Session 部分：

——createsession

主要用setAttribute的方法 不知道你是不是也一样，但是需要考虑一下session的生命周期和什么时候destroy。

HttpSession session=req.getSession();

session.setAttribute("username", username);

——Keepsession.do

原则其实就是你把创建好的session（应该是有一个key用来存用户名，即我们情况中的邮箱）对应的用户信息查出来，封装成一个customer对象，然后转成json的格式传给我，我就可以有办法保证用户的登录状态了。

public class KeepSessionServlet extends HttpServlet {

private static final long serialVersionUID = 1L;

private AdminService adminService;

public AdminService getAdminService() {

return adminService;

}

public void setAdminService(AdminService adminService) {

this.adminService = adminService;

}

protected void doGet(HttpServletRequest req, HttpServletResponse resp)

throws ServletException, IOException {

HttpSession session=req.getSession();

String username=(String) session.getAttribute("username");

resp.setContentType("text/html;charset=UTF-8");

java.io.PrintWriter out = resp.getWriter();

Map<String, Object> result = new HashMap<String, Object>();

List<Administrator> adminList=adminService.keepSession(username);

Administrator admin=adminList.get(0);

result.put("admin", admin);

String result\_str = JSON.toJSONString(result);

out.print(result\_str);

}

public static long getSerialversionuid() {

return serialVersionUID;

}

protected void doPost(HttpServletRequest req, HttpServletResponse resp)

throws ServletException, IOException {

this.doGet(req, resp);

}

}

Filter 部分：

我之前做的filter主要是用来做权限控制的，不知道你用filter做什么。然后你做filter可能会涉及到代理服务器这种。

public class SessionFilter implements Filter {

private AdminService adminService;

public AdminService getAdminService() {

return adminService;

}

public void setAdminService(AdminService adminService) {

this.adminService = adminService;

}

public FilterConfig getConfig() {

return config;

}

public void setConfig(FilterConfig config) {

this.config = config;

}

public FilterConfig config;

public void destroy() {

this.config = null;

}

public static boolean isContains(String container, String[] regx) {

boolean result = false;

for (int i = 0; i < regx.length; i++) {

if (container.indexOf(regx[i]) != -1) {

return true;

}

}

return result;

}

public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain) throws IOException, ServletException {

HttpServletRequest hrequest = (HttpServletRequest)request;

HttpServletResponseWrapper wrapper = new HttpServletResponseWrapper((HttpServletResponse) response);

String logonStrings = "/app/login.html;loginto.do"; //<!--登录登陆页面-->

String notincludeStrings= config.getInitParameter("notincludeStrings");

String includeStrings = config.getInitParameter("includeStrings"); //<!--过滤资源后缀参数-->

String redirectPath = hrequest.getContextPath() + "/login.html";//<!--没有登陆转向页面-->

String redirectPath2 = hrequest.getContextPath() + "/error.html";//<!--没有登陆转向页面-->

String disabletestfilter = config.getInitParameter("disabletestfilter");//<!-- 过滤器是否有效-->

if (disabletestfilter.toUpperCase().equals("Y")) { //<!--过滤无效-->

chain.doFilter(request, response);

return;

}

String[] logonList = logonStrings.split(";");

String[] includeList = includeStrings.split(";");

String[] notincludeList=notincludeStrings.split(";");

if (!this.isContains(hrequest.getRequestURI(), includeList)) {//<!--只对指定过滤参数后缀进行过滤-->

chain.doFilter(request, response);

return;

}

if (this.isContains(hrequest.getRequestURI(), notincludeList)) {

chain.doFilter(request, response);

return;

}

if (this.isContains(hrequest.getRequestURI(), logonList)) {//<!--对登录页面不进行过滤-->

chain.doFilter(request, response);

return;

}

String user = ( String ) hrequest.getSession().getAttribute("username");//<!--判断用户是否登录-->

Map<String,List<String>> roleresource=new HashMap<String,List<String>>();

List<String> resource\_super=new ArrayList<String>();

resource\_super.add("index.html");

resource\_super.add("rzgl\_czrz.html");

resource\_super.add("rzgl\_fwjl.html");

resource\_super.add("rzgl\_ycfw.html");

resource\_super.add("tjfz\_bcfk.html");

resource\_super.add("tjfz\_syqk.html");

resource\_super.add("tjfz\_yhfx\_xxqk.html");

resource\_super.add("tjfz\_yhfx\_yhzz.html");

resource\_super.add("tjfz\_yhfx\_yhsx.html");

resource\_super.add("tjfz\_yhlcl.html");

resource\_super.add("tjfz\_yhzc.html");

resource\_super.add("xxgl\_scgl.html");

resource\_super.add("xxgl\_xjxx.html");

resource\_super.add("xxgl\_xxjl.html");

resource\_super.add("yhgl\_jsqx.html");

resource\_super.add("yhgl\_xtjs.html");

resource\_super.add("yhgl\_ydyh.html");

resource\_super.add("error.html");

roleresource.put("超级用户", resource\_super);

List<String> resource\_admin=new ArrayList<String>();

resource\_admin.add("index.html");

resource\_admin.add("rzgl\_czrz.html");

resource\_admin.add("rzgl\_fwjl.html");

resource\_admin.add("rzgl\_ycfw.html");

resource\_admin.add("xxgl\_scgl.html");

resource\_admin.add("xxgl\_xjxx.html");

resource\_admin.add("xxgl\_xxjl.html");

resource\_admin.add("yhgl\_ydyh.html");

resource\_admin.add("error.html");

roleresource.put("系统管理员", resource\_admin);

List<String> resource\_analysit=new ArrayList<String>();

resource\_analysit.add("index.html");

resource\_analysit.add("yhgl\_ydyh.html");

resource\_analysit.add("tjfz\_bcfk.html");

resource\_analysit.add("tjfz\_syqk.html");

resource\_analysit.add("tjfz\_yhfx\_xxqk.html");

resource\_analysit.add("tjfz\_yhfx\_yhzz.html");

resource\_analysit.add("tjfz\_yhfx\_yhsx.html");

resource\_analysit.add("tjfz\_yhlcl.html");

resource\_analysit.add("tjfz\_yhzc.html");

resource\_analysit.add("error.html");

roleresource.put("系统分析师", resource\_analysit);

if (user == null) {

wrapper.sendRedirect(redirectPath);

return;

}else{

String role=adminService.queryRole(user);

List<String> checkList=roleresource.get(role);

String[] strs = checkList.toArray(new String[checkList.size()]);

for(int i=0;i<checkList.size();i++){

if(!this.isContains(hrequest.getRequestURI(), strs)){

wrapper.sendRedirect(redirectPath2);

return;

}else{

chain.doFilter(request, response);

return;

}

}

}

}

//还是有一些bug 角色可访问资源还没有写完整 而且对于系统管理员来说只控制html并不科学

public void init(FilterConfig filterConfig) throws ServletException {

config = filterConfig;

}

}

如何封装json数据：

我用的是阿里巴巴出品的fastjson.jar架包，我已经传到项目里面了，用到目前为止没出过问题，还是挺神器的。只能说hashmap也是神器。

——单键字符串

Map<String, Object> result = new HashMap<String, Object>();

result.put("code", "密码输入错误！");

String result\_str = JSON.toJSONString(result);

out.print(result\_str);

——List

Map<String, List<Administrator>> result = new HashMap<String, List<Administrator>>();

List<Administrator> adminList=adminService.queryAdminAll();

result.put("adminList", adminList);

String result\_str = JSON.toJSONString(result,SerializerFeature.DisableCircularReferenceDetect);

out.print(result\_str);

依次类推，就是我们项目里用返回字符串（表状态），和返回对象（表示某一个object，如keepsession），返回一个List(数据集)比较多。

建议还是要学一下json的格式，因为虽然大部分地方都是可以直接转的，但是遇到特殊的数据的时候（如时间数据）需要变形的时候，可能还是存在手动拼接的可能性，所以做好心理准备。因为mysql中的时间格式和java中的时间格式是有区别的，你应该也知道，我就不多说了。大致你今天问我的东西就是这些吧。