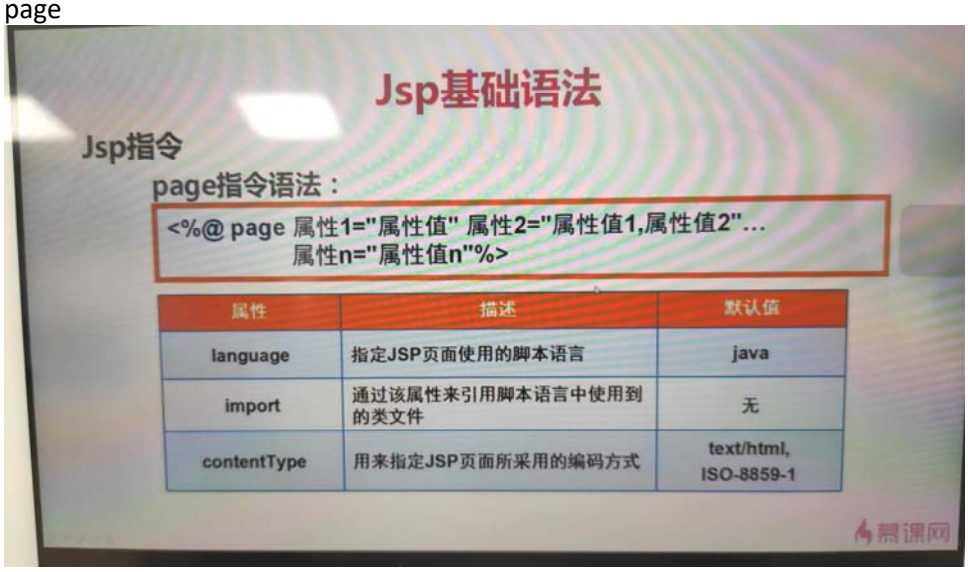
jsp笔记

2019年4月1日　　20:47

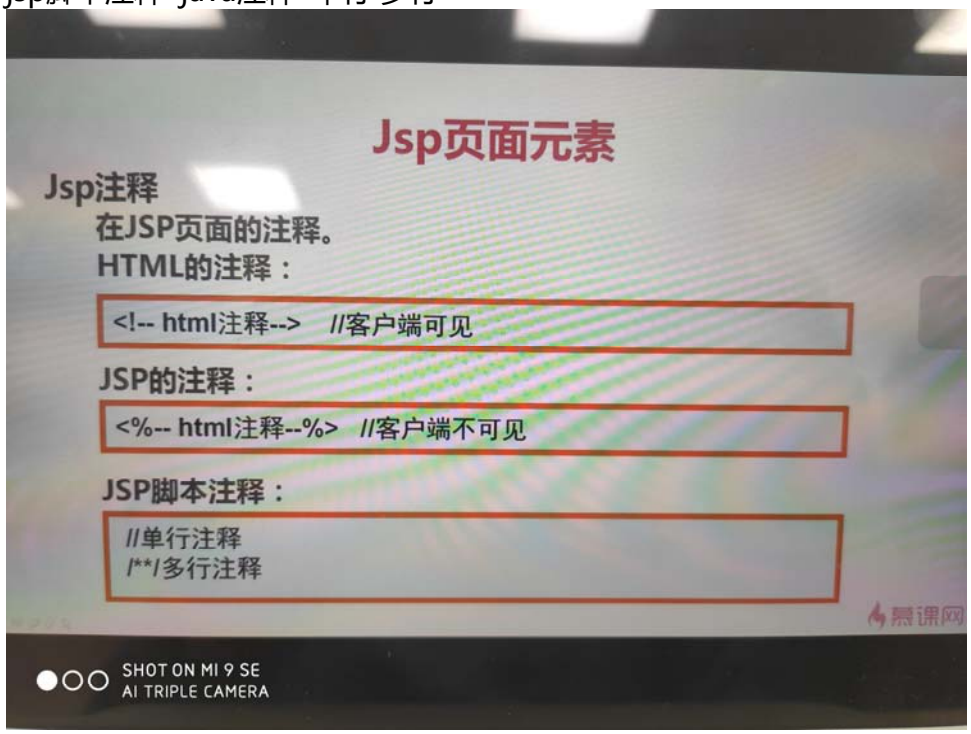
jsp指令

page



注释 3种

- html注释 客户端可见
- jsp注释 客户端不可见
- jsp脚本注释 java注释 单行 多行



jsp脚本

在jsp中加入java代码
<%java代码%>

jsp的声明 在jsp中定义变量或方法 用java脚本格式声明

<%！java声明%>

jsp表达式

<%=表达式%>
注意：表达式不以分号结束

jsp生命周期

两张ipad图片带插入

- **Jsp内置对象**

web容器创建的一组对象，不需要使用new关键字就可以使用的内置对象
如 out对象 out.print();
jsp9大内置对象
常用的有5个：out, request, response, session, application
不常用的4个：page,pageContext,exception,config

内置对象详细介绍

- Web程序的请求响应模式
  - 用户发送请求(request)
  - 服务器给用户响应(response)
  - **out对象**：缓冲区：内存的一块区域用来保存临数据。是JspWriter类的实例，向客户端输出内容常用的对象。常用方法如下：
    1.void println()向客户端打印字符串 ----输出HTML标签
    2.void clear()清除缓冲区的内容，如果在flush之后调用会抛出异常 ---不能在flush之后调用
    3.void clearBuffer():清除缓冲区的内容，如果在flush之后调不用会抛出异常
    4.void flush():将缓冲区内容输出到客户端
    5.int getBufferSize() 返回缓冲区以字节数的大小，如不设缓冲区则为0
    6.int getRemaining() 返回缓冲区还剩余多少可用
    7.boolean isAutoFlush() 返回缓冲区满时，是自动清空还是抛出异常
    8.void close() 关闭输出流
  - **get与post**
    <form name="regForm" action="动作" method="**提交方式**">
    </form>
    action表示该表单提交给哪个动作（可以是jsp文件）去处理，name 表示该表单所起的名

字。method主要有2种方式：get和post

表单的2种提交方式：

1.get:以明文的方式通过URL提交数据，数据在URL中可以看到。提交的数据最多不超过2kb.安全性较低但效率较post高。适合提交数据量不大，安全性不高的数据。比如：搜索、查询等功能。

2.post:将用户提交的信息封装在HTML HEADER内。适合提交数据量大，安全性高的用户信息。比如：注册、修改、上传等功能。

- **request对象**

客户端的请求信息被封装在request对象中，通过它才能了解到客户的需求，然后做出响应。它是HttpServletRequest类的实例。request对象具有请求域，即完成客户端的请求之前，该对象一直有效。常用方法如下：

1.String getParameter(String name)返回name指定参数的参数值--获得单个值

2.String[] getParameterValues(String name)返回包含参数name的所有值的数组 ---获得集合值

3.void setAttribute(String,Object) 存储此请求中的属性

4.object getAttribute(String name)返回指定属性的属性值 ---输入的是属性所对应的键，得到的是属性所对应的值

5.String getContentType()得到请求体的MIME类型

6.Sting getProtocol()返回请求用的协议类型及版本号

7.String getServerName()返回接受请求的服务器主机名

8.int getServerPort()返回服务器接受此请求所用的端口号

9.String getCharacterEncoding()返回字符编码方式

10.void setCharacterEncoding()设置请求的字符编码方式

11.int getContentLength()返回请求体的长度（以字节数）

12.String getRemoteAddr()返回发送此请求的客户端IP地址

13.String getRealPath(String path)返回一虚拟路径的真实路径

14.String request.getContextPath()返回上下文路径

处理中文乱码，最简单的方法是在接收的对象之前加上

```
<%
    Request.setCharacterEncoding("utf-8");  //这里的utf-8必须和对象所在页面的编码一样，无法解决url传参的时候出现的中文乱码问题，解决url传中文参需要配置tomcat的config.xml文件
%>
```

- **response对象**

包含了响应客户请求的有关信息，但在JSP中很少直接用到它。它是HttpServletResponse类的实例。response对象具有页面作用域，即访问一个页面时，该页面内的response对象只能对这次访问有效，其它页面的response对象对当前页面无效。常用方法如下：

1.String getCharacterEncoding()返回响应应用的何种字符编码

2.void setContentType(String type)设置响应的MIME类型

3.PrintWriter getWriter()返回可以向客户端输出字符的一个对象（注意比较PrintWriter与内置对象out的区别）PrintWriter对象的输出流总是提前于out对象，解决方法，在out.println()之后使用out.flush()

4.sendRedirect(java.lang,String location)重新定向客户端的请求

- **请求转发与请求重定向的区别**

  请求重定向：客户端行为，response.sendRedirect()，从本质上讲等同于两次请求，前一次的请求对象不会保存，地址栏的URL地址会改变。

  请求转发：服务器行为，request.getRequestDispatcher().forward(req,resp);是一次请求，转发后请求对象会保存，地址栏的URL地址不会改变。

- **Session对象**：

  一次连接相当于一次session，同意服务器服务不同客户，每一个客户有一个session，直到用户关闭所有连接或者超时session才会断开连接。

JSP内置对象

Session的生命周期

活动：

- 某次会话当中通过超链接打开的新页面属于同一次会话。
- 只要当前会话页面没有全部关闭，重新打开新的浏览器窗口访问同一项目资源时属于同一次会话。
- 除非本次会话的所有页面都关闭后再重新访问某个Jsp或者Servlet将会创建新的会话。

注意事项：注意原有会话还存在，只是这个旧的SessionId仍然存在于服务端，只不过再也没有客户端会携带它然后交予服务端校验。

JSP内置对象

Session的生命周期

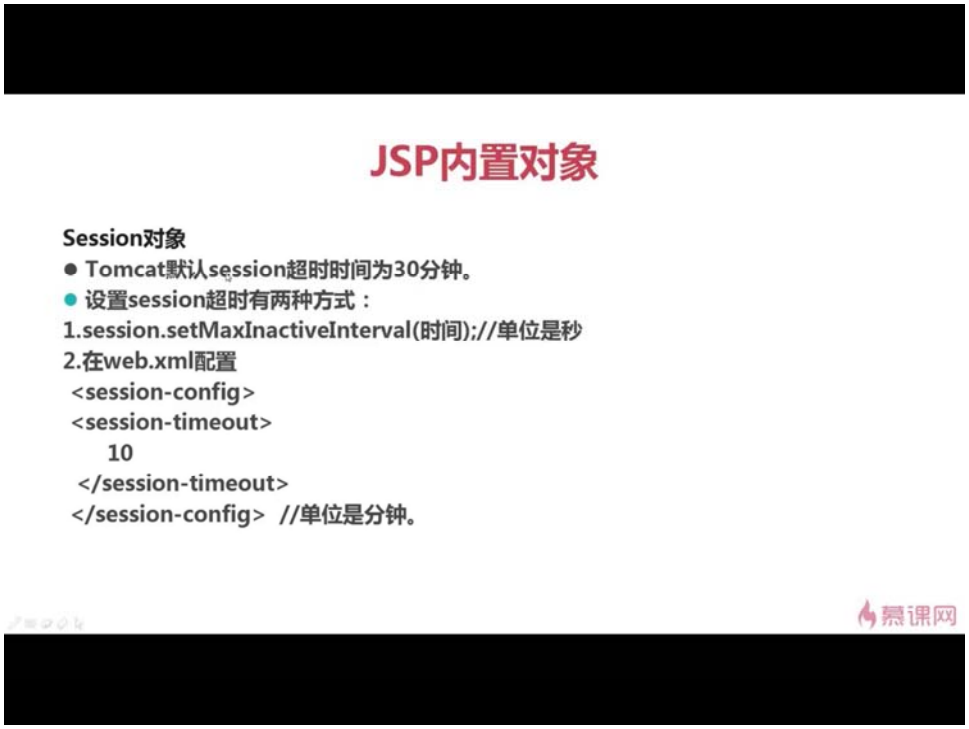
销毁：

Session的销毁只有三种方式：

1、调用了session.invalidate()方法

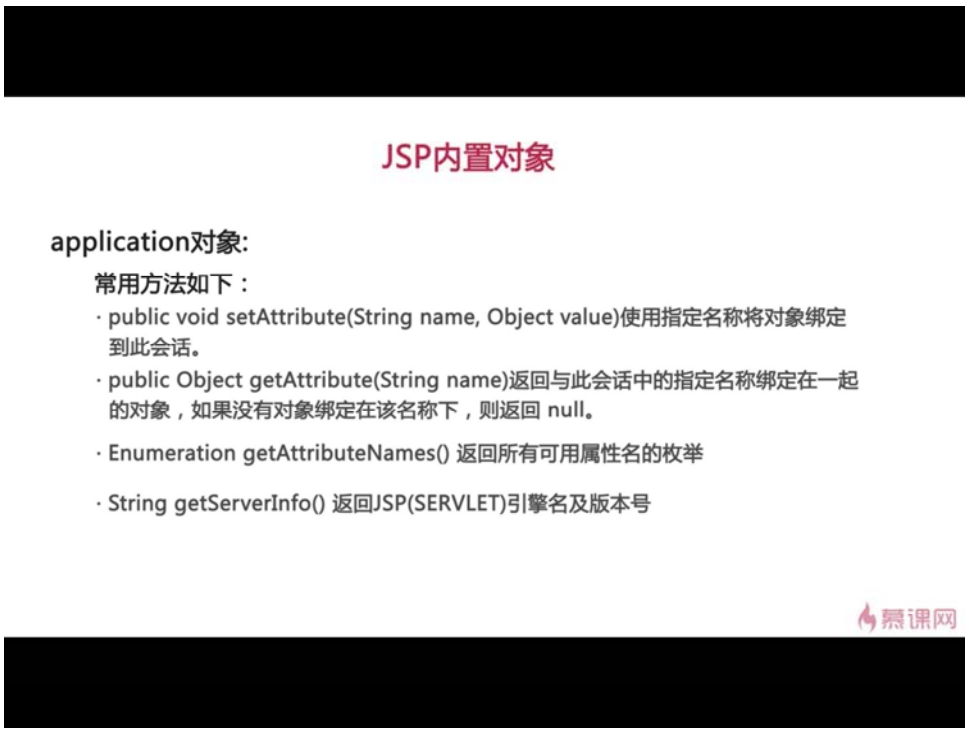
2、Session过期（超时）

3、服务器重新启动



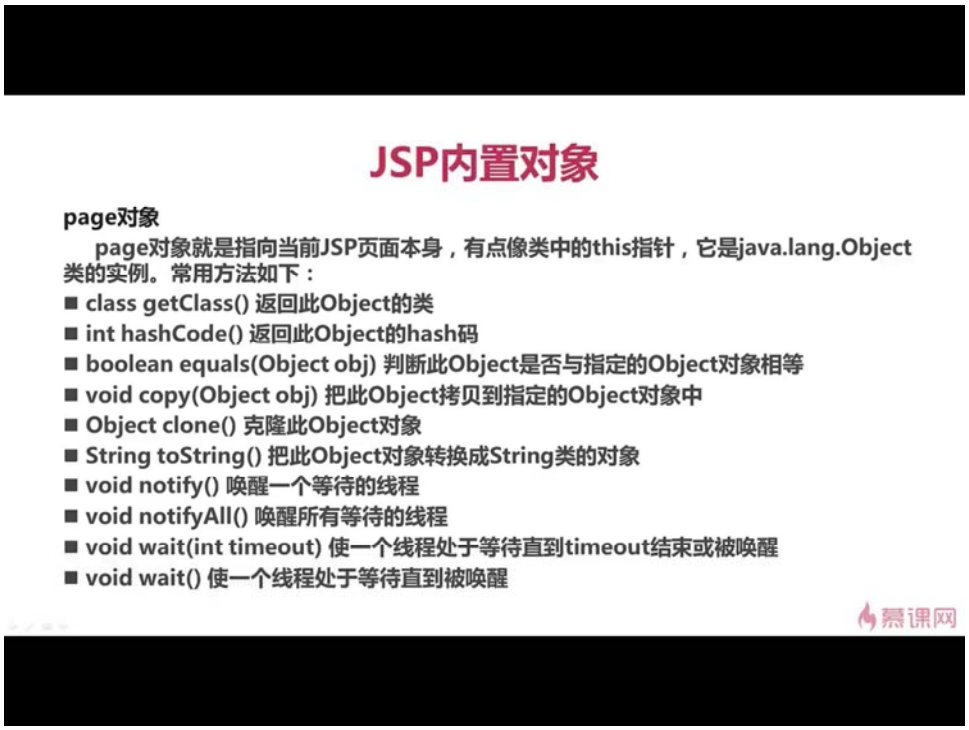
- **application对象**：--类似于Java中的static class
  1.application对象实现了用户数据的共享，可存放全局变量
  2.application对象开始于服务器的启动，种植于服务器的关闭
  3.在用户的前后连接或不同用户之间的连接中，可以对application对象的同一属性进行操作。
  4.在任何地方对application对象属性的操作，都将影响到其他用户对此的访问。
  5.服务器的启动和关闭决定了application对象的生命。
  6.是ServletContext类的实例

  常用方法：

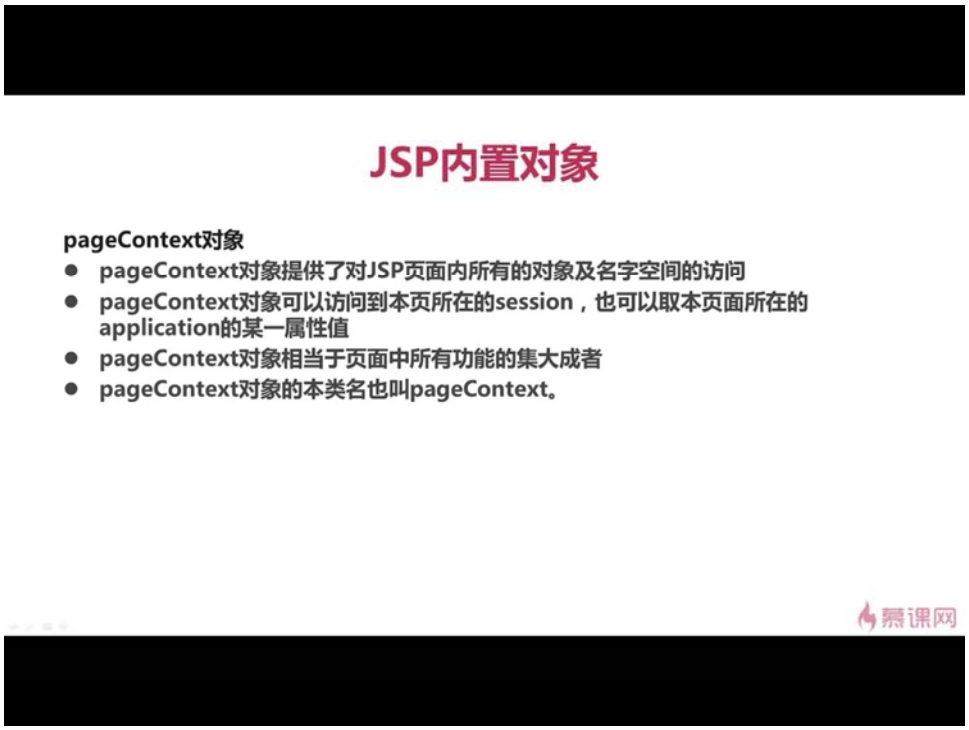


- **page对象**：

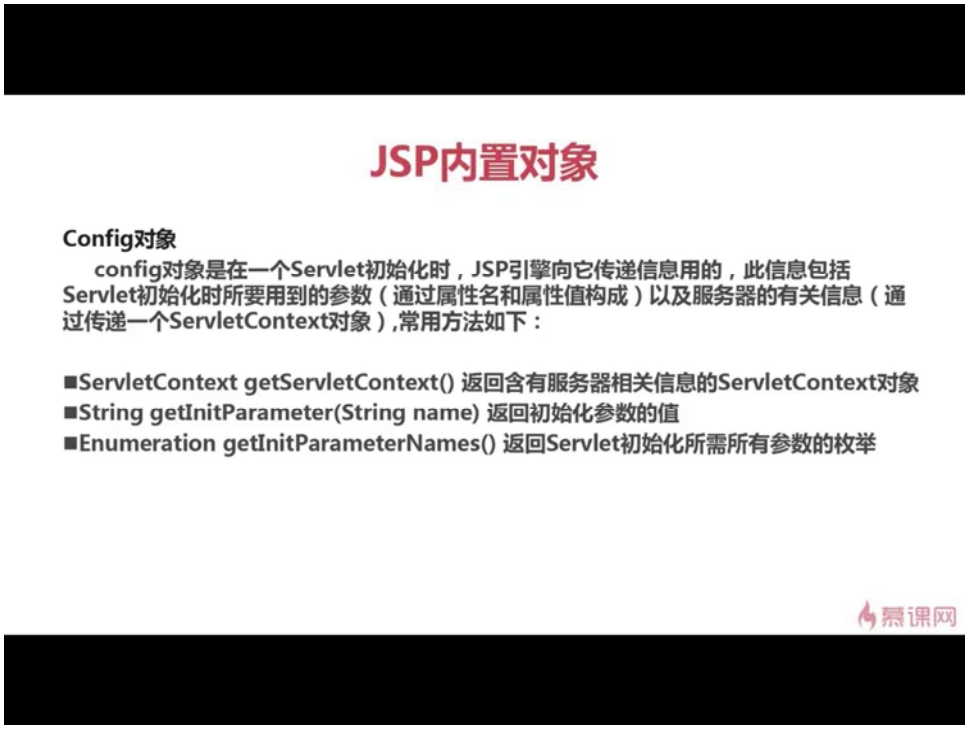
指向当前JSP页面本身，有点像Java中类的this



- **pageContext对象**

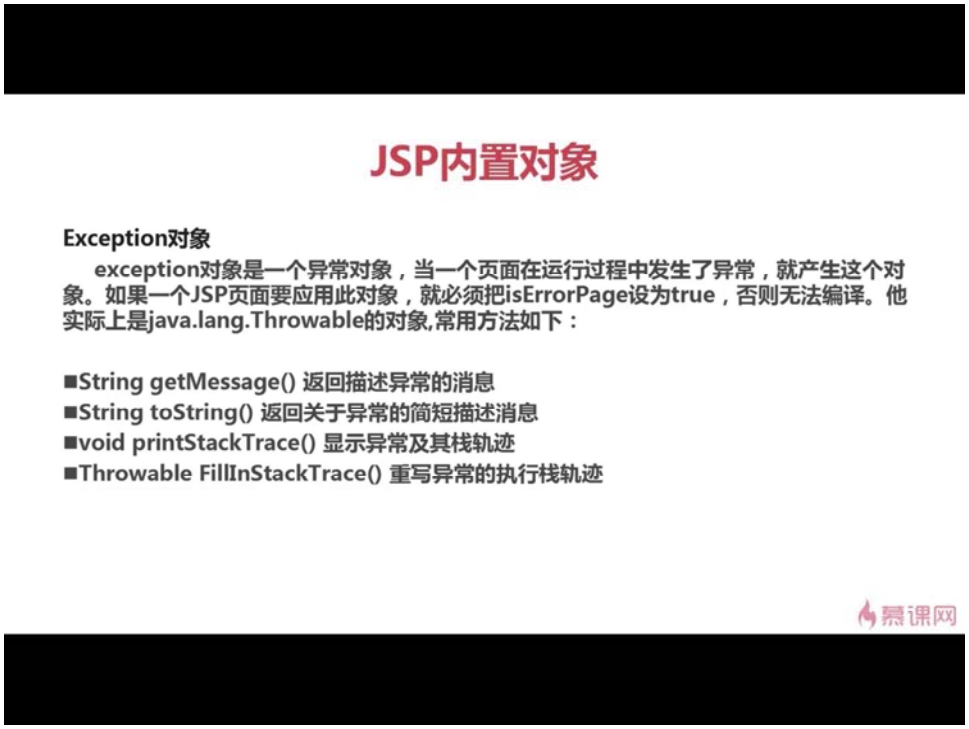


- **Config对象**



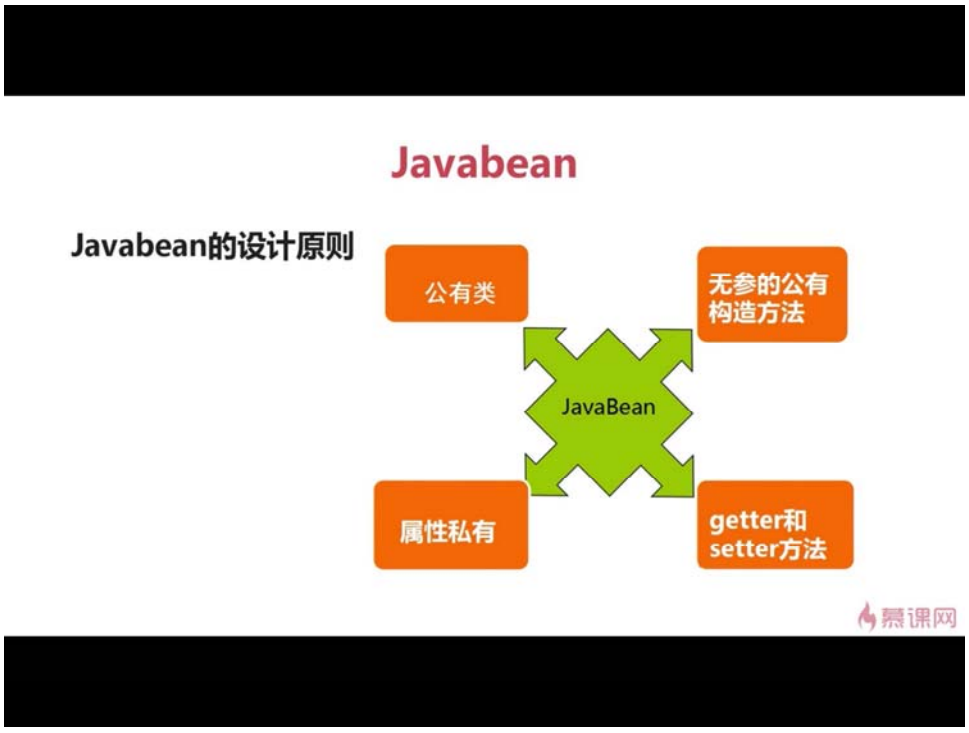
- **exception对象**

  在使用exception对象的 page 中要加入属性 isErrorPage='true'，表示该页面可使用exception对象
  在抛出异常的页面的中的page中加入属性 errorPage="exception.jsp" ----------(这里面的exception.jsp即为自己写的处理异常的jsp页面)



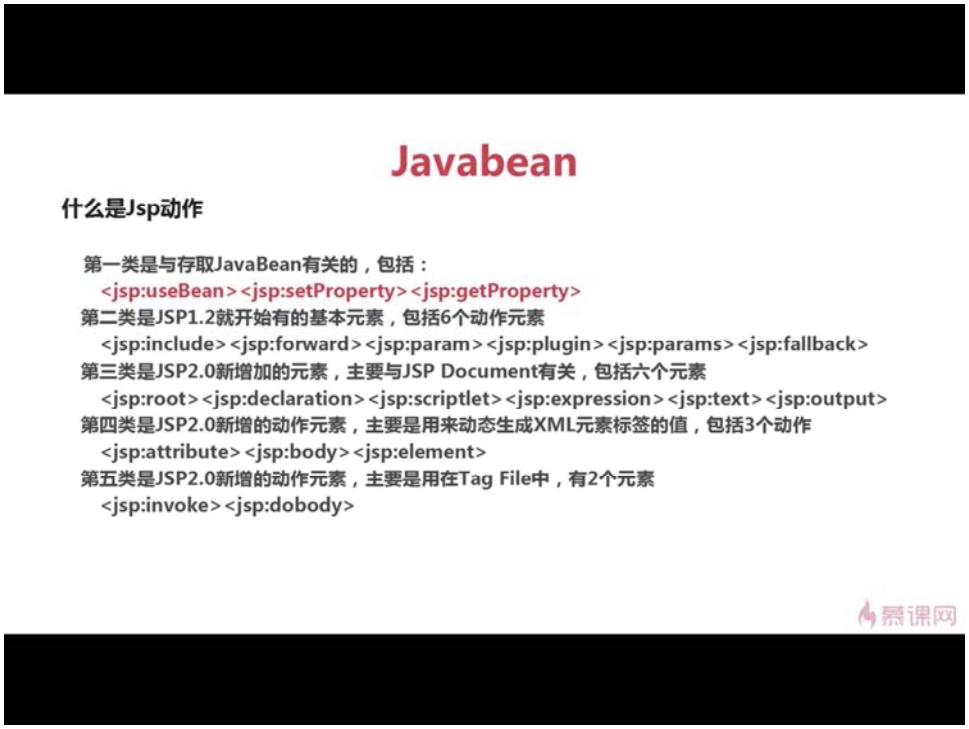
➢ *Javabean*

- Javabean简介：Javabeans就是符合某种特定的规范的Java类。使用Javabeans的好处是解决代码重复编写，减少代码冗余，功能区分明确，提高了代码的维护性。
- 设计原则



- 什么是JSP动作元素

  动作元素为请求处理阶段提供信息。动作元素遵循XML元素的语法，有一个包含元素名的开始标签，可以有属性、可选的内容、与开始标签匹配的结束标签。



第一类：

如何在JSP页面中使用Javabeans

1.像使用普通Java类一样，创建Javabean实例。--需要在page中import相应的包和类

2.在JSP页面中使用jsp动作标签使用Javabean（3种动作标签）：useBranson动作、setProperty动作、getProperty动作

<jsp:userBeans>

作用：在jsp页面中实例化或者在指定范围内使用javabean:

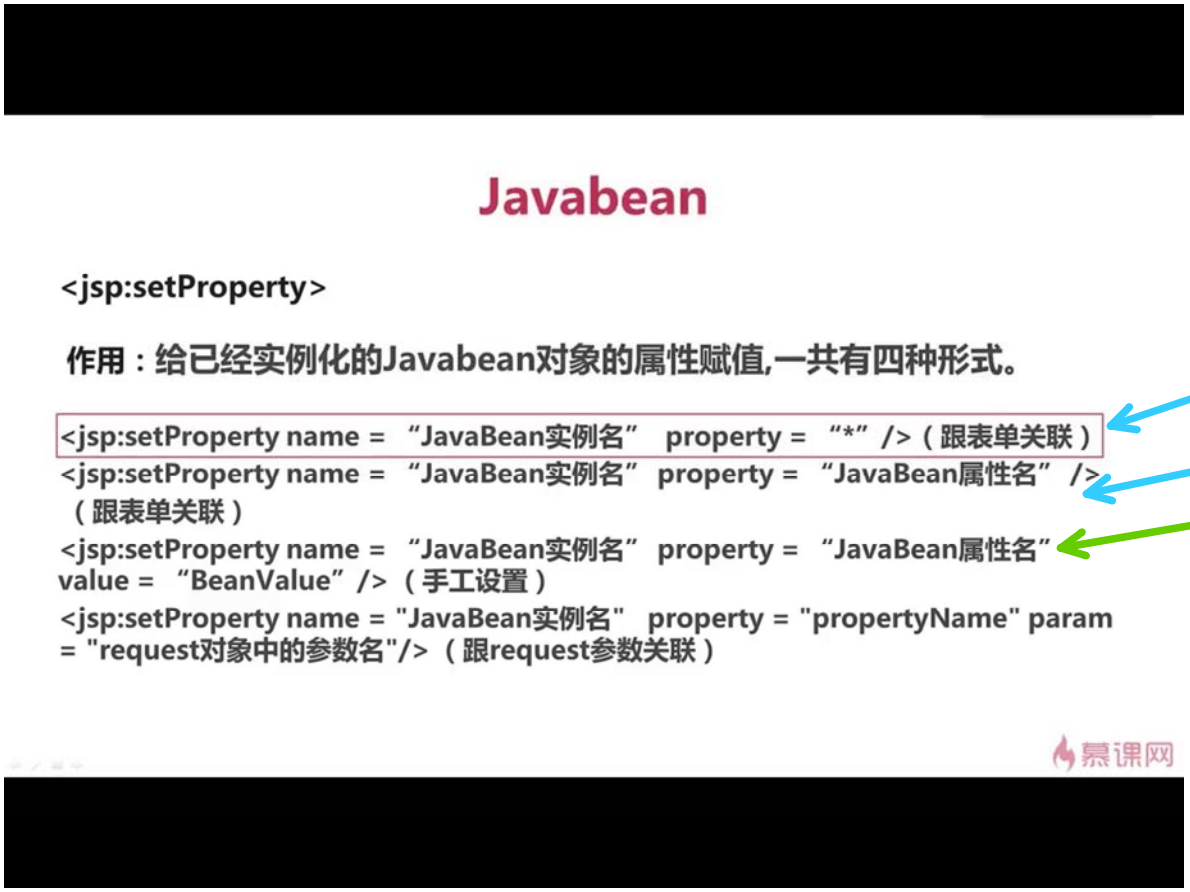
<jsp:useBean id="标识符" class="java类名" scope="作用范围">

标识符：唯一的标识符，也就是 class中的类的实例名，用该实例可以调用class中的方法

Java类名：全名，包括包名和类名

scope可以写成 scope=“page”

<jsp:setProperty>



<jsp:getProperty>

根据表单里面的名字和实例的属性名进行匹配，自动匹配所有的属性

根据表单里面的名字和实例的属性名进行匹配，自动匹配部分的属性

跟表单没有关系，自己给属性设置值，value是自己认为设置的

Javabean的四个作用域范围

说明：使用useBeans的scope属性可以用来指定javabean的作用范围。

- page //仅在当前页面有效
- request //可以通过HttpRequest.getAttribute()方法取得JavaBean对象。
- session //可以通过HttpSession.getAttribute()方法取得JavaBean对象。
- applicatoin //可以通过application.getAttribute()方法取得Javabean对象

➢ Model 1简介

Model1简介

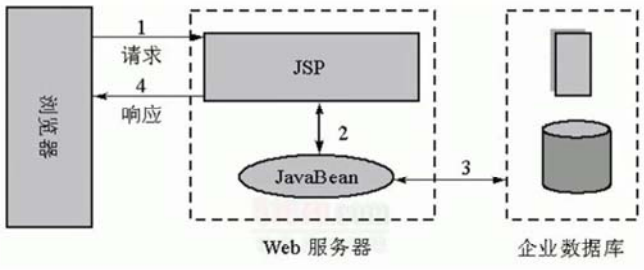
- Model 1模型出现前，整个Web应用的情况：几乎全部由JSP页面组成，JSP页面接收处理客户端请求，对请求处理后直接做出响应。

Model1简介

- Javabean的出现可以使jsp页面中使用Javabean封装的数据或者调用Javabean的业务逻辑代码,这样大大提升了程序的可维护性。

Javabean

Model1简介



➢ Http协议的无状态性

Jsp状态管理

http协议的无状态性

无状态是指，当浏览器发送请求给服务器的时候，服务器响应客户端请求。

但是当同一个浏览器再次发送请求给服务器的时候，服务器并不知道它就是刚才那个浏览器。

简单地说，就是服务器不会去记得你，所以就是无状态协议。

+Cookies

Jsp状态管理

什么是Cookie？

Cookie：中文名称为“小甜饼”，是Web服务器保存在客户端的一系列文本信息。

典型应用一：判定注册用户是否已经登录网站。

典型应用二：“购物车”的处理。

Jsp状态管理

Cookie的作用

- 对特定对象的追踪
- 保存用户网页浏览记录与习惯
- 简化登录

安全风险：容易泄露用户信息

.

Jsp状态管理

Jsp中创建与使用Cookie

创建Cookie对象

```
Cookie newCookie = new Cookie(String key,Object value);
```

写入Cookie对象

```
response.addCookie(newCookie);
```

读取Cookie对象

```
Cookie[] cookies = request.getCookies();
```

Jsp状态管理

Jsp中创建与使用Cookie

●常用方法

方法名称	说 明
void setMaxAge(int expiry)	设置cookie的有效期，以秒为单位
void setValue(String value)	在cookie创建后，对cookie进行赋值
String getName()	获取cookie的名称
String getValue()	获取cookie的值
int getMaxAge()	获取cookie的有效时间，以秒为单位

Session与Cookie

相同点：都是保存用户状态信息的一种机制；都会过期也就是都有一个生存周期

不同点：

Jsp状态管理

Session与Cookie对比

include指令--在一个JSP页面中显示另一个JSP页面，要显示的JSP页面写在URL处。

指令与动作

include指令

语法：

```
<%@ include  file="URL"%>
```

Include 动作

指令与动作

include动作

语法：

```
<jsp:include page="URL" flush="true|false"/>
```

page	要包含的页面
flush	被包含的页面是否从缓冲区读取

指令与动作

include指令与include动作比较

	include指令	jsp:include动作
语法格式	<%@ include file= ".." %>	<jsp:include page= ".." >
发生作用的时间	页面转换期间	请求期间
包含的内容	文件的实际内容	页面的输出
转换成的Servlet	主页面和包含页面转换为一个Servlet	主页面和包含页面转换为独立的Servlet
编译时间	较慢——资源必须被解析	较快
执行时间	稍快	较慢——每次资源必须被解析

forward动作：

指令与动作

forward动作

语法：
<jsp:forward page="URL" />

等同于：
request.getRequestDispatcher("/url").forward(request,response);

param动作：

指令与动作

param动作

语法：
<jsp:param name="参数名" value="参数值">

常常与 <jsp:forward >一起使用，作为其的子标签。