Day12

java课程-李彦伯

- Day12
  - JSP
    - jsp脚本
    - jsp的运行原理
    - jsp的指令
    - jsp九大内置对象
      - pageContext对象
        - 域对象
        - 数据的写入,读取,删除
        - 作用范围当前jsp文件
        - 指定向其他域中存取数据
        - 依次从四大域中获取数据
        - 通过pageContext对象获取其他8大内置对象
      - out对象
      - config对象
      - exception对象
    - 如何运用jsp
    - 作业
    - 404错误总结
  - EL表达式
    - 从域中获取值(重要)
      - 获取四个域中的数据
    - 内置pageContext对象
    - 常用表达式
  - JSTL标签库
    - 下载与安装
    - jstl常用标签
      - c:if判断标签

JSP

> 如果有个需求,不同的用户请求同一个界面在页面的最上方需要显示用户的名字我们可以使用response.getWriter.write(name),但是如果需要将用户的名字放在html界面中,那么我们就需要去拼接一个html界面,会很麻烦.如果能在html界面中直接获取name的值就会方便很多,所以jsp的实质就是在html界面中嵌入java代码

jsp脚本

- `<%java代码%>`相当于写在service方法中的
- `<%=java变量或表达式>`相当于在service内部写了out.println
- `<%!java代码%>`,实质上翻译成servlet,出现在成员变量的位置
- `<!--注释内容-->`,源代码可见,翻译后的java文件可见,最终的html文件可见
- `//单行注释` `/*多行注释*/`源文件可见,翻译后的文件可见,最终的html文件不可见
- `<%--注释内容--%>`源文件可见,翻译后的servlet不可见,最终的html文件不可见

jsp的运行原理

> jsp文件在第一次被访问的时候会被解析成servlet,所以jsp的实质就是一个servlet类,将内部的html标签在servlet内部使用response.getWriter.write()的方式进行输出

jsp的指令

- page指令
  - language：jsp脚本中可以嵌入的语言种类
  - pageEncoding:当前jsp文件的本身编码—内部可以包含contentType
  - contentType:response.setContentType(text/html;charset=UTF-8)
  - import:导入java的包
  - errorPage:当前页面出错后跳转到哪个页面
  - isErrorPage:当前页面是一个处理错误的页面,配置后可以获取异常信息

- include指令,是包含文件的指令`<%@ include file="被包含的文件地址"%>`

- taglib指令,用于导入标签库,`<%@ taglib uri="标签库地址" prefix="前缀"%>`

jsp九大内置对象

名称	描述
pageContext	JSP的页面容器
session	当前会话对象
application	servletContext对象
config	servlet配置,获取servlet信息
out	页面输出对象
page	当前servlet对象
request	request对象
response	response对象
exception	JSP页面所发生的异常，在错误页中才起作用

pageContext对象

域对象

> pageContext对象也是一个域对象,所以也是可以存放数据的

数据的写入,读取,删除

- 向pageContext域中放入数据`setAtrribute(String name,Object obj);`
- 从pageContext域中获取数据`getAttribute(String name);`
- 从pageContext域中删除数据`removeAttribute(String name);`

作用范围当前jsp文件

> 这个域对象的作用范围只在当前的jsp页面中,所以我们一般并不使用这个域对象存放数据

指定向其他域中存取数据

- 向指定域中放入数据`setAttribute(String name,Object obj,int scope)`
- 从指定域中获取数据`getAttribute(String name,int scope)`
- 从指定域中删除数据`removeAttrbute(String name,int scope)`

依次从四大域中获取数据

- 依次从`pageContext`,`request`,`session`,`servletContext`这四大域中获取数据,找到即停止`findAttribute(String name)`

通过pageContext对象获取其他8大内置对象

```
pageContext.getRequest();
pageContext.getResponse();
```

out对象

> 用于向指定页面输出语句,在JSP被翻译后的文件中看到的`out.write()`,还有我们在jsp中使用`<%="abc"%>`使用的都是这个out对象,将数据写入out缓冲区,然后再从out缓冲区刷入response缓冲区(需要就讲解)

config对象

> 获取当前servlet的配置信息,我们一般不使用(需要就讲解)

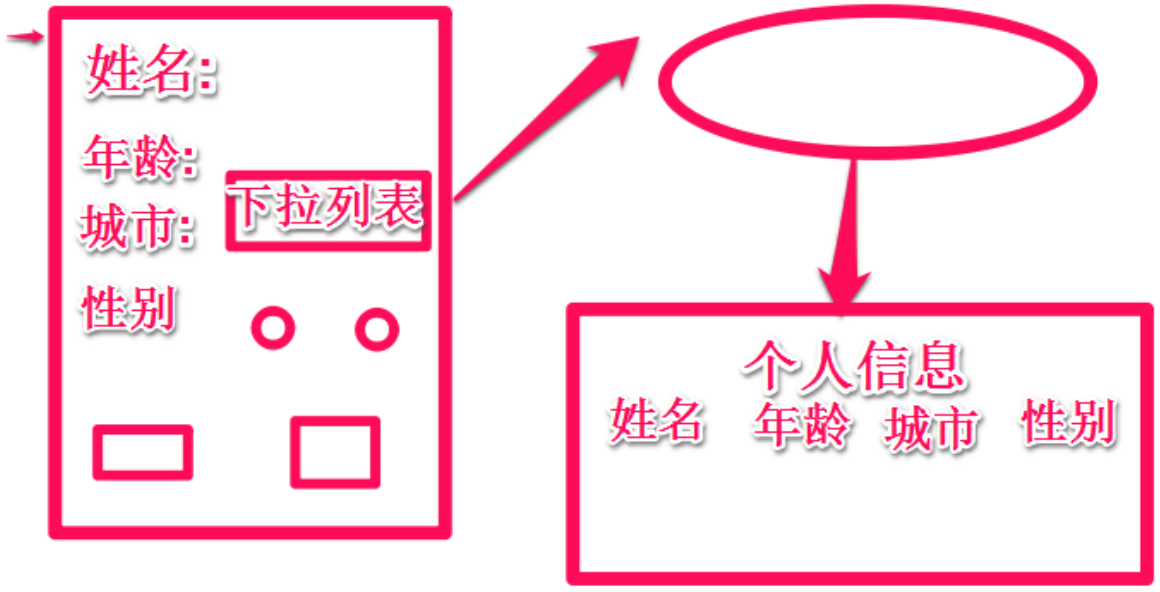
exception对象

> 如果当前页面是一个errorpage,就是在page指令的属性中配置isErrorPage=true,通过这个对象可以获取异常信息`<%= exception.getLocalizedMessage() %>`

如何运用jsp

> 在开发过程中,我们一般使用servlet进行设置数据,使用jsp进行提取数据并在页面进行展示

作业



404错误总结

1.已启动web应用里面的任何资源都是404

|-web应用是否已经部署

|-看localhost:8080,直接出不来,基本可以断定服务器没有启动

|tomcat重新装,1,2,3,将磁盘中的tomcat删除,重新解压新的tomcat,重新配

|- web.xml问题,导致web应用启动不起来,重新看控制台找异常信息

2.一启动单个资源是404

|-如果是servlet,看web.xml中的url-pattern是否和浏览器中的一致

|-如果是html,看文件名称,看这个html在不在webContent

3.界面能显示,一点击提交按钮404

|-看当前form中的action的路径,看是否是对应的servlet的url-pattern

|- 明明正确还是404,有可能是浏览器的缓存问题,在当前网页右键查看源代码,

看路径是否和eclipse中一致,如果不一致,清除浏览器缓存

4.图片不正常显示

|-在浏览器中点击右键查看源代码,看当前img的路径,是否是正确的路径

|- 看你的img放的目录是否是在webContent中

以后凡是遇到写路径先写/,除了内部转发以外,其他的所有路径都是从web应用的名字开始,

内部转发从/从webContent目录下开始

EL表达式

> EL（Express Lanuage）表达式可以嵌入在jsp页面内部，其目的在于**简化从域中获取数据的操作**

从域中获取值(重要)

获取四个域中的数据

- pageScope:能获取pageContext域中的数据
- requestScope:能获取request域中的数据`${requestScope.key}`
- sessionScope:能获取session域中的数据`${sessionScope.key }`
- applicationScope:能获取servletContext域中的数据`${applicationScope.key }`
- 依次从四个域中查找对应的值,如果存在就停止查找`${key}`

```java
public class ELServlet extends HttpServlet {
    private static final long serialVersionUID
= 1L;
    protected void doGet(HttpServletRequest req
uest, HttpServletResponse response) throws Serv
letException, IOException {
        response.setHeader("Content-Type", "tex
t/html;charset=utf-8");
        request.setAttribute("name", "a");
        User us = new User(1, 1, "b");
        request.getSession().setAttribute("use
r", us);
        List<User> li = new ArrayList<User>();
        User us1 = new User(11, 1, "c");
        User us2 = new User(1231, 2, "d");
        User us3 = new User(231, 3, "三");
        User us4 = new User(16, 4, "xx");
        li.add(us1);
        li.add(us2);
        li.add(us3);
        li.add(us4);
        getServletContext().setAttribute("lis
t", li);
        request.getRequestDispatcher("/el.js
p").forward(request, response);
    }

    protected void doPost(HttpServletRequest re
quest, HttpServletResponse response) throws Ser
vletException, IOException {
        doGet(request, response);
    }
```

```
}
```

```
${requestScope.name }<br>
${sessionScope.user.name }<br>
${applicationScope.list[1].name }<br>
```

内置pageContext对象

- 最常用的方法就是通过el获取应用程序的名称 `${pageContext.request.contextPath }`

```
<form action="${pageContext.request.contextPath}/check" method="post">
<input type="text" name="username">
<input type="password" name="pwd">
<input type="submit">
篮球<input type="checkbox" value="lq" name="hobby">足球<input type="checkbox" value="zq" name="hobby">乒乓球<input type="checkbox" value="ppq"name="hobby">
</form>
```

常用表达式

- `${249+1}`
- `${name eq "张三"}` 返回true或者false,比较两个字符串的

    - 对于字符串进行`${name == "张三"}`是比较内存地址

- `${empty user}`返回true或者false
- `${not empty user}`返回true或者false
- `${user==null?true:false}`

  > 注意`${empty ""}`也是true

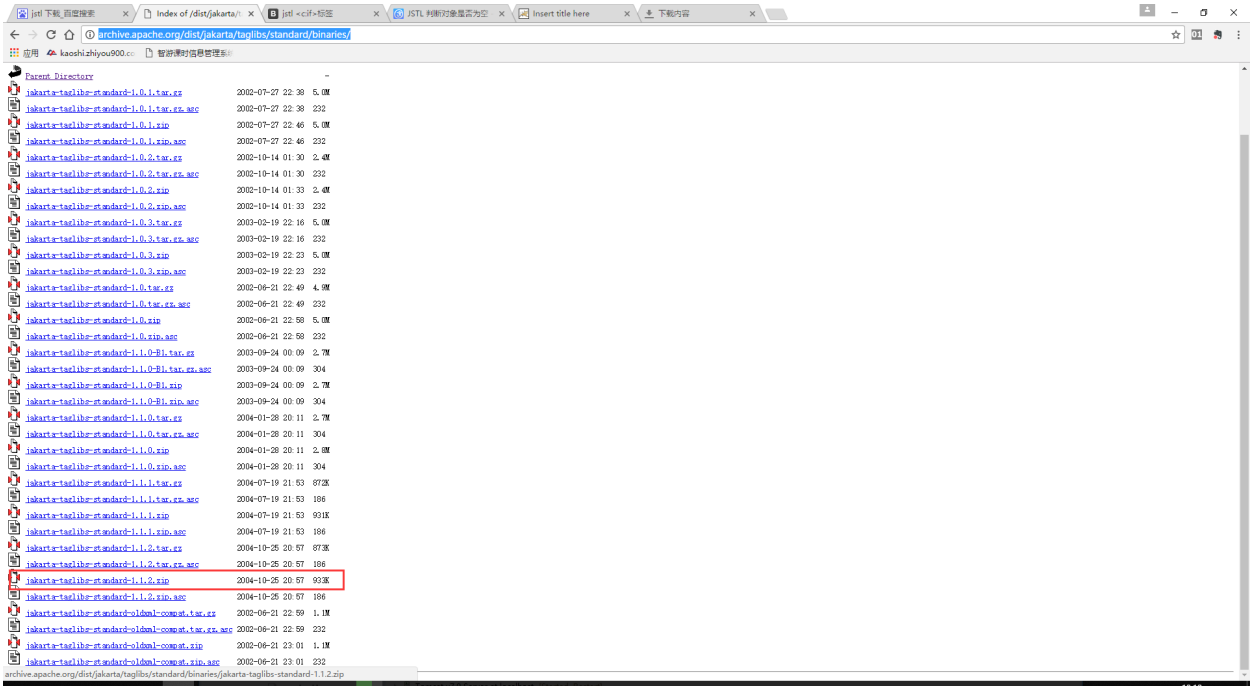
JSTL标签库

> JSTL（JSP Standard Tag Library)，JSP标准标签库，可以嵌入在jsp页面中使用标签的形式完成业务逻辑等功能。jstl出现的目的同el一样也是要代替jsp页面中的脚本代码。JSTL标准标准标签库有5个子库，但随着发展，目前常使用的是他的核心库

标签库	标签库的URI	前缀
Core	http://java.sun.com/jsp/jstl/core	c
I18N	http://java.sun.com/jsp/jstl/fmt	fmt
SQL	http://java.sun.com/jsp/jstl/sql	sql
XML	http://java.sun.com/jsp/jstl/xml	x
Functions	http://java.sun.com/jsp/jstl/functions	fn

下载与安装

- 下载地址为:http://archive.apache.org/dist/jakarta/taglibs/standard/binaries/



- 下载文件是zip,进行解压将lib文件中的两个jar包拖入WEB-INF的lib文件夹中
- 使用导入指令

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

jstl常用标签

c:if判断标签

```
<c:if test="${1==1}">
判断成功
</c:if>
```

c:choose多条件判断标签

```
<c:choose>
    <c:when test="${name == 1 }">
    <h1 style="color: red;">判断正确了</h1>
    </c:when>
    <c:when test="${name == 2 }"></c:when>
    <c:when test="${name == 3 }"></c:when>
    <c:otherwise>上述情况均不满足</c:otherwise>
</c:choose>
```

c:forEach循环遍历标签

- begin表示开始索引
- end表示结束索引
- step表示频率
- varstatus表示当前的状态
    - index属性表示循环的当前索引
    - count属性表示循环的当前次数从1开始

- items表示遍历的集合元素
- var表示每次将集合中的元素赋值给的变量名

普通for循环(使用频率不高)

```
<c:forEach begin="0" end="100" step="1" varStat
us="status">
<!--获取当前循环的索引值-->
${status.index }
<!--获取当前循环的次数-->
${status.count }
</c:forEach>
```

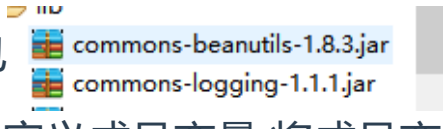
遍历集合

```
<c:forEach items="${list }" varStatus="status"
var="item">
${status.index }:${item.name }
</c:forEach>
```

BeanUtils工具类

> 此工具类可以快速方便的进行将表单提交的数据封装到对应的model对象中

使用步骤

- 导入两个jar包 

- 创建model类,定义成员变量,将成员变量的名称定义成和表单中的name属性相同,生成set和get方法

```java
import java.util.Arrays;
public class People {
    private String name;
    private int age;
    private String city;
    private String sex;
    private int salary;
    private String[] hobby;

    public String[] getHobby() {
        return hobby;
    }
    public void setHobby(String[] hobby) {
        this.hobby = hobby;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public int getAge() {
        return age;
    }
    public void setAge(int age) {
        this.age = age;
    }
    public String getCity() {
        return city;
    }
    public void setCity(String city) {
        this.city = city;
    }
```

```
    public String getSex() {
        return sex;
    }
    public void setSex(String sex) {
        this.sex = sex;
    }
    public int getSalary() {
        return salary;
    }
    public void setSalary(int salary) {
        this.salary = salary;
    }
    @Override
    public String toString() {
        return "People [name=" + name + ", ag
e=" + age + ", city=" + city + ", sex=" + sex +
", salary=" + salary
                + ", hobby=" + Arrays.toStrin
g(hobby) + "]";
    }
}
```

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/h
tml; charset=UTF-8">
<title>Insert title here</title>
</head>
<!-- /beanUtils -->
<body>
<form action="${pageContext.request.contextPath
}/beanUtils" method="post">

姓名:<input type="text" name="name"><br>
年龄:<input type="text" name="age"><br>
工资:<input type="text" name="salary"><br>
城市:<select name="city">
    <option>郑州</option>
    <option>北京</option>
    <option>上海</option>
    <option>广州</option>
    <option>深圳</option>
</select><br>
性别:<input type="radio" name="sex" value="man"
checked="checked">男 <input type="radio" nam
e="sex" value="woman">女<br>
篮球<input type="checkbox" value="lq" name="hob
by">足球<input type="checkbox" value="zq" nam
e="hobby">乒乓球<input type="checkbox" value="pp
q"name="hobby">
<input type="submit">
</form>
</body>
</html>
```

- 在servlet中获取表单数据使用`Map<String, String[]> map = request.getParameterMap();`,并创建对应的model对象`People pe = new People();`
- 调用DBUtils的`BeanUtils.populate(pe, map);`此方法会将表单数据封装到对应的pe的属性中

```
protected void doGet(HttpServletRequest reques
t, HttpServletResponse response)
            throws ServletException, IOExceptio
n {
        request.setCharacterEncoding("UTF-8");
        response.setContentType("text/html; cha
rset=UTF-8");
        People pe = new People();
        Map<String, String[]> map = request.get
ParameterMap();
        /*
         * 会找map中的key和pe中的属性名相同的,将对应
的value赋值给pe的属性
         *
         */
        try {
            BeanUtils.populate(pe, map);
        } catch (Exception e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        System.out.println(pe);
    }
```

> 注意BeanUtils会进行默认的数据类型封装,也就是说如果我们定义的成员变量的类型是int,BeanUtils会自动的将String转换为int类型,但是对于多选框`checkbox`类型的,我们必须将成员变量的类型定义为`String []`否则就会出现转换异常,如果是在有需求是List类型,那么就需要我们自己进行转换