　　Servlet三个要素:   
　　1.必须继承自HttpServlet   
　　2.必须实现doGet()或者doPost()   
　　3.必须在web.xml中配置Servlet   
　　<servlet>   
　　<servlet-name> </servlet-name>   
　　<servlet-class> </servlet-class>   
　　</servlet>   
　　<servlet-mapping>   
　　<servlet-name> </servlet-name>   
　　<url-pattern> </url-pattern>   
　　</servelt-mapping>   
　　  
　　HttpServeltRrequest:请求对象   
　　getParameter():获得表单元素的值   
　　getAttribute():获得request范围中的属性值   
　　setAttribute():设置reqeust范围中的属性值   
　　setCharacterEncoding():设置字符编码   
　　  
　　HttpSerletResponse:相应对象   
　　sendRedirect():外部跳转   
　　getWriter():获得输出流对象   
　　setContentType("text/html; charset=utf-8"):设置相应内容格式和编码   
　　  
　　四种会话跟踪方式:   
　　1.Session   
　　HttpSession session = request.getSession();   
　　session.setAttribute("name", "zhangsan");   
　　session.setAttribute("pwd", "aaa");   
　　String name = (String) session.getAttribute("name");   
　　  
　　2.cookie:   
　　//创建Cookie   
　　Cookie cookie = new Cookie("name", "zhangsan");   
　　//设置Cookie的超时时间   
　　cookie.setMaxAge(24 \* 60 \* 60 \*60);   
　　//把Cookie发送到客户端   
　　response.addCookie(cookie);   
　　  
　　//得到客户端发送的Cookie   
　　Cookie [] cookies = request.getCookies();   
　　for(int i=0; i <cookies.length; i++) {   
　　 Cookie temp = cookies[i];   
　　 String key = temp.getName();   
　　 String value = temp.getValue();   
　　}   
　　  
　　3.隐藏表单域   
　　<input type="hidden" name="name" value="zhangsan" />   
　　request.getParameter("name");   
　　  
　　4.Url重写   
　　问号传参   
　　LoginServlet?username=zhangsan&pwd=123   
　　String name = request.getParameter("username");   
　　String pwd =request.getPareameter("pwd");   
　　  
　　内部跳转:   
　　LoginServlet   
　　request.getRequestDispatcher("index.jsp").forward(request, resposne);   
　　外部跳转:   
　　response.sendRedirect("index.jsp");   
　　内部跳转是一次请求和一次响应   
　　外部跳转是两次请求和两次响应   
　　  
　　ServletContext:Servlet上下文对象   
　　它是一个公共区域,可以被所有的客户端共享   
　　setAttribute():向公共区域里放入数据   
　　getAttribute():从公共区域里取数据   
　　  
　　二:   
　　三:三个标准范围:request, session, ServletContext   
　　 共同点:都有setAttribute(), getAttribute()   
　　 区别:范围不同,request < session < servletContext   
　　四:四种会话跟踪方式   
　　五:服务器上的五大对象   
　　 request, response, servlet, session, servletContext   
　　   
　　Jsp:Java Server Page   
　　页面构成:7种元素   
　　1.静态内容:html   
　　2.指令:page, include, taglib:   
　　<%@ 指令名 属性1="属性值1" 属性2="属性值2" %>   
　　3.表达式: <%=表达式 %>   
　　4.Scriptlet <% Java代码 %>   
　　5.声明: <%! %>:变量和方法   
　　6.动作: <jsp:动作名 属性="属性值"> </jsp:动作名>   
　　7.注释:   
　　客户端看不到的: <%-- --%>   
　　客户端可以看到的: <!-- -->   
　　  
　　  
　　Jsp的执行过程:   
　　1.转译:Jsp--->Servlet   
　　2.编译:Servlet---->.class   
　　3.执行:.class   
　　第一次访问jsp的时候响应速度较慢,后面请求时响应速度快   
　　  
　　脚本:   
　　表达式: <%= %>   
　　Scriptlet: <% %>   
　　声明: <%! %>   
　　  
　　指令:   
　　page:language, import, errorPage, isErrorpage   
　　include:file   
　　taglib:uri:指定标签库描述符的路径 prefix:指定标签的前缀   
　　  
　　隐式对象:   
　　分类:   
　　1.输入和输出对象:request(HttpServletRequest),   
　　 response(HttpServletResponse),   
　　 out(JspWriter), servlet中的out是PrintWriter   
　　2.作用域通信对象:pageContext, request,   
　　 session(HttpSession),   
　　 application(ServletContext)   
　　3.Servlet对象:page(this), config   
　　4.错误对象:exception   
　　   
　　JavaBean:   
　　一个标准的JavaBean有三个条件   
　　1.共有的类   
　　2.具有不带参数的公共的构造方法   
　　3.具有set()和get()方法   
　　4.私有属性   
　　  
　　Jsp中的标准动作:   
　　1.useBean:创建JavaBean的一个实例   
　　<jsp:useBean id="stu" class="com.westaccp.test.Student" scope="page/session/application/request" />   
　　2.setProperty:给JavaBean的属性赋值   
　　<jsp:setProperty name="stu" property="stuName" value="zhangsan" />   
　　<jsp:setProperty name="stu" property="stuName" param="txtName" />   
　　value和param不能同时使用   
　　偷懒的方法: <jsp:setProperty name="stu" property="\*" />   
　　这个时候需要注意的是,表单元素的名字必须和JavaBean的属性值   
　　一模一样   
　　3.getProperty:获得JvaBean的属性值   
　　<jsp:getProperty name="stu" property="stuName" />   
　　4.forward:内部跳转,相当于request.getRequestDispatcher().forward(request, response);   
　　<jsp:forward page="index.jsp" />   
　　5.include:包含   
　　<jsp:include page="header.jsp" flush="true" />   
　　  
　　表达式语言:   
　　EL: Expression Language   
　　语法格式: ${表达式 }   
　　表示式 = 运算符 + 操作数   
　　运算符:跟Java比较,多了一个empty, 少了一个赋值运算符   
　　${empty ""} : true   
　　${empty null} :true   
　　操作数:   
　　-->常量:布尔型(true/false), 整型, 浮点型, 字符串(可以用’’, 还可以用""), Null   
　　-->变量:   
　　 1.指的是放在四个标准范围里的属性(page, request, session, application)   
　　 2.在编准范围内的搜索顺序:page-->request--->session--->application   
　　 3.怎么取得变量值:点运算符., 还以用[]   
　　 <%   
　　 request.setAttribute("name", "lisi");   
　　 %>   
　　 ${requestScope.name}   
　　 或者   
　　 ${requestScope["name"]}   
　　-->隐式对象   
　　 1.pageContext:通过它可以访问request, session, servletContext   
　　 2.跟范围由关的:pageScope, requestScope, sessionScope, applicationScope   
　　 3.跟输入有关的:param, paramValues   
　　 4.其他的:header, cookie, headervalues,   
　　  
　　EL表达式适用的场合:   
　　1.可以在静态文本中使用   
　　2.与自定义标签结合使用   
　　3.和JavaBean结合使用   
　　<jsp:userBean id="stu" class="com.westaccp.test.Student" scope="session" />   
　　<jsp:setProperty name="stu" property="stuName" value="hello" />   
　　${stu.stuName}   
　　  
　　自定义标签:   
　　1.标签处理程序实现   
　　--->实现:继承自BodyTagSupport或者TagSupport   
　　 一般会重写doStartTag(), doEndTag(), doAfterBody()   
　　--->描述:在标签库描述符文件中描述(.tld)   
　　 <taglib>   
　　 <tlib-version>1.0 </tlib-version>   
　　 <jsp-version>2.0 </jsp-version>   
　　 <short-name>simpletag </short-name>   
　　   
　　 <tag>   
　　 <name>showbody </name>   
　　 <tag-class>com.westaccp.test.ShowBodyTag </tag-class>   
　　 <body-content>empty/jsp </body-content>   
　　 <attribute>   
　　 <name>color </name>   
　　 </attribute>   
　　 </tag>   
　　 </taglib>   
　　--->使用: <%@ taglib uri="WEB-INF/mytag.tld" prefix="my" %>   
　　 <my:showbody />   
　　2.标签文件   
　　--->实现和描述   
　　 在.tag文件中实现   
　　 设置主体内容: <%@ body-content="empty/scriptless" %>   
　　 设置属性: <%@ attribute name="name" required="true" rtexprvalue="true" %>   
　　 有主体内容: <jsp:doBody scope="session" var="theBody" />   
　　 <%   
　　 String body = (String) session.getAttribute("theBody");   
　　 %>   
　　--->使用   
　　 WEB-INF/tags/sayhello.tag   
　　 <%@ taglib tagdir="/WEB-INF/tags/" prefix="you" %>   
　　 <you:sayhello />   
　　   
　　标准标签库:   
　　1.核心标签库   
　　-->通用:   
　　 set: <c:set var="" value="" scope="" />   
　　 out: <c:out value="" />   
　　 remove: <c:remove var="" scope="" />   
　　-->条件:   
　　 if: <c:if test="">..... </c:if>   
　　 choose: <c:choose>   
　　 <c:when test="">... </c:when>   
　　 <c:when test="">... </c:when>   
　　 <c:when test="">... </c:when>   
　　 .....   
　　 <c:otherwise>... </otherwise>   
　　 </c:choose>   
　　-->迭代:   
　　 forEach: <forEach var="" items="" varStatus="" begin="" end="">   
　　 foTokens: <foTodens var="" items="" delim=",;|"> </foTodens>   
　　 Java,C#;SQL|C   
　　2.I18N与格式化标签库   
　　-->setLocale:设置本地区域   
　　-->bundle:设置资源包   
　　-->setBundle:设置资源包   
　　-->message:输出消息   
　　3.SQL标签库   
　　-->setDataSource:设置数据源,用于获得与数据库的连接   
　　-->query:执行查询   
　　-->update:执行增,删,改   
　　-->transaction:事务   
　　-->param:参数   
　　4.XML标签库   
　　  
　　过滤器:   
　　生命周期:   
　　1.实例华:   
　　2.初始化:init()   
　　3.过滤:doFilter()   
　　4.销毁:destroy()   
　　5.不可用   
　　  
　　配置:   
　　<filter>   
　　<filter-name> </filter-name>   
　　<filter-class> </filter-class>   
　　</filter>   
　　<filter-mapping>   
　　<filter-name> </filter-name>   
　　<url-pattern> </url-pattern>   
　　</filter-mapping>   
　　  
　　几个重要的接口:   
　　1.Filter:init(), doFilter(), destroy()   
　　2.FilterChain: doFilter(request, response)   
　　3.FilterConfig:getFilterName(), getInitParameter(),   
　　  
　　过滤器链:--->1--->2--->3--->Servlet 请求   
　　 <----1 <---2 <---3 <--- 响应   
　　   
　　MvC设计模式   
　　1.ModelI:jsp+JavaBean   
　　2.ModelII:jsp+Servlet+JavaBean   
　　 jsp---view   
　　 servlet---control   
　　 javabean---model   
　　  
　　MVC:   
　　M--Model:模型:访问后台数据库   
　　V--view:视图:展示   
　　C--control:控制器:控制程序流程   
　　  
　　ModelII和MVC的关系:   
　　MVC是一种设计模式,ModelII它是MVC的一种具体的实现 