# 一. 概念性知识

## HttP :超文本传输协议 客户端与服务器端的信息传输

## html 超文本标记语言 描述网页文档的标记语言

## xml 可扩展标记语言

## java内部使用的编码格式是 unicode，一个java类里面不能有两个public 类，否则报错。但是也可以没有public。

# 1.2 radio checkbox select

radio

<input type=*"radio"* name=*"sex"*checked=*"checked"*/>男

<input type=*"radio"*name=*"sex"* />女</td>

checkbox

<td><input type=*"checkbox"* name=*"ji"* value=*"a"*

select

<select name=”select”>

<option value=”a”>经理</option>

<option value=”b”>服务员</option>

</select>

# 二.style的植入方式

1. <style type="text/css"> </style>

2. <style type="text/css">

@import "mystyledao.css";

</style>

另一个文件 中首行 @charset "utf-8";

3. <link href="myyangshi.css" rel="stylesheet" type="text/css"/>

# 三. 引入JavaScript文件

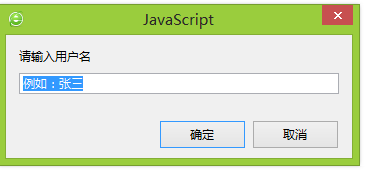
## 1.引入JavaScript文件

**<script type=“text/javascript” src=””/>**

**<script type="text/javascript"> 内容 </script>**

**2. 简单**的**函数及用法**

**1.name=prompt("请输入用户名","例如：张三")；**



**2.document.write("我是第一句<br/>")；**

**3.alert("你刚才输入的是："+name);**

**4.typeof price //检查price的类型**

**5.parseInt(x) 强制转型**

**6.<a href="javascript:b()">请不要点击我，点击之后，后果自负</a>**

**7.if(isNaN(x)==ture) 判断是不是数字 不是数字返回true**

**8. document.getElementById**

**document.getElementByClassName**

**document.getElementsByTagNmae（其实是获取它下面的所有的集合）**

**9.取得数值 var x=document.getElementById('shuzi1').value;**

## 3.JavaScript 函数

**<script type="text/javascript">**

**window.onload=function()**

**{**

**var wo=document.getElementById('zhangyujin');**

**wo.onclick=function() {**

**alert("你好");**

**}**

**}**

**</script>**

## 4.document改变style属性

**<script type="text/javascript">**

**function kaiguan(){**

**if(document.getElementById('down').style.display=="block")**

**{ document.getElementById('down').style.display='none';**

**}**

**else{document.getElementById('down').style.display=='block';}**

**}**

**</script>**

## 5.JavaScript取得java变量

**Request.setAttribute(“name”,”张宇晋”)**

**var  v = "<%=name %>"**

## 6. 隐藏标签元素位置占用与否

1.不占用 display

<div>display：元素的位置不被占用</div>

  if(div1.style.display=='block') div1.style.display='none';    else div1.style.display='block';

2.占用  visibility

 if(div3.style.visibility=='visible') div3.style.visibility='hidden';    else div3.style.visibility='visible';

# 四、jQuery

## 1.jQuery函数

**$(function(){ //加载页面事件**

**$("button").click(function(){**

**//通过jquery选择要操作的元素**

**$(“div”).html(“<h1>更改后的内容<h1>”)**

**$(“div”).text(“更改后的内容”)**

**Wo.innerHtml**

**Wo.html() 打印**

**}}**

## 2.控制css

**$(function(){**

**$("button").css("background-color","red");**

**$(".up").css({height:100px,width:102px,});**

**$("#wo,button").css({font-size:12px});**

**})**

## 3.选择器

### 1.层级选择器

#### 父子 $(“div > ul”).css()

#### 兄弟

**$(“div+ul”).css()**

**//只是将与div最近的ul加上了属性**

**$(“div ~ ul”).css()**

**//将div的所有的ul兄弟（不管是否相邻）全部加上了属性**

### 2.过滤选择器

#### 1.选择第一个和最后一个

**$("#down h2:first").css(“display”,”block”)**

**$("#down h2:final").css(“display”,”block”)**

#### 2.选择奇数偶数

**$("#down h2:even").css(“display”,”block”) 偶数**

**$("#down h2:odd").css(“display”,”block”) 奇数**

#### 3.选择索引d

**$("#down h2:eq(2)").css(“display”,”block”) 索引等于2**

**$("h2:lt(5)").css(“display”,”block”) 索引小于5**

**$(“li:gt(5)").css(“display”,”block”)索引大于5**

**$(“li:has(‘span’) ").css(“display”,”block”) 包括span的标签**

**$(“li:hidden").css(“display”,”block”) 选择隐藏的标签**

### 3.过滤选择器【】

**$(“div[class]).css(“display”,”block”)**

**$(“div[class=‘wo’]).css(“display”,”block”)**

**$(“:text”).css(“display”,”block”) 选择表单元素**

## 4.获取文本框的值

**var wo = $(“input”).val();**

**document.getElementById('文本框ID').value**

## 5.修改图片

$("img").mouseover(function(){

## $(this).attr("src","图片地址")

})

# 六、AJAX

<script>

**var** xmlHttp;

**function** createXMLHttpRequest() {

//创建XMLHttpRequest对象

**if** (window.XMLHttpRequest) {xmlHttp = **new** XMLHttpRequest();}

**else**{xmlHttp = **new** ActiveXObject("Microsoft.XMLHTTP");}

}

//处理从服务器返回的响应

**function** processor() {

//响应完成

**if** (xmlHttp.readyState == 4) {

//响应成功

**if** (xmlHttp.status == 200) { //HTTP状态码，安卓中也有，忘记了吧 哈哈

**var** txt = xmlHttp.responseText;

document.getElementById("div\_msg").innerHTML = txt;

}

}

}

**function** requestData() {

createXMLHttpRequest();

//将状态触发器绑定到函数

xmlHttp.onreadystatechange = processor;

//通过请求参数中使用Math.random()获取随机数来避免使用缓存中的数据

xmlHttp.open("GET","RequestDataServlet?name=wangmingming&address=beijing&t="+ Math.random(), **true**);

xmlHttp.send(**null**);

}

</script>

</head>

<body>

<h2>以GET方式向Servlet提交请求</h2>

<input type=*"button"* value=*"请求数据"* onClick="requestData()"/>

<div id=*"div\_msg"*></div>

</body>

</html>

## 1.xmlhttp.send()

**//Get方式提交：**

**/\* xmlhttp.open("GET","testServlet?name="+userName,true);**

**xmlhttp.send(null);**

**或者可以这样**

**xmlhttp.open("GET","testServlet”,true);**

**xmlhttp.send(name="+userName);**

**//Post方式请求的代码**

**xmlhttp.open("POST","testServlet",true);**

**//POST方式需要自己设置http的请求头**

**xmlhttp.setRequestHeader("Content-Type","application/x-www-form-urlencoded");**

**//POST方式发送数据**

**xmlhttp.send("name="+userName); \*/**

## 2. ReadyState取值 描述

**0 描述一种"未初始化"状态；此时，已经创建一个XMLHttpRequest对象，但是还没有初始化。**

**1 描述一种"发送"状态；此时，代码已经调用了XMLHttpRequest open()方法并且XMLHttpRequest已经准备好把一个请求发送到服务器。**

**2 描述一种"发送"状态；此时，已经通过send()方法把一个请求发送到服务器端，但是还没有收到一个响应。**

**3 描述一种"正在接收"状态；此时，已经接收到HTTP响应头部信息，但是消息体部分还没有完全接收结束。**

**4 描述一种"已加载"状态；此时，响应已经被完全接收。**

## 3. XMLHttpRequest

**创建一个XMLHttpRequest对象的过程因浏览器实现的不同而有所区别。**

**如果浏览器支持XMLHttpRequest对象作为一个窗口属性(所有普通的浏览器都是这样的，除了IE 5和IE 6之外)，**

**那么，代码可以调用XMLHttpRequest的构造器。如果浏览器把XMLHttpRequest对象实现为**

**一个ActiveXObject对象(就象在IE 5和IE 6中一样)，那么，代码可以使用ActiveXObject的构造器。 \*/**

**function** createXMLHttpRequest() {

//创建XMLHttpRequest对象

**if** (window.XMLHttpRequest) {

xmlHttp = **new** XMLHttpRequest();

}

**else**{

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

}

}

## 4.取得AJAX的值

**String name=request.getParameter("name");**

**String address=request.getParameter("address");**

**String str="Welcome!"+name+",your address is "+address+"!";**

**out.println(str);**

**out.flush();**

**out.close();**

# 七、dwr（Direct Web Remoting）

## 1.配置web.xml

**<servlet>**

**<servlet-name>dwr-invoker</servlet-name>**

**<servlet-class>org.directwebremoting.servlet.DwrServlet</servlet-class>**

**//调试的时候用到，调试完毕一定要记得关闭**

**<init-param>**

**<param-name>debug</param-name>**

**<param-value>true</param-value>**

**</init-param>**

**//这个是出现session错误的时候加上的**

**<init-param>**

**<param-name>crossDomainSessionSecurity</param-name>**

**<param-value>false</param-value>**

**</init-param>**

**</servlet>**

**<servlet-mapping>**

**<servlet-name>dwr-invoker</servlet-name>**

**<url-pattern>/dwr/\*</url-pattern>**

**</servlet-mapping>**

## 2.创建一个java文件

**public class Test {**

**/\*\***

**\* 调用没有返回值和参数的Java方法**

**\*/**

**public void testMethod1() {**

**System.out.println("testMethod1()");**

**}**

## 3.编写dwr.xml

**<dwr>**

**<!--allow标签中包含可以暴露给javascript访问的东西-->**

**<allow>**

**<!--create标签中指定javascript可以访问的java类名-->**

**<!--javascript属性指定javascript访问代码对象时使用的名称-->**

**<!--creator属性意味关DWR调用类的默认构造函数来获得实例-->**

**<create javascript="Test" creator="new">**

**<!--指定全限定类-->**

**<param name="class" value="cn.jbit.dwr.test.Test"/>**

**<!--指定要公开的javascript方法。不指定的话就公开所有方法-->**

**<include method="testMethod1"/>**

**<include method="testMethod2"/>**

**<include method="testMethod3"/>**

**<include method="testMethod4"/>**

**<include method="testMethod5"/>**

**<include method="testMethod6"/>**

**<include method="testMethod7"/>**

**<include method="testMethod8"/>**

**<include method="testMethod9"/>**

**</create>**

**<!--convert元素的作用是告诉DWR在服务器端Java对象表示和序列化的JavaScript之间如何转换类型-->**

**<!--这些类型包括Java原生类型和它们各自的封装类表示 ，还有String、Date、数组和集合类型-->**

**<!--match属性指定要转换的javabean名称-->**

**<!--converter属性指定转换的方式采用JavaBean规范-->**

**<!--注意：如果List、Set或者Map中的元素均为简单类型（包括其封装类）或者String、Date、数组和集合类型，则不需要convert标签-->**

**<convert match="cn.jbit.dwr.entity.User" converter="bean"></convert>**

**</allow>**

**<!--signatures用来声明java方法中List、Set或者Map参数所包含的确切类，以便java代码作出判断-->**

**<!--对于java返回值为List的情况，DWR将其转化 为Object数组，传递给javascript;对于java返回值为Map的情况，DWR将其转化为一个Object，其中Object的属性为原Map的key值，Object的属性值为原Map相应的value值-->**

**<signatures>**

**<![CDATA[**

**import java.util.List;**

**import cn.jbit.dwr.entity.User;**

**import cn.jbit.dwr.test.Test;**

**Test.testMethod7(List<User>);**

**Test.testMethod9(Map<String,User>);**

**]]>**

**</signatures>**

**<!-- <dwr>**

**dwr中创建转换对象的方式如下：**

**<allow>**

**因为要使用ssh2框架，所以将bean的创建交给spring，而不是之前的class**

**<create creator="spring" javascript="demo" scope="application">**

**param中name因为要调用bean中的方法，所以使用beanName,value是struts.xml中配置的业务类**

**<param name="beanName" value="userAction"></param>**

**</create>**

**<convert>标签是将converter中定义的转换器映射到具体类型 ，即将match路径下的User转换成bean，供上述的beanName使用**

**<convert converter="bean" match="cn.hfut.ssh.model.User"/>**

**</allow>**

**</dwr> -->**

**</dwr>**

## 4.开始使用

**<script type="text/javascript" src="/DWRSample/dwr/interface/Test.js"></script>**

**<script type="text/javascript" src="/DWRSample/dwr/engine.js"></script>**

**<script type="text/javascript" src="/DWRSample/dwr/util.js"></script>**

**<script type="text/javascript">**

**function callTestMethod1(){**

**Test.testMethod1();**

**alert("123");**

**}**

**</script>**

**</head>**

**<body>**

**<input type="button" value="调用Java方法" onclick="callTestMethod1()" />**

**</body>**

**</html>**

**function callTestMethod2(){**

**Test.testMethod2(callBackFortestMethod2);**

**}**

**function callBackFortestMethod2(data){**

**//其中data接收方法的返回值**

**//可以在这里对返回值进行处理和显示等**

**alert("the return value is "+data);**

**}**

# 八、Ibatis2

## 1.建立配置文件与数据库对应（相当于hibernate中的那个）

**<transactionManager type="JDBC">**

**<dataSource type="SIMPLE">**

**<property name="JDBC.Driver" value="com.mysql.jdbc.Driver" />**

**<property name="JDBC.ConnectionURL" value="jdbc:mysql://localhost:3306/ibatis" />**

**<property name="JDBC.Username" value="root" />**

**<property name="JDBC.Password" value="147094" />**

**<property name="Pool.MaximumActiveConnections" value="10" />**

**<property name="Pool.MaximumIdleConnections" value="5" />**

**<property name="Pool.MaximumCheckoutTime" value="120000" />**

**<property name="Pool.TimeToWait" value="500" />**

**<property name="Pool.PingQuery" value="select 1 from ACCOUNT" />**

**<property name="Pool.PingEnabled" value="false" />**

**<property name="Pool.PingConnectionsOlderThan" value="1" />**

**<property name="Pool.PingConnectionsNotUsedFor" value="1" />**

**</dataSource>**

**</transactionManager>**

**<!-- 映射文件 --> 相当于hibernate中的那个与数据库对应的文件，但是这里的区别就出来了，这里的区别是它是sql语句的集合**

**<sqlMap resource="com/itmyhome/User.xml" />**

## 2. 编写User.xml文件，这里用来编写数据库操作语句

**<?xml version="1.0" encoding="UTF-8"?>**

**<!DOCTYPE sqlMap**

**PUBLIC "-//iBATIS.com//DTD SQL Map 2.0//EN"**

**"http://www.ibatis.com/dtd/sql-map-2.dtd">**

**<sqlMap namespace="User">**

**下面这个user是别名，就是代表后面的type的java文件 ，下文用它来对数据进行草操作**

**<typeAlias alias="user" type="com.itmyhome.User" />**

**</select>**

**<!-- 查询所有的 --> 这里的id在java文件中会调用的**

**<select id="getAllUser" resultClass="user">**

**select \* from user**

**</select>**

**</sqlMap>**

## 3.增删改查

**<sqlMap namespace="User">**

**<typeAlias alias="user" type="com.itmyhome.User" />**

**<!-- 查询 -->**

**<!-- parameterClass 传入参数的类型 int = java.lang.Integer**

**resultClass 返回结果的类型 -->**

**<select id="getUser" parameterClass="int" resultClass="user">**

**select \* from user where id = #id#**

**</select>**

**<!-- 查询所有的 -->**

**<select id="getAllUser" resultClass="user">**

**select \* from user**

**</select>**

**<!-- 插入 -->**

**<insert id="insertUser" parameterClass="user">**

**insert into user(id,name,age) values(#id#,#name#,#age#)**

**</insert>**

**<!-- 更新 -->**

**<update id="updateUser" parameterClass="user">**

**update user SET name=#name#, age=#age# WHERE id = #id#**

**</update>**

**<!-- 删除 -->**

**<delete id="deleteUser" parameterClass="java.lang.Integer">**

**delete from user where id = #id#**

**</delete>**

## 4.取得sqlMap

**String resource = "SqlMapConfig.xml";**

**Reader reader = Resources.getResourceAsReader(resource); //读取配置文件**

**sqlMap = SqlMapClientBuilder.buildSqlMapClient(reader);**

## 5.进行测试

**List<User> userList = sqlMap.queryForList("getAllUser");**

**System.err.println("得到list的大小为"+userList.size());**

**for(User user:userList){**

**System.out.println(user.toString());**

**}**

**user = (User)sqlMap.queryForObject ("getUser", 3);**

**System.out.println(user.getName());**

**sqlMap.insert("insertUser",u);**

**sqlMap.update("updateUser",u2);**

**sqlMap.delete("deleteUser", 3);**

## 6.dynamic

**dynamic可以去除第一个prepend="and"中的字符（这里为and），从而可以帮助你实现一些很实用的功能。具体情况如下：**

**1.使用dynamic**

**1.1 xml  
select \* from Person表  
        <dynamic prepend="where">   
                <isNotNull property="name" prepend="and">  
                    name=#name#  
                </isNotNull>  
                <isNotNull property="sex" prepend="and">  
                    sex=#sex#  
                </isNotNull>                 
        </dynamic>**

**1.2 结果**

**当name、sex都非null时打出如下的sql语句:**

**select    Person表 where (and) name= ? , and sex= ?**

**显然name前的and被自动去除了，很方便吧。**

# 九.hibernate

## 1.建立配置文件与数据库进行连接

<?xml version=*'1.0'* encoding=*'UTF-8'*?>

<!DOCTYPE hibernate-configuration PUBLIC

"-//Hibernate/Hibernate Configuration DTD 3.0//EN"

"http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">

<!-- Generated by MyEclipse Hibernate Tools. -->

<hibernate-configuration>

<session-factory>

<property name=*"dialect"*>

org.hibernate.dialect.MySQLDialect

</property>

<property name=*"connection.url"*>

jdbc:mysql://localhost:3306/hiberate1

</property>

<property name=*"connection.username"*>root</property>

<property name=*"connection.password"*>147094</property>

<property name=*"connection.driver\_class"*>

com.mysql.jdbc.Driver

</property>

<property name=*"show\_sql"*>true</property>

<!-- <property name="format\_sql">true</property>显示的sql语句格式化- -->

<property name=*"myeclipse.connection.profile"*>mysql</property>

<mapping resource=*"com/hlj/domain/Student.hbm.xml"* />

<mapping resource=*"com/hlj/domain/Course.hbm.xml"* />

<mapping resource=*"com/hlj/domain/Stucourse.hbm.xml"* />

</session-factory>

</hibernate-configuration>

## 2.编写java文件与数据库对应

**public** **class** Student **implements** java.io.Serializable {

// Fields

**private** Integer sid;

**private** String sname;

**private** String ssex;

**private** String adept;

**private** Integer sage;

**private** String saddress;

//通过这个集合表明一个学生可以选择多门课程，反过来一门课程可以被多次选择

**private** Set stucourses = **new** HashSet(0);

/\*\* default constructor \*/

**public** Student() {

}

/\*\* minimal constructor \*/

**public** Student(String sname, String ssex, String adept, Integer sage,

String saddress) {

**this**.sname = sname;

**this**.ssex = ssex;

**this**.adept = adept;

**this**.sage = sage;

**this**.saddress = saddress;

}

**public** Set getStucourses() {

**return** **this**.stucourses;

}

**public** **void** setStucourses(Set stucourses) {

**this**.stucourses = stucourses;

}

}

## 3.编写映射文件

<hibernate-mapping>

<class name=*"com.hlj.domain.Student"* lazy=*"false"* table=*"student"* catalog=*"hiberate1"*>

<id name=*"sid"* type=*"java.lang.Integer"* >

<column name=*"sid"* />

<generator class=*"increment"* />

</id>

<property name=*"sname"* type=*"java.lang.String"*>

<column name=*"sname"* length=*"20"* not-null=*"true"* />

</property>

<property name=*"ssex"* type=*"java.lang.String"*>

<column name=*"ssex"* length=*"2"* not-null=*"true"* />

</property>

<property name=*"adept"* type=*"java.lang.String"*>

<column name=*"adept"* length=*"20"* not-null=*"true"* />

</property>

<property name=*"sage"* type=*"java.lang.Integer"*>

<column name=*"sage"* not-null=*"true"* />

</property>

<property name=*"saddress"* type=*"java.lang.String"*>

<column name=*"saddress"* length=*"50"* not-null=*"true"* />

</property>

<set name=*"stucourses"* inverse=*"true"*>

<key>

<column name=*"sid"* not-null=*"true"* />

</key>

<one-to-many class=*"com.hlj.domain.Stucourse"* />

</set>

</class>

## 4.编写映射文件多对一

<?xml version=*"1.0"* encoding=*"utf-8"*?>

<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD 3.0//EN"

"http://www.hibernate.org/dtd/hibernate-mapping-3.0.dtd">

<!--

Mapping file autogenerated by MyEclipse Persistence Tools

-->

<hibernate-mapping>

<class name=*"com.hlj.domain.Stucourse"* table=*"stucourse"* catalog=*"hiberate1"*>

<id name=*"stuCourseId"* type=*"java.lang.Integer"*>

<column name=*"stuCourseId"* />

<generator class=*"increment"* />

</id>

<many-to-one name=*"student"* class=*"com.hlj.domain.Student"* fetch=*"select"*>

<column name=*"sid"* not-null=*"true"* />

</many-to-one>

<many-to-one name=*"course"* class=*"com.hlj.domain.Course"* fetch=*"select"*>

<column name=*"cid"* not-null=*"true"* />

</many-to-one>

<property name=*"grade"* type=*"java.lang.Integer"*>

<column name=*"grade"* not-null=*"true"* />

</property>

</class>

</hibernate-mapping>

## 3.增删改查

增 Employee u=**new** Employee();

u.setEmail("1318830916@qq.com");

u.setHiredate(**new** Date());

u.setName("zhangyujin");

session.save(u);//添加该对象到数据库 //等于==inset

删

Employee employee = (Employee)session.load(Employee.**class**, 3);

session.delete(employee);

改

Employee employee =(Employee)session.load(Employee.**class**, 4);

//下面这个就是相当于更新了

employee.setEmail("13151515@qq.com");

ts.commit();

查

Query query=session.createQuery("select sname,adept from Student");

# 十 Struts

# 八、Struts2

## 1.web.xml中配置

**<filter>**

**<filter-name>struts2</filter-name>**

**<filter-class>org.apache.struts2.dispatcher.ng.filter.StrutsPrepareAndExecuteFilter</filter-class>**

**</filter>**

**<filter-mapping>**

**<filter-name>struts2</filter-name>**

**<url-pattern>/\*</url-pattern>**

**</filter-mapping>**

## 2.配置Struts配置文件config.xml

**<?xml version="1.0" encoding="UTF-8" ?>**

**<!DOCTYPE struts PUBLIC "-//Apache Software Foundation//DTD Struts Configuration 2.1//EN" "http://struts.apache.org/dtds/struts-2.1.dtd">**

**<struts>**

**下面这个主要是提供继承使用，下面这个是包的命名空间，**

**<package name="itcast" namespace="/test" extends="struts-default">**

**<action name="helloworld" class="cn.itcast.action.HelloWorldAction" method="execute" >**

**<!-- 这个相当于是forward -->**

**<result name="success">/WEB-INF/page/hello.jsp</result>**

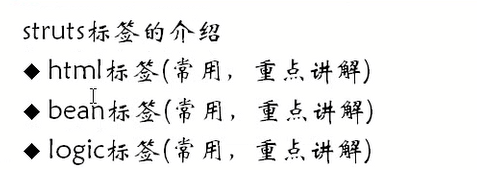
**</action>**

**</package>**

**</struts>**

3.测试

## 3.Struts2标签



1. **<c:if test="${a=='你好啊 if'}">**
2. **<c:choose> 相当于是switch**

**<c:when 相当于是 case**

**<c:otherwise>相当于default**

1. **<c:forEach items="${alluser}"var="u">**

**${u.username}**

**${u.password} <br/>**

**</c:forEach>**

1. **<c:forTokens items="${f }"delims=";"var="tem">**
2. **<c:redirect url="http://www.baidu.com"> </c:redirect>**

**6.<c:import url="c\_if.jsp">**

**6.<c:out value="${hello} " default="没有值"></c:out>**

**7.<c:set var="abc" value="中国,北京" scope="request"></c:set>**

**8.<html:button property="mybutton" value="提交"></html:button>**

**9.<html:image src="images/a.jpg"> </html:image>**

**10 .<bean:write name="a"/>**

**11. 遍历list**

**<s:iterator value="label" id="id">   
    <s:property value="#id.attrName" />   
</s:iterator>**

## 遍历map

**<s:iterator value="map" id="id" status="st">   
     key : <s:property value='key'/>   
     value:<s:property vlaue='value'/>   
</s:iterator>**

## 14表单

**<s:textfield label="用户名" name="uname" </s:textfield>**

**<s:textarea name="rmake" label="备注" ></s:textarea>**

**<s:password label="密码" name="upass"></s:password>**

**<s:file name="file" label="上传文件"></s:file>**

**<s:hidden name="id" value="1"></s:hidden>**

**<s:select list="#{'1':'博士','2':'硕士'}" name="edu" label**

# 十、8种基本数据类型

他们分别是byte、short、int、long、float、double、char、boolean