* JSP基础学习：

1. jsp运行在服务器端
2. jsp：Java server page
3. jsp的基础是servlet
4. jsp是综合技术Jsp=html+java片段+jsp标签（servlet）+javascript（css）
5. jsp无需配置，直接使用，修改jsp文件，不需要重启服务器
6. jsp访问：<http://localhost:8080/web>应用名称/\*\*.jsp

Mvc模式=Jsp+java类（service、javabean）+servlet

1. jsp是一种用于开发动态web资源的技术

* 快速入门：

**练习1：**在jsp页面输出当前时间（用于讲解JSP的 运行原理jsp与servlet的关系）

翻译 编译

ShowTime.class

ShowTime\_servlet.java

showTime.Jsp

如果是第一次访问该jsp文件：web服务器会把showTime.jsp文件翻译成servlet(showTime.java)文件，再将其编译成一个showTime.class，然后把class文件加载到内存

如果是第二次及以后访问，直接访问装载内存中的实例

装载到内存实例

Web服务器

第二次访问

第一次访问

第二次访问浏览器：http://localhost:8080/jsp1/showTime.jsp

第一次访问浏览器：http://localhost:8080/jsp1/showTime.jsp

单例：所有请求都用一个对象来处理，比如我们常用的service和dao层的对象通常都是单例的

多例：每个请求用一个新的对象来处理，比如：action

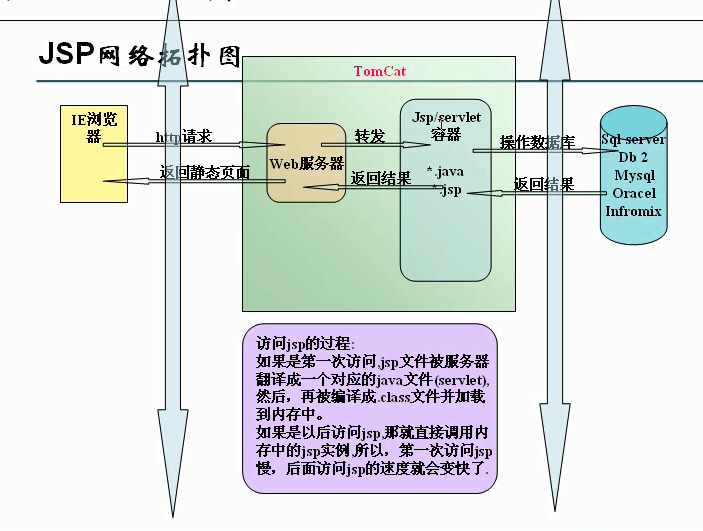
单例模式和多例模式说明：

1. 单例模式和多例模式属于对象模式
2. 单例模式的对象在整个系统中只有一份，多例模式可以有多个实例
3. 他们都不对外提供构造方法，即构造方法都为私有

综上所述：jsp为单例模式

如果jsp文件修改了，则跟第一次访问一样

* Jsp网络拓扑图



* 考虑问题：

1. Web服务器是如何调用并执行一个jsp页面（见上面描述）
2. Jsp页面中的html排版标签是如何被发送到客户端的？（转成servlet文件，out.write()）
3. Jsp页面中的java代码服务器是如何执行的？（转成servlet文件，去掉<%%>原样显示）
4. web服务器在调用jsp时，会给jsp提供一些什么java对象？（提供了九个内置对象：out:向客户端输出数据，字节流、request:接受客户的http请求、response:封装jsp的产生回应、session：用于保存用户的信息，跟踪用户的行为、application：多个用户共享对象、pageContext、exception、page、config）

修改jsp模板：

Window->Preferences->Web->JSP Files->Editor->Templates->New JSP File(html)->Edit

* JSP基础—语法（标签）
  1. 指令元素—用于从jsp发送一个信息到容器，比如设置全局变量，文字编码，引入包等

1. Page指令

<%@ page contentTye=”text/html;charset=utf-8”%>

Page指令中常用属性有：

Language=””jsp中嵌入的代码是XX，通常是java

Import=”包名”在该页面引入包或者某个具体的类

可引入多个包，中间用逗号隔开

Session=[true|false] 是否在jsp页面获取session对象

Buffer=[none|8k|指定大小]给out对象使用的缓存区是多大，默认8K

AutoFlash=[true|false]当buffer满后，是否自动刷新到浏览器

isThreadSafe=[true|false]默认true表示该jsp的线程安全由程序员控制，false则对应的Servlet将实现线程安全接口

errorPage=”相对JSP页面”，当jsp页面出现错误时，自动跳转到指定jsp页面，要求使用相对路径，当“/”开始时，表示当前web应用的根目录。如果不以“/”开头，则表示相对本jsp页面

isErrorPage=[false|true]指定该jsp页面是否专门用于作为error页面，默认false，如果设为true，则可以在该页面直接使用exception内置对象(用的很少)

contentType=”text/html;charset=utf-8”指定网页以什么方式显示页面

pageEncoding=”utf-8”指定servlet引擎以什么方法翻译jsp-🡪servlet并指定网页以什么方式显示页面

1. Include指令

<%@ include file=”filename”%>

该指令用于引入一个文件（通常是jsp文件），jsp引擎会把两个jsp文件翻译成一个servlet文件，因此也称为静态引入

被引入的jsp文件，只需保留page指令即可，<html><body>均可省略

1. Taglib指令---这个指令允许在jsp页面使用自定义的标签

<myTag:yourTag num1=”123”>

* 1. 脚本元素---可以理解就是java的片段scriplet

1. Scriplet <%java代码%>
2. 表达式 <%=java表达式%> <%=rs.getString(1)%>
3. Declaration声明

<%! 成员变量声明 %> 如：<%! Int count=10%>

<%! 函数声明%>

<%! Int jisuanqi(int num){

Int result=0;

For(int i=0;i<num;i++){

Result=result+I;

}

}%>

备注：

使用<% int i=0%>局部变量

使用<%! Int i=0%>成员变量（该jsp对应的servlet的成员变量）

成员变量和局部变量的区别：

根据定义变量位置的不同，可以将变量分为成员变量和局部变量

成员变量是在类范围内定义的变量

局部变量是在一个方法内定义的变量

成员变量可以分为：

实例属性（不用static修饰）：随着实例属性的存在而存在

类属性（static修饰）：随着类的存在而存在

成员变量无需显式初始化，系统会自动对其进行默认初始化

局部变量可以分为：

形参：在整个方法内有效

方法局部变量：从定义这个变量开始到方法结束这一段时间

代码块局部变量：从定义这个变量开始到代码块结束这一段时间

面试题:

<%! Int i=90;%>

<%

Int j=90;

Out.println(“i=”+(++i)+”j=”+(++j));

%>

第一次访问输出：91 91

第二次访问输出：92 91

第三次访问输出：93 91

如果此时另外一台电脑访问此时输出：94 91

备注：函数不能再<%%>中定义

* 1. 动作元素---使用xml语法写的，是jsp规定的一系列标准动作，在容器处理jsp时，就执行相应的操作

<jsp:useBean>//创造一个JavaBean实例

<java:setProperty>//给一个javabean实例设置初始值

<java:param>//给一个jsp设置参数，常常与<jsp:include>结合使用

<jsp:include file=””>

<jsp:param name=”” value=””/>

</jsp:include>

<java:getProperty>//引入一个javabean实例的成员变量

<jsp:include>//引入另一个文件(动态引入：两个文件分开编译)

<jsp:include file=””>

</jsp:include>---动态引入

<%@ include file=””%>---静态引入

相同点：将一个文件引入另一个文件

不同点：静态引入将两个文件编译成一个文件，所被引入的文件不要包含body、html；动态引入时间两个文件分开编译

<jsp:plugin>//使用插件

<jsp:forward>//转发(用的较多)

<jsp:forward page=”0729.jsp”></jsp:forward>

作用：，在开发jsp的过程中，我们通常将jsp放在web-inf目录下，目的是为了防止用户直接访问jsp文件，因此在web-root下有一个入口，目的是转发

使用jsp：forward跳转到对应jsp文件

<jsp:forward page=”/WEB-INF/moveto.jsp”></jsp:forward>

<jsp:fallback>

* 1. Jsp文件的注释

<!-- -->html专用注释可用在jsp中

<%-- 注释内容 --%>jsp专用注释

* JSP九大内置对象：

Out：向客户端输出数据：字节流 out.println()

Request：接受客户端的http请求🡨🡪相当于servlet中的HttpServletRequest

getParameter(String name)//name表示表单的参数名

getParameterValues(String name)//使用得到是String[]

setAttributes(String name,object obj)//设置名字为name的obj,值为obj

getAttribute(String name)//返回由name指定的属性值，如果不存在就返回null

getCookie();

response:封装jsp产生的响应🡨🡪相当于HttpServletResponse

addCookie(Cookie cookie);

sendRedirect(“./welcome.jsp”);

session:用于保护用户的信息，跟踪用户的行为🡨🡪HttpServlet

setAttribute(String name,object obj)

getAttribute(String name)

application:多个用户共享该对象，可以做计算器—servletContext

pageContext:代表jsp页面的上下文，也是一个域对象，可以setAttribute(),作用范围只是本页面

exception//代表运行时的一个异常 getMessage()

page//代表jsp这个实例本身

config//代表jsp对应的servlet的配置，可以得到web.xml中的参数—ServletConfig

练习：jsp版本计算器：

先用两个页面来完成该计算器：myCal.jsp(界面)和myResult.jsp（计算）

1. 如何接收jsp页面提交的数据，并处理

<%

Request.getParameter(“num1”);//获取第一个字符串

%>

1. Jsp如何使用java片段

<%

Request.getParameter(“num1”);//获取第一个字符串

%>

1. Jsp中如何使用javascript完成对数据合法性的校验
2. 引入js<script src=”js/js1.js” type=”text/javascript”></script>
3. Js中定义函数：

Function checkNum(){

Var num1=document.getElementById(“num1”);

Var num2=document.getElementById(“num2”);

//判断输入是不是为空

If(num1 == “”||num2 == “”){

Window.alert(“输入不能为空”);

Return false；

}

Else{

Return true;

}

}

调用函数：

<input type=”submit” value=”提交” onclick=”return checkNum()”>

备注：onclick=”return checkNum()”一定要带return

或者在form中写<form action=”xxx” method=”post” onsubmit=”return checkNum()”>

Js的正则表达式：

正则表达式的创建方式：

字面量创建方式：var reg=/pattern/flags

实例创建方式：var reg=new RegExp(pattern,flags);

Pattern:正则表达式

Flags:标识（修饰符）

标识（修饰符）包括：

1.i 忽略大小写匹配

2.m 多行匹配，即在到达一行文本末尾时还会继续寻找下一行中是否与正则匹配的项

3.g 全局匹配，模式应用于所有字符串，而非在找到第一个匹配项时停止

元字符：

\d : 0-9之间的任意一个数字 \d只占一个位置

\w : 数字，字母 ，下划线 0-9 a-z A-Z \_

\s : 空格或者空白等

\D : 除了\d

\W : 除了\w

\S : 除了\s

. : 除了\n之外的任意一个字符

\ : 转义字符

| : 或者

() : 分组

\n : 匹配换行符

\b : 匹配边界 字符串的开头和结尾 空格的两边都是边界 => 不占用字符串位数

^ : 限定开始位置 => 本身不占位置

$ : 限定结束位置 => 本身不占位置

[a-z] : 任意字母 []中的表示任意一个都可以

[^a-z] : 非字母 []中^代表除了

[abc] : abc三个字母中的任何一个 [^abc]除了这三个字母中的任何一个字符

代表次数的量词元字符

\* : 0到多个

+ : 1到多个

? : 0次或1次 可有可无

{n} : 正好n次；

{n,} : n到多次

{n,m} : n次到m次

验证数字的正则表达式集   
验证数字：^[0-9]\*$   
验证n位的数字：^\d{n}$   
验证至少n位数字：^\d{n,}$   
验证m-n位的数字：^\d{m,n}$   
验证零和非零开头的数字：^(0|[1-9][0-9]\*)$   
验证有两位小数的正实数：^[0-9]+(.[0-9]{2})?$   
验证有1-3位小数的正实数：^[0-9]+(.[0-9]{1,3})?$   
验证非零的正整数：^\+?[1-9][0-9]\*$   
验证非零的负整数：^\-[1-9][0-9]\*$   
验证非负整数（正整数 + 0） ^\d+$   
验证非正整数（负整数 + 0） ^((-\d+)|(0+))$   
验证长度为3的字符：^.{3}$   
验证由26个英文字母组成的字符串：^[A-Za-z]+$   
验证由26个大写英文字母组成的字符串：^[A-Z]+$   
验证由26个小写英文字母组成的字符串：^[a-z]+$   
验证由数字和26个英文字母组成的字符串：^[A-Za-z0-9]+$   
验证由数字、26个英文字母或者下划线组成的字符串：^\w+$   
验证用户密码:^[a-zA-Z]\w{5,17}$ 正确格式为：以字母开头，长度在6-18之间，只能包含字符、数字和下划线。   
验证是否含有 ^%&',;=?$\" 等字符：[^%&',;=?$\x22]+   
验证汉字：^[\u4e00-\u9fa5],{0,}$   
验证Email地址：/^([a-zA-Z0-9]+[\_|\\_|\.]?)\*[a-zA-Z0-9]+@([a-zA-Z0-9]+[\_|\\_|\.]?)\*[a-zA-Z0-9]+\.[a-zA-Z]{2,3}$/  
验证InternetURL：^http://([\w-]+\.)+[\w-]+(/[\w-./?%&=]\*)?$ ；^[a-zA-z]+://(w+(-w+)\*)(.(w+(-w+)\*))\*(?S\*)?$   
验证电话号码：^(\(\d{3,4}\)|\d{3,4}-)?\d{7,8}$：--正确格式为：XXXX-XXXXXXX，XXXX-XXXXXXXX，XXX-XXXXXXX，XXX-XXXXXXXX，XXXXXXX，XXXXXXXX。   
验证身份证号（15位或18位数字）：^\d{15}|\d{}18$   
验证一年的12个月：^(0?[1-9]|1[0-2])$ 正确格式为：“01”-“09”和“1”“12”   
验证一个月的31天：^((0?[1-9])|((1|2)[0-9])|30|31)$ 正确格式为：01、09和1、31。   
整数：^-?\d+$   
非负浮点数（正浮点数 + 0）：^\d+(\.\d+)?$   
正浮点数 ^(([0-9]+\.[0-9]\*[1-9][0-9]\*)|([0-9]\*[1-9][0-9]\*\.[0-9]+)|([0-9]\*[1-9][0-9]\*))$   
非正浮点数（负浮点数 + 0） ^((-\d+(\.\d+)?)|(0+(\.0+)?))$   
负浮点数 ^(-(([0-9]+\.[0-9]\*[1-9][0-9]\*)|([0-9]\*[1-9][0-9]\*\.[0-9]+)|([0-9]\*[1-9][0-9]\*)))$   
浮点数 ^(-?\d+)(\.\d+)?$

练习：

将myCal.jsp(界面)和myResult.jsp（计算）页面合并为一个jsp页面，计算和显示结果都在mycal.jsp全部显示

提交后数据保留<input type=”text” value=”<%=num1%>”

<%

String num1 = request.getParameter("num1");

String num2 = request.getParameter("num2");

String cal = request.getParameter("cal");

**int** num1i = Integer.parseInt(num1);

**int** num2i = Integer.parseInt(num2);

**int** res = 0;

**if** (num1 != **null** && num2 != **null**) {

**if** ("+".equals(cal)) {

res = num1i + num2i;

} **else** **if** ("-".equals(cal)) {

res = num1i - num2i;

} **else** **if** ("\*".equals(cal)) {

res = num1i \* num2i;

} **else** **if** ("/".equals(cal)) {

res = num1i / num2i;

}

}

%>

<form action=*"myCal.jsp"* method=*"post"*>

<span>第一个数：</span>

<input type=*"text"* name=*"num1"* value=*"*<%=num1 %>*"* id=*"num1"*placeholder=*"请输入第一个数字"*>

<br>

<span>第二个数：</span>

<input tyep=*"text"* name=*"num2"* id=*"num2"* value=*"*<%=num2 %>*"* placeholder=*"请输入第二个数字"*> <br>

<span>请选择运算符：</span> <select name=*"cal"*>

<option value=*"+"*>加</option>

<option value=*"-"*>减</option>

<option value=*"\*"*>乘</option>

<option value=*"/"*>除</option>

</select>

<br>

<input type=*"submit"* value=*"提交"* onclick="return checkNum()">

<hr>

<span>计算结果：<%=res %></span>

</form>

对于页面输入验证：

1. 通过js现在客户端验证（js有正则表达式验证）
2. 在服务器端，做二次验证（java有正则表达式验证）

代码格式调整：source—》format

在软件开发中，根据这两门技术的特点，让他们各自负责各的，servlet只负责响应请求产生数据，并把数据通过转发技术带给jsp，数据的显示jsp来做

* Model模式用户管理系统

Model模式

一个简单的用户登录系统

1. 如何查看电脑中是否已经安装mysql

答：在管理-服务中查看是否有对应的服务-状态启用

1. Jane.wang电脑mysql数据库的用户名密码？？？

用户名 ：root

密码：1234

使用Navicat连接本地数据库：mydatabase

在Java ee的开发中有几个重要的开发模式：

1.model1模式（界面和业务逻辑混合在一起） 2.model1.x分层模式（界面、业务逻辑分离）3.MVC模式

Model1模式的缺点：

1. 表现层和业务逻辑混合在一起，乱！
2. 在开发过程中，不利于多人的协同开发
3. 不利于后期的维护

Model1模式的优点：

1. 简单，开发速度比较快
2. 比较适合开发小的项目

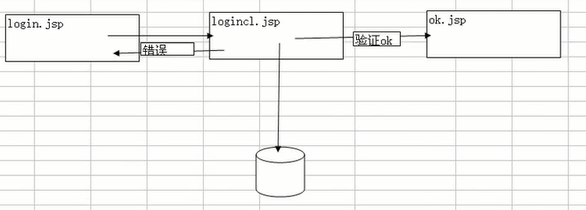
练习：

用户登录系统：model1模式

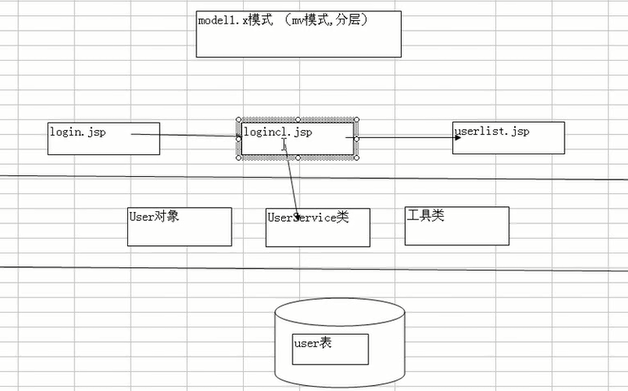
Login.jsp（登录界面）—》logincl.jsp（处理）🡪ok（成功界面）

Login.jsp(登录界面)---》logincl.jsp(处理)—》fail（失败返回登录界面）

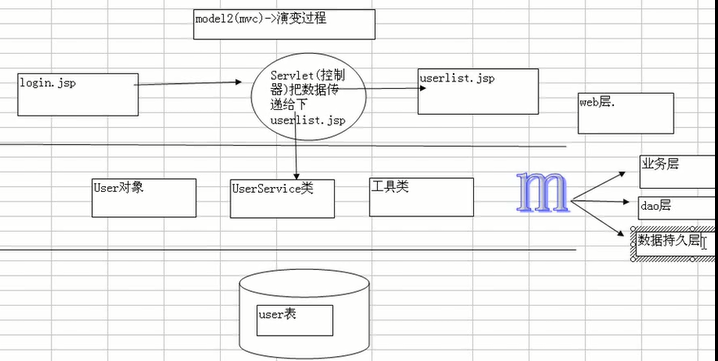
Logincl.jsp可能会暴露数据库信息，ok.jsp可能也会操作数据库，代码重复



升级为model1.x模式：



由于logincl.jsp为jsp专门做页面，在此处不适合，可升级为model2模式即MVC模式：



DAO层：data access object,数据访问对象，即专门对数据库进行操作的类，一般来说dao不含有业务逻辑