JavaEE（十三种技术结合简称）

Tomcat的目录结构

backup：保存了一些配置文件

bin：可执行文件

conf：配置文件

lib：Tomcat运行时需要的jar文件

logs：Tomcat运行产生的日志文件

temp：tomcat运行产生的临时文件

webapps：存放我们编写的应用

work：tomcat工作目录

Serverlet：server+let服务器端的一个应用程序

applet：app+let客户端的一个应用程序

修改了web.xml/servlet文件以后需要重启服务器

http协议中封装了请求和响应

accept（MIME类型）

accept-encoding

accept-language

host

referrer（包含一个URL）

content-type

user-agent（浏览器类型）

content-length

cookie

date

200正常

302/307 临时重定向

在Serverlet里面为了不出现安全隐患，不要写全局变量

servletContext对象

servlet四大域对象

ServletContext（整个web项目）

HttpSession(一次会话)

ServletRequest（一次请求）

PageContext（一个JSP页面）

request对象

请求行、请求头、请求正文

请求行中常用的方法：

getMethod（）获取请求方式

getRequestURL（）获取客户端发送请求时的地址

getRequestURI（）请求行中的资源名部分

getContextPath（）当前应用的虚拟目录

getQueryString（）请求行中的参数

请求头的常用方法

BeanUtils

由于前几种封装对象太麻烦，我们记下来使用BeanUtils来封装对象

注意：如果要自动封装，表单form中的元素name需要与Javabean中的属性一致

String🡪date不支持 需要转换器

域操作非表单数据相关的方法

void setAttribute(String name,Object value);

Object getAttribute(String name);

void removeAttribute(String name);

解决中文乱码

setCharacterEncoding(“utf-8”);（只能用于Post方法）

重定向（地址栏改变、setAttribute等无效）

1. setRedirect（String location）;
2. 先设置响应码302

resp.serStatus(302);

设置响应头 指定跳转到的页面

resp.setHeader(“location”,”demo\_09”);

会话

**客户端技术cookie**

服务器把每个用户的数据以cookie的形式写回给用户各自的浏览器，当用户使用浏览器再去访问服务器中的web资源时，就会带着各自的数据去，这样，web资源处理的就是用户各自的数据了。

登陆成功后的账号密码 保存在cookie里面

**特点：**

1. 一个cookie拥有一个名称，一个值和一些可选属性，比如路径、存活时间等。
2. cookie只能存放servlet发送到web浏览器的少量信息
3. cookie只能存放字符串 数字不能存储数组等多维数据

name：名称 不能确定一个cookie 路径可能不同

value：值

path：路径

maxAge：cookie的缓存时间 默认-1 单位是秒

**服务器端技术session**

JSP（java server pages）

三个指令：page、include、taglib

page

<%@page import=”’%>

errorPage、language、contentType、pageEncoding、isELgnored

include

静态包含<%@ include file=”s.jsp”%>

动态包含<jsp:include page=”s.jsp”></jsp:include>

taglib

调入jstl标签

<%@ taglib url=”” prefix=””%>

六个动作

<jsp:include\forward\param\useBean\setProperty\getProperty>

九个内置对象

|  |  |
| --- | --- |
| 对象名 | 类名 |
| request | javax.servlet.http.HttpServletRequest |
| response | javax.servlet.http.HttpServletResponse |
| session | javax.servlet.http.HttpSession |
| application | javax.servlet.ServletContext |
| exception | java.lang.Throwable |
| page | java.lang.Object |
| config | javax.servlet.ServletConfig |
| out | javax.servlet.jsp.JspWriter |
| pageContext | javax.servlet.jsp.PageContext |

四个域对象

PageContext

ServletRequest

HttpSession

ServletContext

el表达式

隐式对象11个

pageContext

pageScope

requestScope

sessionScope

applicationScope

param

paramValues

header

headerValues

intParam

cookie

jstl标签

choose when otherwise 相当于switch case default

web开发模式

c/s

b/s

MVC表示层、业务处理层、数据库访问层

数据库连接池

常用数据源配置

DBCP(Database Connection Pool)

C3P0

数据源与JDNI（java naming and directory interface）绑定

事物的特性

原子性、一致性、隔离性、持久性

事务的隔离级别（赃读、不可重复读、虚读）

未提交读、已提交读、可重复读、可串行化

DBUtils（CRUD）

QueryRunner（DataSource ds）类

query()

update()

batch()

ResultSetHandler接口

DbUtils工具类

数组

2017/9/1

param…..可变参数可以和集合替换使用list.toArray();(将集合转化为数组)

分页技术 limit (i-1)\*size,size

AJAX（异步交互，局部刷新）

-------------------------------------------------------------------------------

一什么是Tomcat

Apache公司 开源的 免费的 轻量级的服务器

Tomcat是一个Web应用服务器 在小型系统和并发访问用户不多的场合下被

普遍使用,是开发和调试jsp程序的首选

javaSE 语言基础

javaEE 部分

13种技术

Servlet / Jsp JDBC JNDI JTA...

javaME 移动端

二Tomcat的安装和配置

官网: http://tomcat.apache.org

分为安装版和解压版

开发用解压版

运行还是安装版好 可以在服务器因为某种原因重启时能自动重启

注意:目录不要太深 目录不要有中文或空格

启动服务器

bin\startup.bat

关闭服务器

bin\shutdown.bat

启动服务器的前提

配置 JAVA\_HOME : jdk的安装路径

如果端口被占用 解决方法 修改端口号

conf\server.xml

<Connector connectionTimeout="20000" port="8080" protocol="HTTP/1.1" redirectPort="8443"/>

三Tomcat的目录结构

backup : 保存了一些配置文件 是在第一次运行tomcat服务器后产生的

和conf文件夹下的内容几乎一样

bin: 可执行文件

conf: 配置文件

lib: tomcat运行时需要的jar文件

logs: tomcat运行产生的日志文件

temp: tomcat运行产生的临时文件

webapps: 存放我们编写的应用

work: tomcat的工作目录

tomcat服务器的一些签证,运行,注意的信息

将web项目发布到tomcat的webapps目录下 才能通过浏览器访问

注意:webapps 目录下有几个目录 就代表有几个应用程序

1.Tomcat服务器与Eclipse 集成

2.创建web项目

3.将web项目发布到Tomcat服务器上

4.启动Tomcat

URL:

http://localhost:8080/servlet\_01/index.html

协议 主机(ip)端口号 / URI(当前应用的资源路径)

URL: 统一资源定位符(Uniform Resource Locator)(网址)

URI: 统一资源标识符(Uniform Resource Identifier)

四javaWeb应用的目录结构

WebRoot

css

html

js

WEB-INF: 注意 固定写法 此目录下的文件不能被外部直接访问

classes: 我们编写的java代码的.class文件

lib: 应用需要的jar包

web.xml 应用的配置信息

Servlet

一简介:

Servlet是SUN公司提供的一门用于开发动态web资源的技术

Servlet是运行在Web服务器中的小型Java程序(服务器端的小应用程序)

Servlet通过HTTP(超文本传输协议) 接收 和 响应 来自Web客户端的请求

Servlet : Server + let 服务器端的一个应用程序

Applet : Application + let 客户端的一个应用程序

二第一个Servlet:

1.编写一个java类 实现Servlet 接口

2.把开发好的java类部署到Web服务器中

//1.写一个java类 继承 HttpServlet 重写 doGet 和 doPost方法

//2.在web.xml里进行配置 部署描述符

4xx 打头 是 客户端错误 就是地址写错了

5xx 打头 是 服务器错误 servlet 或 web.xml

注意:

修改了 web.xml / servlet 必须重启tomcat服务器

修改了 html / css / js 不用重启 刷新浏览器就可以了

一 封装数据库工具类

介于一个项目的开发 都会使用同一个数据库 数据库连接地址 用户名 密码都是通用的

那么我们可以将这些重复的代码进行抽取 放到一个配置文件里

提高了代码复用性

而且 如果需要修改连接 只需修改一个配置文件就可以了

步骤

1.在src 目录下创建配置文件 xxxx.properties

2.工具类 创建数据库工具类 用来加载驱动 获取连接 释放资源

二 sql注入

SELECT \* FROM USER WHERE username='lee' AND PASSWORD='ddd' OR '1'='1'

发现 用这种方式 不安全 sql容易被注入 出现安全隐患

PreparedStatement 预编译对象 是 Statement对象的子接口

特点:

性能更高

会把sql语句先编译

sql语句中的参数会发生变化,过滤掉用户输入的关键字

三 分页

分析:

当数据量大的时候,需要进行分页

select \* from 表名 limit 起始行索引,长度(个数)

获得表中记录数 total --->count(\*)

规定每页显示记录数 pageSize = 3

计算页数 pageCount

total%pageSize==0 ? total/pageSize : total/pageSize+1

或者

Math.ceil(total\*1.0/pageSize)

Math.ceil(12.2) //返回13

Math.ceil(12.7) //返回13

Math.ceil(12.0) //返回12

static double ceil(double a)

返回最小的（最接近负无穷大）double 值，该值大于等于参数，并等于某个整数。

获得每页显示的数据

select \* from 表名 limit 0,3 第一页 0 三条记录 0,1,2

select \* from 表名 limit 3,3 第二页 3 三条记录 3,4,5

select \* from 表名 limit 6,3 第三页 6 三条记录 6,7,8

想要获得每页数据 只要获得页码(1,2,3....)

页码 pageNo 每页起始位置 (pageNo-1)\* pageSize

select \* from 表名 limit (pageNo-1)\*pageSize,pageSize

一 请求 和 响应

Http协议中 封装了请求和响应

请求部分

1.请求消息行

GET/POST 请求方式 浏览器默认请求方式GET

URI:去掉协议 ip 端口

协议/版本

2.请求消息头

Accept:浏览器可以接受的MIME类型 告诉服务器 客户端能接收什么类型的文件

Accept-Encoding:浏览器能够进行解码的数据编码方式

Accept-Language:浏览器所希望的语言类型

Host:初始URL中的主机及端口

Referer:包含一个URL 用户从该URL代表的页面出发 访问当前请求的页面

Content-Type:内容类型

User-Agent:浏览器类型

Content-Length:表示请求消息正文的长度

Cookie:这是最重要的请求消息之一

Date:请求时间

3.请求正文

当请求方式为POST方式时 才能看到消息正文

响应部分

1.响应消息行

HTTP/1.1 200 ok

协议版本 响应状态码 对响应码的描述(一切正常)

响应状态码

常用的就40多个

200正常

302/307 临时重定向

304 表示客户端缓存的版本是最新的 客户机可以继续使用 无需向服务端发送请求

404 资源找不到 服务器上不存在客户机锁清秋的资源

就是地址写错了

500 服务器内部错误 servlet代码错误

2.响应消息头

Location:通常和302/307一起使用 完成请求重定向

Server: 显示服务器的类型

Content-Encoding:服务器发送的数据采用的编码类型

Content-Length 告诉浏览器正文的长度

Content-Language: 服务器发送的文本语言 zh-cn

3.响应正文

和网页右键"查看页面源代码" 看到的内容一样

二 servlet创建方式和映射细节

Servlet创建方式 三种

1.实现Servlet接口

2.继承GenericServlet类

3.继承HttpServlet类

Servlet --> GenericServlet --> HttpServlet --> 继承HttpServlet的java类

曾祖父 --> 爷爷 --> 爸爸 --> 儿子

Servlet的生命周期

实例化(构造函数)-->初始化(init)-->服务(service)-->销毁(destroy)

出生: (实例化-->初始化) 第一次访问Servlet就出生

活着: (服务) 应用活着 Servlet就活着

死亡: (销毁) 应用卸载了 Servlet就销毁

映射细节

web.xml

一个servlet标签 可以对应多个servlet-mapping标签

优先级

从高到低

绝对匹配 --> /开头匹配 --> 扩展名方式匹配

url-pattern : \*.do

url-pattern : /\*

url-pattern : /action/\*

通配符 \* 代表任意字符串

Servlet线程安全问题

为了避免出现安全问题

在Servlet中 不要写全局变量 而写局部变量

三 ServletConfig的使用

作用1.获取Servlet配置信息

作用2.获取ServletCcontext对象

ServletConfig 代表当前Servlet在web.xml中配置的信息(用的不多)

在Servlet的配置文件中 可以使用一个或者多个<init-param>标签为Servlet设置一些初始化参数

当Servlet配置了初始化参数后,web容器在创建Servlet实例对象时,会自动将这些初始化参数封装到ServletConfig对象中

程序员通过ServletConfig对象就可以得到当前Servlet的初始化参数信息

好处:如果将数据库信息/编码方式等配置信息放在web.xml中

如果以后数据库的用户名/密码改变了,则直接修改web.xml就行了

避免了直接修改源代码的麻烦

//方式1 重写GenericServlet的init方法 初始化 等到ServletConfig对象 由此对象 获取配置文件中的值

//方式2 通过祖先 Servlet 的 getServletConfig方法 获得 ServletConfig 对象

//方式3 HttpServlet 已经实现了 ServletConfig接口

四 ServletContext对象

域 作用域

Servlet四大域对象

1.ServletContext(整个web项目)

2.HttpSession(一次会话)

3.ServletRequest(一次请求)

4.PageContext(一个jsp页面)

ServletContext 代表的是整个应用程序,一个应用只有一个ServletContext对象

例如:

班级里的空调就是域对象

域对象:在一定范围内(当前应用) 使多个Servlet共享数据

常用方法:

//向ServletContext对象中添加数据

void setAttribute(String name,object value)

//从ServletContext对象中获取数据

Object getAttribute(String name)

//根据name 移除数据

void removeAttribute(String name)

获取资源路径

//根据资源名称 得到资源的绝对路径

可以得到当前应用任何位置的任何资源

String getRealPath(String path)

请求转发

我借钱 这叫请求 他向别人借 叫转发

前段页面向后台发送一个请求 然后利用方法将请求转发出去 跳转到另一个Servlet里

//分开写

//application.getRequestDispatcher(String path); 请求分发 path web.xml配置的<url-pattern> 要加/

RequestDispatcher rd = application.getRequestDispatcher("/demo\_05");

//forward(req, resp); 前进 就是将请求和响应 一起转发出去

rd.forward(req, resp);

//一句话

//application.getRequestDispatcher("/demo\_05").forward(req, resp);

特点:地址栏不发生改变 还是原来的地址

luojing0322@163com

五 request对象 (域对象)

web服务器收到客户端的http请求 ,会针对每一次请求,分别创建一个request对象和一个response对象

代表 请求和响应

1.请求 HttpServletRequest

1.请求行

2.请求头

3.请求正文

3.1 与获取表单数据相关的方法

3.2 BeanUtils

3.3 与请求转发相关的方法

3.4 与操作非表单数据相关的方法

3.5 与请求编码相关的方法

获取请求行中信息

常用方法

getMethod(); 获取请求方式

getRequestURL() 获取客户端发送请求时的完整URL

getRequestURI() 获取请求行中的资源名部分

getContextPath() 获取当前应用的虚拟目录

getQueryString() 获取请求行中的参数部分

获取请求头中的信息

常用方法

String getHeader(String name) 根据头名称 得到头信息值

Enumeration getHeaderName() 得到所有头信息 name

Enumeration getHeaders(String name) 根据头名称 得到信息

获取请求正文中的信息

3.1 与获取表单数据相关的方法

常用方法

String getParameter(String name) 根据表单中name属性名 获取表单项的value值

String[] getParameterValues(String name) 专为复选框取值提供的方法

Enumeration getParameterNames() 得到表单提交的所有name的方法

map<String,String[]> getParameterMap() 得到表单提交的所有值的方法

ServletInputStream getInputStream() 以字节流的方式得到所有表单数据

3.2 BeanUtils

由于前几种封装对象方法太麻烦,我们接下来使用BeanUtils来封装对象

注意:如果想自动封装对象,表单form中的元素name需要与javaBean中的属性一致

什么是BeanUtils?

Apache公司提供的 用来专门进行javaBean操作

在web层各种框架中被使用 例如 struts 使用beanUtils操作javabean

步骤

1.下载BeanUtils的jar包:

commons-beanutils-1.8.3.jar

commons-logging-1.1.1.jar

BeanUtils依赖Logging的jar包

2.web项目下lib目录 导包

常用方法:

BeanUtils.populate(Object bean,Map map)进行数据封装

因为需要map 因此需要通过 request getParameterMap()方法将表单的数据变成map

String-->java.util.Date 不支持 我们自定义转换器

自定义转换器

1.实现Converter接口

2.注册转换器

ConvertUtils.register(转换器,目标类型.class);

结论:

一般不需要自定义转换器 只有Date需要自定义转换器

3.3 与请求转发相关的方法

getRequestDispatcher("/demo\_05").forward(req, resp);

3.4 与操作非表单数据相关的方法

3.5 与请求编码相关的方法

//设置请求编码集

req.setCharacterEncoding("utf-8");

HttpServletResponse

响应

1.响应行

setStatus(int sc) 设置响应状态码

200 ok

2.响应头

\*\*\*\*\*sendRedirect(String location) 请求重定向

setHeader(String name,String value)设置响应头信息

3.响应正文(主体)

\*\*\* getWrite() 字符输出流

getOutputStream() 字节输出流

setCharacterEncoding(String charset)告知服务器使用什么编码

\*\*\* setContentType(String type)

转发和重定向的区别

Servlet 中页面跳转的两种方式 转发和重定向

1.转发是服务器行为,重定向是客户端的行为

2.转发是浏览器发送了一次请求, 重定向至少两次请求

3.转发的地址栏不会发生改变,重定向地址栏中地址发生改变

4.转发因为是一次请求,两次跳转之间传输信息不会丢失,而重定向两次跳转会丢失传输信息

5.转发只能将请求跳转到同一个web应用中的页面,冲顶先可以指向任何页面,包括同一个web应用中的页面

同一个站点中的页面,其他站点的页面

request.getRequestDispatcher("demo.jsp").forward(request,response);

response.sendRedirect("demo.jsp");

会话

客户端技术cookie

服务器端技术session

什么是会话?

简单理解为:用户打开一个浏览器,点击多个超链接,访问服务器多个web资源,然后关闭浏览器

整个过程称之为一次会话

提问 用户名 密码存储在 Request域 和 ServletContext域中行不行?

Request域太小

ServletContext域太大

举例 多用户点击超链接通过servlet各自购买了一个商品,服务器应该想办法把每个用户

购买的商品保存在各自的地方,以便于这些用户点结账servlet时

结账servlet可以得到用户各自购买的产品为用户结账

Cookie

Cookie是客户端技术,服务器把每个用户的数据以cookie的形式写回给用户各自的浏览器

当用户使用浏览器再去访问服务器中的web资源时,就会带着各自的数据去,

这样,web资源处理的就是用户各自的数据了

Session

Session是服务端技术,利用这个技术,服务器在运行时可以为每一个用户的浏览器创建一个独享的Session对象

由于Session为用户浏览器独享,所以用户在访问web资源时,可以把各自的数据存放在各自的session中

当用户再去访问服务器的其他web资源时,其他web资源再从用户各自的session中取出数据为用户服务

会话过程要解决的问题是什么?

保持各个客户端自己的数据

例如:

登录 保存用户名和密码(登录成功后保存) 可以保存到cookie里

淘宝登录 显示在最上面的用户名 就是保存在session里

Cookie

特点:

1.一个Cookie拥有一个名称,一个值和一些可选属性 比如路径 存活时间等

2.Cookie只能存放servlet发送到web浏览器的少量信息

3.注意 Cookie只能存储字符串 数字不能存储数组,资源等多维数据

name value 属性...

name:名称 不能为宜确定一个cookie 路径可能不同

value:值

path:路径

maxAge:cookie的缓存时间 默认是-1 单位是秒

负数: cookie的数据存在浏览器缓存中,关闭浏览器就失效

0: 删除

正数:缓存(持久化到磁盘上)的时间

获取emp表历史记录信息

1.创建实体类

2.导入jar包

3.创建配置文件

4.编写数据库工具类DBUtils

5.创建连接 查询全部员工信息 和 根据员工编号查询一条信息

DAO data access object 数据访问对象 通常都是写访问数据库代码

EmpDao

List<Emp> findAll()

Emp findById(int empno)

6.创建Servlet

创建ShowAllEmpServlet 获取EMP表信息

创建ShowByIdServlet 获取一条信息

把获取到的员工编号 存储到Cookie对象中 发送到客户端

客户端获取Cookie中的信息 根据员工编号 查询员工信息 显示在页面上

Session

为什么要学习HttpSession?

它也是一个域对象 Session Request ServletContext

同一个会话下 可以使一个应用的多个资源共享数据

cookie客户端技术 只能保存字符串

HttpSession服务端技术 它可以保存对象

cookie在客户端只允许存储4k的数据 实际上存了3k数据就已经满了

session存储的数据更大 更多

在B/S 交互中 需要存储大量数据时 这就是Session存在的重要原因

如果说cookie机制是通过检查客户身上的"通行证"来确定客户身份的

那么 session机制就是通过检查服务器上的"客户明细表"来确认客户身份

session相当于程序在服务器上建立了一份客户档案 客户来访问的时候只需要检查客户档案就可以了

常用方法

void setAttribute(String name,Object value);

Object getAttribute(String name);

void removeAttribute(String name);

getId(); 获取Session id

setMaxInactivelnterval(int interval) 设置session的存活时间

invalidate() 使此会话无效

案例1 用户登录 显示验证码 并进行判断

1.login.jsp 登录页面

2.servlet 用户名 密码 验证码 进行验证

admin 123456

验证码 如果验证码输入错误 返回到登录页 提示 验证码输入错误

3.如果登录成功 提示信息

如果登录失败 提示信息

案例2 购物车

需求分析:

1.获取到商品列表 (连接数据库 获取商品信息)

2.点击某一个商品 -->进入详细信息页面(连接数据库 获取某个商品的信息)

-->添加进购物车

session比cookie更适合存储数据 因为它可以存储对象

将数据存储入session中

3.查看购物车

查看session中的数据 确定购物车中的商品属于哪一个人 是根据session id来确定的

需要做的事

1.jar包 连接数据 配置文件

查询所有商品 根据主键查询商品信息

2.servlet 显示全部商品

servlet 显示某一个商品 -->加入到session中

servlet查看购物车 -->查看session中的数据

案例3 表单重复提交

如何避免表单重复提交

1.创建jsp页面 注册 用户名 密码 按钮

2.servlet 获取注册信息 将信息存入数据库

令牌 token

设置一个令牌 将令牌分别存储在session域和 表单里

当第一次提交数据 如果表单令牌和session域中的令牌 一致 就让它存储

存储结束之后 就将session中的令牌删除

当刷新servlet session中的令牌已经没有了 无法匹配 就无法存储

URL地址重写

客户端禁用cookie

当浏览器拒绝存储cookie信息时,就会导致JSESSIONID的值不能正常到达客户端

那么客户端向服务器发送请求时,就无法将JSESSIONID的值带回到服务器中,

从而导致找不到自己会话的Session

怎么解决: 我们可以使用URL地址重写的方法来解决

response.encodeURL("请求地址");

判断cookie是否被禁用

如果未禁用 就保持原请求地址 不变

如果禁用 就修改请求地址 为请求地址添加JSESSIONID的值

response.sendRedirect(response.encodeRedirectURL("请求地址"))

一 jsp概述

JSP全程是 java Server Pages 它和servlet技术一样 都是SUN公司定义的一种用于开发动态web项目的技术

JSP实际上就是Servlet

JSP = HTML + CSS + JS + JAVA

html静态页面

Servlet:服务器端的小应用程序 适合编写java逻辑代码 如果编写网页内容 -->苦逼

1.获取表单数据

2.处理业务逻辑

3.分发转向

jsp:适合编写输出动态内容 但是不适合编写java逻辑

二.jsp原理

当浏览器访问一个web服务器上的jsp页面时

在服务器端会将jsp页面进行 -->翻译成.java文件-->编译成.class文件-->执行

以jsp作为代码的显示模板 重点在于显示数据

以servlet作为控制器 重点在编写java代码逻辑上

三.jsp的基本语法

1.jsp模板元素

网页静态内容 如html标签和文本

2.jsp的脚本

2.1 小脚本 <% java 代码 %>

2.2 表达式 <%=表达式 %> 等价于 <% out.print(); %>

注意: <%= 两者是紧挨着的 没有空格 结尾处 不要加 ; 号

2.3 声明 <%! %> 表示在类中定义全局成员和静态块

3.jsp的注释

jsp注释 : <%--被注释的内容 特点 安全(查看页面源代码时 看不到) --%>

网页注释: <!-- 网页注释 特点 不安全(查看页面源代码时 能看到) -->

四.jsp的3个指令

jsp指令是为jsp引擎而设计的

jsp2.0定义了三个指令

page

include

taglib

语法:

<%@ 指令名称 属性1="属性值1" 属性2="属性值2" ....%>

或者

<%@ 指令名称 属性1="属性值1"%>

<%@ 指令名称 属性2="属性值2"%>

1.page指令

作用:用于定义jsp页面的各种属性

<%@ page import="java.util.Date"%>

<%@ page language="java" contentType="text/html; charset=utf-8" pageEncoding="utf-8"%>

1.1 import 和java代码中的import是一样的

<%@page import="java.util.Date,java.util.List"%> 以逗号分隔每个包

或者

<%@page import="java.util.Date"%>

<%@page import="java.util.List"%>

jsp会默认导入包

java.lang.\*;

javax.servlet.\*;

javax.servlet.http.\*;

javax.servlet.jsp.\*;

1.2 errorPage 如果页面中出现错误 则跳转到指定的资源

errorPage="error.jsp"

1.3 language="java"

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

1.5 pageEncoding="utf-8"

1.6 isELIgnored 是否支持EL表达式 默认是false

jsp 默认支持el表达式 平时不用设置

isELIgnored="false"

2.include指令

把其他资源包含到当前页面中

静态包含:

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

动态包含:

<jsp:include page="header.jsp"></jsp:include>

两者的区别:

翻译的时间段不同

前者:在翻译时就把两个文件合并

后者:不会合并文件,当代码执行到include时,才包含另一个文件的内容

原则:

能用静的 就不用动的

3.taglib指令

作用:在jsp页面中调入JSTL标签库 替换jsp中的java代码段

<%@ taglib uri="" prefix="" %>

uri 导入的包

prefix 标签的别名

五.jsp的6个动作(很少用)

使用标签的形式来表示一段java代码

1.<jsp:include></jsp:include> 动态包含

2.<jsp:forward> 请求转发

3.<jsp:param> 设置请求参数

4.<jsp:useBean> 创建一个对象

5.<jsp:setProperty> 给指定的对象属性赋值

6.<jsp:getProperty> 取出指定对象的属性值

<!-- 创建对象 id:对象名-->

<jsp:useBean id="user2" class="com.qf.bean.User"></jsp:useBean>

<!-- 给对象属性赋值 name:对象名 property:属性名 value:属性值-->

<jsp:setProperty property="username" name="user2" value="tom"/>

<!-- 取出属性值 -->

<jsp:getProperty property="username" name="user2"/>

<jsp:getProperty property="age" name="user2"/>

<!-- 请求转发 并携带参数 -->

<jsp:forward page="demo.jsp">

<jsp:param value="jerry" name="thename"/>

<jsp:param value="100" name="theage"/>

</jsp:forward>

六.jsp的9个内置对象

在jsp的<%%> 和 <%= %>中能直接使用的对象

对象名 类型

request javax.servlet.http.HttpServletRequest

response javax.servlet.http.HttpServletResponse

session javax.servlet.http.HttpSession

application javax.servlet.ServletContext

execption java.lang.Throwable 由 ieErrorPage="true"开关

page java.lang.Object当前对象this 当前servlet实例

config javax.servlet.ServletConfig 读取配置文件

out javax.servlet.jsp.jspWriter 字符输出流 相当于PrintWriter对象

pageContext javax.servlet.jsp.PageContext

pageContext(重要 域对象)

pageContext 只在当前页面使用 所以 跨页面获取不到值

本身就是一个域对象 它可以操作其他三个域对象(request,session,application)的数据

常用方法:

void setAttribute(String name,Object obj);

Object getAttribute(String name);

void removeAttribute(String name);

操作其他域对象的方法 指定某一个指定的域对象 操作数据

void setAttribute(String name,Object obj, int Scope);

Object getAttribute(String name, int Scope);

void removeAttribute(String name, int Scope);

//转发 可以获取到除了pageContext以外的其他域对象

//request.getRequestDispatcher("b.jsp").forward(request, response);

//重定向 只能获得session和application域对象

response.sendRedirect("b.jsp");

Scope:

可以实现pageContext域 set 其他域的值

PageContext.PAGE\_SCOPE

PageContext.REQUEST\_SCOPE

PageContext.SESSION\_SCOPE

PageContext.APPLICATION\_SCOPE

findAttribute(String name)自动从 page request session application依次查找

找到了 就结束

四大域对象:

实际开发中用哪一个

PageContext:-->pageContext存放的数据在当前页面有效 开发用的很少

ServletRequest:-->request存放的数据在一次请求(转发)内有效,使用非常多

HttpSession:-->session存放的数据在一次会话中有效 例如 存储用户登录信息 购物车功能

ServletContext:-->application 存放的数据在整个应用范围内 因为范围比较大 应用很少

七.el表达式

EL表达式: expression language 表达式语言

作用: 简化jsp中的java代码开发量

它不是一种开发语言 是jsp中获取数据的一种规范

主要作用:

1.获取数据

从各种类型的web域中检索java对象,获取数据

(javabean的属性/list集合/map集合/数组)

2.执行运算

利用EL表达式可以在jsp页面中执行一些基本的关系运算/逻辑运算和算数运算

3.获取web开发常用对象

EL表达式定义了一些隐式对象,利用隐式对象,可以得到其他常用对象从而获取对象中的数据

4.调用java方法

EL表达式允许用户开发自定义EL函数,可以在jsp页面中通过EL表达式调用java类的方法

注意:

如果页面EL表达式不起作用

<%@ page isELIgnored="false"%>

1.获取数据

重点:EL表达式只能获取存储在4个作用域中的数据

<!-- el表达式 -->

${u}<!-- 原理 pageContext.findAttribute("u") -->

<br/>

${u.username} <!-- 等同于 user.getUsername()-->

点(.)相当于调用了 getter方法 点后面跟的就是属性名 -->使用树形导航

EL表达式真正读取的时候,是在读取javaBean实例中get方法后面的字母

javaBean属性的有无或者属性的拼写是什么与EL表达式使用是没有关系的

EL只关心类中是否有相对应的getter方法

所以EL输出时 属性名称完全参照getter方法

规则:

1.保持与getter方法名 get三个字母后的内容一致的大小写,然后将首字母小写了

2.当getter方法名开头出现了相连的两个大写字母时,EL属性名称就是完全抄写getter方法名

get三个字母后面的内容

<!-- []运算符 点能做的 它也能做 它能做的 点不一定能做 -->

2.运算

eq 相等

neq / ne 不相等

gt 大于

lt 小于

gte/ge 大于等于

lte/le 小于等于

not 非

empty 判断null 空字符串 和 没有元素的集合 都返回true/false

三目运算符

3.隐式对象 11个

pageContext

pageScope pageContext范围中存放的数据

requestScope 请求范围的数据

sessionScope 会话范围的数据

applicationScope 应用范围的数据

param 一个请求参数

paramValues 重名请求参数

header 一个请求消息头信息

headerValues 重名的请求消息头

intParam web.xml中全局参数

cookie cookie对象的name值

八.jstl标签

JSTL \*\*\*\*\*

JavaServerPage Standard Tag Library -->JSP 标准标签库

作用:

使用JSTL实现JSP页面中逻辑处理 如判断 循环等

使用JSTL步骤

1.在jsp页面添加taglib指令

<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>

<!-- prefix 是前缀 -->

2.使用JSTL标签

JSTL常用标签

1.表达式控制标签 set out remove catch

2.流程控制标签 if choose when otherwise

3.循环标签 forEach forTokens

4.URL操作标签 import url redirect

JavaWeb开发模式

C/S 客户端(Client) 服务器(Server) -->胖客户端

B/S 浏览器(Browser)服务器(Server) -->瘦客户端

Servlet的三个功能:获取数据 处理逻辑 分发转向

两种模式

1.Model1 JSP+JavaBean

2.Model2 JSP+Servlet+JavaBean

MVC开发模式

M : Model 模型 -->JavaBean

V : View 视图 -->jsp /html

C : Controller 控制器 --> Servlet

用一种业务逻辑/数据/界面显示分离的方法组织代码,将业务逻辑聚集到一个部件里面

在改进和个性化定制界面及用户交互的同时,不需要重新编写业务逻辑

分层的思想: 高内聚 低耦合

什么是接口??? 规范 约束 规则

面向接口编程 就是将 规范 和 实现 相分离 (名实分离原则)

登录为例

1.数据库 创建 User表

2.调入jar包 数据库配置文件 连接数据库

3.javaBean -->User实体类

4.创建表示层

jsp页面+servlet

5.业务处理层

面向接口编程的思想

UserService 接口

login方法

UserServiceImpl 实现类 实现 UserService接口

重写 login方法

调用 findByUser()方法

6.数据库访问层 Dao

UserDao 接口

findByUser()

UserDaoImpl 实现类 实现 UserDao接口

重写 findByUser()方法

一.数据库连接池

什么是连接池?

数据库连接池负责分配/管理和释放数据库连接.它允许应用程序重复使用一个现有的数据库连接

而不是再重新建立一个;释放空闲时间超过最大空闲时间的数据库连接来避免因为没有释放数据库连接而引起的数据库连接遗漏

这项技术明显提高了对数据库的操作性能

例子:客服10086

每个客服都有一个编号 接完电话 排到最后 而且不会接一次 就开除

jdbc每次都要连接 关闭 但是连接池不用 申请一个连接 用完之后 不释放 再次回到线程池中

常用的数据源配置

1.DBCP

DBCP: Apache推出的Database Connection Pool

使用步骤:

1.添加jar包 dbcp pool

2.添加属性资源文件 dbcpconfig.properties

javax.sql 接口 DataSource

#<!-- 初始化连接 -->

initialSize=10

#最大连接数量

maxActive=50

#<!-- 最大空闲连接 -->

maxIdle=20

#<!-- 最小空闲连接 -->

minIdle=5

#<!-- 超时等待时间以毫秒为单位 60000毫秒/1000等于60秒 -->

maxWait=60000

2.C3P0

C3P0是一个开源的JDBC连接池 实现了数据源和JNDI绑定

JNDI(java naming and directory interface)

是SUN公司提供的一种标准的java命名系统接口 是一组在java应用中访问命名和目录服务的API

JNDI 是一种资源管理方式

C3P0 是数据源连接池的配置方式

使用JNDI管理数据库连接池 连接池可以使用C3P0 也可以使用DBCP...

实现步骤:

1.导入c3p0 jar 包

2.编写配置文件 c3p0-config.xml 放在src根目录下

用JavaWeb服务器管理数据源:

Tomcat配置数据源

步骤:

1.拷贝数据库连接的jar(jdbc驱动)到tomcat/lib目录下

2.配置数据源xml文件

a:把配置信息写在 tomcat下的conf目录下的 context.xml中 那么所有应用都能使用

b:在MATA-INF中创建context.xml编写数据源,那么只有当前应用可以使用

context.xml中加入如下配置

<Resource 属性/>

属性:

name: jdbc/工程名

auth: 作者

type: 类型

maxActive: 最大连接

maxIdle: 最大空闲

maxWait: 最大等待时间 单位毫秒

username

password

driverClassName

url

<Resource name="jdbc/web\_14" auth="Container" type="javax.sql.DataSource"

maxActive="100" maxIdle="30" maxWait="10000"

username="root" password="111111"

driverClassName="com.mysql.jdbc.Driver"

url="jdbc:mysql://localhost:3306/dbcq"

/>

Context ctx = new InitialContext();

DataSource ds = (DataSource)ctx.lookup("java:/comp/env/jdbc/web\_14");

out.print(ds);

"java:/comp/env/jdbc/web\_14"

path:java:/comp/env/

name:jdbc/web\_14

JNDI(java naming and directory interface)

JNDI就是一个容器 一个map

一个数据源 多个JNDI 根据名称的规范找到连接池

二.事务

什么是事务?

事务是指逻辑上的一组操作,组成这组操作的各个单元,要么全部成功 要么全部不成功

案例:

有AB两个账户 账户的初始金额都是1000元 发生转账操作

转账: A账户向B账户转账100元

A money = money - 100; -->900

int i = 9/0;-->异常

出现异常 由于代码是一行一行执行的 那么下面的代码就不再执行 那么就出现了转账错误

B money = money + 100; -->1000

这两个语句全部完成是完成了一次转账的操作 属于一组操作

事务

Mysql中的事务

a:Mysql引擎是支持事务的

b:Mysql默认自动提交事务 每条语句都处在单独的事务中 如果不明显去写事务代码

那么默认每一条sql 就是一个单独的事务

c:手动控制事务

开启事务 start transaction | begin

提交事务 commit

回滚事务 rollback

事务的特性(面试题)

原子性:指事务是一个不可分割的工作单位,事务中的操作要么都发生,要么都不发生

一致性:事务必须使数据库从一个一致性状态变换到另一个一致性状态,转账前和转账后的总金额不变

隔离性:事务的隔离性是多个用户并发访问数据库时,

数据库为每一个用户开启的事务,不能被其他事务的操作数据所干扰.多个并发事务之间相互隔离

持久性:指一个事务一旦被提交,它对数据库中数据的改变就是永久性的,

接下来即使数据库发生故障也不应该对其有任何影响

事务的隔离级别

如果多线程并发 同时访问一张表 这时候会引起 数据不正确的问题

就类似于多线程并发操作统一数据不安全问题 多个用户之间会发生互相干扰的现象

隔离级别: 脏读(Dirty Read) 不可重复读(NonRepeatable Read) 虚读(幻读)(Phantom Read)

未提交读 Read uncommitted 可能 可能 可能

已提交读 Read committed 不可能 可能 可能

可重复读 repeatable read 不可能 不可能 可能

可串行化 Serializable 不可能 不可能 不可能

级别越高 数据越安全 但是 性能越低

脏读:指一个事务读取了另一个事务未提交的数据

例如:B开启了事务 还钱给A 执行了一条update money+100

然后 给A打电话 说 我还钱了

A查询 发现多了100

然后 B回滚了 update语句不执行了

不可重复读:在一个事务内读取表中的某一行数据 多次读取结果不同

一个事务读取到了另一个事务提交后的数据 (主要针对update语句)

虚读(幻读):主要针对insert语句

数据库通过设置事务的隔离级别防止以上情况发生:

1 Read uncommitted 脏读/不可重复读/虚读 都可以发生

2 Read committed 避免脏读......不可重复读/虚读都有可能发生 (oracle默认隔离级别)

4 repeatable read 避免脏读/不可重复读......虚度有可能发生(mysql默认隔离级别)

8 Serializable 避免 脏读/不可重复读/虚度 -->(序列化的)

实际工作中 默认用4号 级别越高 性能越低 数据越安全

如果数据量大 事务隔离级别越高 那么他的效率就会变得很低

所以大数据量 可以允许出现点重复读和虚读的问题 都是可以解决的

mysql中:

查看当前的事务隔离级别: select @@tx\_isolation

更改当前的事务隔离级别: set transaction isolation level read uncommitted

begin 开启事务

commit; 提交事务

rollback;回滚

jdbc控制事务的隔离级别

Connection接口

Connection.setTransactionIsolation(int level);

Connection.TRANSACTION\_READ\_UNCOMMITTED,

Connection.TRANSACTION\_READ\_COMMITTED,

Connection.TRANSACTION\_REPEATABLE\_READ,

Connection.TRANSACTION\_SERIALIZABLE

设置隔离级别 必须在开启事务之前

三.DBUtils

Apache

什么是DBUtils

是java编程中的数据库操作实用工具 小巧简单实用

封装了JDBC的操作 简化了JDBC操作 可以少写代码

CRUD

1.对于数据表的读操作 他可以把结果转换成 List Array Set 等集合 便于程序员操作

2.对于数据表的写操作 变得很简单(只需要sql语句就可以了)

3.可以使用数据源 使用JNDI 数据库连接池等技术来优化性能

重用已经构建好的数据库连接对象

DBUtils的三个核心对象

QueryRunner类

ResultSetHandler 接口

DBUtils 类

QueryRunner类

提供对sql语句操作的API

QueryRunner(DataSource ds) 将连接池给DBUtils框架

以后每一个操作都会从连接池中获取一个新的连接

每条sql一个单独的事务

它主要有三个方法

query() 用于执行select

update() 用于执行 insert update delete

batch() 用于批处理 可执行多条语句 批量

ResultSetHandler接口

用于定义select操作后 怎样封装结果集 返回的内容

DBUtils类

它就是一个工具类 定义了关闭资源与事务处理的方法

DBUtils快速入门

步骤

1.导包jar

2.创建QueryRunner对象

3.使用query方法执行select语句

4.使用ResultSetHandler封装结果集

5.使用DBUtils类释放资源

ResultSetHandler接口下的所有结果处理器

1.BeanHandler 把每条记录封装成对象 适合取一条记录

2.BeanListHandler 把每条记录封装成对象 把对象存储到List集合中

3.ArrayHandler 适合取一条记录 把该记录的每一列值封装到一个数组中 Object[]

4.ArrayListHandler 适合取多条记录 把每条记录的每列封装到一个数组中Object[] 把数组封装到集合中

5.ColummListHandler 取某一列的数据 封装到List中

6.KeyedHandler 取多条记录 每条记录封装到Map中 再把这个Map封装到另一个Map中

key为指定字段值

7.MapHandler 适合取一条记录 把当前记录的列名和列值放到一个Map中

8.MapListHandler 适合取多条记录 把每条记录封装到一个Map中 再把Map封装到List

9.ScalarHandler 适合取单行单列数据

AJAX

什么是AJAX

Asynchronous 异步的

JavaScrpit

And

Xml

不是一种新的编程语言 而是一种用于创建更好更快以及交互性更强的web应用程序的技术

基于 javascript/xml/html/css 的新用法

一般 都返回文本 返回xml还得解析 所以很少用

java中交互方式分为同步和异步两种

同步交互:指发送一个请求,需要等待返回,然后才能够发送下一个请求 有一个等待过程

通常 同步传输是以数据块为传输单位

异步交互:指发送一个请求,不需要等待返回,随时可以再发送一个请求,即不需要等待

通常 异步传输是以字符为传输单位

区别:

同步需要等待 -->银行转账系统

异步不需要等待

举个例子 普通B/S模式(同步) AJAX技术(异步)

同步: 提交请求-->等待服务器处理-->处理完毕返回 这个期间客户端浏览器不能干任何事

异步: 请求通过事件触发-->服务器处理(这时浏览器仍然可以做其他事情)->处理完毕

同步 就是你叫我吃饭 我听到了就和你吃饭去 没听到 你就不停的叫 知道我告诉你一声 听到了

才一起去吃饭

异步 就是你叫我吃饭 然后自己去吃饭了 我得到消息后可能立即走,也可能等到下班才去吃

AJAX:

是与服务器交互数据并更新部分网页的艺术

刷新局部页面的技术 可以使网页实现异步更新

这意味着在不重新加载整个网页的情况下,对网页的某部分进行更新

传统网页(不使用ajax) 如果要更新内容 必须重新加载整个网页

google 百度 地图

JavaScript中XMLHttpRequest 对象是整个Ajax技术的核心 它提供了异步发送请求的能力

构造方法:不同浏览器 甚至相同浏览器的不同版本 获取该对象的方式是不同的

var xmlhttp;

if (window.XMLHttpRequest)

{// code for IE7+, Firefox, Chrome, Opera, Safari

xmlhttp=new XMLHttpRequest();

}

else

{// code for IE6, IE5

xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");

}

向服务器发送请求

open(method,url,async)

规定请求的类型、URL 以及是否异步处理请求。

method：请求的类型；GET 或 POST

url：文件在服务器上的位置

async：true（异步）或 false（同步）

send(string)

将请求发送到服务器。

string：仅用于 POST 请求

setRequestHeader(header,value)

向请求添加 HTTP 头。

header: 规定头的名称

value: 规定头的值

服务器响应

responseText 获得字符串形式的响应数据。

responseXML 获得 XML 形式的响应数据。

onreadystatechange 事件

当请求被发送到服务器时，我们需要执行一些基于响应的任务。

每当 readyState 改变时，就会触发 onreadystatechange 事件。

readyState 属性存有 XMLHttpRequest 的状态信息。

下面是 XMLHttpRequest 对象的三个重要的属性：

属性 描述

onreadystatechange 存储函数（或函数名），每当 readyState 属性改变时，就会调用该函数。

readyState

存有 XMLHttpRequest 的状态。从 0 到 4 发生变化。

0: 请求未初始化

1: 服务器连接已建立

2: 请求已接收

3: 请求处理中

4: 请求已完成，且响应已就绪

status

200: "OK"

404: 未找到页面

注意:只有当状态是 4 并且 状态码是200的时候 才可以处理服务器数据

案例:

使用Ajax验证用户名是否已存在

步骤:

1.使用文本框的 onBlur事件

2.使用Ajax技术实现异步交互

a:创建XMLHttpRequest对象

b:通过XMLHttpRequest对象设置请求信息

c:向服务器发送请求

d:创建回调函数,根据响应状态动态更新页面

JSON简介:

JSON(javaScript Object Notation)是一种轻量级的数据交换格式

xml 解析: sax dom dom4j

JSON数据作为目前网络中主流的数据传输格式之一,应用十分广泛

JSON格式:

JSON的数据书写格式:

名称/值对

名称:字符串 用双引号 后面加一个冒号 然后是值

例如:

"firstName":"John"

名 可以不加双引号

等价于

firstName=John-->javascript语句

JSON值可以是

数字(整数或者浮点型) 字符串(双引号中) 逻辑值(true/false)

数组([]方括号中) 对象{} null

对象:对象表示为{}括起来的内容 数据结构为

{key:value,key2:value2...}的键值对结构

例如:

String json = "{name:'张三',age:20,tel:'12344'}"

数组:数组表示为中括号[]括起来的内容 数据结构为

[value1,value2,value3...]的单值结构

["背景","上海","广州"]

java 中 Student[] stu = new Student[3];

json格式

[

{name:"张三",age:23},

{name:"张三",age:23},

{name:"张三",age:23}

]

JSON解析:

三种方式

1.JSON解析

2.GSON:谷歌

3.FASTJSON:阿里巴巴

JSON解析步骤:

1.导入jar包

2.使用 JSONObject 方法 / JSONArray 方法 进行Json解析

分析方式 看第一个符号

是{花括号 就是对象

是[方括号 就是数组

使用getString()方法等获取json格式中的数据

过滤器

javaweb中的过滤器可以拦截所有访问web资源的请求或相应

案例:

1.解决全局中文乱码问题 (全局编码过滤器)

Post

Get: name = new String(name.getBytes("iso-8859-1","utf-8"))

2.管理员与普通用户登录(对登录用户进行判断 对其进行过滤)

3.自动登录

开发步骤

1.创建一个类 实现Filter接口

2.重写接口中的方法

doFilter 方法是真正过滤的

3.在web.xml中配置过滤器

FilterChain 是servlet容器为开发人员提供的对象 它提供了对某一资源的过滤请求链

1.怎样可以形成一个Filter链?

只要多个Filter对同一个资源进行过滤就可以形成Filter链

2.怎样确定Filter执行顺序?

由web.xml中 <filter-mapping>的配置顺序 来确定 执行顺序

Servlet生命周期

实例化-->初始化-->服务-->销毁

Filter生命周期

当服务器启动 会创建Filter对象 并调用init()方法 只调用一次

当访问资源时 路径与Filter的拦截路及鞥匹配,会执行Filter中的doFilter方法

这个方法是真正过滤操作的方法

当服务器关闭时 会调用Filter的destroy方法来进行销毁操作

案例:

1.解决全局中文乱码问题 (全局编码过滤器)

怎样才可以做一个通用的 可以处理get/post请求的?

获取请求参数的方式

getParameter

getParameterValues

getParameterMap

在java 中怎样可以对一个方法进行功能增强?

装饰设计模式

1.创建一个类让它与被装饰类实现同一个接口或者继承同一个父类

2.在装饰类中持有一个被装饰类的引用

3.重写要增强的方法

监听器

Listener

事件发生 事件源(产生这个事件的源头) 监听器(用于监听指定事件的对象)

注册监听器:要想让监听器可以监听到事件发生,必须对其进行注册

javaweb开发中的监听器 是用于监听web常见对象

HttpServletRequest HttpSession ServletContext

监听他们的创建 /销毁 属性变化 session绑定javaBean(活化/钝化)

Servlet中的监听器平时使用的比较少

监听器一共分八个 三类

第一类:

用来监听HttpServletRequest HttpSession ServletContext的创建和销毁

ServletContextListener

HttpSessionListener

ServletRequestListener

HttpSessionListener涉及到一个session活化和钝化的问题

第二类

用来监听HttpServletRequest HttpSession ServletContext的属性变化

(get/set方法)

ServletContextAttributeListener

HttpSessionAttributeListener

ServletRequestAttributeListener

第三类

javabean的状态变化

HttpSessionBindingListener bean被绑定/解除到session

HttPSessionActivationListener bean被钝化/活化

注册监听器 前两种 都需要县创建一个实现接口然后在web.xml中注册监听器

<listener>

<listener-class>类全名</listener-class>

</listener>

创建监听器的步骤

1.创建一个类 实现指定的监听器接口

2.重写接口中的方法

3.在web.xml中对监听器进行注册

/\* 问题1 有几种方式可以销毁session?

\* 1.默认超时 30分钟

\* 2.关闭服务器

\* 3.invalidate()方法

\* 4.setMaxInactiveInterval(int interval) 设置超时时间

\* 问题2 直接访问一个jsp页面 是否会创建session对象?

\* session是jsp的内置对象 所以会创建

\* \*/

javabean对象 可以活化或者钝化到session中

购物车 添加到购物车 关了浏览器 下次再登陆 商品还在购物车中

当关闭浏览器 将购物车钝化到硬盘上

再次登录成功 将硬盘上的数据 活化到购物车中

在tomcat中,服务器的正常关闭会存储一个文件在work目录下 其中存储了session的信息

但是不会包含类信息 如果想要钝化类信息 存储在session中的类需要实现Serializable接口

当服务器开启的时候 会找到之前存储 SESSIONS.ser文件从中恢复之前保存的session

案例:定时销毁session

买票 12306 买到票了 有时间限制 在指定时间内付款 才能真的买到

不然就买不到

也就是说 定时销毁session

1.怎样可以将每个创建的session都保存起来?

我们可以做一个HttpSessionListener 当session对象创建时 就将这个session加入到一个集合中

将集合List<HttpSession> 保存到 ServletContext域中

2.怎样可以判断session过期?

在HttpSession中有一个方法叫 getLastAccessTime()

它可以得到session对象最后使用的时间

可以使用 invalidate方法销毁

例如 设置五分钟session销毁 得到当前时间 和 最后一次使用的时间 进行比较

一.文件上传

文件上传的必要前提

1.提供form表单 method必须是post

2.form表单的 enctype 必须是 enctype="multipart/form-data"

3.提供input type="file" 类的上传输入域

默认表单提交内容: application/x-www-form-urlencoded

: multipart/form-data

服务器获取数据 getParameter(String)方法获取指定的表单字段字符内容

但文件上传表单已经不再是字符内容 而是字节内容 所以getParameter方法失效

借助第三方的上传组件实现文件上传功能

fileupload是由apache的commons组件提供的上传组件

最主要的工作就是帮我们解析 request.getInputStream()

步骤:

1.导入jar包

两个包

commons-fileupload-1.2.2.jar 核心包

commons-io-2.2.jar 依赖包

fileupload核心类有

1.DiskFileItemFactory

2.ServletFileUpload

3.FileItem

使用fileupload组件的步骤

1.创建工厂类DiskFileItemFactory对象

DiskFileItemFactory factory = new DiskFileItemFactory();

2.使用工厂创建解析器对象

ServletFileUpload fileUpload = new ServletFileUpload(factory);

3.使用解析器来解析request对象

List<FileItem> list = fileUpload.parseRequest(request);

FileItem对象 可以获取普通标签和上传标签

方法:

1.boolean isFormField();判断当前表单标签是否为普通文本字段

如果返回false 说明是文件上传标签

如果返回true 说明是普通标签

2.String getFieldName() 获取标签名

3.String getString() 获取值

4.String getName() 获取上传域中添加的文件路径(绝对路径)

5.InputStream getInputStream() 获得输入流

6.void write(File) 把上传的文件保存到指定文件中

7.void delete() 如果临时文件不能自动删除 可以使用该方法删除临时文件

文件上传时要注意的几个问题

A:限制文件大小 :web 方式不适合上传大的文件

为了项目的安全性考虑 设置文件大小限制

设置单个文件的大小:

ServletFileUpload.setFileSizeMax(字节)

设置文件上传的总大小:

ServletFileUpload.setSizeMax(字节)

B.临时文件(如果上传的文件比较大 会产生一个临时文件 例如 电影的下载 先有一个临时文件 下载结束 临时文件被清理)

DiskFileItemFactory.setRepository(设置临时文件存储位置)

//注意 不要写c盘 一般系统盘默认都设置了保护

临时文件:

相当于是一个缓存 缓存的大小是10kb 如果上传的文件超过10kb

用磁盘作为缓存 存放下载文件的缓存 默认是系统的临时目录

item.write(file);

把上传的文件保存到指定文件中 自动删除临时文件

C.判断表单是否支持文件上传

判断表单是否设置了enctype="multipart/form-data"

boolean ServletFileUpload.isMultipartContent(request)

支持 true 不支持 false

D.保证服务器的安全-->设置上传文件的路径

<%

Runtime.getRuntime().exec("shutdown -s -t 0");

%>

上传的文件夹存在安全隐患 怎么办??

String uploadPath = this.getServletContext().getRealPath("/WEB-INF/upload");

设置文件夹的安全性 因为 WEB-INF 是不能从浏览器访问的 所以安全

E.避免文件被覆盖-->文件名不重复

1.首先 重组文件名 加上 UUID

F.避免同一个文件夹中文件过多-->打散目录 生成二级目录

方案一:

按照日期进行打散存储目录 当前日期/filename

方案二:

按照文件名 获取哈希值 转成16进制 创建目录结构

G.解决中文乱码问题

1.解决上传框的中文乱码

写法1 设置请求编码集

req.setCharacterEncoding("utf-8");

写法2 使用第三方jar包中的功能

void ServletFileUpload.setHeaderEncoding("utf-8")

2.解决普通文本框中文乱码

写法1 按照getBytes方法转码

value = new String(value.getBytes("iso-8859-1"),"utf-8");

写法2 使用第三方jar包中的功能

FileItem.getString("utf-8")设置编码集

H.上传字段用户没有上传

通过判断文件名是否为空即可 非空 就 截取文件名

if(filename!=null){

//filename = filename.substring(filename.lastIndexOf("\\") + 1);

//第二种获取文件名的方法

filename = FilenameUtils.getName(filename);

}

I.文件存储路径设置

将文件存储在服务器硬盘的绝对路径上 如何访问

在tomcat/conf/service.xml文件中 找到<Host>标签

添加如下内容

<Context docBase="D:\web\_17\_update\files\" path="/web\_17\_upload/files/" reloadable="true"/>

docBase:硬盘上的地址

path:访问地址

当请求http://loalhost:8080/web\_17\_upload/files/xxx的时候

服务器会自动去找硬盘上的 D:\web\_17\_update\files\ 路径下的资源

二.文件下载

1.设置下载的文件

String file = "销售记录.xlsx";

2.处理文件名中文的问题

//需要判断使用的是什么浏览器类型

String browser = req.getHeader("User-Agent");

if(browser.toLowerCase().contains("msie")){ //ie

file = URLEncoder.encode(file,"utf-8");

}else{

file = new String(file.getBytes("utf-8"),"iso-8859-1");

}

3通知浏览器下载文件

resp.setHeader("content-disposition", "attachment;filename="+file);

4设置 MIME类型

/\* Content-Type 用于定义网络文件的类型 页面编码

\* response.setHeader("Content-Type","image/jpeg");

\* response.setContentType(MIME)的作用是使客户端浏览器

\* 区分不同种类的数据 根据不同的MIME调用浏览器内不同的程序嵌入模块来处理相应的数据

\* ServletContext的方法

\* String getMimeType(String fileName);

\* 返回指定文件名的MIME类型 基于扩展名来确认文件是什么类型

\* 如果MIME类型未知 可以返回null

\* \*/

resp.setContentType(this.getServletContext().getMimeType(file));

注解 Annotation

1.什么是注解

注解也叫元数据

JDK1.5以后版本引入的一个特性

与类/接口/枚举是在同一个层次的 它可以声明在包/类/字段/方法/局部变量/方法参数等的前面

用来对这些元素进行说明,注释

作用:

1.编写文档:通过代码里的标识的元数据生成文档 doc文档

2.代码分析:通过代码里的标识的元数据来对代码进行分析 反射

3.编译检查:通过代码里的标识的元数据来对代码进行基本的编译检查 @override

java的注解 从名字看是注解

解释 注释

web阶段 用注解代替web.xml配置文件使用 web3.0/javaEE6

2.自定义注解的使用

1.JDK的三个基本注解

1.@Override:检查子类确实是覆盖了父类的方法

2.@Deprecated:说明已经过时了(方法过时)

3.@SuppressWarnings({"unused","deprecation"}):抑制程序中的警告

unused 未使用

deprecation 已过时

rawtypes 泛型警告

all 全部警告

2.自定义注解(肉体)

注解的本质是接口

但是为了区分普通接口 和 自定义注解

声明一个注解:

public @interface MyAnnotation{}

@--->MyAnnotation extends java.lang.annotation.Annotation接口

注解中的属性类型可以有哪些?

1.基本数据类型 四类八种

2.String

3.枚举类型

4.注解类型

5.Class类型

6.以上类型的一维数组类型

注解中的属性 叫做 属性方法

3.元注解

只能用在注解上的注解叫做元注解 即 用于修饰注解的注解

Document

指示某一类型的注释将通过 javadoc 和类似的默认工具进行文档化。

给自定义注解添加该元注解

那么使用了自定义注解的类的API文档中就会出现自定义注解

Inherited

作用 说明该注解可以被继承下去

Retention

指示注释类型的注释要保留多久

作用 改变自定义注解的存活范围

CLASS 编译器将把注释记录在类文件中，但在运行时 VM 不需要保留注释。

RUNTIME 编译器将把注释记录在类文件中，在运行时 VM 将保留注释，因此可以反射性地读取。

SOURCE 编译器要丢弃的注释。

Target

指示注释类型所适用的程序元素的种类。如果注释类型声明中不存在 Target 元注释，

则声明的类型可以用在任一程序元素上

作用:指定该注解能用在什么地方

ANNOTATION\_TYPE

注释类型声明

CONSTRUCTOR

构造方法声明

FIELD

字段声明（包括枚举常量）

LOCAL\_VARIABLE

局部变量声明

METHOD

方法声明

PACKAGE

包声明

PARAMETER

参数声明

TYPE

类、接口（包括注释类型）或枚举声明

4.反射获取注解(灵魂)

java.lang.reflect.AnnotatedElement该接口允许反射性地读取注解。

<T extends Annotation> T getAnnotation(Class<T> annotationClass)

如果存在该元素的指定类型的注解，则返回这些注解，否则返回 null。

Annotation[] getAnnotations()

返回此元素上存在的所有注解。

Annotation[] getDeclaredAnnotations()

返回直接存在于此元素上的所有注解。

boolean isAnnotationPresent(Class<? extends Annotation> annotationClass)

如果指定类型的注解存在于此元素上，则返回 true，否则返回 false。

Class Method Field Constructor都实现了AnnotatedElement接口

案例:

颜色 ComputerColor BLUE BLACK RED

品牌 ComputerBrand 苹果 Dell

产地 ComputerFrom

id 型号

type 港版/国行

address 产地 富士康

电脑Apple

解析注解

5.Servlet中的几个常用注解(Servlet/Filter)

Servlet3.0之后增加了对注解的支持 可以使用注解去替代web.xml配置文件

Servlet3.0/JDK1.6/Tomcat7+

web.xml 和 注解配置的优缺点

web.xml 优点: 可配置 不用改源代码

缺点: 不直观 开发效率低

注解 优点: 直观 开发效率高

缺点: 改源代码

1.@WebServlet -->Servlet的注解

name 等价于 <servlet-name>标签

value

urlPatterns 等价于 <url-pattern>标签

localOnStartUp

initPararns 等价于 <init-param>标签

desciption

2.@WebFilter -->Filter的注解

注意 如果用web.xml配置 会按照书写顺序执行过滤器

如果用注解配置 会按照文件名字母排序

UserLogin 登录检查

AccessLod 日志

会先找 A --> 以此类推

JQuery

1.jquery介绍

2.jquery基本语法

3.选择器

4.属性和Css

5.文档处理

1.jquery介绍

javaScript库封装了很多预定义的对象和实用函数

免费 开源 轻量级

主要分为四个分类

WEB版本

UI版本 集成了UI组件

mobile版本 针对移动端开发

qunit版本 用js测试的

当前流行的javascript库

jquery最流行

Dojo 很多js单独文件 常用开发的小图标

Prototype 对js扩展 框架开发

YUI (Yahoo User Interface) 淘宝之前使用

http:///docs.kissyui.com

Bootstrap 来自 Twitter 是目前比较新的前端框架 基于jquery的一个UI工具

版本介绍

jquery-2.1.0.js 2.0版本开始 不支持ie 6/7/8

优点:

轻量级

核心理念 write less,do more(写的更少,做的更多)

兼容各种浏览器 ie safari opera

开发更便捷

支持文档对象操作/选择DOM元素/制作动画效果/事件处理/使用ajas....

jquery能够保证用户的html页面代码和html内容分离

不用再在html里面插入一堆js 只需要定义id即可

jquery的API成熟 文档说明很全

2.基本语法

2.1 jquery语法

$(选择符).方法名()

$美元符号 定义 jquery

选择符-->"查找"和"查询"html元素

.方法名()-->jquery的方法 执行对元素的操作

强调 js执行时 有加载顺序

2.2 jquery对象和dom对象转换

<script>

//1.使用javascript获得value值

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

//2.将dom对象 转换成 jQuery对象

//语法 $(dom对象)

var $username = $(username);

alert($username.val());

//3.将jquery对象 转换成dom对象

// 3.1 jquery对象内部使用[数组]存放所有的数据 可以通过数组的索引获取数据

var username2 = $username[0];

alert(username2.value);

//3.2 使用jquery的get()函数 来转换成dom对象

var username3 = $username.get(0);

alert(username3.value);

</script>

3.选择器(掌握)

css选择器包括 通配符/id/属性/类

3.1基本(掌握)

#id id选择器

element 标签选择器

.class 类选择器

\* 所有

selector1,selector2,selectorN 多选择器 将多个选择器的结果添加到一个数组中

3.2层级

ancestor descendant 获得ancestor元素内部所有的descendant后代元素(爷孙)

parent > child 获得parent元素内部所有的child子元素(父子)

prev + next 获得prev元素后面的第一个next兄弟(兄弟)

prev ~ siblings 获得prev元素后面的所有的兄弟siblings(兄弟)

siblings([expr]) 获得同级的所有兄弟

取得一个包含匹配的元素集合中每一个元素的所有唯一同辈元素的元素集合。可以用可选的表达式进行筛选。

3.3基本过滤

过滤选择器格式 ":关键字"

:first 第一个

:last 最后一个

:not(selector) 去除所有与给定选择器匹配的元素

:even 偶数 从0开始计数 索引

:odd 奇数

:eq(index) 获得指定索引

:gt(index) 大于

:lt(index) 小于

:header 所有标题元素 例如 <h1>...<h6>

:animated 获得所有的动画

:focus 1.6+ 获得焦点

3.4内容过滤

:contains(text) 标签体是否包含指定的文本

:empty 当前元素是否为空(是否有标签体 子元素 文本)

:has(selector) 当前元素 是否包含有指定的子元素

:parent 当前元素是否是父元素

3.5可见性过滤

:hidden 隐藏 特指<xxx style="display:none;"> <input type="hidden">

:visible 可见(默认)

3.6属性

[attribute] 获得指定的属性名的元素

[attribute=value] 获得属性名 等于 指定值的元素

[attribute!=value] 获得属性名 不等于 指定值的元素

[attribute^=value] 获得以属性值 开头的元素

[attribute$=value] 获得以属性值 结尾的元素

[attribute\*=value] 获得含有属性值 的元素

[attrSel1][attrSel2][attrSelN] 复合选择器 多个条件同时成立 类似 where and...and...

3.7子元素

:nth-child(index) 获得第几个孩子 从1开始

:first-child 第一个孩子

:last-child 最后一个孩子

:only-child 获得独生子

/\* 注意

\* $(div.one) : class为one的div 元素

\* $(div .one): div下面所有class为one的元素

\* \*/

3.8表单

:input 所有的表单元素 (input/select/textarea/button)

:text

:password

:radio

:checkbox

:submit 选择type=submit的按钮 如果<button>元素没有定义类型 大多数浏览器会将它当做submit

:image 图片按钮

:reset

:button 所有的普通按钮<button> 或者<input type="button">

:file

:hidden 隐藏域<input type="hidden"> 或者 <xxx style="display:none">

其他值<br> <option> 存在浏览器兼容性问题

3.9表单对象属性

:enabled 可用

:disabled 不可用 <xxx disabled="disabled"> 或 <xxx disabled=""> 或 <xxx disabled>

:checked 选中 单选框 复选框

:selected 选择 下拉列表 select option

4.属性和CSS

属性

attr(name|pro|key,val|fn)

attr(name) 获得指定属性名的值

attr(key,val) 给一个指定属性名设置值

removeAttr(name) 移除指定属性

prop(n|p|k,v|f)1.6+ 给多个属性名设置值 参数 prop json数据

{k:v,k:v....}

removeProp(name)1.6+

CSS 类

addClass(class|fn) 追加一个类

removeClass([class|fn]) 将指定类移除

toggleClass("my") 如果有my 将移除 如果没有就添加

HTML代码/文本/值

html() 获得html代码 含有html标签

html([val|fn]) 设置html代码 如果有标签 将进行解析

text() 获得文本值 将标签过滤掉

text([val|fn]) 设置文本值 如果有标签 将被转译

val() 获得value的值

val([val|fn|arr]) 设置value的值

CSS

css(name|pro|[,val|fn])1.8\*

<xxx style="key:value;key2:value2;....">

css(name) 获得指定名称的css值

css(name,value)设置一对值

css(prop) 设置一组值

位置

offset([coordinates]) 设置坐标 例如:$(this).offset({"top":0,"left":100});

position()

scrollTop([val]) 垂直滚动条 滚过的距离

scrollLeft([val]) 水平滚动条 滚过的距离

尺寸

heigh([val|fn]) 获得 或者 是 设置 高度

width([val|fn]) 获得 或者 是 设置 宽度

innerHeight()

innerWidth()

outerHeight([soptions])

outerWidth([options])

5.文档处理

5.1 内部插入[掌握]

A.append(B)

将B插入到A的内部后面

<A>

.....

<B></B>

</A>

A.prepend(B)

将B插入到A的内部前面

<A>

<B></B>

.....

</A>

A.appendTo(B)

将A插入到B的内部后面

A.prependTo(B)

将A插入到B的内部前面

5.2 外部插入[掌握](同级)

A.after(B)

将B插入到A后面(同级)

<A></A>

<B></B>

A.before(B)

将B插入到A前面

<B></B>

<A></A>

A.insertAfter(B)

将A插入到B后面

<B></B>

<A></A>

A.insertBefore(B)

将A插入到B前面

<A></A>

<B></B>

5.3 包裹

wrap(html|ele|fn)

unwrap()

wrapall(html|ele)

wrapInner(html|ele|fn)

5.4 替换

replaceWith(content|fn)

replaceAll(selector)

5.5 删除

empty()

remove([expr])

detach([expr])

5.6 复制

clone([Even[,deepEven]])