html的规范，则浏览器可以正确解析

href:表示链接到的目的地

target:表示打开另外一个页面来展示目的地页面

img:引入一张视图，一般仅设置图片的宽度即可

src:source的简写表示源，也就是图片从哪里被引入进来

路径问题:src="etoak.jpg"

src="文件名"

src="目录1/文件名"

src="目录1/目录2/文件名"

../:表示本文件跳出当前目录

src="../目录1/文件名"

<img src="etoak.jpg" width="200px" />

<img src="image/etoak2.jpg" width="200px" />

<img src="image/image2/etoak3.jpg" width="200px" />

3:列表

ul:无序列表

<ul>

<li><a href="#">列表1</a></li>

</ul>

ol:有序列表

<ol>

<li>列表1</li>

</ol>

3:表格

table:表示表格的开始

tr:表示一行

td:表示一列

注意以上三者缺一不可

caption:表示表格标题 不是必须

width:宽度

height:高度

border:设置边框的粗细

cellspacing:设置表格单元格的边框和表格的边框之间的距离，若设置为0则两者重合

cellpadding:设置表格内的内容和单元格边框之间的距离

align:水平对齐

<td align="left">列1</td>

<td align="right">列2</td>

<td align="center">列3</td>

valign:垂直对齐

<td valign="top">列1</td>

<td valign="middle">列2</td>

<td valign="bottom">列3</td>

表格中一行有几列，第二行也要有几列，可以通过合并单元格的方式来改变单元格的个数

<td colspan="3">列1</td>

注意此处两个标签单独使用,没有任何作用，一般配合js使用

<label>我是一个便签</label>

<span>我也是一个便签</span>

转义字符:在html当中，部分字符不能直接书写，必须使用转义字符

空格:&nbsp;

大于号:&gt;

小于号:&lt;

form 表单:在页面中将一部分数据，放置在form标签内，当提交表单时可以将这些数据发送到action属性设置到目的地，数据以键值对的方式发送，表示是JavaEE中web工程的开始

action:通过表单项收集数据最终要发送到的目的地

method:发送数据的方式有get和post两种

1:单行文本输入框

type:text固定写法表示单行文本输入框

name:表示key值，一般可以随意书写注意凡是说随意书写的不包含

1)中文 2)空格 3)- 4)特殊字符，注意$不是特殊字符 5)数字开头

value:一般不书写，用户填入的内容就是value属性

placeholder:h5新加入用来书写提示信息

2:单行文本密码框

type:password

name:同上

value：同上

3:单选框

type:radio

name:随意书写

value:注意单选框一般书写value值，表示选择之后提交的值

checked:默认选中

4:多选框

type:checkbox

爱好:<input type="checkbox" name="hobby" value="basketball" />篮球

<input type="checkbox" name="hobby" value="read" />看书

<input type="checkbox" name="hobby" value="game" checked />游戏

<input type="checkbox" name="hobby" value="study" />学习

5:下拉列表框

<select name="location">

<option value="0">===请选择您的归属地===</option>

<option value="1">德州</option>

<option value="2">哈尔滨</option>

<option value="3">佳木斯</option>

<option value="4">株洲</option>

</select>

6:隐藏域

type:hidden，隐藏域一般是开发人员用来在用户不知情的情况下传递值使用，多用于分页等，用户无法在页面查看隐藏域

7:上传控件，一般用来上传文件，value值不限定字符串

上传文件:<input type="file" name="up" />

8:普通按钮

type:button，点击不能提交按钮，一般用来绑定js使用

9:可点击图片

type:image，注意点击后此控件可以提交表单

<input type="image" src="etoak.jpg" width="100px" />

10:多行文本输入框

name:为key

cols:一行可以书写多少个字符

rows:可以书写几行

11:提交按钮

type:submit

value:按钮上面的文本，注意按钮一般不书写name属性而书写value属性

12:取消按钮

type:reset

target:对应iframe中的name属性，当打开连接时页面覆盖iframe中引入的页面

<li><a href="1.html" target="etoak">列表1</a></li>

iframe:表示引入一个外部页面到本页面使之成为一个页面。

src:引入外部页面的路径

name:对应链接中target

width:引入页面的宽

height:引入页面的高

<iframe src="main.html" name="etoak" width="100%" height="100%" frameborder="0"></iframe>

#表单提交时，get方式和post方式有什么区别

get:速度快，安全性较低，通过浏览器的地址栏进行值的传递，格式:?key1=value1&key2=value2&keyN=valueN，最多传递256个字符,不支持中文，链接传递值默认使用get方式

post:速度慢，安全性高，通过封装进请求的消息体进行值的传递，不通过浏览器地址栏，没有大小限制，一般上传操作肯定使用post

#简述html和xml文件有什么不同？

html不区分大小写，容错性很高，标签后可以没有"/"标记预定义，不可自己指定，通常用来显示数据

=======================================================================================

~CSS~

~Cascading Style Sheet~

~层叠样式表~

1996年由网景公司(netscap)发布，专门用来渲染页面的样式，被誉为世界上最美丽的语言，大大降低了html同时渲染样式和结构的复杂性。css提出的选择器机制被后来的很多前端语言借鉴。

Css技术区分大小写，以.css为后缀可以直接将css代码书写在html页面，也可以书写独立页面。Css的宗旨是将结构和样式解耦，如果都使用html来对结构和样式进行渲染，则会导致以下问题：

冗余代码严重，代码重用性低下，浏览器解析缓慢

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Css格式：

选择器(从全文中选择要被渲染的元素){

属性1:属性值1;

属性2:属性值2;

属性3:属性值3;

}

孟诚{

年龄:22;

家乡:青岛;

性别:男;

爱好:滑板 女;

}

引入Css的三种方式:

1:内嵌式(写在head中)

直接将Css代码书写在head标签内，type="MIME类型"，所谓MIME类型是指用来提示浏览器内部是使用的何种技术，浏览器会根据设置的技术来对标签内的内容进行解析

内嵌式初步的将结构和样式解耦，但是并没有解耦彻底，并且这种方式代码重用性较低

2:外链式

<link rel="stylesheet" type="text/css" href="css/style.css">

引入一个外部独立的css文件到本页面

rel:固定写法为stylesheet，表示引入的是一个样式表

type:MIME格式

href:独立的css文件路径，将结构和样式完全解耦，代码重用性较高

3:行内式

直接将css样式书写在标签内，这种方式严重违背css将解耦和样式解耦的原则

当多种css引入方式出现冲突时优先级问题：

行内式>内嵌式和外链式谁放在后面，出现冲突时后面书写的会覆盖之前书写的出现冲突的样式

基本选择器：

1:标记选择器;

tagName{

直接使用标签名作为选择元素的依据，这种选择器一般极少单独使用，非常容易引起误选情况发生

}

2:类别选择器;

.className{

在标签中添加一个class属性，直接使用.属性名的形式拿取指定的元素

}

3:id选择器;

#idName{

直接使用#id属性名的方式拿取元素

}

#Css层叠特性：

当多种基本选择器出现冲突时优先级问题：

id选择器>类别选择器>标记选择器

不受顺序影响，但是如果存在行内式，则一切以行内式为准

#CSS继承特性

在没有任何冲突的前提下，子元素会完全继承父元素的所有css渲染效果

复合选择器：

1:交集选择器;

tagName#idName{

}

tagName.className{

}

由一个标记后面紧跟一个类别或者一个id，则必须同时满足两个条件才可以被成功选取

2:并集选择器;

sel1,sel2,selN{

}

多种基本或者复合选择器用逗号隔开，只要符合其中任意一个就可以被成功选取，没有个数限制

3:后代选择器;

sel1 sel2 selN{

1 2 N 必须具有祖先关系 左祖先 右后代

}

根据左祖先右后代的原则，用空格将基本或者复合选择器隔开，则可以精确选取具有特定祖先元素的子元素，没有个数限制

4:全选选择器;

\*{

拿取全部元素

}

页面元素类型：

在css中将元素分为很多类别，其中主要有如下三种类型的元素：

1:内联元素（inline）

label span img a

以上几种元素，元素的开始和结尾没有换行，一行可以存在多个，设置此元素的盒子模型以及对齐方式无效。

（img元素属于内联元素，但是具有块元素的特性，它是比较特殊的元素）

2:块元素（block）

div p h1~h6 ul li

以上元素，元素的开始和结尾自带换行，一行不能存在第二个元素，设置盒子模型有效，设置对齐方式有效

3:空元素（empty）

这种元素一般进行一些参数的设置以及结构的渲染

<br /> <meta /> <hr />

通过display属性可以切换元素的类型

在css页面中的元素都存在一个边框，在默认情况下这个边框是透明的没有显示，我们可以通过border属性来显示元素的边框

盒子模型：

在css中所有元素都像一个盒子一样从上到下摆放在页面上，通过设置一些参数可以更改盒子四周的距离，间隔等，这些参数被称之为盒子模型。

边框:边框类型 边框粗细 边框颜色;

类型:solid单实线

double双实线

dotted点状线;

浮动:float:改变元素的排列顺序使之重新进行排列

left从左到右 right从右到左;

标准文本流：

是指元素按照元素本身的默认的类型，在没有任何css样式的改动下，默认从上到下排列的一种状态

定位：

相对定位：position:relative;

根据元素原先所在位置的左上角进行定位，定位之后原来的位置依然被占用，不改变元素的类型。

偏移量 top 和 bottom只能书写一个，left和right只能书写一个

绝对定位: position:absolute;

元素根据距离它最近的定位过的父元素的左上角进行定位，定位之后元素不再占满一行，原来的位置不再保留;注意如果祖先没有任何一个定位过，则根据body也就是浏览器的左上角进行定位

============================================================================================

~JS~

~Javascript~

最早由numbas公司发布，原名C--，后更名为livescript

在与微软争夺桌面市场的大战中获得胜利，之后被SUN公司收购，最终定名为JavaScript，是一门运行在本地客户端的解释型的弱语言，与面向对象的Java语言就如同雷锋和雷锋塔一样基本没有任何关系，JavaScript同样是面向对象的（本质是面向过程），区分大小写，独立的文件以.js

结尾，js的安全性较低，曾经一度发展缓慢，但是随着js的众多类库不断的发布，目前几乎可以说没有任何一个网页不使用js技术了

js的常见类库有:ajax jquery Protype dojo Extjs EasyUI Angular VUE 等等

Js语法:

基本数据类型：

Java:

short byte int long double float char boolean

Js:

String number booleannull undefined

复合数据类型

Array Object

赋值:

Java: int i;

String str;

Js: var i = true;

在js中没有默认值的概念，必须事先赋值

标识符统统使用var

流程控制和Java基本一致，循环和Java基本一致

js的运行过程：

书写在页面上，或者引入一个独立的.js文件到页面，在页面的某些元素上绑定激发事件，这些事件触发后会执行绑定在上面函数，从而运行函数内的脚本

js基本：

BOM:全称Browse Object Model

浏览器对象模型，对浏览器打开的窗口分别定义七个对象，用来封装窗口的信息

window:表示窗口

document:表示页面的正文

history:表示浏览器历史记录以及缓存等信息

location:表示浏览器地址栏信息

frame:表示页面的一种框架

screen:表示打开当前窗口的显示信息

navigator:表示当前使用的浏览器信息

onload:激发事件之一，表示整个页面载入成功，等号后面为函数名

DOM：全称Document Object Model

文档对象模型简称document浏览器从上向下解析，当全文解析无误之后会根据结构和样式生成当前实际页面的一个模型，浏览器从上到下解析页面，如果解析无误，则会生成一个dom模型，全称为DOM文档对象模型，JavaScript技术不能对html进行操作，只能对这个模型进行增删查改，修改之后模型和现实页面发生了偏差，则浏览器重新刷新页面使页面和模型保持一致，dom模型起到了一个中间件的功能，并不是js直接对页面的html进行了修改

DOM文档对象模型是js和html的一个桥梁，当我们去掉window.onload语句时，脚本运行到document.getElementById()时还没有创建好模型，无法根据id从模型中拿取元素，所以js脚本无法执行

document:

getElementById();

根据id拿取一个元素

createElement()

创建一个元素

appendChild()

给指定元素添加子元素，原先的子元素不受影响

innerHTML在指定的开闭合元素中间添加超文本

引入js的几种其他方式：

1：书写在head标签中

2：书写在body里面

3：引入外部.js文件<script src="script/myjs1.js"></script>

onfocus:表示获得焦点，函数内部的值叫做实参

onblur:失去焦点

onmouseover:鼠标滑过

onmouseout:鼠标滑出

在js中可以通过style属性使用js脚本来修改元素的css样式

对象.style.css属性名 = css属性值

document.getElementsByTagName("标签名")根据标签名拿取多个元素，返回值是一个数组

onsubmit:表示表单在提交的时候return后面的函数如果返回true，则表单可以提交，否则点击无效

onkeyup:键盘键位抬起

onkeydown:键盘键位落下

oncopy:return false;禁止复制

oncut:禁止剪切

onpaste:禁止黏贴

对象字面量：

在js中存在和java中类似的对象概念，通过对象字面量和构造方法可以创建js中的对象，在这些对象中存在属性通过对象.属性名可以拿取属性值

1：通过对象字面量来创建一个属性值固定的对象

var 对象名 = {属性名1:属性值1,属性名2:属性值2,属性名3:function(){

这里表示特殊的属性值，因为

这个属性值是一个函数，那么这种特殊的属性我们称之为方法

}}

Js中创建数组的方法

1：var ar1 = new Array();

ar1[0]= \*\*\*

ar1[1]= \*\*\*

\*\*\*

2: var ar2 = new Array(数组长度)

3：var ar3 = new Array([第0个元素],[第1个元素],[第2个元素\*\*\*]);

var a = new Array(["b"], [2], ["a"], [4]);

4：var ar4 = ["b", 2, "a", 4];

创建数组并赋值的简写

=====================================================================================

~DOM4j~

是一种使用Java代码来解析xml文件或者创建xml文件的技术，dom4j并不是Java内置的解析xml文件的解决途径，但是因为其易用性已经完全取代了java内置的解析xml文件的方案，成为标准的JavaAPI

功能：导入dom4j的jar包，使用Java语言

1:解析已经存在.xml中的数据（拆包裹）

2:创建一个.xml文件用来封装数据（打包裹）

API:Application Programming Interface：应用程序编程接口，是由官方提供的一系列函数，方法，接口等，使用这些函数 方法 接口等工具可以完成我们自己的需求

1:拿取根元素

getRootElement()

拿取元素名

getName()

2:拿取子元素

List elements()

3:拿取属性

List attrbutes()

4:拿取属性名

getName()

拿取属性值

getValue()

5:拿取嵌套的文本

getText()

使用DOM4j技术通过Java来读取xml文件中的数据：

1：拿取解析器 SAX:Sample Api Xml

Sample API Xml类似一台扫描仪，用来扫描线程xml文件，是用来解析xml的API

SAXReader sax = new SAXReader();

2：使用拿取的解析器解析xml文件，拿取它的文档对象模型

Document doc = sax.read(new File("etoak.xml"));

3：拿取根元素

Element root = doc.getRootElement();

4:拿取多个一级子元素

List<Element> firstChildList = root.elements();

5:遍历多个一级子元素组成的集合，拿取每一个一级子元素的多个属性

List<Attribute> firstChildAttr = firstEle.attributes();

6:拿取二级子元素

List<Element> secondEleList = firstChildEle.elements();

7:拿去二级元素内部嵌套的值

使用DOM4j技术通过Java来把数据写入xml中：

1:创建一个文档对象模型

Document doc = DocumentHelper.createDocument();

2:创建根元素

Element root = doc.addElement("students");

3:创建一个一级子元素

Element firstElement = root.addElement("student");

4:给一级子元素添加属性值

firstElement.addAttribute("id", "001");

5:给一级子元素添加二级子元素

Element email = firstElement.addElement("email");

给子元素添加内部文本

email.setText("et01@etoak.com");

6：设置输出文件的目录，全名

OutputStream os = new FileOutputStream("etoak2.xml");

7：设置xml文件输出格式为标准格式

OutputFormat format = OutputFormat.createPrettyPrint();

8：设置格式的编码

format.setEncoding("utf-8");

9：根据以上输出目的地和格式输出xml文件传递两个参数目的地和格式

XMLWriter xw = new XMLWriter(os,format);

10：将组装好的模型输出

xw.write(doc);

============================================================================================

~JDBC~

~Java DataBase Connectivity~

JavaEE十三种标准技术规范之一

为不同品牌的数据库提供了一套完全一致的Java解决方案使用相同的Java代码，就可以对多种品牌的数据库进行增删查改等一系列操作。JDBC从ODBC发展而来，是最早的秉承ORM思想的数据库操作语言，

（ORM Object Relation Mapping 对象 映射 关系），JDBC是全手动ORM技术，所有的SQL语句需要我们进行书写之后在其基础上诞生了半自动的MyIbatis，以及全自动的Hibernate

JDBC是内置的Java技术，所有的JDBCAPI放置在java.sql包中

使用JDBC的步骤：

1):创建关系表

通过开发工具建立链接创建关系表，一般放置在com.公司名.sql包中，通常将sql语句放置在.sql文件中

2):创建与关系表对应的实体类

a:创建实体类注意使用包装类

b:创建实体类必须存在空参的构造方法

c:创建实体类一般覆盖toString

方法一般放置在com.公司名.po包或者com.公司名.bean包，尽量使用包装类，防止歧义，必须书写空参构造方法，酌情书写带参数的构造方法，生成getter和setter方法，自定义数据类型为了方便打印，则一般覆盖toString方法

3):建立链接

通过各个数据库厂商提供的驱动使Java程序可以识别各个品牌的数据库，并且建立链接

不同品牌数据库提供了不同的驱动程序，可以使用Java代码使用相同的代码来操纵数据库

4):获取链接

设置三个参数数据库相应品牌的地址，用户名和密码三个参数设置好之后可以使用Java代码来链接相应的数据库

5):拿取执行器

执行器用来使用Java代码来执行sql语句，调用相应的方法，可以对数据库进行增删查改操作

#JDBC当中execute() executeUpdate() executeQuery()的具体使用方式？

boolean execute()

\* 执行DQL语句，如果返回结果集则返回true，反之返回false

\* 执行DML语句，不管是否可以成功执行都返回false

int executeUpdate()

\* 执行DML语句，返回的数字，是变更的记录数，执行DQL语句立刻报错

ResultSet executeQuery()

\* 专门用来执行DQL语句，返回一个结果集,结果通过ResultSet来进行封装

\* 注意不能通过判断ResultSet是否为Null来判断是否存在返回数据

我们不能通过rs是否为null来判断是否存在有效数据，因为rs类似一个封装数据的表格，表格的第一行是表头，里面封装了所有的字段名，所以表格永远存在第一行，rs永远不为null，可以通过.next()方法来判断是否存在有效数据，指针默认指向表格的第一行，当执行next()时，如果返回true，则指针向下移动一行，如果返回false则无法移动，可以通过.next()来判断是否存在有效数据

executeBatch() 批处理，返回更改的记录数

#PreparedStatement 与 Statement的使用区别？

1)Statement执行时，执行sql语句和服务器就会有交互，一次执行一条sql语句，PreparedStatement存在强大缓存区域，可以对多条sql语句进行预编译，同样的sql语句与数据库仅仅只有一次交互

2)PreparedStatement可以使用?作为占位符，不再需要对sql语句进行拼接，使用占位符可以杜绝sql注入安全隐患，并且使开发更加简便，代码更加条理

3)如果使用占位符过多，Statement的占用资源与数据库的交互次数较PreparedStatement少

\*注意：Date类型不能直接进行拼接，需要转换成String类型才能进行拼接

\*注意：java.util.Date转换成java.sql.Date的方法，通过getTime()方法可以将其转换，java.util.Date是java.sql.Date的父类 java.sql.Date(per.getBirth().getTime())

#如何使用两种方式来拆开封装的ResultSet中的数据:

1:通过get数据类型(列数)

2:通过get数据类型(列名)

============================================================================================

~Servlet~

~Server+Applet~

~一个运行在服务器端的Java小程序~

JavaSE

JavaEE

JavaME ~~> Andriod

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

CS开发模式和BS开发模式

CS：Client~Server 客户端服务器模式

用户事先通过网路或者光碟等媒介安装一个客户端到本地，通过使用客户端提供的功能完成自己的需求，我们平时运行的各种大型游戏，office、qq、迅雷等常见app都是CS架构，这种架构开发出来的程序功能强大，能够充分发挥用户计算机的能力，但是前期后期维护困难，成本较高。CS架构并不是Java擅长的领域

BS：Browse~Server 浏览器服务器模式

用户本地的浏览器就是一个客户端，不再需要用户安装或者进行任何的升级维护，运行商提供一个地址，用户通过输入地址打开相应的网页，输入提供安全认证证书获得相应的权限，基本与我们上网浏览网站类似，这种模式专业性较强，基本不需要进行本地的维护，所有的维护工作全部放置在服务器端，但是这种模式对于网络依赖性较高，功能较为单一。随着网络的不断发展BS架构的缺点渐渐被掩盖，这种开发模式也是Java擅长的，因为

Java的跨平台性，在很多特殊领域，Java占据绝对主导地位

一个普通类如果继承了HttpServlet抽象类，那么我们就称这个类为Servlet类

此方法为Servlet专门用来处理业务逻辑的方法

HttpServletRequest:接口，封装了所有提交过来的请求

request.getParameter(String);

request.getParameterValues(String);

request.getMethod();

request.getRealPath("/");

request.getContextPath();

request.setCharacterEncoding(String);

request.setAttribute(String,Object);

request.getAttribute(String);

HttpServletResponse:接口，封装了所有要发送走的响应

response.setContentType("");

response.getWriter();

ServletConfig：封装某个Servlet的私有参数设置等信息

config.getServletName();

config.getInitParameter(String);

ServletContext：封装了整个工程中所有Servlet的设置信息

application.getContextPath();

application.getRealPath("/");

application.setAttribute(String,Object);

application.getAttribute(String);

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

喜闻乐见的三种错误代码：

404

1:404后面跟随一个url地址

表示根据开发者提供的路径没有找到资源，路径错误，例如action提交之后，

与web.xml的url-pattern节点不对应

2:404后面没有任何路径

tomcat开启失败

开启失败原因

a:tomcat文件缺失

b:部署的工程web.xml存在错误

c:已经开启了一个tomcat了

405

1:post请求提交doGet接受

get请求提交doPost接受

2:表单的method属性书写错误

500

编译失败

1:没有找到web.xml中设置的servlet实例的地址

2:Servlet编译失败

#Servlet的生命周期

Servlet是一个单实例多线程的Java程序（存在并发问题）自始至终只有一个Servlet实例，不会因为请求过多创建多个Servlet

执行顺序：

1:构造方法（1次）~~》init()（仅仅执行一次，一般用来处理一些事先需要准备的参数的读取）

2:service()（根据请求执行多次，如果用户为get，则调用doGet()，如果用户为post，则调用doPost()）~~》doGet()||doPost()（执行多次，专门用来完成业务逻辑）

3:destory()（执行一次，在关闭tomcat之前执行）

#Servlet中四种范围的局限性是怎样的?

范围有效是指，setAttribute(String,Object),跳转之后可以getAttribute(String)取出来

PageContext:跳转立即失效，不能进行值的传递

HttpServletRequest:重定向失效，当进行重定向跳转时，无法传递值，因为重定向不是同一次请求

HttpSession:销毁失效，session失效则无法传值，否则只要session有效则可以一直传递值

ServletContext:只要tomcat不关闭永远有效

#Servlet中两种跳转方式有什么不同?

请求转发:

由浏览器发出请求，servlet转发这个请求到目的地，由request调用，因为是同一次请求所以request范围有效，浏览器地址栏显示的是跳转之前的路径，跳转之前是do\*\*,跳转之后依然是do\*\*

重定向：

Servlet返回响应到浏览器，由浏览器再次发送请求到目的地，

因为不再是同一次请求，所以request范围无效，浏览器地址栏

显示的就是最终跳转的目的地，跳转之前是do\*\*，跳转之后是doGet()

request.sendRedirect("/工程名/servlet/类名");

request.sendRedirect("/工程名/\*\*.html");

#Servlet中两种会话跟踪机制Cookie和HttpSession的区别是?

Cookie:

由浏览器提供的一种会话跟踪机制，最终保存在客户端本地，不支持中文，只能存储字符串类型。Cookie从创建开始设置有效周期，时间开始倒计时，在倒计时结束之前，用户关闭计算机，关闭浏览器等操作都不会对cookie内部的参数造成影响，cookie的安全性能较低，一般不用于存储极其重要的数据。

Cookie cookie = new Cookie(key,value);

key和value都是字符串

HttpSession:

由服务器端的web容器提供的一种会话跟踪机制，安全性较高，支持中文，通过request.getSession()来创建，创建之后有一个默认最大不活动周期，之后开始倒计时，如果倒计时结束之前都没有请求提交，则session销毁，如果有新的请求提交，则session倒计时重置，所以理论上如果用户一直在活动则session永远有效，通过setAttribute来设置参数，参数为Objective

session有一个唯一的身份标识sessionid,一个32位的随机码

#Cookie和HttpSession如何销毁?

Cookie

1:周期到期

2：cookie.setMaxAge(0)立刻销毁

3：cookie.setMaxAge(负数)关闭浏览器失效

HttpSession

1：超过最大不活动周期，没有请求提交

2：session.invalidate()立刻销毁当前的session

3：关闭浏览器（关闭浏览器其实并不是销毁了session，而是当我们重新打开浏览器时，会再次给我们一个session，而原先的session无法拿取了）

#HttpSession默认是依靠Cookie来维持的？如果Cookie禁用呢？

HttpSession默认是使用cookie来维持的，当我们使用浏览器禁用cookie功能时，HttpSession无法正常工作。

Cookie cookie = new Cookie("jsessionid","46464465464654646465");

HttpSession会话跟踪机制是依靠cookie来维持的，当用户书写request.getSession()时,首先检查当前cookie中是否存在sessionid，如果没有则创建一个新的session，并将一个sessionid绑定给当前session，跳转到另一个页面后，当运行request.getSession()时，从cookie中拿取sessionid，根据这个id拿取session，因为id和跳转前一致，所以session是同一个session，封装的数据可以成功拿取

#如果Cookie被禁用HttpSession还能使用其它方式来维持吗？

使用重写URL的方式

String path = "/ServletDay3\_cookie\_session/servlet/TestSession2";

重写url，通过浏览器的地址栏来传输sessionid

String newPath = response.encodeURL(path);

response.sendRedirect(newPath);

============================================================================================

jsp基础:

<%--这是jsp页面的注释方式--%>

1:带!有;

生成的代码是全局级别的声明,在service方法外 <%! String str1 = "我是声明的参数"; %>

2:不带!有;

生成的代码是局部的变量，在service方法内 <% String str2 = "我是局部变量"; %>

3:带= 没有;

表达式的输出方式，直接输出值，生成的源码也在service方法内

include指令元素：

使用include指定元素 file:表示引入到本页面的辅页

<%@ include file="data2.jsp" %>

page指令元素：

<%@ page

language="java"

import="java.util.\*"

pageEncoding="UTF-8"

contentType="text/html; charset=utf-8"

session="true"

buffer="8kb"

info="thisisinfo"

isELIgnored="false"

isThreadSafe="true"

errorPage="error.jsp"

isErrorPage="false"

autoFlush="true"

%>

language:表示本页面使用的脚本语言，一般是java

注意很多的书籍将jsp页面书写的java代码称之为脚本，这里不要和js搞混

import:页面脚本导包，多个包用逗号隔开

pageEncoding:jsp转换为servlet的编码，注意默认iso-8859-1不支持中文

contentType:软编码的一部分，一般在指令元素外再添加request.setCharacterEncoding("编码")这两句话就是Servlet中的软编码，在jsp中不管get和post都支持中文了

session:默认为true，表示支持session会话跟踪机制如果设置为false，则打开jsp页面时不自动创建当前的session

buffer:页面的缓存，默认8kb，这些内容会保存在用户本地，提高浏览速度

info:嵌套在jsp中的一段信息，一般保存用户的签名等，通过getServletInfo()取出

isELIgnored:是否忽略EL表达式，默认不忽略

isThreadSafe:当前线程是否安全，默认true，单实例多线程，不安全！

更改为false，单线程模式，线程安全，底层实现了SingleThreadModel接口效率低下

errorPage:当前页面出现异常时自动跳转到哪个页面，一般用来设置错误友好化

isErrorPage:当前页面是否可以使用exception内置对象来显示异常，true可以使用false无法使用

autoFlush:当前页面是否自动实时刷新，默认不支持，不读取缓存，设置刷新时间还可以自动根据时间间隔刷新

EL表达式(Express Language)：

表达式:${要输出的值}

所谓表达式就是指只能用来显示数据，不能进行复杂的业务逻辑的一种显示方式，EL表达式就是其中的一种

使用EL表达式拿取范围属性：

${范围.key值}

注意：范围.可以省略，.后面肯定是key值

范围：pageScope requestScope sessionScope applicationScope

如果不指定范围，则默认拿取现存范围最小的

接受客户端提交过来的值：

${param.key}

注意：这里的param.不能省略，param就是parameter的简写

拿取自定义数据类型中的属性值：

${范围.key.属性名}

空验证:

${empty 范围.key}

取不到值返回true，取到值返回false

注意：EL表达式并不是任何值都能直接显示，必须根据提供的范围和key值从范围中拿取，

如果无法拿取值，则什么都不输出

JSTL(Jsp Standard Tag Lib)jsp第二代标签：

1:范围赋值

var:key值

value:表示value值

scope:范围

page request session application

<c:set var="elena" value="elenaPage" scope="page"></c:set>

2:取值

<c:out value="直接输出值" default="无法取值"></c:out>

default:表示value如果取不到，默认输出default的值

3:删除值

var:要删除的key

scope:要删除的范围

注意这里如果不指定删除的范围，则默认删除全部符合的

<c:out value="${requestScope.elena}" default="删除成功！！！"></c:out>

4:遍历

items:表示循环体，可以从Java脚本或者EL表达式中拿取

var:表示每次遍历时的key值，没有固定写法，默认为pageContext范围不能更改

begin:开始遍历的索引值

end:结束遍历的索引值

step:步长，以几条记录为单位进行遍历

<c:forEach items="${sessionScope.mylist}" var="etoak" step="2">

5:流程控制

if属性如果为true则进入内部代码

<c:if test="${empty pageScope.徐伟 }">

<c:if test="${3>1}">

<c:out value="输出这里了~~~"></c:out>

</c:if>

</c:if>

if-else:

c:choose必须和c:when在一起，不能单独书写c:when也不能单独书写c:otherwise

<c:choose>

<c:when test="${3>10}">

<c:out value="输出这里啦~~"></c:out>

</c:when>

<c:otherwise>

<c:out value="最终输出这里~~~"></c:out>

</c:otherwise>

</c:choose>

6:重定向

<c:redirect url="/show.jsp"></c:redirect>

7:引入外部页面

<c:import url="http://www.baidu.com"></c:import>

jsp内建动作(jsp中第一代标签，已经内置在jsp页面中):

1:"新建"一个对象

id:表示新建对象的key值，注意这里仅仅是一个String类型的key值，但是这个key值肯定和新建的对象名重名，也就是说看到了这个id属性就知道新建对象的叫什么了

class:新建对象的类

scope:新建对象的范围

四个范围：page request session application

以下标签类似

Student stu = new Student();

request.setAttribute("stu",stu);

<jsp:useBean id="stu" class="com.etoak.po.Student" scope="request"></jsp:useBean>

2:接受提交过来的值并且赋值

name:给哪个key对应的实体类赋值

对应usebean的id属性

property给哪个属性赋值

如果页面提交过来的key值和属性名正好一一对应,则使用\*表示给所有属性赋值

String name = request.getParameter("myname");

String pass = request.getParameter("mypass");

Student stu = (Student).getAttribute("stu");

stu.setMyname(name);

stu.setMypass(pass)

<jsp:setProperty property="\*" name="stu" />

只给单一属性赋值

<jsp:setProperty property="myname" name="stu" />

只给单一属性赋特定值

value:表示赋此值，用户再次输入任何值无效

<jsp:setProperty property="myname" name="stu" value="孟诚" />

当页面提交过来的key和实体类的属性名不对应时，设置param属性来对应页面提交过来的key,param就是parameter的简写

3:拿取属性值：

property:要拿取的属性名

name:对应jsp:useBean中的id属性，同样是一个key值

<jsp:getProperty property="myname" name="stu" />

#Jsp九个内置对象分别有哪些?它们的作用是?

所谓内置对象是指在源码中已经准备好的九个对象，可以直接使用，不需要再次创建

pageContext：

pageContext对象提供了对JSP页面内所有的对象及名字空间的访问，也就是说他可以访问到本页所在的SESSION，也可以取本页面所在的application的某一属性值，他相当于页面中所有功能的集大成者，它的本类名也叫pageContext。

request

客户端的请求信息被封装在request对象中，通过它才能了解到客户的需求，然后做出响应。它是HttpServletRequest类的实例

session

session对象指的是客户端与服务器的一次会话，从客户连到服务器的一个WebApplication开始，直到客户端与服务器断开连接为止。它是HttpSession类的实例.

application

application对象实现了用户间数据的共享，可存放全局变量。它开始于服务器的启动，直到服务器的关闭，在此期间，此对象将一直存在；这样在用户的前后连接或不同用户之间的连接中，可以对此对象的同一属性进行操作；在任何地方对此对象属性的操作，都将影响到其他用户对此的访问。服务器的启动和关闭决定了application对象的生命。它是ServletContext类的实例。

config

config对象是在一个Servlet初始化时，JSP引擎向它传递信息用的，此信息包括Servlet初始化时所要用到的参数（通过属性名和属性值构成）以及服务器的有关信息（通过传递一个ServletContext对象）

response

response对象包含了响应客户请求的有关信息，但在JSP中很少直接用到它。它是HttpServletResponse类的实例。

out

out对象是JspWriter类的实例,是向客户端输出内容常用的对象

page

page对象就是指向当前JSP页面本身，有点象类中的this指针，它是java.lang.Object类的实例

exception

exception对象是一个例外对象，当一个页面在运行过程中发生了例外，就产生这个对象。如果一个JSP页面要应用此对象，就必须把isErrorPage设为true，否则无法编译。他实际上是java.lang.Throwable的对象

#Jsp中指令元素有哪些?

page指令

include指令

taglib指令

注意上传操作必须使用Post提交方式

enctype:这里multipart/form-data，表示上传文件必须使用字节流而不能使用字符流

stl全称是？请简述你所使用过的jstl标签

jstl(jsp standard tag lib)jsp标准标签库

#17:在jsp分页中如何拿取分页四要素

分页四要素

一共有4个参数，当分页时，如果可以拿取，其中四个数据则分页就可以完成

1:总记录数 dao.getCount()

2:每页记录数 自己订

3:总页数 总记录数+每页记录数-1）/每页记录数

4:当前页 默认是1，但是是一个变量

总记录数：根据dao层的dao.getCount()方法得到

每页记录数：根据需求设定

总页数：（总记录数+每页记录数-1）/每页记录数

当前页：初始值设为1，第一次判断隐藏域value值为空，继续向下走，js写一个change()方法传参，点到下一页就+1，然后把当前页面值赋值给隐藏域的value，函数中提交表单再次循环，用request.getParameter()方法，通过隐藏域的key值得到隐藏域的value值，判断如果隐藏域value值不为空，则继续向下走，依次循环。

============================================================================================

~Asynchronized Javascript And Xml~

~Ajax~

~异步的js和xml~

ajax:

1：荷兰甲级联赛冠军阿贾克斯队

2：web2.0时诞生了异步js技术，大大改变互联网用户的体验，由一些老技术组成，诞生于2004年

同步技术:

HttpServletRequest

用户发出请求，必须等待响应的返回，在响应返回之前，用户只能傻傻的等待在浏览器前，无法进行任何操作，当响应返回，整个页面会被刷新。导致不必要的资源损耗

异步技术:

用户发出请求，不需要等待响应的返回，（响应正常情况下应该返回），用户发出请求后可以继续完成自己的其他操作，当响应在未来一段时间返回时，仅仅局部渲染页面，不会导致整个页面的刷新

Ajax使用到的技术：

1:Javascript：核心语法

2:dom4j：解析或者读取xml

3:xml：封装数据

4:dhtml：(css和html等前端技术的统称)

5:json：2006年加入ajax技术体系，取代了xml在数据封装解析领域的统治地位，封装传输大量数据

ajax能够传输的三种数据类型

\* 1：String text/plain

\* 2：xml text/xml

\* 3：json text/plain

给回调函数发送响应，注意这里只能使用print，如果使用println则返回的字符串自动添加\n

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

<script type="text/javascript">

var request;

//A:创建异步请求

function create(){

//直接创建异步请求

request = new XMLHttpRequest();

以上版本为level2版的异步请求，如果用户的浏览器为ie6、7、8，则可能需要使用level1版本的书写方式

if(window.XMLHttpRequest){

request = new XMLHttpRequest();

}else{

request= new ActiveXObject("Microsoft.XMLHttp");

}

}

//B:主函数

function checkName(value){

1:根据浏览器的不同创建异步请求

create();

2:向服务器发送异步请求

request.open(method, url, async, username, password)

a)method:表示发送请求的方式get或者post

b)url:发送请求的目的地，注意如果使用的是get，则可以使用?传递值

例如servlet/类名?key=value

c)async:表示是否使用异步请求，默认为true，更改为false，则使用同步请求，当响应未返回，则浏览器锁死无法进行任何操作

d)username和password：服务器安全策略，有些web容器必须输入正确的用户名和密码才可以接受请求

request.open("post","servlet/Check",true);

3:设置使用字符流，get方式不需要书写此句，get默认使用字符流

request.setRequestHeader("Content-Type","application/x-www-form-urlencoded");

4:声明回调函数，注意这里是声明没有括号，函数名可以随便书写，不能有括号，带上括号为调用

request.onreadystatechange = etoak;

5:设置向服务器传输的值，post在此处传递值，get也必须书写此句，但是填写null

request.send("key="+value);

}

function etoak(){

//保证接受数据完整

if(request.readyState==4){

//保证服务器运转正常

if(request.status==200){

var value = request.responseText;

//根据id拿取label

var dom\_lb = document.getElementById("name\_msg");

var dom\_sub = document.getElementById("sub");

if(value=="bingo"){

dom\_lb.innerHTML="<font color='red'>用户名已经被占用</font>";

dom\_sub.disabled = true;

return;

}

dom\_lb.innerHTML="<font color='green'>用户名可以使用</font>";

dom\_sub.disabled = false;

}

}

}

</script>

==========================================================================================

JSON

Javascript Object Notation

专门用来封装大量数据的一种特殊格式的字符串，使用Json必须首先导包，它是Java的技术

JSON两种格式：

1：{key1:value1,key2:value2,key3:value3,keyN:valueN}

key:必须是字符串

value:八种基本数据类型

\* 数组

\* null

\* json格式1

\* json格式2

2：[value1,value2,value3,valueN]

value:

\* 八种基本数据类型

\* 数组

\* null

\* json格式1

\* json格式2

JSON的封装

1:封装方式1

\* 可以封装任意数据类型，封装之后呈现格式1的json

JSONObject jo = new JSONObject();

jo.put("key1",value1);

2:封装方式2

\* 专门用来封装自定义数据类型和Map，封装之后呈现第一种格式的json

Map<Integer,String> map = new HashMap<Integer,String>();

map.put(1, "王者荣耀");

map.put(2, "皇室战争");

map.put(3, "部落冲突");

map.put(4, "阴阳师");

JSONObject js = JSONObject.fromObject(map);

3：封装方式

\* 专门用来封装数组 Set 和 List，封装之后呈现第二种格式的json

List<String> list = new ArrayList<String>();

list.add("Javascript");

list.add("Java");

list.add("C++");

list.add("C");

list.add("Objective-C");

list.add("Php");

JSONArray ja2 = JSONArray.fromObject(list);

//["Javascript","Java","C++","C","Objective-C","Php"]

System.out.println(ja2);

==========================================================================================

~jQuery~

~Write Less Do More~

2006年由John Resig独立发布。jQuery仅仅需要从网站下载.js文件导入到我们自己的页面中就可以使用，它提供了功能强大的选择器机制，使用代码链书写，基本解决了各个浏览器的差异性。

jQuery是JavaScript众多类库中使用最为广泛的一个。

由John Resig发布，是世界上最早的Javascript类库之一，使用先进的选择器机制，解决了浏览器差异性，支持代码链书写代码，可以说是js众多类库中应用最为广泛。

下载的源码文件命名规范：

jQuery\_版本号.js

版本号：1.x支持大部分浏览器

2.x增强了稳定性添加了部分新特性，但是不支持ie678

在版本号后面如果存在min后缀表示迷你版，在投产阶段使用类似jre，而一般在开发阶段使用不带min后缀的完全版，类似jdk。

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Servlet\_jsp的中文编码问题

Servlet：

post

软编码

response.setContentType("text/html;charset=编码")

request.setCharacterEncoding("编码");

get

软编码

response.setContentType("text/html;charset=编码")

request.setCharacterEncoding("编码");

A):

修改tomcat默认设置

添加useBodyEncodingForURI="true"

B):

硬编码

public void doGet(HttpServletRequest request,HttpServletResponse response)throws Exception{

软编码

String name = request.getParameter("key");

//使用硬编码

String newName = change(name);

}

public String change(String old){

try{

return new String(old.getBytes("iso-8859-1"),"编码");

}catch(Exception ex){

ex.printStackTrace();

return null;

}

}

jsp

page指令元素

contentType="text/html;charset=编码"

再从page指令元素外添加<% request.setCharacterEncoding("编码") %>

jQuery:基本选择器：

#js中的dom元素和jquery中的jquery元素是同一种元素吗？如果不是为什么？

jquery元素是对js中的dom元素一个轻度的封装，一般来说js元素拿到的就是本元素，而jquery元素将其封装进一个数组中

dom\_lb：<label id="mylb">\*\*\*</label>

$jq\_lb：[<label id="mylb">\*\*\*</label>]

jQuery元素只能使用自己的函数和方法，dom元素只能使用自己的函数和方法，两者不能互相彼此使用对方的函数、方法、属性等

dom\_lb.innerHTML = "\*\*"

$jq\_lb.html(\*\*\*)

#两种元素能否互相转换

jquery~~》js

$("#mylb")[0] = dom\_lb

$("#mylb").get(0) = dom\_lb

js~~》jquery

$("#mylb") = $(dom\_lb)

注意：两种元素中的方法 函数 激发事件等不能互相使用

Jquery选择器的书写方式：

$("sel")

基本选择器：

1：$("tagName")

根据标签名拿取元素

2：$(".className")

根据类别名拿取元素

3：$("#idName")

根据id名拿取元素

4：$("tagName.className")

$("tagName#idName")

必须同时满足标记中存在类别或者id

5：$("sel1,sel2,selN")

凡是符合其中任意一个就可以被成功选取，没有个数限制

6：$("sel1 sel2 selN")

根据左祖先右后代的原则拿取子元素，没有个数限制

7：$("\*")

拿取全部元素

8：$("sel1 > sel2")

拿取sel1的子元素sel2，注意仅仅拿取子元素，设置拿取的元素的css样式，.css("样式属性名","样式属性值")，此方法可以调用多次，相当于js中的style属性

9：$("sel1 + sel2")

a:向下选取

b:必须紧邻

c:拿取兄弟

以上三个条件缺一不可

10：$("sel1 ~ sel2")

a:向下选取

b:互为兄弟

基本过滤选择器：

1:$(":first")

拿取某个特定的第一个元素

注意一般冒号之前存在限定条件

2:$(":last")

拿取最后一个元素

3:$(":even")

拿取索引值是偶数的特定元素

4:$(":odd")

拿取索引值是奇数的特定元素

5:$(":eq(index)")

拿取索引值匹配的特定元素

6:$(":gt(index)")

拿取索引值大于index的特定元素

7:$(":lt(index)")

拿取索引值小于index的特定元素

.text():相当于js中的innerText方法，覆盖元素中原先的文本元素，书写新的文本元素

8:$(":header")

拿取所有的标题元素:h1~h6

9:$(":contains(text)")

拿取包含特定文本的元素

10:$(":has(sel)")

拿取包含特定元素的文本,内部的sel是另外一个选择器

11:$(":empty")

拿取不包含任何文本也没有子元素的空元素

.html():要放置在特定元素中的超文本，支持标签，它的前身就是js中的innerHTML

12:$(":not(sel)")

拿取一个不符合内部括号的指定元素

13:$(":hidden")

拿取页面的隐藏元素

注意此选择器谨慎使用因为不同浏览器打开页面后隐藏元素是不同

text()如果括号内不书写任何内容，则表示拿取元素的文本

14:$(":visible")

拿取所有可见元素

15:$("[属性名=属性值]")

拿取含有特定属性名=属性值的元素

16:$("[属性名^=属性值]")

拿取以特定属性值开头的元素

17:$("[属性名$='属性值']")

拿取以属性值结尾的元素

18:$("[属性名\*='属性值']")

拿取包含特定属性值的元素

属性过滤选择器：

1：$("div[id]")

选取存在id属性的div元素

2：$("input[name='myname']")

选取存在name属性是myname的input元素

3：$("input[name!='myname']")

\*\*\*不是myname

4：$("input[name^='news']")

选取存在name属性并且属性开头为news的input元素

5：$("input[name$='news']")

\*\*\*\*属性值结尾为news\*\*\*\*

6：$("input[name\*='man']")

\*\*\*\*属性值中有man的input元素

7：$("input[id][name$='man']")

选取存在id属性并且存在name属性，而且name属性值结尾必须是man的input元素

修改元素属性：

1: css("css样式属性名","css样式属性值");

2: addClass("要添加的class属性名")

removeClass("要删除的class属性名")

toggleClass("存在class属性删除，反之添加")

3: attr("要拿取的属性的属性名");

attr("要修改的属性的属性名","要修改到的属性值")

4: html() innerHTML

拿取元素内部的超文本

text() innerText

拿取元素内部的文本

~~~~~~~~~~~~~~~~~~~~~~~~~

html(要覆盖原先内容的超文本)

text(要覆盖原先内容的文本)

这里主要的区别就是text是不支持标签的

5: A.append(B)

AB A是B的父元素，A原先的子元素不受任何影响，B排列在A的子元素后面

A.appendTo(B)

BA B是A的父元素

其它同上与append相反

6: A.before(B)

BA两者互为兄弟

A.after(B)

AB同上以上变为紧邻元素

7: A.replaceWith(B)

A被B取代

A.replaceAll(B)

B被A取代

8： clone()

复制一个元素，克隆体和本体完全一致，包括样式结构等

9: 遍历：索引值从0开始

sel.each(function(索引值){

});

10：empty()删除元素的子元素及其内容

remove()删除元素本地以及子元素及其内容

事件：

jQuery四大核心函数

1:$(sel)

内部书写选择器，也可以指定第二个参数表示使用此选择器的范围，如果没有第二个参数表示此选择器从全文中拿取

2:$(html)

内部书写html标签，一般配合部分方法使用

3:$(dom元素)

用来转换js元素

$(document.getElementById("idName"))

4:$(document).ready(function(){

ready函数表示当全文结构加载完毕，当存在事件绑定时，则全部的函数书写在此函数内，有点类似java的主方法

});

1:为特定元素绑定多个激发事件

$(sel).bind("事件1 事件2 事件N",function(){

满足任何一个激发事件会执行的脚本

});

2:为特定元素绑定

固定激发事件

$(sel).事件(function(){

});

3:hover()

$(sel).hover(

function(){

鼠标滑过执行此代码

},

function(){

鼠标滑出执行此代码

}

);

4：

$(sel).one("事件1 事件2 事件N",function(){

});

与bind函数类似，但是此函数有效期仅仅激发一次

破坏性操作：

当执行某个方法之后，原来已经拿取的元素结果集和执行某方法之后拿取的元素结果集不一致，则我们称这个方法为破坏性操作

children(""):破坏性操作，拿取特定的子元素，括号内为指定元素名

slideDown():卷帘显示

slideUp():卷帘隐藏

end():返回执行一次破坏性操作之前的元素结果集，如果没有破坏性操作则返回null

siblings():破坏性操作，拿取当前元素的兄弟元素，不包含本元素，如果添加andSelf()则包含本元素

show():显示元素

jQuery使用Ajax：

$.post(url,data,callback)

url:要发送异步请求到的目的地，注意get方式可以使用?传递值

eg:"servlet/CheckName?etoak="+$jq\_input.val()+"&time="+new Date()

data:表示传递的值，get和post都可以从此处传递值

格式:{key1:value1,key2:value2,keyN:valueN}

callback:表示回调函数括号内的对象，表示从服务器返回的数据，回调函数，如果服务器出现问题没有响应

data:表示服务器返回的值，function(data){}

注意jquery的ajax支持以下五种数据类型：

text xml html script json

$.ajax({

使用的提交方式

type:"post",

发送到的目的地,get方式可以从此处传递值

url:"servlet/Check",

返回值类型，支持五种类型：text、script、html、json、xml，注意服务器端的MIME设置作废

dataType:"text",

是否使用异步请求,如果设置false，则不使用，响应返回前浏览器锁死

async:true,

要发送的值

data:"etoak="+$jq\_input.val(),

success:function(value){

服务器无误回调函数

value表示从servlet中接受的值

},

error:function(){

服务器出现问题回调函数

}

});

#js中的onload和jquery中的ready()有什么区别吗？

A: onload必须加载完全部元素才可以激发

ready()仅仅加载完结构即可

B: onload上下文仅仅执行一次，如果书写多次，则只执行最后一次

ready()执行多次从上到下依次执行