雷丰阳

从入门到精通

建议零售价：$9.9

JavaWeb知识

从入门到精通

目录

[一、jQuery 7](#_Toc457904477)

[1、简介： 7](#_Toc457904478)

[2、HelloWorld 7](#_Toc457904479)

[3、$：jquery的核心函数 8](#_Toc457904480)

[4、核心函数用处： 8](#_Toc457904481)

[5、dom对象？jquery对象？ 8](#_Toc457904482)

[6、jquery选择器 9](#_Toc457904483)

[7、dom操作（增删改查）； 11](#_Toc457904484)

[1、dom查询 11](#_Toc457904485)

[2、dom属性操作 12](#_Toc457904486)

[3、dom增删改 12](#_Toc457904487)

[8、css操作 13](#_Toc457904488)

[9、事件 13](#_Toc457904489)

[1）、文档加载 13](#_Toc457904490)

[2）、事件绑定 14](#_Toc457904491)

[3）、事件对象 14](#_Toc457904492)

[10、动画 15](#_Toc457904493)

[二、XML 16](#_Toc457904494)

[1、xml简介 16](#_Toc457904495)

[2、xml基本语法 16](#_Toc457904496)

[3、xml解析 17](#_Toc457904497)

[1、解析体系 17](#_Toc457904498)

[2、dom4j解析 18](#_Toc457904499)

[4、xpath 20](#_Toc457904500)

[三、Tomcat-Web环境搭建 21](#_Toc457904501)

[1、下载： 21](#_Toc457904502)

[2、安装： 21](#_Toc457904503)

[3、启动排错： 22](#_Toc457904504)

[4、tomcat目录结构 23](#_Toc457904505)

[5、eclipse整合tomcat 24](#_Toc457904506)

[6、eclipse整合tomcat的镜像 25](#_Toc457904507)

[四、HTTP 29](#_Toc457904508)

[1、简介： 29](#_Toc457904509)

[2、数据格式 29](#_Toc457904510)

[报文格式： 29](#_Toc457904511)

[请求报文： 29](#_Toc457904512)

[响应报文： 30](#_Toc457904513)

[响应状态码： 30](#_Toc457904514)

[MIME类型 31](#_Toc457904515)

[五、Servlet 32](#_Toc457904516)

[1、简介 32](#_Toc457904517)

[2、HelloWorld 32](#_Toc457904518)

[3、生命周期 33](#_Toc457904519)

[4、init（ServletConfig config） 33](#_Toc457904520)

[5、ServletContext 34](#_Toc457904521)

[6、HttpServlet 34](#_Toc457904522)

[7、HttpServletRequest 35](#_Toc457904523)

[8、HttpServletResponse 36](#_Toc457904524)

[9、转发和重定向 36](#_Toc457904525)

[10、路径问题 36](#_Toc457904526)

[11、编码问题 37](#_Toc457904527)

[六、JSP 38](#_Toc457904528)

[1、jsp简介 38](#_Toc457904529)

[2、运行原理 38](#_Toc457904530)

[3、JSP基本语法 39](#_Toc457904531)

[1）、模板元素： 39](#_Toc457904532)

[2）、JSP脚本片段 39](#_Toc457904533)

[3）、JSP表达式 39](#_Toc457904534)

[4）、JSP声明 40](#_Toc457904535)

[5）、JSP指令 40](#_Toc457904536)

[6）、JSP标签（动作元素，Action元素） 41](#_Toc457904537)

[4、九大隐含对象 42](#_Toc457904538)

[七、EL 45](#_Toc457904539)

[1、简介 45](#_Toc457904540)

[2、El语法 45](#_Toc457904541)

[3、EL内置11个对象 45](#_Toc457904542)

[1、PageContext： 45](#_Toc457904543)

[2、四个域对象相关 46](#_Toc457904544)

[3、5个和HTTP协议相关 46](#_Toc457904545)

[4、initParam 46](#_Toc457904546)

[八、JSTL 47](#_Toc457904547)

[1、简介 47](#_Toc457904548)

[2、JSTL标签库 47](#_Toc457904549)

[3、核心库常用标签 47](#_Toc457904550)

[c:set： 47](#_Toc457904551)

[c:out： 48](#_Toc457904552)

[c:remove： 48](#_Toc457904553)

[c:if： 49](#_Toc457904554)

[c:choose c:when c:otherwise 49](#_Toc457904555)

[c:forEach 49](#_Toc457904556)

[4、函数库 50](#_Toc457904557)

[九、Cookie 51](#_Toc457904558)

[1、简介 51](#_Toc457904559)

[2、cookie的创建 51](#_Toc457904560)

[3、Cookie的读取 51](#_Toc457904561)

[4、修改cookie 52](#_Toc457904562)

[5、cookie的生存时间； 52](#_Toc457904563)

[6、删除cookie 52](#_Toc457904564)

[7、cookie设置路径 53](#_Toc457904565)

[8、cookie运行原理： 53](#_Toc457904566)

[十、Session 54](#_Toc457904567)

[1、简介： 54](#_Toc457904568)

[2、Session的创建及获取 54](#_Toc457904569)

[3、会话控制原理： 54](#_Toc457904570)

[4、session生存时间 55](#_Toc457904571)

[5、活化钝化： 56](#_Toc457904572)

[6、session的应用 56](#_Toc457904573)

[十一、Filter 57](#_Toc457904574)

[1、简介： 57](#_Toc457904575)

[2、HelloWorld 57](#_Toc457904576)

[3、FilterConfig、FilterChain 58](#_Toc457904577)

[4、url-pattern 58](#_Toc457904578)

[5、Filter生命周期 58](#_Toc457904579)

[6、dispatcher 59](#_Toc457904580)

[7、Filter链 59](#_Toc457904581)

[十二、AJAX 60](#_Toc457904582)

[1、简介： 60](#_Toc457904583)

[2、js原生ajax请求 60](#_Toc457904584)

[1）、Get： 60](#_Toc457904585)

[2）、post 61](#_Toc457904586)

[3、Jquery发送AJAX 62](#_Toc457904587)

[4、JSON数据 62](#_Toc457904588)

[1、JSON对象转字符串；方便传输 63](#_Toc457904589)

[2、JSON字符串转为对象；方便解析 63](#_Toc457904590)

[3、java操作json 63](#_Toc457904591)

[5、jQuery发送ajax 63](#_Toc457904592)

[1）、$.get(url,[data],[callback],[type])： 63](#_Toc457904593)

[2）、$.post(url,[data],[callback],[type]); 64](#_Toc457904594)

[3）、$.getJSON(); 64](#_Toc457904595)

[4）、$.ajax(); 64](#_Toc457904596)

[十三、国际化 65](#_Toc457904597)

[1、简介 65](#_Toc457904598)

[2、国际化的几个核心类 65](#_Toc457904599)

[1）、Locale 65](#_Toc457904600)

[2）、日期格式化 DateFormat 65](#_Toc457904601)

[3）、数字格式化 NumberFormat 66](#_Toc457904602)

[4）、货币格式化 NumberFormat 66](#_Toc457904603)

[5）、ResourceBundle 管理资源文件 66](#_Toc457904604)

[6）、信息格式化 MessageFormat 66](#_Toc457904605)

[3、国际化在WEB中的应用 67](#_Toc457904606)

[十四、文件上传下载 67](#_Toc457904607)

[1、文件上传 67](#_Toc457904608)

[2、文件下载 69](#_Toc457904609)

[书城项目 70](#_Toc457904610)

[一、第一阶段 使用jquery进行表单验证 70](#_Toc457904611)

[二、第二阶段 完成登陆注册功能； 71](#_Toc457904612)

[软件架构： 71](#_Toc457904613)

[项目架构： 71](#_Toc457904614)

[开发项目 73](#_Toc457904615)

[三、第三阶段 项目优化 76](#_Toc457904616)

[1、将HTML页面转为jsp页面 76](#_Toc457904617)

[2、将菜单抽取出来。 76](#_Toc457904618)

[3、将页面的公用的引入信息也抽取出来 76](#_Toc457904619)

[4、页面加入错误提示信息 77](#_Toc457904620)

[5、一个Servlet处理多个请求。 78](#_Toc457904621)

[6、自动化封装对象 79](#_Toc457904622)

[四、第四阶段 后台图书管理功能 前端图书展示 80](#_Toc457904623)

[五、第五阶段 登陆、注销、注册验证码、购物车 86](#_Toc457904624)

[1、登陆 86](#_Toc457904625)

[2、注销 86](#_Toc457904626)

[3、验证码 86](#_Toc457904627)

[4、购物车 87](#_Toc457904628)

[六、第六阶段 订单模块完成 90](#_Toc457904629)

[1、订单模块 90](#_Toc457904630)

[2、功能分析 90](#_Toc457904631)

[七、第七阶段 优化、引入Filter 96](#_Toc457904632)

[1、问题一、库存销量问题： 96](#_Toc457904633)

[2、结账完成直接刷新，会生成新的订单。 96](#_Toc457904634)

[3、和用户订单相关的所有操作都需要验证用户登陆 96](#_Toc457904635)

[4、使用filter控制事务 97](#_Toc457904636)

[八、第八阶段 使用ajax 101](#_Toc457904637)

[1、注册检查用户名 101](#_Toc457904638)

# 一、jQuery

## 1、简介：

JavaScript的第三方库、第三方框架。有一个外部的jquery.js文件。

没有min的是未压缩版 的，有min的是压缩的。

优点：

1、简化js开发

2、隐式迭代

3、链式操作

## 2、HelloWorld

要求，点击按钮弹出HelloWorld

1）、导入jquery库

<script src=”jquery.js”> 不能编写代码 </script>

2）、在新的script标签中编写jquery代码

<script type=””>

//jquery代码编写

</script>

3）、

$(function(){//和window.onload功能一样

$(“#btn”).click(function(){

//

alert(“HelloWorld”);

});

})

## 3、$：jquery的核心函数

alert($);//function(){};

## 4、核心函数用处：

1）、传入一个选择器字符串；按照选择器查找出元素 $(“#btn”)

2）、传入一个函数：文档加载完成以后执行 $(function(){ //文档加载完成以后执行方法体})

3）、传入一个HTML片段：创建出一个HTML元素节点；$(“<p>你好</p>”)

4）、传入一个dom对象：把他包装成一个jquery对象。

## 5、dom对象？jquery对象？

dom对象：使用原生的js查找以及创建的对象都叫dom对象

getElementsByXXX

jquery对象：使用jquery的方法找到以及创建的都是jquery对象

dom对象只能用原生的js方法。jquery定义了很多丰富的简化开发的方法。

我们为了让dom对象也能用jquery的方法：

将dom对象转为jquery对象：使用核心函数包装。$(dom对象)

我们也可以将jquery对象转为dom对象：

jquery对象是把所有的dom封装成了一个数组：

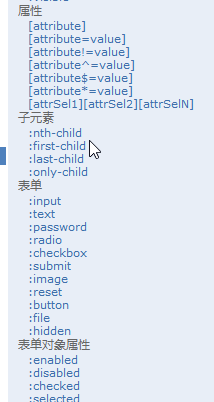
使用数组下标的形式就行了：$(“#btn”)[0]

## 6、jquery选择器

选择器？：快速查找元素。

语法：$(选择器字符串)





选择器：

写法：$(选择器字符串)；

1）、连着写：选择器字符串1选择器字符串2选择器字符串n

效果：多个选择器选出的元素的交集

2）、按照层级关系写：

选择器1 选择器2：后代

选择器1 > 选择器2 > 选择器3：父子

选择器1 + 选择器2：下一个兄弟选择器

~：所有后面的兄弟们

选择器1,选择器2,选择器3…….：并集关系

选择器能写哪些？：jquery文档中的选择器字符串全部能写

## 7、dom操作（增删改查）；

### 1、dom查询

jquery还提供了一些查询方法。

熟练掌握：

<div>

<p>

<b>你好</b>

<a>我是链接</a>

<h1>

<a id=”a1”>链接2</a>

</h1>

</p>

</div>

[children([expr])](mk:@MSITStore:C:\Users\lfy\Desktop\尚硅谷WEB_雷丰阳_JAVAEE版\文档\jQueryAPI_1.7.1_CN.chm::/children.html)；查找子元素

$(“p”).children(”a”):选出p标签中所有的子元素但是他是a标签

结果：<a>我是链接</a>

[find(expr|obj|ele)](mk:@MSITStore:C:\Users\lfy\Desktop\尚硅谷WEB_雷丰阳_JAVAEE版\文档\jQueryAPI_1.7.1_CN.chm::/find.html) ；查找后代元素

$(“div”).find(“a”):选出div中所有的后代a

结果：<a>我是链接</a>、<a>链接2</a>

[parent([expr])](mk:@MSITStore:C:\Users\lfy\Desktop\尚硅谷WEB_雷丰阳_JAVAEE版\文档\jQueryAPI_1.7.1_CN.chm::/parent.html)；查找父元素

$(“#a1”).parent();

结果：找到h1元素

[parents([expr])](mk:@MSITStore:C:\Users\lfy\Desktop\尚硅谷WEB_雷丰阳_JAVAEE版\文档\jQueryAPI_1.7.1_CN.chm::/parents.html)；查找祖先元素

$(“#a1”).parents();

结果：h1、p、div、body….

$(“#a1”).parents(“p”);

结果：p

### 2、dom属性操作

* + [attr(name|pro|key,val|fn)](mk:@MSITStore:C:\Users\lfy\Desktop\尚硅谷WEB_雷丰阳_JAVAEE版\文档\jQueryAPI_1.7.1_CN.chm::/attr.html)
    - $（”input:eq(1)”）.attr();
    - 两种用法：
      * 1、直接传递属性名。获取属性值
      * 2、传递key-value，设置属性值
  + [removeAttr(name)](mk:@MSITStore:C:\Users\lfy\Desktop\尚硅谷WEB_雷丰阳_JAVAEE版\文档\jQueryAPI_1.7.1_CN.chm::/removeAttr.html) //移除某个属性

[prop(name|pro|key,val|fn)](mk:@MSITStore:C:\Users\lfy\Desktop\尚硅谷WEB_雷丰阳_JAVAEE版\文档\jQueryAPI_1.7.1_CN.chm::/prop.html)1.6+

//获取属性值。

//区别：prop他经常用来获取元素的原生属性比较好

<a href=<http://www.baidu.com> aid=”1”>链接</a>

自定义的属性用attr比较好，他可以返回正确的值。$(“a”).attr(“aid”)返回1;

$(“a”).prop(“aid”)返回undefined

* + [removeProp(name)](mk:@MSITStore:C:\Users\lfy\Desktop\尚硅谷WEB_雷丰阳_JAVAEE版\文档\jQueryAPI_1.7.1_CN.chm::/removeProp.html)1.6+ //移除属性
  + [html([val|fn])](mk:@MSITStore:C:\Users\lfy\Desktop\尚硅谷WEB_雷丰阳_JAVAEE版\文档\jQueryAPI_1.7.1_CN.chm::/html.html) :

1、$(“a”).html(“<b>链接</b>”)：把a中的内容替换成b标签。浏览器会显示成加粗的链接

2、$(“a”).html()：获取a中的html代码

* + [text([val|fn])](mk:@MSITStore:C:\Users\lfy\Desktop\尚硅谷WEB_雷丰阳_JAVAEE版\文档\jQueryAPI_1.7.1_CN.chm::/text.html)

1、$(“a”).text(“<b>链接</b>”)：把a中的内容替换成文本。浏览器会显示链接的名就叫

<b>链接</b>

2、$（”a”）.text()；获取a中的文本。只会拿出所有文本

* + [val([val|fn|arr])](mk:@MSITStore:C:\Users\lfy\Desktop\尚硅谷WEB_雷丰阳_JAVAEE版\文档\jQueryAPI_1.7.1_CN.chm::/val.html)

1、$(“input[name=’username’]”).val()：获取文本框的value值

2、$(“input[name=’username’]”).val(“admin”)：设置文本框的value为admin

3、val()可以传递一个数组：用于checkbox,select,radio的选中

### 3、dom增删改

1）、dom增加：

a.append(b)：a里面添加b，会把b放在a内部的后面

a.appendTo(b)：a添加到b里面，会把a放在b内部的后面

a.after(b)：a的后面添加b，把b加到a后面

a.insertAfter(b)：把a插入到b的后面。

2）、dom删除

$(“p”).remove();直接删除p。整个元素节点都没了

$(“p”).empty();掏空p元素。p元素还在，里面的东西都清空了

3）、dom修改

a.replaceWith(b)：用b替换a

a.replaceAll(b)：用a替换所有的b

## 8、css操作

css();

1)、$(“div”).css(“background-color”)：返回样式值

2)、$(“div”).css(“background-color”,”blue”)：设置样式值

## 9、事件

### 1）、文档加载

1、window.onload = function(){} & $(function(){})

window.onload：文档加载完成，后完成

$(function(){})：dom树绘制完成，先完成的

2、window.onload不能写多次，写多次只有最后一次生效。

$(function(){})可以写多次都可以执行。本质是方法调用

### 2）、事件绑定

1、事件：表示用户和浏览器的交互行为。

jquery中的事件。文档中有所有的事件列表。

绑定事件：

1）、常规绑定：

$(“a”).事件名(回调函数)

2）、绑定多事件：

链式调用：$(“a”).click(function(){}).mouseover(function(){});

bind()方法：

$(“a”).bind(“事件名 事件名2”,回调函数)；用空格隔开多事件

3）、one()：绑定一次性事件。

4）、live()：为后来动态添加的元素也能绑定上事件。

事件移除：

unbind()：

1、$(“a”).unbind(“事件名”)：移除指定事件

2、$(“a”).unbind()：移除所有事件

### 3）、事件对象

事件对象：浏览器在每次发生事件的时候都会把当前事件的详细信息封装为一个对象。

获取事件对象：

浏览器在每次调用回调函数的时候会把事件对象自动传入。但是我们以前在定义回调函数的时候没有参数列表。我们要使用只需要在声明回调函数的时候参数列表中加上一个参数。这个参数就是用来接收我们浏览器传递过来的事件对象的

$(“a”).click(function(event){

//我们以前没有参数表

//event：代表当前事件的详细信息的对象

// event.type :事件类型

//alert(event.type)

alert(event.pageX+”🡪”+event.pageY)

});

## 10、动画

show()：显示元素

hide()：隐藏元素

toggle()：切换显示/隐藏状态

sildeUp()：向上收缩

sildeDown()：向下展开

sildeToggle()：切换展开/收缩状态

fadeIn()：淡入

fadeOut()：淡出

fadeTo()：淡化到某个透明度

fadeToggle()：切换淡入/淡出状态

animate()：高级动画

传入元素最终的样式。传入时间。在规定时间将元素变为最终的样子

# 二、XML

## 1、xml简介

xml：使用统一的格式保存数据。

xml：可扩展标记语言。html：超文本标记语言。标记语言=标签语言

用途：

程序配置文件：c3p0-config.xml、

网络传输：即时通讯（QQ）。openfire（即时通讯服务器软件）。

传递消息：

来自哪里、去哪里、消息内容、发送时间、回执

<message>

<to>tom</to>

<from>jerry</from>

<content>你好</content>

</message>

## 2、xml基本语法

1、语法规范：

1）、第一行是文档声明，可以没有，如果有必须顶格写

2）、标签区分大小写 <a></a>

3）、标签必须正确关闭<br/> <a></a>，而且可以嵌套但不能交叉嵌套

4）、标签可以有属性，但是如果有属性就必须有值<a href x>，而且值必须用双引号括起来

5）、xml文档中只能有一个根标签。根标签是包含其他所有标签的（除过文档声明、处理指令）

2、其他语法：

注释：<!-- -->

转义字符：和html中转义字符是通用的 < 🡪 &lt;…..

CDATA区：

功能：原封不动的输出CDATA区中的所有内容，即使是有转义字符，翻译显示的时候都不会转义。

写法： <! [CDATA[数据区]]>

处理指令： <?处理指令?>

## 3、xml解析

### 1、解析体系

两个分支：

DOM解析：官方标准。

原理：一次性将文档全部加载进来。生成一个DOM对象。这个对象里面就是详细信息

优点、缺点：

优：可以增删改

缺：耗内存、效率低（只要文档第一行，都会读取所有）

SAX解析：第三方标准。纳入官方文档

原理：一行一行读取，每读取一行调用回调函数进行处理。他是基于回调机制的。

优点、缺点：

优：查询速度快。最坏的情况才是全部读取。

缺：不能增删改。只能查询。只能前进不能倒退。

DOM体系

---🡪标准的DOM解析：

---🡪第三方DOM4j解析。

SAX体系

---🡪标准的SAX解析

---🡪第三方pull解析。

### 2、dom4j解析

1、解析步骤：导包、dom4j-1.6.1.jar

1）、创建解析器

SAXReader reader = new SAXReader();

2）、解析文件,返回DOM对象

Document document = reader.read(“文件的路径”);

3）、获取根节点

Element rootElement = document.getRootElement();

4）、遍历查询修改。。。

常用方法：

1、获取元素

List<Element> eles = rootElement.elements();获取所有的子元素

List<Element> eles = rootElement.elements(“student”);获取所有student子元素

List<Attribute> attrs = element.attributes();//获取当前标签的所有属性

Iterator<Element> iterator = rootElement.elementIterator();获取元素的迭代器，也是找子元素。可以通过递归迭代的方法获取所有元素

2、读取元素内容：

String name = ele.getName();//获取元素名，标签名

String text = ele.getText();//获取标签体内的文本内容

String attrValue = ele.attributeValue(“属性名”);//获取标签对应的属性

几个快捷方法：

element.elementText(“标签名”);//获取标签名为传入的参数的子标签的文本值

3、整个dom操作中会有三个对象

Document：整个xml文档对象

document.getRootElement();//获取根标签

Element：代表元素对象。元素=标签

getName()：获取元素名。标签名

getText()：获取标签体重文本值

Attribute：代表属性节点

getName()：获取属性名

getValue()：获取属性值

ele.attributeValue(“属性名”)：快速获取属性值

## 4、xpath

作用：快速的查询元素。

语法格式：类似与写路径的格式。

/AAA：从根目录下获取AAA。获取AAA这个根标签

//BBB：获取文档中所有的BBB

/AAA/BBB：根AAA下的BBB标签

参照xpath高手必备

用法：

1、导包：jaxen-1.1-beta-6.jar

2、使用：

之前的dom4j操作都是一样的。只不过在查询元素的时候可以使用两个快速的查询方法：

Node：节点；元素节点、属性节点、文本节点Element Attribute

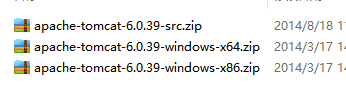
List<Element> ele.selectNodes(xpath表达式)：返回符合表达式的所有节点

Element ele\_return =(Element) ele.selectSingleNode(xpath表达式)：返回符合表达式的单个元素

# 三、Tomcat-Web环境搭建

## 1、下载：

去百度搜索tomcat,点击相应连接下载



## 2、安装：

1、解压压缩包到指定的位置：解压到当前文件夹

2、tomcat是java编写的，所以tomcat软件的运行依赖java运行环境

1）、需要配置JAVA\_HOME环境变量：

右键计算机--🡪属性-🡪高级系统设置-🡪环境变量

配置java-home

没有：新建环境变量---》变量名：JAVA\_HOME

-🡪变量值：jdk/jre的安装根目录

C:\Program Files\Java\jdk1.7.0\_07千万不能进其他文件夹

CATALINA\_HOME：指tomcat的安装根目录

2）、服务器就能启动了：

启动服务器：

进入tomcat的bin目录；双击startup.bat

怎么才是启动成功呢？

1）、

2）、访问服务器：

localhost:8080

3）、服务器的启动和停止：

启动：双击startup.bat

停止：双击shutdowm.bat

总结：

先配置好JAVA\_HOME。双击startup.bat启动服务器。访问localhost:8080看看是否可以成功访问. 127.0.0.1:8080(本机地址) 192.168.10.165:8080(外部访问地址)

[www.jd.com（域名-ip地址](http://www.jd.com（域名-ip地址) 211.58.16.98:80）如果是访问80端口是不用输端口号的

## 3、启动排错：

1、改端口

默认是访问8080：

可以在conf文件夹中修改server.xml：Connector port=8080

将port改为新的。不能与外部冲突

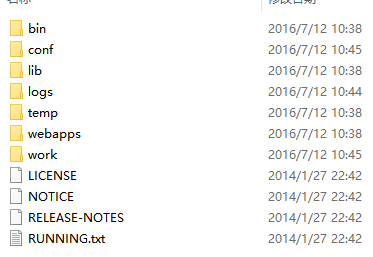
现象：双击starup.bat一闪而过。

原因：很多，如何查看到底是哪个原因导致的；JAVA\_HOME配置问题。其他问题

双击startup.bat其实最终会运行catalina.bat。catalina.bat会有错误报告。

可以在cmd窗口使用catalina run命令来观察错误。

## 4、tomcat目录结构



bin：这里面都是tomcat的脚本文件，可执行文件，批处理.bat文件，这里就包括服务器启动和停止

conf：保存tomcat的配置文件。

比如：，保存服务器配置信息的。包括服务器端口号，以及服务器其他处理命令等。

lib：都保存的是tomcat运行需要的第三方jar包。tomcat他是使用java编写的

logs：保存tomcat运行期间的日志信息

temp：这里面都是保存一些临时文件。这里的文件是可以删除的

webapps：这里保存的都是每个web项目。一个服务器上可以运行很多项目。把每个项目应该都放在webapps。webapps中的每个文件夹都代表一个项目。默认是访问ROOT下的index.html。localhost:8080就会访问root文件夹下的index.html;

访问其他：访问webapps/docs 怎么访问？

localhost:8080/docs 返回docs下的index.html

192.168.10.165:8080/项目名 返回对应项目下的index.html

work：保存tomcat在工作期间会翻译和编译一些文件。保存工作期间产生的一些文件

## 5、eclipse整合tomcat

开发是在eclispe下开发。每次如果都在外界去启动和停止服务器会很麻烦。

eclispe可以整合tomcat；tomcat就可以在eclipse中方便的启动和运行。

java透视图：这种模式下，适合开发javaSE程序

web透视图：这种模式下，适合开发静态网页程序。静态web工程

javaEE透视图：这种模式下，适合开发动态服务器程序。动态web工程

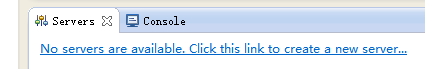
动态？静态？

静态：不具备调用java代码，访问数据库..这些能力。页面写的什么就是什么

动态：页面是可以和数据库进行沟通，调用java。。。。

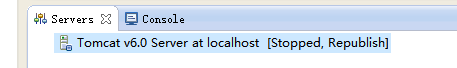
以后使用javaEE透视图;

整合服务器：



点击链接---🡪创建新服务器的对话框-🡪选择apache文件夹-🡪选择与安装版本对应的服务器-🡪指定tomcat的安装路径(指定tomcat的安装根目录)-🡪点击finish

至此，tomcat将服务器就整合完毕。



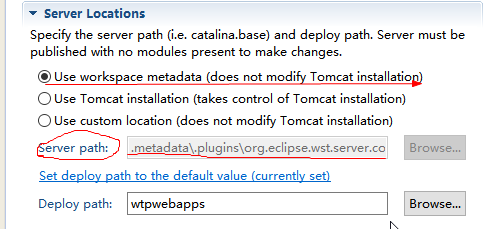
在eclipse中启动停止服务器：

启动---》点击右键-🡪start  
 停止-🡪点击右键🡪stop

## 6、eclipse整合tomcat的镜像

其实eclipse把tomcat整合进来以后。他实际上是额外的保存了一个tomcat的镜像（副本）。这个副本和外部我们安装的这个tomcat没关系了。

副本的存放位置：

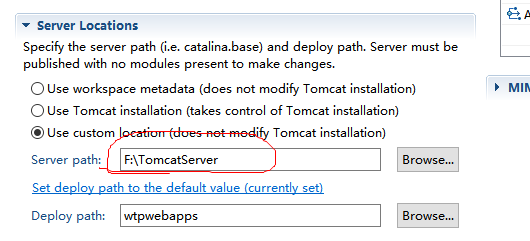


工作空间下\.metadata\.plugins\org.eclipse.wst.server.core\tmp0

1、服务器的镜像位置。

2、默认应用程序都在wtpwebapps。

我们可以修改服务器镜像位置。为了以后方便使用和查找



7、创建工程并运行

1、静态web工程：static web project

2、动态web工程：

注意点：必须选择web版本选择2.5

3、访问web工程。

1）、访问某个具体资源的时候：

右键-🡪run as🡪run on server

2）、在工程身上点击右键🡪run as 🡪 run on server

默认返回工程下的index.html页面。如果没有首页就会404（资源找不到）

3）、工程发布在服务器那里了？

动态web工程部署到服务器的时候，只会把WebContent下的所有内容发布上去。还会把java代码编译成.class文件放在WEB-INF/classes/全包名



.settings ：每个工程都会有，eclipse自动创键的，里面都是eclipse对当前工程的描述信息

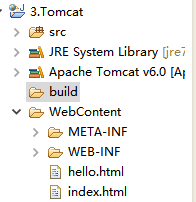
build：存放所有编译后的.class文件。这里的内容最后都会发布服务器下项目名/WEB-INF/classes

src：存放源码

WebContent：存放当前工程需要的文件。html,….,img

一般切换到包视图下；

搜索:package。



以后java代码都是在src下，网页文件等资源放在WebContent；

当工程运行发布后；



特别注意：

发布的时候没有WebContent文件夹

META-INF：描述工程的一些信息。xxx

WEB-INF：工程的所有配置文件，以及java代码翻译后的class文件

classes：存放.class文件

lib：里面存放工程运行时要使用的所有的jar包。所以我们以后编写工程的时候如果引入第三方包，直接把第三方包放到lib下。自动添加到build-path

web.xml：这个是当前web工程的配置文件。

WEB-INF下的内容不能通过浏览器访问。WEB-INF下的所有内容都是受保护的。

# 四、HTTP

## 1、简介：

HTTP 超文本传输协议

规定浏览器和服务器传输数据的格式

## 2、数据格式

浏览器-🡪服务器：发送请求，请求报文

服务器-🡪浏览器：发送响应，响应报文

### 报文格式：

HTTP协议规定，报文的格式如下

报文首部

空行

报文主体

### 请求报文：

请求首行： 请求方式 请求地址 协议版本号

请求头： key-value 回车 key-value

空行： 分割报文首部信息和报文体

请求体： 封装发送的数据的。表单提交

get：

没有请求体。如果有发数据，get请求会将数据放在请求首行的请求地址后面

url?key=value&key=value 多个数据都用&连接，数据的开始必须有一个问号。

这个问号就代表后面所有都是提交过来的数据

post：

有请求体，如果有提交数据，post请求会把这些数据放在请求体中

HTTP：超文本。 HTTPS：安全的传输协议

get： 提交的数据量有限制，原因url地址最多256个字符。不能提交大量数据

post：把提交的数据都放在请求体中了，所有数据量显示，可以提交大量数据

### 响应报文：

服务器-🡪浏览器：响应报文

响应首行：协议版本号 响应状态码 状态提示

响应头：key：value 多个用回车分隔

空行：

响应体：响应的内容都在这儿。比如浏览器要index.html，响应体中是页面的源代码。

这个代码浏览器就会解析。一解析页面就展示出来了

### 响应状态码：

1\*\*：信息

2\*\*：成功 200 响应成功

3\*\*：重定向：浏览器会去重新访问别的页面

4\*\*：客户端错误： 404：资源未找到

5\*\*：服务端错误 500：服务器内容错误

### MIME类型

表示内容类型。

格式：大类型/小类型

常见的类型。去查文档。

# 五、Servlet

## 1、简介

1）、运行在服务器上的小java程序

2）、接受请求、处理请求、完成响应。

3）、实际上Servlet是sun公司定义的一个接口。所有对这个接口的实现类都具有Servlet的功能，都可以称为Servlet。服务器也可以叫Servlet容器。

## 2、HelloWorld

1）、编写Servlet的实现类。

init()、service()、destory()

2）、去web.xml中注册Servlet的信息

<Servlet>

<servlet-name>HelloServlet</servlet-name>指定Servlet的别名，一般使用类名

<servlet-class>com.atguigu.servlet.HelloServlet</servlet-class>指定Servlet全类名

</Servlet>

<servlet-mapping>

<servlet-name>HelloServlet</servlet-name>

<url-pattern>/hello</url-pattern>配置servlet的url映射信息，浏览器访问对应的url就会启动servlet进行处理，servlet就是动态资源。

</servlet-mapping>

3）、浏览器去访问当前项目下/hello。servlet执行service()方法进行处理

## 3、生命周期

生命周期：从创建到销毁的过程。

第一次访问会创建对象并进行初始化

构造器、 init()初始化、service()：

以后每次访问都执行service()；

项目移除/服务器停止：destory()销毁：

## 4、init（ServletConfig config）

ServletConfig：

代表：当前servlet的配置。每一个servlet都对应一个自己的ServletConfig

获取：服务器调用初始化方法的时候会传参进入。

功能：

1）、获取Servlet的别名： config.getServletName();

2）、获取Servlet的初始化参数：

在web.xml中对应的servlet配置类信息

<servlet>

<servlet-name>HelloServlet</servlet-name>

<servlet-class>com.atguigu.servlet.HelloServlet</servlet-class>

<init-param>

<param-name>属性名</param-name>

<param-value>属性值</param-value>

</init-param>

</servlet>

String value = config.getInitParameter(“属性名”);

3）、获取ServletContext：

ServletContext sc = config.getServletContext();

## 5、ServletContext

代表：当前web项目

获取：ServletContext sc = config.getServletContext();

功能：

1、获取项目路径：String path = sc.getContextPath();// /5.Servlet

2、获取web应用的初始化参数：

<context-param>

<param-name>user</param-name>

<param-value>root</param-value>

</context-param>

String value = sc.getInitParameter(“user”);//root

3、获取资源的真实路径：

url称为虚拟路径

真实路径：资源在磁盘中的路径 在服务器中的真实磁盘位置

index.html <http://localhost:8080/5.Servlet/index.html>

String path = sc.getRealPath(“/index.html”) f://tomcatServlet/wtpwebapps/…

4、作为最大的域对象共享数据。

域对象的功能：可以在不同资源之间共享数据

四大域对象。

## 6、HttpServlet

eclipse自动创建的sevrlet都是继承HttpServlet。

HttpServlet---继承-🡪GenericServlet---实现--🡪Servlet

doGet()；处理get方式提交过来请求

doPost()；处理post方式提交过来的请求

运行原理：

要运行一个Servlet。就是调用Servlet的service()方法。

GenericServlet实现了Servlet接口，他里面有一个service()方法但是他是一个抽象方法。

HttpServlet是GenericServlet的子类，他实现了service(ServletRequest,ServletResponse)方法（将传进来的两个参数强转为HttpServletXXX,调用重载的service()方法）。而且添加了一个重载的service(HttpServletRequest,HttpServletResponse)方法。

在重载的service中区分了对应的请求类型，不同的请求类型就调用对应的doXXX方法。

我们创建的Servlet继承了HttpServlet’，所以调用doXXX会调用我们自己编写的。

## 7、HttpServletRequest

代表：当前的请求对象，将当前请求的所有信息都封装在了HttpServletRequest对象中

获取：服务器传参进入的。直接拿来使用

功能：

1、获取请求参数（用户从页面传递过来的参数）request.getParameter(“username”);

2、获取项目路径 String path= request.getContextPath();

3、转发到一个资源（动态资源、静态资源）

request.getRequestDispatcher(“/index.html”).forward(request,response);

4、作为域对象共享数据。

## 8、HttpServletResponse

代表：当前响应对象，将要发给浏览器的响应信息都封装在这个对象中

获取：服务器传参进入。要给浏览器写响应信息直接使用

功能：

1、给浏览器响应一串内容

response.getWriter().write(“<h1>HelloWorld</h1>”);

2、重定向到一个资源（动态资源，静态资源）

response.sendRedirect(“url”);

## 9、转发和重定向

转发：当前请求我已经处理完毕，转交给别人继续处理。

重定向：当前请求处理完毕，告诉浏览器去重新访问新的资源

转发 重定向

请求次数 1 2

地址栏变化 不变 变化

web-inf下的资源是否可以访问 可以访问 不可以

## 10、路径问题

写路径url；就会有相对路径和绝对路径两种

相对路径：

写法：不以/开始

解析：以当前资源为参照标准去解析url

绝对路径：

写法：以/开始

解析：

1）、浏览器解析。以主机路径为开始往下找 <http://localhost:8080/>

2）、服务器解析。以项目路径为开始往下找

推荐使用绝对路径。绝对路径安全。转发可能会导致页面上的相对路径发生错误

注意点：

1、在页面写绝对路径都需要加上项目名。

2、response.sendRedirect(request.getContextPath+”/url”)

base标签：为页面上所有相对路径指定新的参照标准

<base href=”http://localhost:8080/5.path/”>

## 11、编码问题

原因：编码解码的字符集不统一

请求乱码：

1）get：需要修改server.xml文件。在8080端口标签配置处加上一个属性

URIEncoding="utf-8"

2）post：

在获取请求参数之前。request.setCharacterEncoding("utf-8");

响应乱码

解决：

在响应内容之前。设置响应的内容类型以及内容的编码格式

response.setContentType(“text/html;charset=utf-8”);

# 六、JSP

## 1、jsp简介

Java Server Page：

本质是一个Servlet。但是在JSP页面既能写HTML标签也可以写Java代码。

## 2、运行原理

xxx.jsp页面----翻译---🡪.java文件----编译----🡪.class

运行jsp页面的时候：

1、index.jsp： <http://localshost:8080/01.JSP/index.jsp>

2、web.xml中有一个配置，他是拦截所有以.jsp结尾的请求。JspServlet

3、找到jsp页面对应的编译后的java代码。运行即可

4、.class。\_jspService();作用：将之前在jsp页面定义好的内容一点点的写出来。out.write();//功能就是给浏览器写出数据

翻译和编译后的文件位置：镜像服务器/work/…/对应的项目

只有在第一次运行的时候才执行翻译和编译。以后每次运行直接调用.class里面的方法。

## 3、JSP基本语法

### 1）、模板元素：

语法：就是HTML标签的写法

作用：JSP页面里面的所有HTML代码都称为模板元素。定义页面结构

翻译：out.write(“<h1>test</h1>”)；所有的模板元素都会被原封不动的write出来

### 2）、JSP脚本片段

语法：<% 写java代码 %>

作用：java代码会被直接复制在对应的java文件的jspService方法中。

翻译：将原来定义的java代码直接复制进去

注意：

1）、因为java代码放在方法中。所以方法中不可以编写的语法都不能写

### 3）、JSP表达式

语法：<%= 表达式内容 %>

作用：在页面输出表达式的内容

翻译：表达式的内容会被原封不动的作为参数传入out.print()方法中

注意：

1）、因为表达式内容是作为参数，所以特别注意不能加分号

### 4）、JSP声明

语法：<%! 声明内容 %>

作用：在类中定义方法或属性

翻译：声明内容直接复制进类中

注意：

1）、可以声明私有变量….

2）、这个经常不太使用

### 5）、JSP指令

语法：<%@指令名 属性=值 属性=值%>’

作用：告诉服务器如何翻译和解析页面

指令分类：

1）、page指令

作用：定义页面的相关属性

Content-Type=”text/html;charset=utf-8”：告诉服务器如何将页面的所

有内容写出去

pageEncoding=”UTF-8”：告诉服务器页面的编码格式。服务器就按照

指定的格式去解码页面

import=”java.util.\*”：在页面导包

errorPage=”/error.jsp”：页面发生错误以后跳转到error.jsp

2）、include指令

语法：<%@include file=””%> file来指定被包含的页面的路径

作用：静态包含，将被包含的页面和当前页面合并起来

3）、taglib指令

### 6）、JSP标签（动作元素，Action元素）

1）、jsp:forward

作用：转发

语法：<jsp:forward page=”/url”></jsp:forward>

注意：

1）、如果转发的时候不带参数标签体中就不能有任何字符

2）、如果转发的时候带参数。携带方法：

使用<jsp:param name=”key” value=”val”/> 标签放在

forward标签体中

2）、jsp:include

作用：动态包含。作用和静态包含是一样的，都是将被包含的页面和当前页面合并起来

语法：<jsp:include page=”/url”></jsp:include>；

page指定被包含页面的路径

区别：

动态包含 静态包含

方式： 把页面的运行结果合并进来 将整个页面内容直接复制到方法中

编译文件： 编译被包含的页面和当前页面 只编译当前页面

使用： 被包含页面的内容经常变化建议 被包含页面的内容不经常变化

使用动态包含

## 4、九大隐含对象

1）、四大域对象

域对象：

作用：在不同资源之间共享数据

方法：

域对象.setAttribute(“key”,value);在域中保存一个数据

域对象.getAttibute(“key”);从域中取出一条数据

域对象.removeAttribute(“key”);从域中移除key对应的数据

PageContext pageContext：

当前页面可以共享数据，离开当前页面数据就不能获取

HttpServletRequest request：

同一次请求共享数据，转发是同一次请求

HttpSession session：

同一次会话共享数据：浏览器打开-关闭。

ServletContext application：

当前web应用共享数据。只要web应用不卸载或者服务器不关闭都可以

2）、其他对象

Throwable exception:

当jsp页面的isErrorPage=”true”，exception就可以使用

HttpServletResponse response：

代表当次响应

ServletConfig config:

代表Servlet配置信息。页面的config。JspServlet拦截所有\*.jsp

Page page = this;

代表当前页面。不如直接去使用this

JspWrite out;

代表给浏览器输出数据的对象。

现象：

out.write(“a”);

out.write(“b”);

response.getWriter.write(“c”);

response.getWriter.write(“d”);

//在页面显示数据的顺序 cdab

为什么？

out这个对象会将数据写到jsp的缓冲区。

response会把数据写到response缓冲区。

当给浏览器完成响应的时候就是将缓冲区的数据全部刷出去。

将jsp缓冲区的数据刷到response缓冲区中，将response缓冲区的内容交给浏览器

可以手动的将jsp缓冲区的数据刷新出来

out.write("a");  
out.write("b");  
out.flush();  
response.getWriter().write("c");  
response.getWriter().write("d");

会出现ABCD

page buffer="0kb";定义页面缓冲区大小。没有缓冲

# 七、EL

## 1、简介

EL expression Language（表达式语言）el表达式

作用：简化在页面上的取值操作。

## 2、El语法

1、可以使用 对象.属性

2、可以使用 对象[“属性”]

特别注意：

获取特殊的属性名。Hello-kitty 对象[“hello-kitty”]

## 3、EL内置11个对象

我们发现El表达式不能取出在页面之前声明的变量

<% String str = “Hello”%>

<%=str%> ${str}

El表达式只能获取到这11个对象中的属性。

### 1、PageContext：

可以获取到jsp页面的其他八个隐含对象，然后获取对应对象中的属性

### 2、四个域对象相关

pageScope：pageContext域中的所有属性都封装在了pageScope中。他是一个Map

requestScope：request域中所有的属性都封装在了requestScope中。他是一个Map

sessionScope：将session域中的所有内容都放在sessionscope中。他是一个Map

applicationScope:将application域中所有内容都放在applicationScope中。他是一个Map

### 3、5个和HTTP协议相关

param：获取请求参数，将所有的请求参数都封装在param中。他是一个Map

paramValues：获取某个参数的多个值。

header：获取请求头信息，将所有的请求头信息封装在header中。他是一个Map

headerValues：获取某个请求头的多个值

Cookie：学到cookie’再说

### 4、initParam

获取web应用的初始化参数<context-param>

以后能使用el的地方就尽量不用jsp表达式

# 八、JSTL

## 1、简介

JSP Standard Tag Library（JSP标准标签库）；

有两个外部的jar包。taglibs-standard-spec-1.2.1.jar、taglibs-standard-impl-1.2.1.jar

JSTL：简化开发，简化和替代脚本片段。

## 2、JSTL标签库

五个标签库；

核心库 c

函数库 fn

格式化库 fmt’

数据库 sql

xml x

使用标签库的步骤：

1）、导包taglibs-standard-spec-1.2.1.jar、taglibs-standard-impl-1.2.1.jar

2）、在页面引入标签库

使用taglib指令 <%@taglib prefix=”标签前缀” uri=”标签库的唯一标识”%>

## 3、核心库常用标签

### c:set：

1）、<c:set property=”username” target=”${student}” value=”张三”>

修改某个对象的属性值。

property：要修改的属性的名

target：要修改的目标对象。target=”<%=stu%>”

value：修改后的值

2）、<c:set var=”stu” value=”张三” scope=”page”>

作用：在域中保存一个属性

var：域中保存的key

value：保存的值

scope：指定保存在哪个域。

四种取值：page,request,session,application

### c:out：

在页面输出内容

<c:out value=”${hello}” default=”你好” escapeXml=”true”>

value：页面要输出的内容

default：如果内容为null默认值

escapeXml：是否对内容进行转义

true：转义，原封不动的输出<h1>你好</h1>

false：不转义，如果是标签浏览器就会解析。<h1>你好</h1>会输出一个标题

### c:remove：

从域中移除一个属性

<c:remove var=”” scope=””>

var：要移除的key

scope：从哪个域中移除

注意：如果不指定scope，从所有域（4个域）移除相关属性

### c:if：

执行判断。类似与if()

<c:if test=””></c:if>

test：判断条件

如果条件为真，执行标签体

### c:choose c:when c:otherwise

<c:choose>

<c:when test=”判断条件”></c:when>

<c:when test=”判断条件”></c:when>

<c:otherwise></c:otherwise>//以上when都不成立的时候执行otherwise

</c:choose>

### c:forEach

<c:forEach items=”${students}” var=”stu” begin=”1”

end=”10” step=”2” varStatus=”status”><c:forEach>

可以有以下属性  
items:指定要遍历的集合

var:每次遍历的数据存在var指定的key中

begin:遍历开始的索引

end:遍历结束的索引

step:步长。表示一次跨几个

varStatus：每次遍历的状态的信息

当前遍历到第几个，当前遍历的元素是什么....

将每次遍历当前的状态信息保存到status遍历中

current：当前遍历取出的对象

index：当前遍历的索引

count:当前遍历的数量

first：是否是第一个

last是否最后一个

begin遍历开始的索引。只有标签指定了begin才会有值

end遍历结束的索引

step遍历的步长

## 4、函数库

函数的前缀fn；

语法：${fn:函数名(参数表)}

${fn:length(object)}

功能：

1）、统计字符串的长度

2）、计算集合的长度。

# 九、Cookie

## 1、简介

Cookie：

1）、浏览器端保存数据的一种技术

2）、Cookie是一个键值对。

3）、服务器发给浏览器的。浏览器按照要求保存

4）、Cookie有限制，不管是在单个cookie’的大小上，还是在保存的数量上。

5）、浏览器访问服务器的时候会带上相应的cookie

6）、cookie不直接支持中文

## 2、cookie的创建

1）、服务器创建的cookie’，发给浏览器。

Cookie cookie = new Cookie(“name”,”value”);

response.addCookie(cookie);//发给浏览器

## 3、Cookie的读取

1）、浏览器和服务器沟通的时候会带上cookie，所以我们可以通过request对象来获取到cookie

Cookie [] cookies = request.getCookies();

for(Cookie cookie:cookies){

cookie.getName();//获取到cookie名

cookie.getValue();//获取到cookie’的值

}

## 4、修改cookie

因为浏览器认为name相同的cookie就是同一cookie，可以使用同名cookie进行替换。

Cookie cookie = new Cookie(“user”,”123456”)；//创建一个同名cookie；

response.addCookie(cookie);

## 5、cookie的生存时间；

1、默认创建的cookie，整个会话期间都存在。浏览器关闭就没有了。

2、修改cookie的存活时间。

cookie.setMaxAge()//以秒为单位

负数：默认就是会话期间。

0：cookie不保存，删除

正数：保存多长时间。

## 6、删除cookie

1、创建一个同名cookie

Cookie cookie = new Cookie(“user”,”123456”);

2、设置cookie的生存时间

cookie.setMaxAge(0);//cookie不会被保存

3、将cookie再发给浏览器

response.addCookie(cookie);

## 7、cookie设置路径

1、并不是说访问服务器就会带上所有cookie，每一个cookie都有一个默认路径。

默认路径：是按照哪个Servlet创建的，就是哪个servlet的路径

/abc/CookieServlet cookie路径：项目下/abc

2、可以为cookie设置路径

cookie.setPath(request.getContextPath+“/abc”);浏览器解析这个路径，所以加上项目名

## 8、cookie运行原理：

1）、浏览器发请求给服务器，服务器可以创建一个Cookie让浏览器保存

2）、浏览器收到响应，发现响应头中会有Set-Cookie字段。浏览器就会按照要求保存内容

3）、浏览器以后访问对应的项目就会带上cookie

# 十、Session

## 1、简介：

他是一个域对象，作用就是在整个会话期间共享数据的。

## 2、Session的创建及获取

1）、创建：

每一次新会话进来，服务器都会为其创建一个新的session对象。服务器内部使用SessionMap的东西来管理这些session对象。

2）、获取：

HttpSession session = request.getSession();

## 3、会话控制原理：

1）、当新会话进来，服务器创建一个session对象，每个session对象有一个id与他对应。

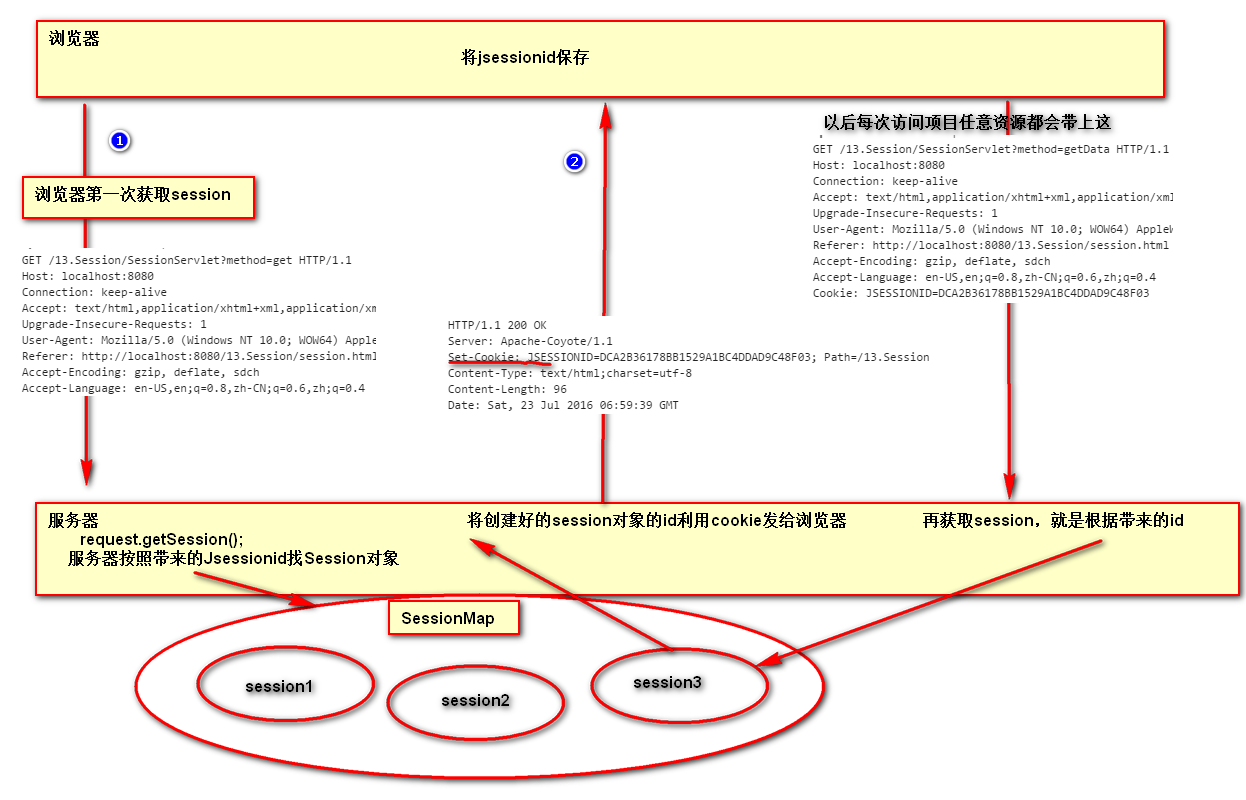
2）、服务器创建好session对象后，将session的id作为cookie发给浏览器。

new Cookie(“JSESSIONID”,”sessionidxxxxxxxx”);

3）、浏览器保存JSESSIONID这个cookie

4）、以后只要访问这个项目，就会带上JSESSIONID。

5）、服务器收到任何一个请求，如果要获取session，服务器会在请求头中获取JSESSIONID的值，按照值去找对应的Session对象。如果找到了，使用找到的对象。如果没找到，创建新的session对象。回到第一步。



## 4、session生存时间

1、session默认30分钟。在服务器的web.xml中配置

2、我们可以在当前项目下的web.xml中修改session过期时间。

<session-config>

<session-timeout>15</session-timeout>分钟

</session-config>

3、session.setMaxInactiveInterval();session最大不活动的失效时间，以秒为单位

负数：session永不过期

0、正数：session在多少秒不使用后就过期。

过期：session会被服务器销毁

## 5、活化钝化：

钝化：

session对象从内存序列化到磁盘中的过程

1）、服务器停止

2）、session大量存在，会将一些不常用的session钝化

3）、session一段时间不使用就会钝化，不能超过30分钟。

活化：

session对象从磁盘中反序列化到内存中的过程。

1）、服务器开启；

2）、session要使用了。

特别注意：

session中的内容要能同session对象一起活化钝化必须实现序列化接口

## 6、session的应用

1）、用户长时间不操作要求重新登陆

2）、验证码

3）、表单重复提交

4）、会话期间共享数据

# 十一、Filter

## 1、简介：

filter过滤器。与servlet,listener称为javaWeb的三大组件

1）、Filter实际上是一个接口

作用：

过滤请求。可以在请求到达目标资源之前过滤请求（修改），可以在响应来到浏览器之前，过滤响应。

## 2、HelloWorld

1）、编写一个Filter的实现类

doFilter()//过滤请求

2）、在web.xml中进行注册

<filter>

<filter-name>HelloFilter</filter-name>filter别名

<filter-class>com.atguigu.filter.HelloFilter</filter-class>filter全类名

</filter>

<filter-mapping>

<filter-name>HelloFilter</filter-name>

<url-pattern>/index.jsp</url-pattern> 指定过滤哪个请求

</filter-mapping>

3）、当浏览器请求index.jsp的时候，HelloFilter的doFilter方法就会执行过滤

放行请求：chain.doFilter()

## 3、FilterConfig、FilterChain

1）FilterConfig：filter的配置信息

作用：可以获取Filter别名、获取Filter初始化参数、获取ServletContext对象  
 2）FilterChain chain：

作用：放行请求

## 4、url-pattern

1）、精确匹配

语法格式： 精确的路径地址 /pages/cart.jsp

2）、模糊匹配

1)、路径匹配

语法格式 /路径名/\* 访问路径下的所有资源都会拦截 /pages/\*

/\* 所有请求

2)、后缀匹配

语法格式 \*.后缀名 访问以后缀名结束的所有请求都会拦截 \*.jsp

## 5、Filter生命周期

1、服务器启动： 会执行构造器已经初始化(init)。 创建对象并初始化

2、每次请求对应的资源： 会执行doFilter()方法进行过滤

3、服务器停止或项目卸载：执行destory();

## 6、dispatcher

用来告诉服务器拦截哪种类型过来的请求。

REQUEST：拦截直接请求过来的。

FORWARD：拦截转发来目标资源的

INCLUDE：拦截动态包含的

ERROR：拦截通过全局错误配置的请求

<error-page>

<error-code>500</error-code>

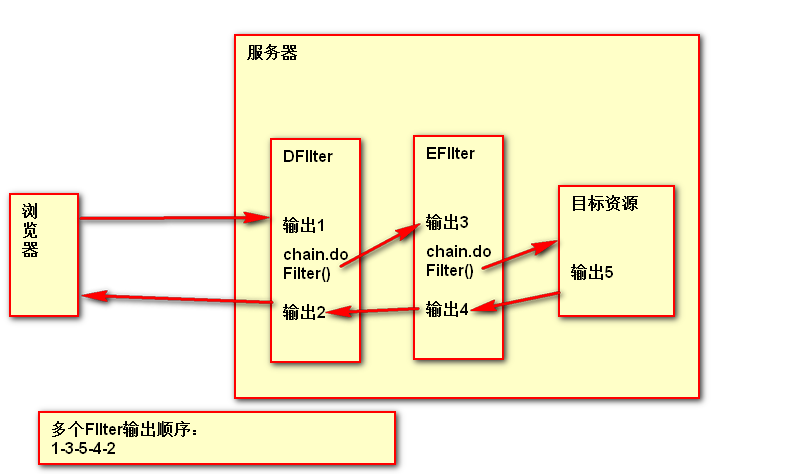
<location>/error.jsp</location>

</error-page>

## 7、Filter链

同一个资源可以被多个filter进行过滤。

哪个filter先执行，按照web.xml中配置的先后顺序执行。



# 十二、AJAX

## 1、简介：

异步的javascript和xml;

使用javascript与服务器进行异步通信，传输数据可以使用xml。

无刷新页面的技术

## 2、js原生ajax请求

主要就是在使用XmlHttpRequest对象。js中有这个对象，这个对象可以给服务器发送异步请求，与服务器沟通。

### 1）、Get：

创建XmlHttpRequest。他的兼容性不好。我们将其抽取成一个方法。

**function** getXMLHttpRequest(){

**var** xhr;

**try**{

//大部分浏览器都支持

xhr = **new** XMLHttpRequest();

}**catch**(e){

**try**{

//如果不支持，在这里捕获异常并且采用IE6支持的方式

xhr = **new** ActiveXObject("Msxml2.XMLHTTP");

}**catch**(e){

//如果还不支持，在这里捕获异常并采用IE5支持的方式

xhr = **new** ActiveXObject("Microsoft.XMLHTTP");

}

}

**return** xhr;

}

1、获取XMLHttpRequest对象

var xhr = getXMLHttpRequest();

2、打开链接

xhr.open(“GET”,”ajaxServlet?id=1”,true);

//请求参数、请求地址、请求是否异步

//异步：不会阻塞浏览器，不用等待请求全部完成再去执行其他方法

//同步：会阻塞浏览器，需要等待请求全部完成再去执行其他方法

3、发送请求

xhr.send();//get请求send方法不用传参数。

4、接受响应数据。

1）、监听请求状态

xhr.onreadystatechange = function(){

//2）、请求全部完成

// readyState: 0 1 2 3 4

//status：响应状态码

if(xhr.readyState==4 && xhr.status==200 ){

//3）、提取服务器给浏览器的响应内容

xhr.responseText;//提取内容为文本

xhr.responseXML;//提取内容为xml。将响应给浏览器的xml封装为对象。使用原生的dom增删改查都可以。getElementByXXX() . ele.innerHTML

}

}

### 2）、post

var xhr = getXMLHttpRequest();

xhr.open(“POST”,”ajaxServlet?id=1”,true);

xhr.setRequestHeader("Content-type", "application/x-www-form-urlencoded");//\*\*\*\*\*

xhr.send(“name=admin&age=18”);//send中发送数据。数据格式：key=value&key=value

xhr.onreadystatechange = function(){

if(xhr.readyState ==4 && xhr.status ==200){

var result = xhr.responseText;

//对结果的处理

}

}

## 3、Jquery发送AJAX

1）、发送请求服务器和可以给浏览器响应数据，我们如果要响应一个有组织关系的数据。

我们以前可以响应xml字符串，但是解析有点麻烦。

## 4、JSON数据

新的数据的表示方式：JSON；JavaScript对象表示法。

格式：

要求。key必须带双引号。

key:value,key:value

对象：{“name”:”张三”,”age”:18}

数组：[值1，值2]

value取值：

字符串、数字、布尔、数组、对象、null

**json解析**

1）、浏览器

1、JSON对象、JSON字符串

### 1、JSON对象转字符串；方便传输

var json = {“name”:”admin”,”age”:18};

var str = JSON.stringify(json);//将json对象转为json字符串

### 2、JSON字符串转为对象；方便解析

var json2 = JSON.parse(str);//将json字符串转为json对象。

### 3、java操作json

1）、用第三方工具 fastjson；

2）、

//将对象转为json字符串

String jsonstr = JSON.toJSONString(obj);

//将字符串转对象

Student stu = JSON.parseObject(jsonstr,Student.class);

List<Student> list = JSON.parseArray(jsonstr,Student.class);

## 5、jQuery发送ajax

$.get();$.post(),$.getJSON();,$.ajax();

### 1）、$.get(url,[data],[callback],[type])：

说明：

url：请求地址

data：请求数据。数据的格式：

1)、k-v字符串key=value&key=value

2)、json对象{“name”:”admin”,”age”:18}

callback：回调函数。请求完成后会执行这个回调函数。接受服务器响应过来的数据只需要给回调函数中写一个参数来接受即可。

type：服务器响应的内容类型。常用：text,json

示例：

$.get(“ajaxServlet”,”name=admin&age=18”,function(data){

alert(data);

},”json”);

### 2）、$.post(url,[data],[callback],[type]);

同get发送。

get发送的是data中的数据最终会被带在url？后面，。所以get不能携带大量数据

post发送的是data中的数据最终会被带在请求体中。可以携带大量数据

### 3）、$.getJSON();

将服务器响应的数据自动封装为json对象，是下面方法的简写

$.get(“ajaxServlet”,”name=admin&age=18”,function(data){

alert(data);

},”json”);

### 4）、$.ajax();

最底层的发送ajax请求方法。所有的项都是可配置的。详见jquery文档

示例：

$.ajax({

[url:”ajaxServlet](url:)”,

type:”GET”,

data:”name=admin&age=18”,

success:function(data){

alert(data);//服务器响应的数据

},

error:function(msg){

alert(msg);//错误信息

}

});

# 十三、国际化

## 1、简介

i18n，使应用程序能兼容多国语言。

## 2、国际化的几个核心类

### 1）、Locale

Locale对象表示区域信息。

获取：

1、Locale china = Locale.CHINA

2、Locale china = new Locale(“zh”,”CN”);

国际化就是根据不同的区域信息(Locale)来显示不同的内容

### 2）、日期格式化 DateFormat

DateFormat format =DateFormat.getDateTimeInstance(DateFormat.FULL,DateFormat.FULL,locale);

//第一个参数代表日期的格式，FULL显示全部信息，第二个代表时间格式，第三个代表区域信息

String str = format.format(new Date());

### 3）、数字格式化 NumberFormat

NumberFormat nf = NumberFormat.getInstance(locale);

String str = nf.format(123456798.123);

### 4）、货币格式化 NumberFormat

NumberFormat nf = NumberFormat.getCurrencyInstance(locale);

String str = nf.format(123456798.123);

### 5）、ResourceBundle 管理资源文件

用法：

1）、创建资源文件：命名规格： 基础名\_语言代码\_国家代码.properties

2）、使用ResourceBundle加载

ResourceBundle bundle = ResourceBundle.getBundle(基础名)

3）、获取资源文件中对应的值

String value = bundle.getString(key);

4）、我们在编写key对应的值的时候可以使用占位符

{索引号} 索引从0开始

占位符可以重复写多个 {0}使用第一个参数填充当前位置

### 6）、信息格式化 MessageFormat

MessageFormat 我们用来格式化一些字符串，填充占位符

String str = MessageFormat.format(value,”占位符参数”)

## 3、国际化在WEB中的应用

我们在web中要国际化可以使用fmt标签库；

1）、编写资源文件

2）、导包、导入标签库

3）、使用标签

fmt:setLocale：绑定区域信息 <fmt:setLocale value="zh\_CN"/>

fmt:setBundle：绑定资源文件 <fmt:setBundle basename="bookstore"/>

fmt:message：获取资源文件中的信息

1、<fmt:message key="username" />

2、<fmt:message key="welcome" >

<!--fmt:param指定占位符参数 -->

<fmt:param>张三</fmt:param>

</fmt:message>

fmt: formatDate ：日期格式化 <fmt:formatDate value="<%=new Date() %>"/>

# 十四、文件上传下载

## 1、文件上传

1）、创建一个文件上传表单

<form action=”upload” method=”post” enctype=”multipart/form-data”>

<input type=”file” name=”headerimg”/>

<input type=”submit” value=”提交”>

</form>

2）、使用第三方工具上传，导包commons-fileupload.jar，commons-io.jar

3）、编码

1）、创建一个磁盘文件项工厂

FileItemFactory factory = new DiskFileItemFactory()

2）、创建一个文件上传解析器

ServletFileUpload fu = new ServletFileUpload(factory);

3）、解析请求

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

4）、遍历解析每个部件。表单上传会将每个表单项封装成一个部件（FileItem）

for(FileItem item:list){

if(item.isFormField()){//普通的表单项

String name = item.getFieldName();//获取表单项的name

String value = item.getString(“utf-8”);//获取表单项的value

}else{

item.getName();//获取文件名。ie浏览器是全路径，其他浏览器是文件名

InputStream stream = item.getInputStream();//获取到文件流

upload(item);

}

};

5）、文件上传到服务器

public void upload(FileItem item){

//1、获取到文件名

String fileName = item.getName();

//2、文件名在ie下是全路径所以需要截取

fileName = fileName.subString(fileName.lastIndexOf(“\\”)+1);

//3、文件重名问题可以通过添加前缀

fileName = UUID.random().toString().replace(“-”,””) +”\_”+ fileName

//4、获取到要保存到的位置

String path = getServletContext().getRealPath(“/upload/”+fileName);

FileOutputStream os = new FileOutputStream(path);

//5、保存流

InputStream is = item.getInputStream()

IOUtils.copy(is,os);

//6、关流

is.close();os.close();

}

## 2、文件下载

1）、只需要告诉浏览器文件的内容类型和处理方式即可

response.setContextType(“application/octet-stream”);//告诉浏览器这是一个流

response.setHeader(“Content-Disposition”,”attachment;filename=”+filename);

2）、filename会中文乱码。

String fn =new String(filename.getBytes(“GBK”),”ISO8859-1”);

response.setHeader(“Content-Disposition”,”attachment;filename=”+ fn);

# 书城项目

## 一、第一阶段 使用jquery进行表单验证

1）、把书城项目导入到工程里面

先创建一个静态web工程。

把页面全部复制到webContent下

2）、使用jquery进行表单验证，注册的表单验证

1）、给注册按钮绑定单击事件

2）、提取用户填写的信息，进行验证。

3）、如果验证成功就跳转到页面，否则弹出错误提示、

使用正则表达式进行验证

用法：表达式.test(字符串) 测试字符串是否符合要求

## 二、第二阶段 完成登陆注册功能；

分析：

1、创建一个动态web工程。

2、将第一阶段做好的页面复制进来

### 软件架构：

表示层（UI(User interface)）

用户界面及交互

软件的架构

业务逻辑层（业务代码）

业务逻辑代码

数据库访问层

表示层：里面是用户界面，直接展示给用户，用户通过表示层的界面来访问项目

业务逻辑层：里面都是处理的业务逻辑代码，。登陆（写相应的处理代码）

数据库访问层：称为持久化层，保存数据，访问数据。对数据的增删改查

### 项目架构：

表示层：就是WebContent下面的所有页面

业务逻辑层：

1、Dao层：专门用来执行数据库操作的。集中操作数据库的代码（DataBase Access Object）数据库访问对象。UserDao BookDao

2、service层：专门编写逻辑代码。登陆（从数据库查询，返回成功）

3、servlet层：用来控制项目的页面跳转。用来接收请求和完成响应的。用service层定义的代码来处理请求。

数据库访问层：Dao层

1、bean层：模型层。功能：将表中的数据和javaBean建立起关系

数据库表。。。。

我们将查出的数据我们经常需要封装对象。一张表对应JavaBean类，一条数据对应一个对象。

分层的目的：为了解耦、为了分模块。分模块易于分工合作

通过建立不同的包，在项目中区分每个包都是哪个模块。

用户界面

(WebContent)

Service层

网站业务逻辑

Servlet层

控制网站跳转逻辑

Dao层

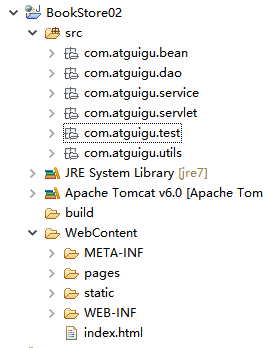
数据库操作

Bean层

javaBean

1、交互逻辑：

用户从界面发起请求--🡪Servlet层处理请求并控制跳转-🡪Servlet调用相应的业务逻辑（Service层）--🡪调用Dao层对数据操作-🡪将数据封装为对象



分层后代码就会很有条理、。解耦。

### 开发项目

1、需求分析，建立模型。需求分析需要提前建立好。

做什么功能？用户的登陆注册

1）、建立Bean层：

抽取用户模型：

数据用户表：

一般创建表的时候，每个表都应该有个前缀。来快速识别这个表属于个哪个模块

bs\_user .

主键:id、用户名:username、密码:password、邮箱:email

CREATE DATABASE bookstore\_0519;

CREATE TABLE bs\_user(

id INT(11) PRIMARY KEY AUTO\_INCREMENT,

username VARCHAR(200) UNIQUE NOT NULL,

PASSWORD VARCHAR(200) NOT NULL,

email VARCHAR(200)

)

对应的javaBean模型：

**public** **class** User {

//主键

**private** Integer id;

//用户名

**private** String username;

//密码

**private** String password;

//邮箱

**private** String email;

2）、Dao层

//1、打通数据库JDBCUtils，获取和关闭数据库连接

//BaseDao：将基本的增删改查方法都放在BaseDao中，BaseDao作为其他dao的父类

//BaseDao<T>：BaseDao希望他定义的方法如果是UserDao继承那么就可以直接操作User表，如果是BookDao继承就可以直接操作Book表。

//queryRunner update方法可以增删改

定义所有dao我们都可以采用面向接口编程。

我们希望在UserDao（接口）只定义方法。

UserDao:

User getUserByUserNameAndPassword(User user);

boolean saveUser(User user);

3）、Service层

UserService 真正的业务逻辑。

业务逻辑层，也使用面向接口编程。易于网站架构。

UserService：处理和用户相关的业务逻辑。

登陆、注册：

User login(User user) ,boolean regist(User user);

4）、Servlet层

LoginServlet RegistServlet

分析交互逻辑：

登陆逻辑：

用户来到登陆页面（填写信息，点击登陆）

🡪LoginServlet来处理登陆请求

🡪调用UserService.login()方法

🡪返回User对象。

🡪返回的是null；登陆失败（用户名密码错误），应该再回到登陆页面

🡪返回的是User对象。登陆成功。来到登陆成功页面

注册逻辑：

用户来到注册页面（填写信息，点击注册按钮）

🡪RegistServlet处理请求

-🡪UserService.regist()进行注册

🡪返回true/false;

注册成功：true。来到成功页面

注册失败：false。回到注册页面继续填写

## 三、第三阶段 项目优化

### 1、将HTML页面转为jsp页面

jsp、html：只需要将jsp的page头复制进入html里面，将html后缀名改为jsp

1、先将头复制进去，2、再改后缀名

### 2、将菜单抽取出来。

可以使用静态包含，将菜单包含到相应的页面。

优点：

1、将菜单提取出来以后，修改连接只需要修改提取后的文件

没登录前的菜单： 

登陆后的菜单：



管理员的菜单：



### 3、将页面的公用的引入信息也抽取出来

因为每个页面都引入了相同的css文件。以及都需要base。以及也需要jquery文件

我们将这些也抽取

<base href=*"*<%=request.getScheme() %>*://*<%=request.getServerName() %>*:*<%=request.getServerPort() %><%=request.getContextPath() %>*/"*/>

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

<script type=*"text/javascript"* src=*"static/js/jquery-1.7.2.js"*></script>

base中的href属性应该是动态获取到的；

request.getScheme();获取协议名 http

request.getServerName();获取主机名 localhost

request.getServerPort();获取端口号 8080

request.getContextPath();获取项目路径 /BookStore03

href值可以通过以上值拼串

拼串需要注意的几点：

1、协议名和主机名加上://

2、主机名和端口号加上:

3、端口号和项目路径之间一定不要加/

4、项目路径后面必须加上/

### 4、页面加入错误提示信息

页面发送请求登陆、注册---🡪Servlet进行处理（登陆、注册成功还是失败）

---🡪失败以后回到之前页面。希望有错误消息提示。我们就应该将错误消息放在域中

--🡪来到页面从域中取出数据

注意点：页面取数据第一次进来都是null。应该做一个非空判断

5、表单回显

页面表单上的数据提交以后如果再回到这个页面（转发、重定向）。表单的数据都会消失，我们希望回到页面后表单上填写的数据还是存在的。（表单回显）。

注册：

用户点击注册（/RegistServlet）---》Servlet处理-🡪转发到regist.jsp

我们可以利用转发特性，转发的整个过程都是同一次请求。我们直接可以在页面获取请求参数。在页面request.getParameter(“username”)和在servlet中调用request.

getParameter(“username”)获取到的数据是一样的

只有转发可以这么做。要重定向还要回显数据，只能放在域对象中，去页面获取。

### 5、一个Servlet处理多个请求。

现象：一个Servlet处理一个请求。

我们希望。UserService login regist 和用户相关的业务都在里面

UserServlet 和用户有关的请求都用它来处理，登陆、注册…

解决：希望一个Servlet来处理多个请求

我们可以在页面定义一个参数，这个参数就代表我是哪个请求。服务器获取到这个参数的值，进行判断对其做相应的处理

//页面处理

<input type="hidden" name="method" value="login"/>

//代码处理

**if** ("login".equals(method)) {

login(request, response);

}

**if** ("regist".equals(method)) {

regist(request, response);

}

**if** ("updatePwd".equals(method)) {

}

问题：

if判断太多，我们可以使用反射来动态调用对应的方法。

解决办法：

我们编写了一个BaseServlet。我们在doPost方法中。利用反射区动态的调用子类的方法。

特别注意：

1、以后我们写的Servlet都来继承BaseServlet

2、以后在我们编写的Servlet中添加方法的时候

方法名(HttpServletRequest request,

HttpServletResponse response)

因为BaseServlet在查找方法的时候，按照方法名(请求，响应)

3、以后子类一定要删除doGet，doPost方法。这两个方法应该继承父类的。

### 6、自动化封装对象

问题：在每个Servlet进行处理的时候，都需要获取到页面的数据，然后封装成对应的对象。

现象：

1、我们将获取数据封装对象都放在Servlet开始执行

2、如果字段很多，我们需要request.getParameter();造成多余代码太多。

3、封装数据的时候可能还要转化类型。

解决：我们希望我们能将获取数据、封装对象的这个过程做成一个自动化的工具。

User user =WebUtils.param2Bean(request,new User());

T t = param2Bean(HttpServletRequest request,T t);

工具类。commons-beanutils-1.8.0.jar、commons-logging-1.1.1.jar

BeanUtils.setProperty(bean,name,value);

BeanUtils.populate(bean,map);将map中的数据映射到javaBean里面

## 四、第四阶段 后台图书管理功能 前端图书展示

分层分析每层该怎么做。图书

1、Model层

数据模型：

1）、建立对应的图书模型



主键：id

书名：title

作者：author

价格：price

销量：sales

库存：stock

图片封面地址：imgPath.一般都是在数据库保存图片路径

2）、把数据表建立

CREATE TABLE bs\_book(

id INT(11) PRIMARY KEY AUTO\_INCREMENT,

title VARCHAR(200) NOT NULL,

author VARCHAR(200) NOT NULL,

price DOUBLE(11,2),

sales INT(11),

stock INT(11),

img\_path VARCHAR(200)

)

建立对应的javaBean

**public** **class** Book {

**private** Integer id;//主键

**private** String title;//书名

**private** String author;//作者

**private** **double** price;//价格

**private** **int** sales;//销量

**private** **int** stock;//库存

**private** String imgPath;//图片封面地址

3）、dao

BookDao：

int addBook(Book book);//添加图书

int deleteBook(Book book);//删除图书，按照id删除

int updateBook(Book book);//按照id修改，book中封装的其他字段代表的就是修改后的值

Book getOne(Book book);//获取一本图书

List<Book> getAll();//查询出所有图书

业务模型：

BookService;定义图书操作的业务逻辑。

boolean add(Book book);//添加图书

boolean delete(Book book);//删除图书

boolean update(Book book);//修改图书

Book getOne(Book book);//根据id查询

List<Book> getAll();//获取所有图书

2、Controller层

Servlet：

BookManagerServlet

BookClientServlet

1）、分析业务流程BookManagerServlet

1、点击图书管理-----🡪

查出所有图书(Servlet)

----🡪在页面进行显示book\_manager.jsp

2、点击删除---🡪

删除对应的图书

--🡪返回图书列表页面

3、点击修改--🡪

查询出要修改的图书的详细信息

--🡪来到页面显示图书的详情book\_edit.jsp

-🡪用户修改相应信息，点击提交

--🡪修改图书信息

-🡪还是回到图书列表页面book\_manager.jsp

4、点击添加

---🡪直接跳到book\_edit.jsp

--🡪用户输入所有信息

--🡪点击提交

--🡪图书保存

--🡪回到图书列表页面book\_manager.jsp

3、View层

页面

显示多条图书的时候，会一次性全部查询出来。

导致问题：

1）、影响数据库性能

2）、会有大量的等待时间影响用户体现

解决办法：

分页；点击下一页或者对应的页面，只是在数据库中取出部分数据。



数据库端：查询部分数据。

sql：select \* from bs\_book where xxx=? limit index,size;

index:查询开始的索引数，索引对应的行号-1;

size:代表查询多少条数据。

页面显示：

如果是第一页数据：

select \* from bs\_book where xxx=? limit 0,4;

如果是第二页数据：

select \* from bs\_book where xxx=? limit 4,4;

做图书分页查询功能：

M：

图书模型；

分页模型：



整个流程应该是，用户点击第几页--🡪从数据库中查出页码对应的数据--🡪封装到Page中----🡪将Page对象交给页面----🡪页面进行解析显示

业务模型：

dao:

添加分页查询的方法：

List<Book> getBooksByPage(int index,int size);

service:因为Servlet是直接调用service。

当前页码：pageNo String

每页的记录数：pageSize String

Page<Book> getPage(String pageNo,String pageSize);

BaseDao需要定一个可以查出单个值的方法

getSingleValue

BookDao

int getTotalCount();

V：

C：

前端图书显示：

分页进行显示。连续显示多页。连续显示5页。

当前页码-2开始到当前页码+2结束。

总页码小于5

全部显示

只有总页码大于5

1、如果当前页码1，2，3，显示前五页

1 2 3 4 5

2、如果当前页码>3

前两页 当前页 后两页

3、如果当前页码接近尾页，显示后五页

总页码-4 总页码

前端带条件查询的分页：

M：

数据模型：

业务模型：

service-dao

dao：

BookDao-🡪

getPageByPrice(int index,int size,double min,double max);

V：

页面：

用户输入价格区间，点击查询--🡪Servlet查询对应区间的图书

--🡪来到页面显示，即使是翻页，也是价格区间的图书

C：

在编写的开始按照MVC分层模型先来分析每一层该做什么。

根据页面真正要显示的东西以及业务逻辑

M：

数据模型：

业务模型：

service --- dao

先定义业务中有可能用到的数据库的操作。把这些操作放在dao中。先实现出来。根据页面正在的逻辑去编写。写service方法的时候有一个诀窍。

1、如何确定要做哪些方法。就是根据页面，直接来定。页面的操作就是方法的内容。比如按照价格查询。getPageByPrice();

2、如何确定方法的参数列表。service就像一个中间层。

service方法的参数列表的确定，应该就是页面传过来是什么就是什么。如果有自动化工具能封装对象就能整对象，如果没有按照页面传递过来的参数列表来定。

dao的参数列表怎么定？因为service是一个中间层，所以说。service应该去做所有的事情，包括转型封装对象…，dao方法的参数列表按照数据库查询语句需要的数据类型及参数表来定。dao中我们不希望有任何的转型操作。

bean🡪dao🡪service🡪servlet->view

当业务方法定义完成后就可以顺着逻辑来写。

模拟整个页面流程进行处理。

用户点击登陆

----🡪先改好连接

---🡪再去编写对应的Servlet

---🡪调用service

---🡪将要给用户的数据放在域中，

--🡪去向一个页面

--🡪改页面

## 五、第五阶段 登陆、注销、注册验证码、购物车

### 1、登陆

用户登陆成功，将登陆成功的用户保存在session中，转发到成功页面。

### 2、注销

清除sesssion即可

### 3、验证码

1)、导入

2)、配置web.xml文件

<servlet>

<servlet-name>kaptcha</servlet-name>

<servlet-class>com.google.code.kaptcha.servlet.KaptchaServlet</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>kaptcha</servlet-name>

<url-pattern>/code.jpg</url-pattern>

</servlet-mapping>

3)、服务器验证

从session中取出验证码，和浏览器带来的进行验证

4）、点击更换验证码

只需要改变img src属性的值即可，但是如果几次的src都是相同的值，浏览器会从缓存中加载，或者直接不发请求。

### 4、购物车

#### 1）、购物车的实现；

1）、基于session的实现，将购物的数据保存在session中。方便在其他资源处共享

2）、基于数据库，将购物车中的数据保存在数据库。

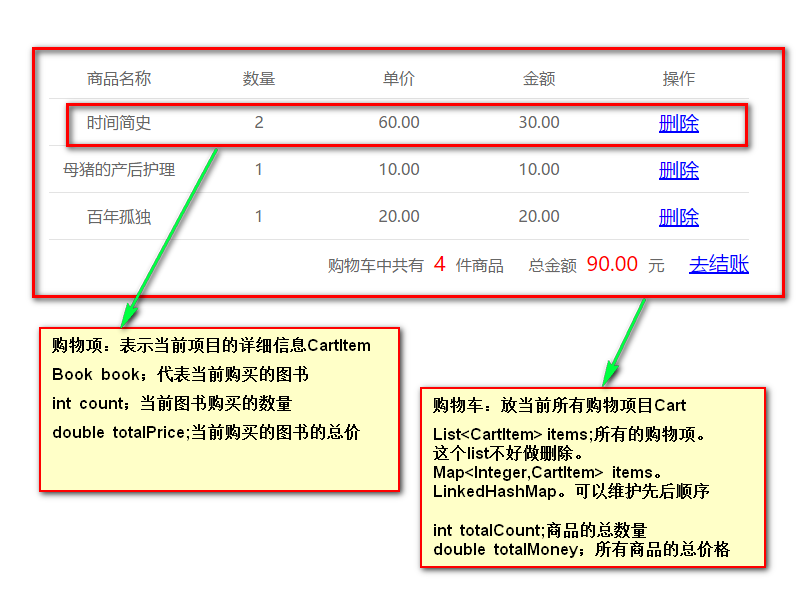
我们使用第一种来实现。

#### 2）、按照功能分层分析

M：

数据模型：

购物车、购物项模型



不用创建对应的数据库表。

创建对应的javaBean

业务模型：

service—dao（不需要），我们将对购物车的操作可以放在Cart类中

和购物车相关的方法：

加入购物车：void addBook2Cart(Book book);

删除某个购物项：void deleteItem(String bookid);

修改购物项数量：void updateCount(String bookid,String count);

清空购物车：void clear();

获取所有购物项：getAllItems（）;

V：

view层：

C：

Controller层：

功能流程：

1）、加入购物车

用户点击加入购物车

-🡪CartServlet处理请求

🡪哪来的回哪儿去。referer。

2）、用户点击购物车，查看所有内容

用户点击购物车

--🡪直接来到Cart.jsp

🡪取出session中保存的购物车数据进行显示

3）、清空购物车

4）、删除购物项

用户点击删除，带上要删除的id

--🡪CartServlet处理。

-🡪回到购物车页面

5）、修改购物项的数量

用户填写完数量后

--🡪给CartServlet发请求来修改数量

-🡪回到购物车页面进行显示

5、运算精度

1）、大整数运算。

会出现运算精度问题。数据溢出。

2）、浮点数运算。

会出现运算精度问题。浮点数在计算机内容保存的是一个近似数。

如果这些数据只是保存和获取没一点问题，但是运算就会出问题

解决：

BigDecimal。

## 六、第六阶段 订单模块完成

### 1、订单模块

1）、管理员端

订单管理，查出所有订单，查看详情，发货

2）、用户端

1）、去结账：

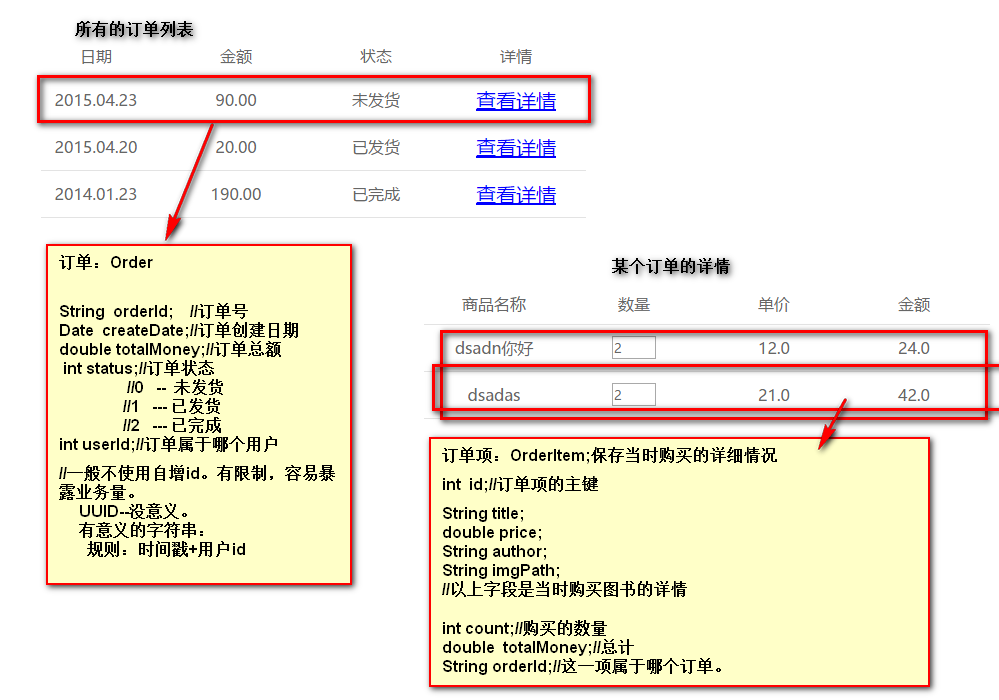
2）、点击我的订单，查出（自己）所有订单，可以查看详情

### 2、功能分析

M：

数据模型：

订单。



用户和订单的关系：

一（用户）对多（订单）的关系。可以使用外键。

Order表有一个外键 userid。

外键的直观体现：就只是表中的一个字段

外键关联是一种约束。order表有个userid关联到user表id的时候。

如果建立了关联关系userid的值一定都是在user表中id字段有的。

如果检查没有，保存失败。

订单和订单项的关系：

一（订单）对多（订单项）关系；

一般都在多的一端去保存一的数据

1）、创建表

CREATE TABLE bs\_order(

id VARCHAR(50) PRIMARY KEY,

create\_date DATETIME,

total\_money DOUBLE(11,2),

STATUS INT(2),

userid INT(11),

FOREIGN KEY(userid) REFERENCES bs\_user(id)

)

CREATE TABLE bs\_order\_item(

id INT(11) PRIMARY KEY AUTO\_INCREMENT,

title VARCHAR(200) NOT NULL,

price DOUBLE(11,2),

author VARCHAR(200),

img\_path VARCHAR(200),

amount INT(11),

total\_money DOUBLE(11,2),

order\_id VARCHAR(50),

FOREIGN KEY(order\_id) REFERENCES bs\_order(id)

)

2）、创建JavaBean

业务模型：

service-dao

dao:

OrderDao：

1、保存订单 saveOrder(Order order);

2、查询所有订单 getAllOrders();

3、查询某个用户的所有订单 getMyOrders(User user);

4、修改订单状态 updateOrder(Order order);

OrderItemDao：

1、查出某个订单对应的所有订单项 getItems(Order order);

2、保存某个订单项 saveItem(OrderItem item)

service：

观察页面写对应的方法；

OrderService;

1、用户点击我的订单应该去调用查出用户对应订单的方法

List<Order> getMyOrders(User user);

2、查看某个订单详细

List<OrderItem> getOrderInfo(Order order);

3、用户点击确认收货

void received(Order order);

4、管理员点击订单管理查出所有订单

List<Order> getAllOrders();

5、点击发货

void send(Order order);

6、用户点击去结算，结账。返回订单号

String checkout(User user,Cart cart);

OrderItemService;

1、保存某个订单项

void saveItem(OrderItem item);

2、获取某个订单的所有订单项

List<OrderItem> getItems();

V：

C：

OrderManagerServlet

1）、管理员端：

1、点击订单管理查出所有订单

点击订单管理-🡪 OrderManagerServlet查出所有订单-

🡪来到pages/manager/order\_manager.jsp显示

2、点击发货

显示所有订单-🡪如果状态为0显示点击发货，否则显示对于的状态信息

-🡪管理员点击发货-🡪OrderManagerServlet处理🡪回到订单列表页面

OrderClientServlet

2）、用户端

1、结账

用户点击结账-🡪OrderClientServlet处理请求

-🡪结账之前必须判断用户是否登陆。

🡪结账完成后来到checkout.jsp页面显示订单号

2、订单列表

用户点击我的订单-🡪 OrderClientServlet处理请求

-🡪查出当前用户的所有订单。放在域中

🡪来到pages/order/order.jsp页面显示所有订单

3、确认收货

用户点击确认收货按钮-🡪 OrderClientServlet处理请求

-🡪

注意点：

1）、乱用referer.

判断refer,就看发请求的时候浏览器的地址栏。

referer

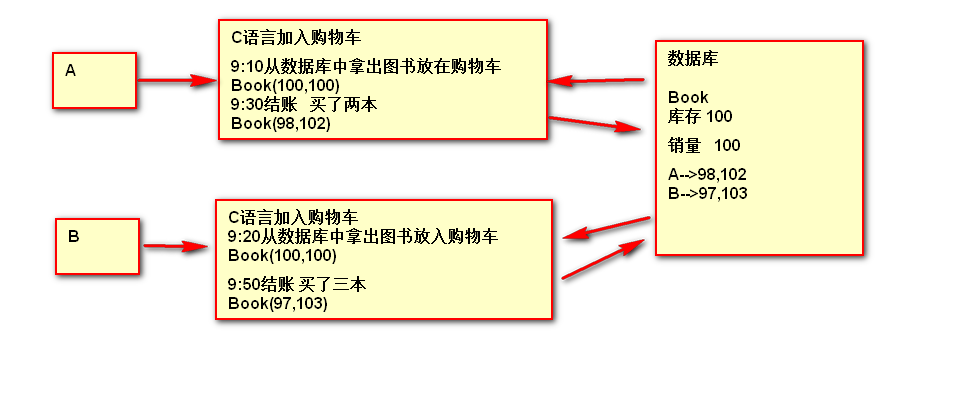
## 七、第七阶段 优化、引入Filter

### 1、问题一、库存销量问题：

现象：当我们两个用户同时购买图书的时候，发现库存和销量最后修改的有问题。

原因：我们在修改图书库存和销量的时候都是从购物车中拿出图书。修改他的库存和销量再保存。

解决：我们在修改图书的时候一定要从数据库实时获取



### 2、结账完成直接刷新，会生成新的订单。

问题：表单重复提交的问题

### 3、和用户订单相关的所有操作都需要验证用户登陆

问题：我们很多请求是需要验证用户身份，也就是验证用户是否登陆。

解决：使用Filter，只要是和用户订单相关的操作全部验证用户是否登陆。

### 4、使用filter控制事务

事务：

一系列操作，比如先去a表保存数据,再去b表修改数据，又去c表保存数据。

我希望，整个操作过程如果都成功，这个操作才算成功，如果有一个失败，整个全部失败。

回滚之前的操作。

现象：当某一个操作失败，其他的都没有受影响。我们希望的是，一个操作失败，全部失败，不能生成订单。

事务：

事务控制的原理：不让这些操作自动提交

1）、一系列操作应该使用同一条数据库链接才行

2）、在连接获取开始设置为手动提交，当整个全部执行完成以后，没有错误，再去手动提交。

否则回滚

3）、关闭连接。整个操作完成以后再关闭。

修改结账方法：让他满足以上条件。

结账：

3个操作：

1）、保存订单bs\_order

2）、保存订单项

3）、修改库存和销量

满足第一个条件：

1、一系列操作使用用一条连接：

1）、在一系列操作之前，获取一条连接。以后的操作，将这个连接传进去让其使用即可。

就得修改所有的方法了。

2）、其实就是用户点击结账-🡪调用结账方法--- service ,dao-🡪结账成功；

在结账开始到结账完成使用同一链接就行。

服务器每次都会为一个请求开辟一个线程去执行直到请求完成。

用户点击发请求开始-----🡪整个响应结束(使用同一线程)

服务器每次都会为一个请求开辟一个线程去执行。我们可以一个线程对应同一链接。4

我们可以在JDBCUtils获取连接的时候来获取当前线程对应的连接。

我们也需要在JDBCUitls中将每个线程对应的连接可以保存起来。

底层map来保存每一个线程对应的连接。

只要是同一线程获取到的都是同一链接。

2、设置手动提交

3、整个操作完成以后再关闭连接

将BaseDao修改，将最后关闭连接删除。我们应该一系列操作完成以后在关闭。

4、问题，BaseDao将数据库操作的异常全都处理了，导致业务逻辑方法，没办法感知到哪个操作发生了错误。

我们如果发生异常，我们可以抛出去。

1）、不能删除try catch，如果删除会导致之前调用他的所有方法都得try-catch

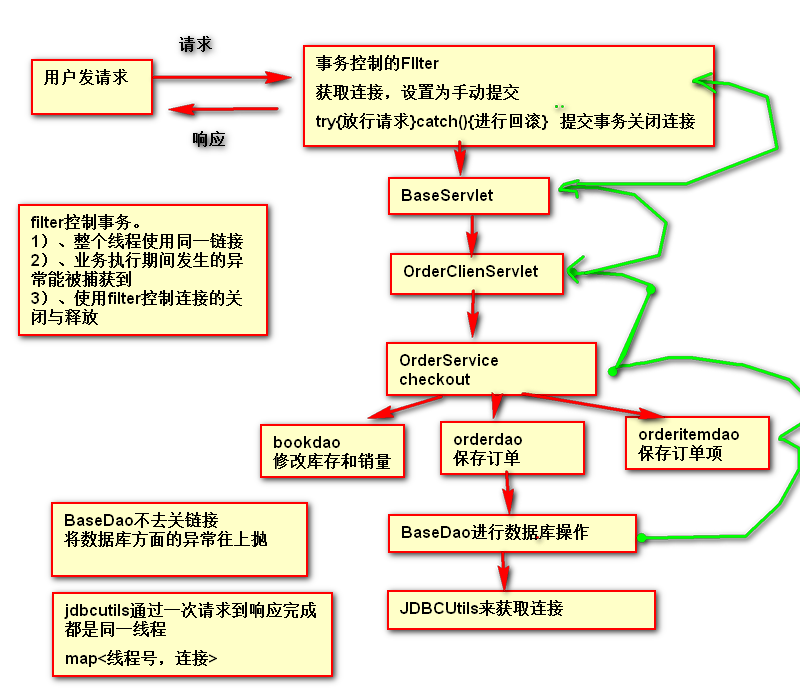
我们可以做一个新的异常抛出去。我们需要做一个运行时异常抛出去。

异常就是一层层往上抛直到有人处理为止。

现在我们将事务控制放在service中。导致其他方法关不了连接

导致其他方法要使用事务控制的时候需要在方法中获取连接、设置手动提交、回滚。

使用Filter进行事务控制



我们使用了一个Map<Long,Connection>。

一个请求到响应完成，服务器会开启一个线程进行执行。

ThreadLocal：

用来在同一线程之间共享数据

5、解决库存为负数的问题

引入了事务控制，我们只需要修改数据库底层的约束，将库存和销量改为只能保存正数。

一但购买的图书量很多。数据库在修改库存和销量的时候，就会出现异常并抛出。filter捕获到就会回滚，其他的修改就不会生效。

## 八、第八阶段 使用ajax

### 1、注册检查用户名

用户输完用户名-🡪就需要知道用户名是否可用。

M：

service –dao

我们在dao层添加一个检查用户名是否可用的方法。根据用户名获取用户，。

service🡪isExist(User user)

dao🡪getUserByUserName(User user)

V：

一会添加ajax请求

用户名输入框失去焦点-

🡪发送ajax请求-

🡪servlet返回0,1-

🡪根据返回值动态的只去修改页面部分内容

C：

2、购物车我们也可以使用ajax

购物车只是修改了部分内容。我们现在是刷新整个页面。

M：

service-dao

V：

C：

只需要每次修改数量的时候发送ajax请求。🡪

servlet收到请求🡪

将修改的那部分数据交给浏览器--?>

浏览器显示部分数据