# 初识web项目

## 1、tomcat服务器

1、安装tomcat： 在tomcat.apache.org下载解压缩版本，随便解压到一个目录即可

1. 配置tomcat环境变量： 新建一个CATALINA\_HOME值为tomcat文件夹目录

在path环境变量下添加 tomcat里bin文件夹目录

依赖jdk所以，需要把jdk目录(不是bin)弄到JAVA\_HOME里

然后在tomcat bin目录startup.bat打开。在网页里用https://localhost:8080/测试即可。

http://localhost:8080/NEWS/ url组成部分 协议，服务器域名或ip有时端口号，路径

## **1.2、tomcat目录作用**

/bin： 存放各种平台下用于启动和停止tomcat的脚本文件

/conf: 存放Tomcat服务器的各种配置文件

/lib: 存放tomcat服务器所需的各种jar文件

/logs: 存放tomcat的日志文件

/temp: tomcat运行时用于存放临时文件

/webapps: web应用的发布目录

/work: tomcat把由JSP生成的servlet存放于此目录下

## 2、eclipse创建web项目

**1、修改工作空间所有编码格式**

在window-preferences里搜jsp 把编码全部格式改为utf-8

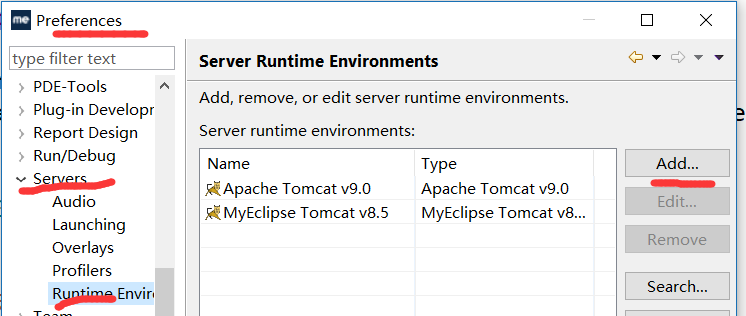
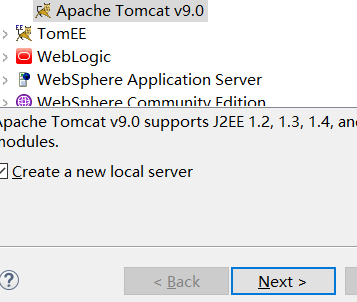
搜content type全utf-8

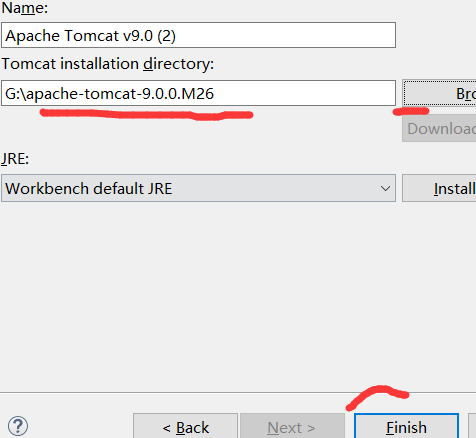
搜eclipse->window->preferences->General->workspaceTypes->Other->UTF-8->OK

Tomcat里conf里有个server.xml里Connector节点可以改端口号 加一个URIEncoding="UTF-8"

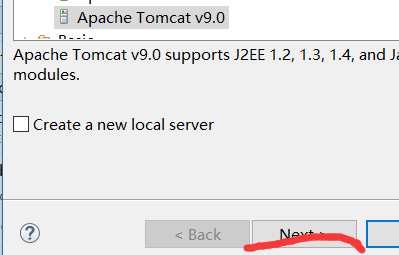
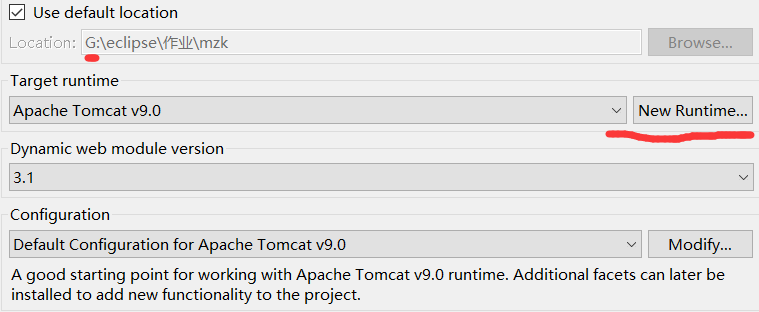
创建jsp页面上方有<%@ page language="java" contentType="text/html; charset=utf-8" pageEncoding="UTF-8"%> 即可

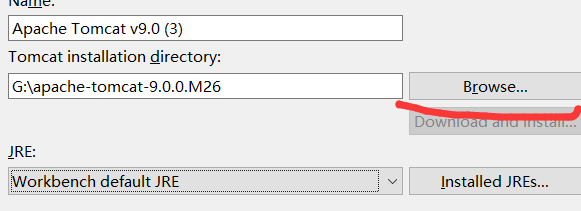
**2、给工作空间添加tomcat(以后只要在这个空间所有项目都采用这个tomcat)**



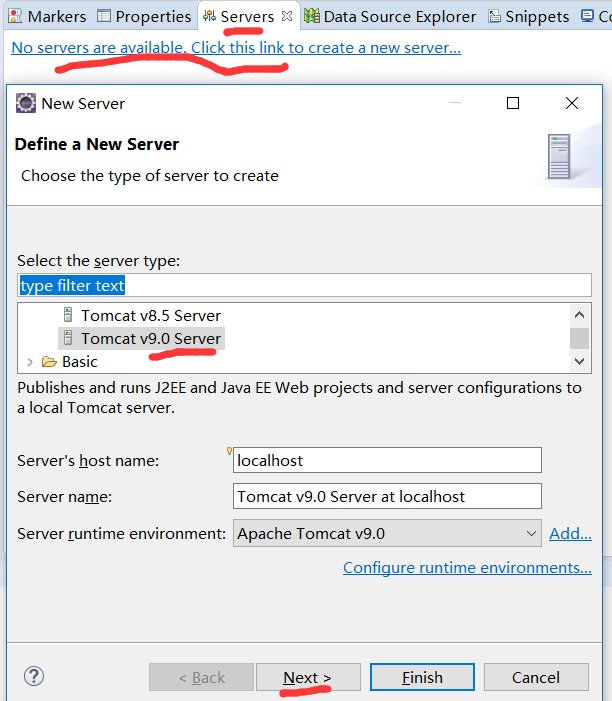
1. **或者创建项目时选择tomcat(只是这个项目用)**





**4、file 新建项目选择Dinamic web project**

**5、把项目添加到tomcat容器**



**6、web项目里各目录作用**

WEB-INF 里的文件不能被执行；存放应用程序所需的各种资源web.xml等

/WEB-INF-lib 存放web应用使用的JAR文件

需要右键项目新建一个名为resource文件夹，在右键java resource新建一个source folder文件夹选择resource文件夹，这个文件夹用来存放静态配置文件。

## **3、jsp简单用法**

嵌套java代码： <% int a=5; int b=4; int c=a+b; %>

输出结果 <%=c %>

输出结果 out.println();

声明方法 <%!public void show(){} %>

## 4、单词

content 内容

preferences 参数选择

General 普通

workspace 工作/作业，空间

Other 其他的

Encoding 编码

server service servlet 服务器，服务，控制器

language 语言

import 进口/输入

Client 客户

Browser 浏览器

# Cs架构与bs架构

c/s:(Client/Server,客户端/服务器端)架构的系统，其分为客户端和服务器两层，用户在本地安装客户端软件，通过网络与服务器端相互通信。

b/s:(Browser/Server,浏览器/服务器)架构下，应用系统完全放在应用服务器上，并通过应用服务器同数据库服务器进行通信，系统界面是通过浏览器来展现的。

# 深入了解webProject

## jsp内置对象

### request内置对象:

request.getParameter("name"); 获取页面提交的数据(返回String)

request.getParameterValues("aihao"); 获取一组以相同名称命名的表单提交的数据(返回String数组)

request.setCharacterEncoding("UTF-8"); 解决中文乱码问题

request.getRequestDispatcher("index.jsp").forward(request, response); 跳转到index并且保留原页面接收的数据

转发跳页面引用的css是根据原网站路径来调用的

getparameter获取的是表单和?后带过来的数据，而getattr只获取setattr的数据

### response内置对象:

response.sendRedirect("index.jsp"); 将请求重新定位到一个新的url上 重定向

<a href="color\_1.jsp?color=绿色&id=1">绿色</a>

1、从数据共享上

Forword是一个请求的延续，可以共享request的数据

Redirect开启一个新的请求，不可以共享request的数据

2、从地址栏

Forword转发地址栏不发生变化，资源路径根据上一个路径来获取

Redirect转发地址栏发生变化

### session内置对象: 存在服务端

String getId() 获取sessionid

void setMaxInactiveInterval(int interval) 设定session的活动时间，单位是秒0永远不结束

int getMaxInactiveInterval() 获取session的有效非活动时间(以秒为单位)

void invalidate() 设置session对象失效

void setAttribute(String key, Object value) 以key/value的形式保存对象值

Object getAttribute(String key) 通过key获取对象值

void removeAttribute(String key) 从session中删除指定名称(key)所对应的对象

会话超时或手动invalidate都会使session失效,失效之后只要刷新或重新进一次这个网址数据就访问不到(除非页面的倒退和前进)

会话超时是指两次请求时间间隔超过了服务器允许的最大时间间隔；在时间内关闭窗口照样session失效

### application内置对象:

void setAttribute(String key,Object value)

Object getAttribute(String key);

### 对象的作用域:

request作用域与客户端的请求绑定在一起，只要一直用转发url不改变数据就能一直用

session数据可以被本次会话中的其他页面所访问

application系统的全局变量，用于一个应用内所有用户之间的数据共享

getAttr和setAttr来存取数据，都是存object类型，get返回object

，

request.getParameter是获取input和？getAttri是获取set的数据

### cookie内置对象：

cookie用于保存一些小数据String在客户端永久保存

Cookie a= new Cookie(String name,String value);

response.addCookie(a)

Cookie[] cookies=request.getCookies();

cookie[i].getName().equals(name)

setMaxAge(int) 参数等于0删除

Cookie c=new Cookie("name","123"); 账号密码存到里面 然后登陆页面如果找到 就把值赋给String变量然后在加到文本框里(简单的记住密码)



### base标签和include

<base href="<%=basePath%>"> 所有的href都是根据项目根路径来找的 而不是根据本页面

<%@ include file="NewFile1.jsp" %> 插入一个文件

## 2、单词

请求 request

Dispatcher 调度器

响应 response

会话 session

Inactive 不活动的

Interval 间隔

invalidate 使无效无价值

include 包含

cookie 饼干

base 基础/底部

# 数据源与连接池

## 1、在tomcat发布一条消息

1、需要在tomcat conf文件夹 或 server里修改context.xml里的<Context>标签

2、加一个 <Environment name="a" value="hello" type="java.lang.String"/>

3、然后再jsp页面javax.naming下的Context ctx=new InitialContext(); 负责检索对象

String ss=(String)ctx.lookup("java:comp/env/a"); 方法根据名称检索制定对象

## 连接池简介

1、负责管理数据库连接，让程序使用现有的连接而不是重新创建一个，

2、连接池机制 创建一定数量的连接放到池里，数量由最小数据库连接数来设定，用户请求时看空闲池有没有，有的话分配给用户，没有则检查是否达到最大连接数，没有达到就新建一个连接，如果达到请求将被加入等待队列中，等待时间有链接释放就给用户用，超过等待时间则返回null

## jndi数据源

1. 需要在tomcat conf文件夹 或 server里修改context.xml里的<Context>标签

<Resource auth="Container" 表示用容器application用web应用管理resource

driverClassName="com.mysql.jdbc.Driver"

maxActive="20" maxIdle="5" maxWait="10000" 最多20个连接 最多5个空闲连接

name="jdbc/news" password="123456"

type="javax.sql.DataSource"

url="jdbc:mysql://localhost:3306/newsmanagersystem?useUnicode=true&amp;characterEncoding=utf-8" username="root"/>

2、在web.xml文件配置

<resource-ref>

<description>news DataScource</description>

<res-ref-name>jdbc/news</res-ref-name>

<res-type>javax.sql.DataSource</res-type>

<res-auth>Container</res-auth>

</resource-ref>

1. java代码连接

Context ctx=new InitialContext();

DataSource ss=(DataSource)ctx.lookup("java:comp/env/jdbc/news");

Connection con=ss.getConnection();

javax.sql.DataSource 获取的数据源 必须在web里用(意思就是不运行tomcat你测试不出来)

## 4、dbcp数据源

1. 导入dbcp三个jar包和mysql 包
2. 再把datasource配置文件导入到src
3. dataSource.getConn

|  |
| --- |
| private static Properties properties=new Properties();  private static DataSource dataSource;    // 加载DBCP配置文件  static {  try {  InputStream is = DBUtil.class.getClassLoader().getResourceAsStream("database\_dbcp.properties");  properties.load(is);  dataSource = BasicDataSourceFactory.createDataSource(properties);  }catch (Exception e) {  // TODO Auto-generated catch block  e.printStackTrace();  }  } |

## 5、单词

Initial 初始

lookup 仰望/查阅

Container 容器

Active 积极/活跃

Idle 闲置的

# 分页与上传

## 分页

根据传过来的页码来用sql limit （行数-1）\*行显数，行显数

Daoimpl 写求新闻总条数方法和分页查询方法 (参数为service计算好的从哪开始 和 行显数)

Serviceimpl 写求总共几页方法和分页查询方法(传入行数和行显数，并计算（行数-1）\*行显数 )

然后页面当前页数默认为1 第一次传处理页默认是第一页 返回首页总页数和当前页数。 第二次传过去当前页数返回相应集合

## 上传

|  |
| --- |
| request.setCharacterEncoding("utf-8");  String fieldName = ""; //表单字段元素的name属性值  //传过来什么字段就声明多少个变量  String name="";  String tupian;  //设置上传缓存区  File temp=new File("C:\\Users\\lenovo\\Desktop\\示例4：文件上传的实现");  if(!temp.exists()){  temp.mkdirs();  }    //请求信息中的内容是否是multipart类型  boolean isMultipart = ServletFileUpload.isMultipartContent(request);  //上传文件的存储路径（项目发布的路径）  String uploadFilePath = request.getServletContext().getRealPath("uploadImages/");  if (isMultipart) {  //upload依赖于factiry才能使用  DiskFileItemFactory factory = new DiskFileItemFactory();  factory.setSizeThreshold(1024\*1024); //设置缓存区大小  factory.setRepository(temp); //设置缓存区路径  //这个类是用来实现文件上传操作的  ServletFileUpload upload = new ServletFileUpload(factory);  upload.setSizeMax(1024\*1024); //设置一个完整请求最大允许大小    try{  //解析form表单提交过来的数据 返回一个fileitem集合  在表单中的每一个元素均对应一个FileItem对象  List<FileItem> items = upload.parseRequest(request);  Iterator<FileItem> iter = items.iterator();  while (iter.hasNext()) {  FileItem item = (FileItem) iter.next();    if (item.isFormField()){ //判断存储的内容是否为普通表单  fieldName = item.getFieldName(); //获取表单字段的name属性值  //如果名字是name的话就赋值给声明的name用来传给方法  if (fieldName.equals("name")){  name = item.getString("UTF-8") ;  }  }else{  String fileName = item.getName(); //获取文件名  String fileName1 = item.getFieldName(); //获取字段名  List<String> fff=Arrays.asList("jpg"); //限制文件格式  //获取传过来文件的后缀  String ext=fileName.substring(fileName.indexOf(".")+1);  //如果包括此后缀才继续上传文件  if(!fff.contains(ext)){  System.out.println("没有此格式");  }else{  File f=new File(fileName); //把文件愿路径传给file  //把发布路径传到saveFile(f.getName=xxx.jpg)  File saveFile = new File(uploadFilePath,f.getName());  //将item中保存的内容保存到这个路径  item.write(saveFile);  //发送到数据库里的路径为tupian  if(fieldName1.equals("tupian")) {  tupian = "uploadImages/"+uploadFileName;  } |

# Servlet

## servlet简介

servlet是一个符合特定规范的java程序。用于在发送的请求和响应之上提供网络服务。在服务器上解释和执行浏览器请求，充当客户端和其他层中间层

就不用jsp页面来接收请求并返回数据了， jsp方便开发表示层组件，servlet适合编写流程控制代码

可以直接右键新建 选择servlet创建

## 2、serlet使用方法

1、创建servlet包 HttpServletTest类继承HttpServlet 实现doGet和doPost方法 把代码都写到get然后post调用get

2、在get方法里写上 处理乱码问题

response.setContentType("text/html;charset=UTF-8");

request.setCharacterEncoding("UTF-8");

3、web.xml配置如下代码

<servlet>

<servlet-name>HttpServletTest</servlet-name>

<servlet-class>mzk.HttpServletTest</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>HttpServletTest</servlet-name>

<url-pattern>/awm</url-pattern>

</servlet-mapping>

容器根据在web.xml文件中定义的url映射查找servlet servlet-mapping中url-pattern到servlet-

name再到servlet里的servlet-name到servlet-class

4、也可以不用再webxml配置，直接采用注解方式

在HttpServletTest类上加 @WebServlet(name="user",urlPatterns="/aa")

Name表示servlet-name，urlPatterns 路径

5、效果就是这样子的



## \*\*\*\*\*\*\*3、servlet生命周期:

1.当容器启动或客户端发送请求，就会找内存中是否有servlet实例,没有则创建一个

2.实例化后将调用init方法进行初始化 加载servlet时执行一次

3.接到请求时调用service方法根据请求调用doget或dopost

4.容器关闭或需回收资源调用destroy方法该方法指定哪些资源可以被系统回收

定义servlet初始化参数 在servlet标签里使用init-param 包括param-name param-value 一个初始化参数名字 一个值

然后直接getInitParameter(param里定义的name) 获取以name为名的初始化参数值 不需要前缀.

在web-app 里用context-param 包括param-name param-value 一个初始化参数名字 一个值

this.getServletContext().getInitParameter(param里定义的name) 获取servlet上下文引用

一个为单个servlet用 一个为该项目的所有servlet用

**Servlet里获取内置对象**

Session和apli需要如下获取，其他如request,response直接在doxxx方法里里面用即可

PrintWriter out=response.getWriter(); 用于向客户端发送文本

HttpSession s=request.getSession();

ServletContext application=this.getServletContext();

HttpServletRequest

# JSTL 和 EL

## EL使用方法

1、[]操作符访问某个属性 ${names.one} ${names["one"]} 第二个方式能包含特殊字符

2、里面可以用java的一些基本关系操作符 ，但必须保证判断两方类型一样, 如果想两个${}做对比 则${a==b}

3、 还有${empty user}用于检测一个对象是否为空，字符串长度为0，集合size为0 为空返回true

4、隐式对象 pageScope requestScope sessionScope applicationScope param paramValues pageContext可以访问jsp八大内置对象，提供对jsp内置对象的访问 ${pageContext.request.contextPath}

如果在request和session都存在一个name为凯凯的，那么就必须加前缀隐式对象${sessionScope .name}

## 2、JSTL使用步骤

1、导入两个jar包

2、jsp页面顶部加 <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>

uri引用某个标签库文件 prefix指定的前缀即可访问该标签库中的某个标签功能如 <c:标签名>

## 3、JSTL标签库

### 1、通用标签库

<c:set var="name1" value="asdsad" scope="page"/> 相当于pageContext.setAttribute("name1","asdsad");

或<c:set target="${user}" property="name" value="马志凯"/> 操作的对象 对应对象属性名 赋予的值

<c:out value="${param.name}" escapeXml="false"/> esca为false则不处理特殊符号 el不能处理特殊符号

<c:remove var="user" scope="request"/> 删除的名字和所属域

### **2、条件标签库**

<c:if test="${empty user}"> 主题内容 </c:if> test里面为true那么就输出主题内容可以用el表达式

<c:choose> <c:when test=""></c:when> <c:otherwise></c:otherwise> </c:choose> when里写测试条件

### **3、迭代标签库**

<c:forEach var="user" items="${list}" varStatus="status"></c:forEach> begin end step 开始结束和间隔

var 为循环到的当前对象 items被迭代的集合 varstat存储var引用相关信息 当前索引 index 总数 count

### 4、日期格式化标签库

<%@ taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt" %> 导入标签

<fmt:formatDate value="<%=new Date() %>" pattern="yyyy-MM-dd"/>

## 4、单词

destroy 破坏/消灭

Scope 范围

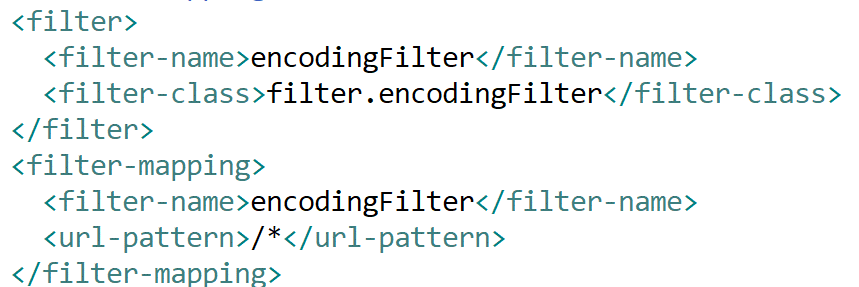
choose 选择

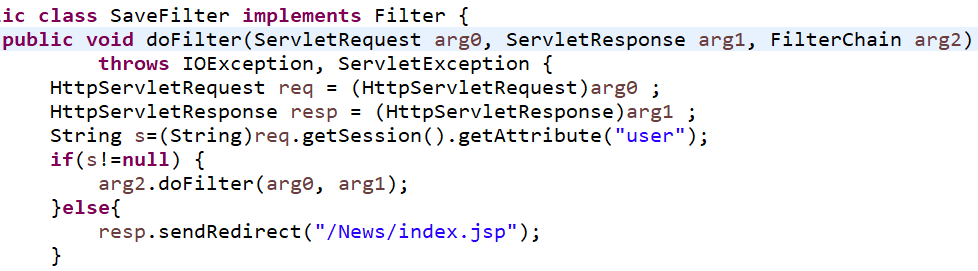
when 当。。时

otherwise 否则

format 格式

# 过滤器

所有路径都要进入encodingFilter过滤器

判断有没有登陆

调用doFilter方法是将请求转发给过滤器链上下一个对象(必须都满足传过来的url地址)，如果没有了就发送给相应servlet；

**编码格式过滤：**

在Filter包里写一个encodingFilter类实现Filter接口在里面写

arg0.setCharacterEncoding("UTF-8");

arg2.doFilter(arg0, arg1);

**登陆过滤：**

url写成/admin/\* 然后dofilter写上面截图的代码

# Ajax

## ajax简单介绍

ajax为javaScript xml等现有的技术整合而成，可以根据返回结果只更新局部页面

1. 用户界面触发事件调用javaScript
2. 通过ajax引擎-xmlHttpRequest对象异步发送请求到服务器，

3、服务器返回json，text等格式数据，

4、然后利用返回的数据使用dom和css技术局部更新用户界面

## 2、原生ajax

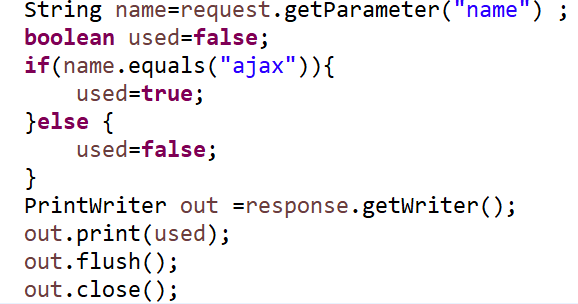


使用原生post方法为上方图片，get方法为：

var url="DemoServlet?name="+name; post传的参数是写到send里

xmlHttpRequest.send(null);

xmlHttpRequest.setRequestHeader("Content-Type","application/x-www-form-urlencoded"); 这一行去掉



## 3、使用jquery实现ajax

### 1、语法：

$.ajax({})

url:String 发送请求的地址，默认为当页地址

type:String 请求post或get，默认get

data:String 发送到服务器的参数

dataType:String 预期服务器返回的数据类型 json，text，html等

html就在servlet里拼好html代码 但是html跟text一样效果

success:Function 请求成功后调用的函数 直接是写好的方法的话不用加括号

error:function 请求失败后调用的函数

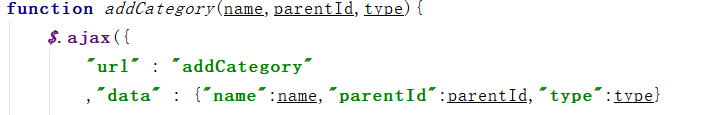
beforeSend:function 发送请求前调用的函数 如果请求较长可以提前显示一条提示信息

complete:function 请求完成后调用的函数 把那个正在等待小东西关闭

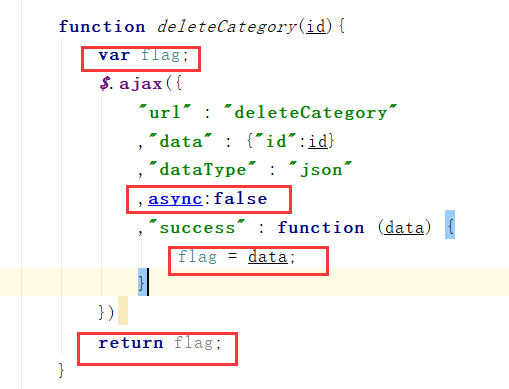
timeout:number 设置请求时间

global:boolean 默认为true，表示是否触发全局Ajax事件





将async设置为false，调整为同步请求，即可在外部获取到ajax数据



## \*\*\*\*\*4、ajax交互扩展

$.get("DemoServlet","name="+name,callBack,[预计返回类型]); 类型默认为text,也可以自己设置

$.ajax({

"url" : "DemoServlet",

"type": "GET",

"data": "name="+name,

"success" : callBack

})

$.post("DemoServlet","name="+name,callBack,[预计返回类型]);

$.getJSON("DemoServlet","name"+name,callBack); 只能以get方式发送请求

$("#nameDiv").load("DemoServlet","name="+name); 该方法不用回调函数，默认用get，除非data为一个对象，则使用post发送 并且他不是全局函数而是针对选择器匹配元素执行

可以在servlet拼html再用out输出，也可以转到一个html页面输出需要插入的html代码，html页面没有其他页面结构

url 可以通过空格来决定加载什么内容 如 load("DemoServlet #a"); 就是只显示#a里的内容

## 5、定义json对象

### **json类型对象**

var user={

"name":"张三",

"age":30,

"pwd":"000"

};

### **json类型数组**

var ary=["中","没","饿"];

### json格式的user对象数组

var userAry=[

{"name":"张三","age":30},

{"name":"张s","age":30},

{"name":"张w","age":30}

];

$sadInputs=$("#sad").find(":input"); 获取所有表单对象

var queryString=$sadInputs.serialize(); 把表单内容转换为String序列化

读取表单内容一定要用button按钮进入ajax方法，submit会直接刷新页面导致跳不到servlet

元素不能被禁用，必须有name属性,选中的checkbok和radio才有效，不能序列化file

String json=JSON.toJSONString(a,true); 将java对象序列化成json对象(传入集合maplist都行)，true生成的事带格式的

Object o = JSON.toJSON(a); 直接返回对象类型

JSON.toJSONStringWithDateFormat(jsonMap,"yyyy-MM-dd"); 里面的date会按这个类型显示

后面可以在加参数SerializerFeature.WriteMapNullValue 会把值为null的也转换为json格式

SerializerFeature.WriteMapNullValue 将实体类获取的值为null的字符串输出为””

SerializerFeature.WriteNullNumberAsZero 将实体类获取的值为null的数值输出为

## 单词

Filter 过滤器

success 成功

error 错误

serialize 序列化

exception 例外

# Jsp内置对象深入

在jsp开发中会频繁使用到一些对象,如ServletContext HttpSession PageContext等.如果每次我们在jsp页面中需要使用这些对象都要自己亲自动手创建就会特别的繁琐.SUN公司因此在设计jsp时,在jsp页面加载完毕之后自动帮开发者创建好了这些对象,开发者只需要使用相应的对象调用相应的方法即可.这些系统创建好的对象就叫做内置对象.

pageContext对象： 提供了对jsp页面所有对象以及命名空间的访问。 PageContext

request对象： 封装了来自客户端、浏览器的各种信息。 HttpServletRequest

response对象： 封装了服务器的响应信息。 HttpServletResponse

session对象： 用来保存会话信息。也就是说，可以实现在同一用户的不同请求之间共享数 HttpSession

application对象： 代表了当前应用程序的上下文。可以在不同的用户之间共享信息。 ServletContext

exception对象： 封装了jsp程序执行过程中发生的异常和错误信息。 Throwable

out对象： 用于向客户端、浏览器输出数据。 JspWriter

config对象： 封装了应用程序的配置信息。 ServletConfig

page对象： 指向了当前jsp程序本身。

# 重点

jsp页面修改后tomcat会自动重新加载  
java文件修改后如果没有设置自动重载是需要重启的

String path = request.getContextPath(); /mzk

String basePath = request.getScheme()+"://"+request.getServerName()+":"+request.getServerPort()+path+"/"; http://localhost:8080/mzk/

String uploadFilePath = request.getServletContext().getRealPath ("uploadImages/" );

G:\eclipse\作业\mzk\.metadata\.plugins\org.eclipse.wst.server.core\tmp0\wtpwebapps\mzk1\uploadImages\ 项目发布的路径,找里面的文件

在控制台server里修改默认发布路径(先delete所有) 因为上传图片在发布路径而不是eclipse路径

<base href="<%=basePath%>"> 只对href有效，对重定向转发无效

<a href="color\_1.jsp?color=绿色&id=1">绿色</a> 1页面转发a到2，2可以这样a=3覆盖原来转发过来的a在转向3

转发跳页面引用的css是根据原网站路径来调用的如果样式不管用的话把引用css的路径都改成绝对/News/admin/assets/css/

response.setContentType("text/html;charset=UTF-8"); servlet里改变格式的和setCharacterEncoding一起 servlet的类要 extends HttpServlet

上传图片 enctype="multipart/form-data"

biancheng\案例，项目\java\案例新闻发布\news\src\BaseDaoimpl 里有分页代码

servlet不响应东西 那么ajax也会执行回调函数

${product.imgUrls[${idx.index}] } ${product.imgUrls[idx.index] } 如果el嵌套el不需要写${}

如果用json data传的数据为null 接收到的为null字符串 而不是null

el表达式只能获取setAttrbute里面的

el表达式没获取到值的话 啥也不输出

HttpServletRequest