进销存管理系统实训课件

主讲：汤燕

* 课程回顾
* 登录日志
* 操作日志
* 异常日志
* 日志查询

报表制作？

什么是闭包？

什么是立即执行函数？

什么是延迟执行函数？

什么是定时器？

面试非技术类问题汇总

==============================================================

进销存系统在这些业务系统中算是比较复杂的。

这个算是有进销存系统开发经验或者说是小型ERP系统开发经验。

==============================================================

# 日志模块数据库结构

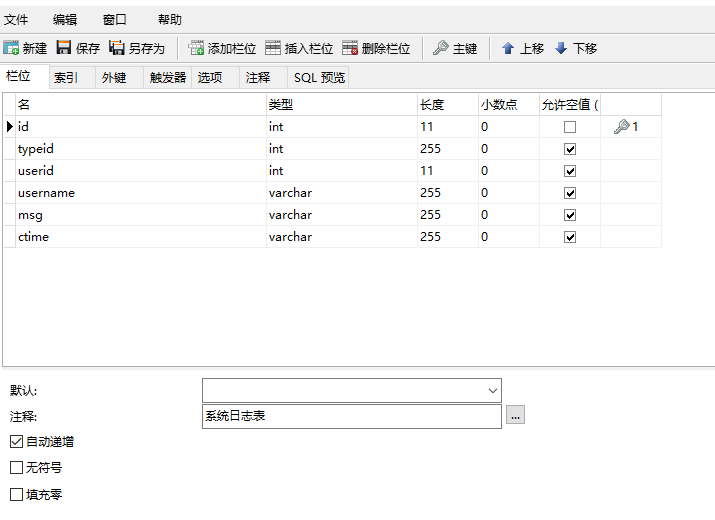
## 日志模块需求

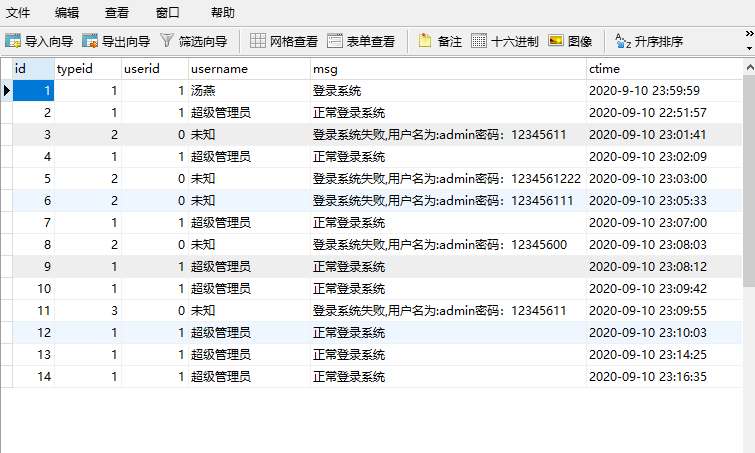
日志不需要表单添加，是后端系统根据用户的操作自动记录的行为。

项目完整编译。有的同学的eclipse没有把自动编译否选上。



## 数据库表结构





## 创建数据库表的脚本

|  |
| --- |
| DROP TABLE IF EXISTS `tblog`;  CREATE TABLE `tblog` (  `id` int(11) NOT NULL AUTO\_INCREMENT COMMENT '系统日志表',  `typeid` int(255) DEFAULT NULL COMMENT '日志类型：1登录日志 2操作日志 3异常日志',  `userid` int(11) DEFAULT NULL COMMENT '用户id',  `username` varchar(255) DEFAULT NULL COMMENT '用户名称',  `msg` varchar(255) DEFAULT NULL COMMENT '消息内容',  `ctime` varchar(255) DEFAULT NULL COMMENT '创建时间',  PRIMARY KEY (`id`)  ) ENGINE=InnoDB AUTO\_INCREMENT=15 DEFAULT CHARSET=utf8; |

# 自己封装一个日志操作类

|  |
| --- |
| package com.rz;  import java.text.SimpleDateFormat;  import java.util.ArrayList;  import java.util.Date;  import java.util.List;  import java.util.Map;  import javax.servlet.http.HttpServletRequest;  public class tblogopt {    public static void addmsg(int typeid,String msg,HttpServletRequest request)  {  Date t=new Date();  SimpleDateFormat df1 = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");  String ctime=df1.format(t);  Object currentmember=request.getSession().getAttribute("currentuser");  Map<String, Object> obj=(Map<String,Object>)request.getSession().getAttribute("currentuser");  String memberid="0";  String truename="未知";  if(!(currentmember == null||currentmember == ""||currentmember.equals(""))){  memberid=obj.get("id").toString();  truename=obj.get("truename").toString();  }  String strSql=" insert into tblog (typeid,userid,username,msg,ctime) values (?,?,?,?,?)";  //定义参数对象  List<Object> params = new ArrayList<Object>();  params.add(typeid);  params.add(memberid);  params.add(truename);  params.add(msg);  params.add(ctime);  DBHelper Dal=new DBHelper();  Dal.excuteSql(strSql, params);  }      } |

# 使用类名调用静态方法



# 日志列表和查询

## 前端页面

|  |
| --- |
| <%@ page language=*"java"* contentType=*"text/html; charset=UTF-8"*  pageEncoding=*"UTF-8"*%>  <%@ page language=*"java"* import=*"java.util.\*"* %>  <%@ page language=*"java"* import=*"java.sql.\*"* %>  <%@ page language=*"java"* import=*"java.util.regex.\*"* %>  <!DOCTYPE html>  <html>  <head>  <meta charset=*"UTF-8"*>  <title></title>  <link rel=*"stylesheet"* type=*"text/css"* href=*"./css/comm.css"*/>  </head>  <body>  <%  List<Map<String, Object>> list = (List<Map<String, Object>>)request.getAttribute("list");  String pagestr = (String)request.getAttribute("pagestr");  %>  <div class=*"path"*>当前位置:/夜鹰进销存系统/系统管理/日志查询 <span class=*"rightbox"*><input type=*"text"* id=*"tbkey"*/><input type=*"button"* id=*"btnsearch"* value=*"查询"*/></span></div>  <table border=*"0"* cellspacing=*"0"* cellpadding=*"0"* class=*"tb tblist"*>  <tr>  <td style="width:*50px*;">ID</td><td style="width:*150px*;">姓名</td><td style="width:*150px*;">类别</td><td>内容</td><td style="width:*180px*;">时间</td>  </tr>  <%**for** (Map<String, Object> m : list) { %>  <tr>  <td><%=m.get("id")%></td>  <td><%=m.get("username")%></td>  <td data-typeid=*"*<%=m.get("typeid")%>*"* class=*"typename"*></td>  <td><%=m.get("msg")%></td>  <td><%=m.get("ctime")%></td>  </tr>  <% } %>  </table>    <div class=*"pager"*>  <%=pagestr%>  </div>    <script src=*"./js/jquery-1.11.0.js"* type=*"text/javascript"* charset=*"utf-8"*></script>  <script type=*"text/javascript"*>  $("td.typename").each(**function**(index,item){  **var** typeid=$(item).attr("data-typeid");  **switch**(typeid)  {  **case** "1":$(item).text("登录日志");**break**;  **case** "2":$(item).text("操作日志"); $(item).parents("tr").css({"background-color":"red","color":"#fff"});**break**;  **case** "3":$(item).text("异常日志");**break**;  }  });    **var** key = localStorage.getItem("keyprosearch");  $("#tbkey").val(key);  $("#btnsearch").click(**function** () {  **var** key = $("#tbkey").val();  localStorage.setItem("keyprosearch", key);  location.href = "tbloglist?key=" + key;  });    </script>  </body>  </html> |

## 后端代码

|  |
| --- |
| package com.rz;  import java.io.IOException;  import java.sql.SQLException;  import java.util.ArrayList;  import java.util.List;  import java.util.Map;  import javax.servlet.ServletException;  import javax.servlet.annotation.WebServlet;  import javax.servlet.http.HttpServlet;  import javax.servlet.http.HttpServletRequest;  import javax.servlet.http.HttpServletResponse;  /\*\*  \* Servlet implementation class tbloglist  \*/  @WebServlet("/tbloglist")  public class tbloglist extends HttpServlet {  private static final long serialVersionUID = 1L;    /\*\*  \* @see HttpServlet#HttpServlet()  \*/  public tbloglist() {  super();  // TODO Auto-generated constructor stub  }  /\*\*  \* @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)  \*/  protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  int currentpage = 1;  request.setCharacterEncoding("utf-8");  String key=request.getParameter("key");  if(!(key==null||key.equals("")))  {  key = new String(key.getBytes("iso-8859-1"), "utf-8");//解决中文乱码的问题  }  try {  String p = request.getParameter("p");  currentpage = Integer.valueOf(p);  } catch (Exception e) {  currentpage = 1;  }  DBHelper Dal = new DBHelper();  String strSql = " select id from tblog order by id desc ";  if(!(key==null||key.equals("")))  {  strSql = " select id from tblog where username like '%"+key+"%' or msg like '%"+key+"%' order by id desc ";  }    List<Map<String, Object>> listall = null;  List<Object> params = new ArrayList<Object>();  try {  listall = Dal.executeQuery(strSql, params);  } catch (SQLException e) {  e.printStackTrace();  }    Pager pageobj = new Pager();  pageobj.allrecordcount = listall.size();  pageobj.pagesize = 10;  pageobj.currentpage = currentpage;  pageobj.urlname = "";  pageobj.w="key="+key+"&classid=0";    int startindex = pageobj.pagesize \* (pageobj.currentpage - 1);  String strSqlpager = " select \* from tblog order by id desc limit "+startindex + "," + pageobj.pagesize + "";  if(!(key==null||key.equals("")))  {  strSqlpager = " select \* from tblog where username like '%"+key+"%' or msg like '%"+key+"%' order by id desc limit "+startindex + "," + pageobj.pagesize + "";  }    List<Map<String, Object>> listpage = null;  try {  listpage = Dal.executeQuery(strSqlpager, params);  } catch (SQLException e) {  e.printStackTrace();  }  String pagestr = pageobj.GetPageInfo();  request.setAttribute("pagestr", pagestr);  request.setAttribute("list", listpage);  request.getRequestDispatcher("/admin/tbloglist.jsp").forward(request,  response);    }  /\*\*  \* @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)  \*/  protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  // TODO Auto-generated method stub  }  } |

# 你平时如何调试代码

前后端代码的调试思路。

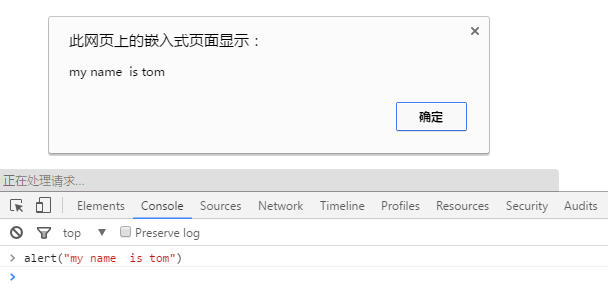
## 为什么掌握这些调试技巧

调试技巧，在任何一项技术研发中都可谓是必不可少的技能。掌握各种调试技巧，必定能在工作中起到事半功倍的效果。譬如，快速定位问题、降低故障概率、帮助分析逻辑错误等等。而在互联网前端开发越来越重要的今天，如何在前端开发中降低开发成本，提升工作效率，掌握前端开发调试技巧尤为重要。

这些前端JS调试技巧，也许你已经熟练掌握，那让我们一起来温习，也许有你没见过的方法，不妨一起来学习，也许你尚不知如何调试，赶紧趁此机会填补空白。

## 骨灰级调试Alert

那还是互联网刚刚起步的时代，网页前端还主要以内容展示为主，浏览器脚本还只能为页面提供非常简单的辅助功能的时候。那个时候，网页主要运行在以IE6为主的浏览器中，JS的调试功能还非常弱，只能通过内置于Window对象中的alert方法来调试，那时候看起来应该是这个样子：



需要说明一点，当年还没有这么高级的控制台，而alert的使用也是在真实的页面JS代码中。alert的调试方式很原始，但当时确实有它不可磨灭的价值，甚至到今天，已然有其用武之地。

缺点：

1. 弹出模态的调试信息窗口，不能操作其他地方。
2. 窗口着实不太美观，而且会遮挡部分页面内容。
3. 窗口会阻塞程序继续执行，要确定之后才进行下一步。

## 新一代调试王者Console

随着JS在Web前端中能做的事情越来越多，责任越来越大，而地位也越来越重要。传统的alert调试方式已经渐渐不能满足前端开发的种种场景。

### 常用的方法

**console.log(object[, object, ...])，使用频率最高的一条语句**：向控制台输出一条消息。支持 C 语言 printf 式的格式化输出。

console.info(object[, object, ...]) ，向控制台输出一条信息，该信息包含一个表示“信息”的图标，和指向该行代码位置的超链接。

console.error(object[, object, ...])在控制台中输出一条错误信息。

console.warn(object[, object, ...])在控制台中输出一条警告信息。

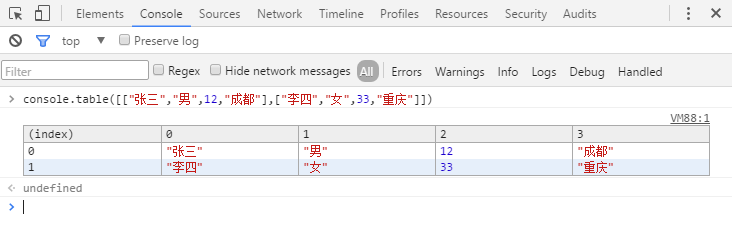
**console.dir(object) 输出一个对象的全部属性**（输出结果类似于 DOM 面板中的样式）。

**console.clear() 清空控制台。**

以Chrome浏览器为首的后起之秀，为Console扩展了更丰富的功能：

### 输出表格

例1： console.table([["张三","男",12,"成都"],["李四","女",33,"重庆"]])



### 美化console.log的文本

先看两个效果

1、打开百度按F12，就会出现下图效果：

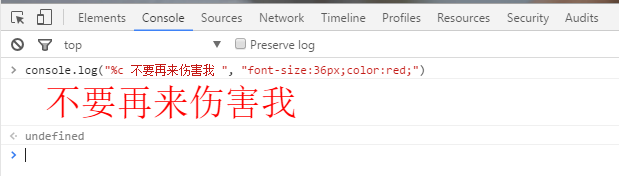


再看支付宝：

<https://auth.alipay.com/login/index.htm>



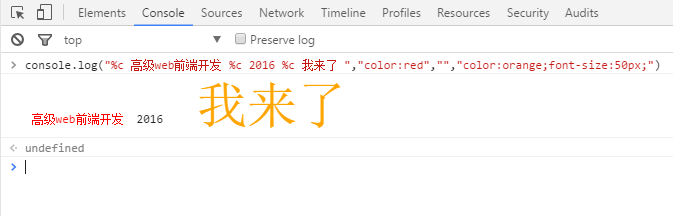
例1：一行文本单个样式。



console.log("%c 不要再来伤害我 ", "font-size:36px;color:red;")

例2：一行文本多个样式，每个%c 对应一个样式

console.log("%c 高级web前端开发 %c 2016 %c 我来了 ","color:red","","color:orange;font-size:50px;")



Console.log输出文字样式支持css3

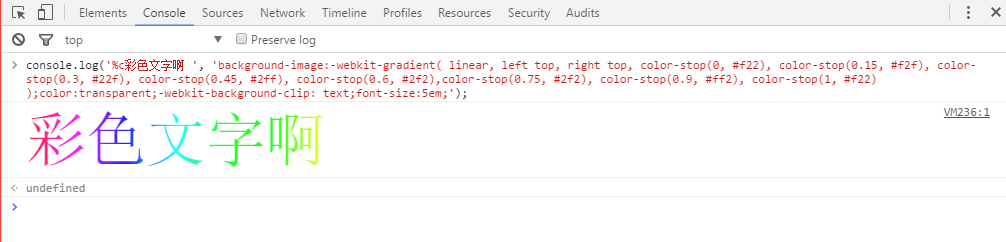
例3：给文字加阴影。

console.log("%c两情若是久长时，又岂在朝朝暮暮！", "text-shadow: 3px 1px 1px grey;font-size:36px");



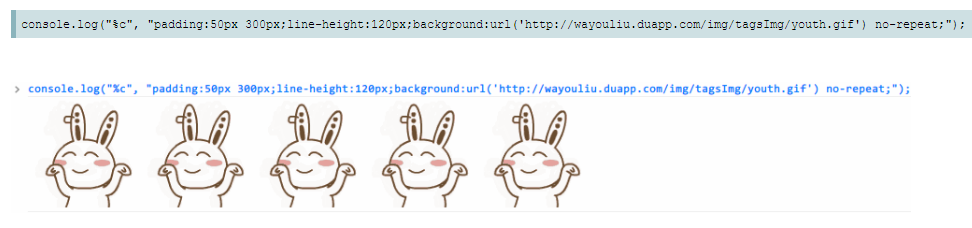
例4：彩色文字。

console.log('%c彩色文字啊 ', 'background-image:-webkit-gradient( linear, left top, right top, color-stop(0, #f22), color-stop(0.15, #f2f), color-stop(0.3, #22f), color-stop(0.45, #2ff), color-stop(0.6, #2f2),color-stop(0.75, #2f2), color-stop(0.9, #ff2), color-stop(1, #f22) );color:transparent;-webkit-background-clip: text;font-size:5em;');



例5：输出背景图片，甚至是gif动画作背景图片。

console.log("%c", "padding:50px 300px;line-height:120px;background:url('http://wayouliu.duapp.com/img/tagsImg/youth.gif') no-repeat;");



### JS断点调试

JS断点调试，即是在浏览器开发者工具中为JS代码添加断点，让JS执行到某一特定位置停住，方便开发者对该处代码段的分析与逻辑处理。为了能够观察到断点调试的效果，我们预先随意准备一段JS代码：

例如，最终想输出m5,结果把m5写成了m2,传统调试可以在代码里加很多个alert,但是这样调试效率低，老弹出确认框，此时就可以打断点调试。

<script type="text/javascript">

function add(a,b)

{

var c=a+b

return c;

}

$(function(){

$("#btn1").click(function(){

var m1=100;

var m2=m1+200;

var m3=m1-m2;

var m4=add(m1,m3);

var m5=m4;

$("#box1").text(m5);

console.log(m5);//例如m5不小心写成m2了，就可以打断点调试

});

});

</script>