泛微移动端为了安全性的考虑，引入了token机制，无法直接获取用户登录的账号和密码，因此不能直接采用简单的单点登录方案来实现。

由于能获取到用户的账号，与泛微移动端的集成有两种方案：

1、基于token验证后获取的用户名直接在报表系统中登录，不使用密码，实际客户案例惠而浦

2、基于token验证后获取的用户名，从用户账号密码数据对照表中获取用户的登录密码，进行单点登录。

方案1

**一、所需环境准备**

**1、手机浏览器报表插件**

微信集成的企业应用中打开的只能是HTML5页面，不能使用FR新版移动app的原生解析功能。 FR开发了手机浏览器HTML5浏览方式，实现手机浏览器中报表的自适应、参数界面分离等效果。 移动终端HTML5浏览方式仅对分页报表进行了处理，详细请查阅[移动端Html5报表](http://help.finereport.com/doc-view-781.html" \o "移动端Html5报表)

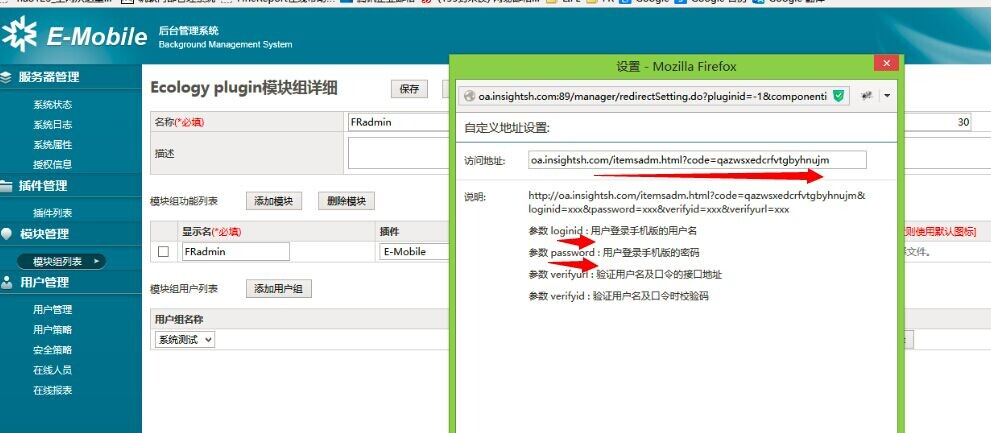
设计器插件安装方法参照[插件的安装管理](http://help.finereport.com/doc-view-793.html)

服务器安装插件方法参照[服务器上安装插件](http://help.finereport.com/doc-view-790.html)

**2、需要commons-codec-1.9.jar包**

需要将[commons-codec-1.9.jar](http://www.finereporthelp.com:8021/download/FR%E4%B8%8E%E5%BE%AE%E4%BF%A1%E9%9B%86%E6%88%90/commons-codec-1.9.jar)包拷贝到报表服务器WEB-INF/lib下面，如果已经存在其他版本，例如我们设计器工程里已存在commons-codec-1.6.jar，则可以不需要再导入；

**二、泛微老版本app集成**



**图1-1**

如图泛微老版本的app会传用户名loginid，密码参数password，所以思路大致和PC端集成没太大区别，ajax的方式即可，在此不再赘述。

相应的访问页面添加到泛微app模块下提供的访问地址处即可。

**三、泛微新版本app集成**

注：如果泛微PC端也改为基于token认证的话，此方法应该同样适用于PC端单点登录，可供参考



**图2-1**

如图泛微新版本应该是出于安全考虑，不在提供登录用户的password系统参数，所以之前的ajax方式不适用了。

**1、解决思路：**

**A.报表可以实现不传密码单点认证**

1、把编译后的filter文件（具体代码见下文）部署到报表工程的WebReport/WEB-INF/classes/com/fr下面

2、修改报表工程WebReport/WEB-INF/下的web.xml文件添加：

|  |
| --- |
| <filter>   <filter-name>filter</filter-name>   <filter-class>com.fr.filter.loginfilter</filter-class>   </filter> |

3、将访问文件hello.html放在WebReport目录下，同时web.xml添加如下标签：

|  |
| --- |
| <filter-mapping>  <filter-name>filter</filter-name>  <url-pattern>/hello.html</url-pattern>  </filter-mapping> |

hello.html并无什么直接意义，内容可随意。参考代码如下：

|  |
| --- |
| <html>  <body>  <script type="text/javascript">  var myvalue = "aaa";  function fun1() {   return "function return test";   }  document.write("Hello World!");  </script>  </body>  </html> |

4、将http://ip:端口/WebReport/hello.html 添加到泛微app模块下提供的访问地址处

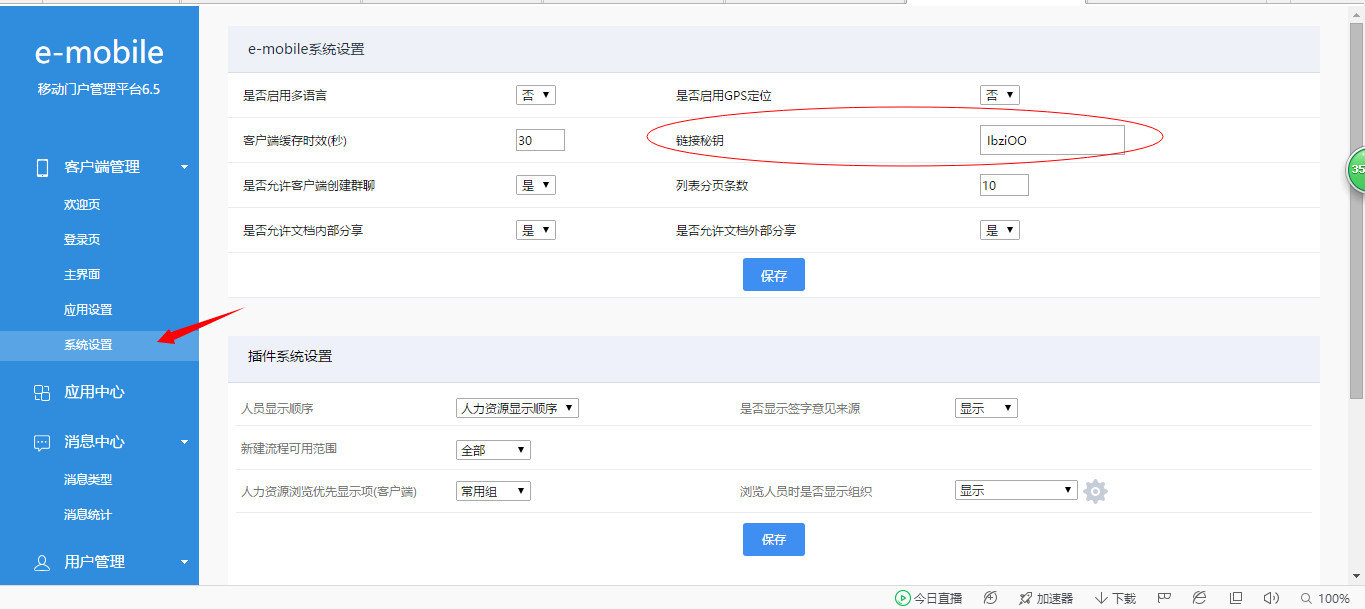
**B.token机制**

虽然不认证密码可以实现只通过用户名认证实现单点登录，但是并不安全，避免不了PC端有人通过get传参的方式直接等数据决策系统，因此需要引入泛微的token机制。

token的认证机制见图2-1，token = SHA1(链接秘钥+loginid+stamp)，SHA1为相应的算法。

stamp为时间戳变量，随时间增长单调递增。loginid为用户名参数。

链接秘钥需要在泛微的平台中提前查到，认证过程中为常量，获取位置见下图2-2



**图2-2**

所以最终的认证方式为：

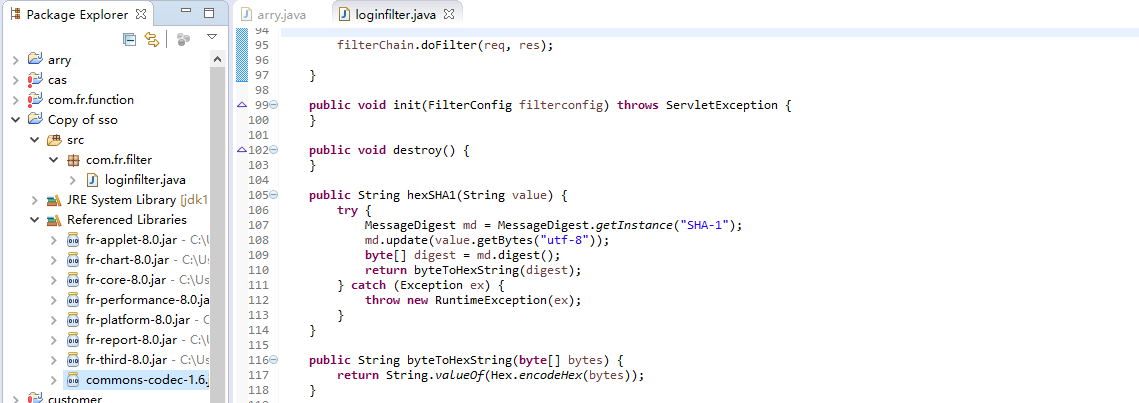
 先进行泛微的token认证。即泛微传过来的token值与报表filter中通过传过来的其他参数计算的token值相等，则验证通过，否则返回跳转地址。

认证通过后进行报表用户名认证，认证通过则跳转到决策系统。

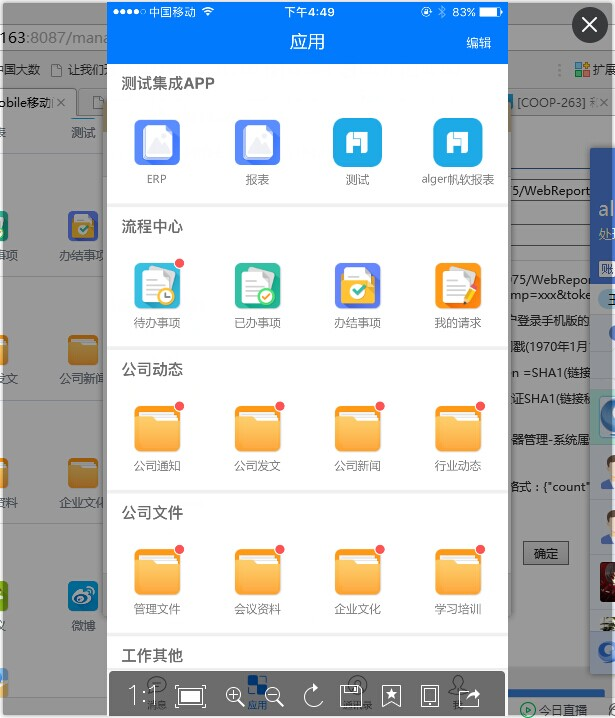
filter代码如下（示例客户app集成实现代码）

|  |
| --- |
| package com.fr.filter;  import com.fr.fs.base.entity.User;  import com.fr.fs.base.entity.UserInfo;  import com.fr.fs.control.UserControl;  import com.fr.fs.privilege.auth.FSAuthentication;  import com.fr.fs.privilege.base.FServicePrivilegeLoader;  import com.fr.fs.privilege.entity.DaoFSAuthentication;  import com.fr.fs.web.FSConstants;  import com.fr.privilege.session.PrivilegeInfoSessionMananger;  import javax.servlet.\*;  import javax.servlet.http.HttpServletRequest;  import javax.servlet.http.HttpServletResponse;  import javax.servlet.http.HttpSession;      import org.apache.commons.codec.binary.Hex;  import java.io.IOException;  import java.security.MessageDigest;      public class loginfilter implements Filter {  public void doFilter(ServletRequest req, ServletResponse res,   FilterChain filterChain) throws IOException, ServletException {  // System.out.println("#####最开始的filter####");   HttpServletRequest re = (HttpServletRequest) req;   HttpServletResponse resp = (HttpServletResponse) res;  String username = re.getParameter("loginid");   username = username.toUpperCase();// 无论用户名登录的时候是大些还是小写，泛微传过来的loginid都给转成了小写。示例客户报表中的用户为大写，此处做toUpperCase处理。其他客户看情况处理。   HttpSession session = re.getSession(true);  // 增加oa token验证   String loginid = re.getParameter("loginid");// 登陆用户   String stamp = re.getParameter("stamp");// 提交时间   String token = re.getParameter("token");// token   String secrect = "IbziOO"; //图2-2处可查询到   String newtoken = hexSHA1(secrect + loginid + stamp);   System.out.println("newtoken------------------------------" + newtoken+"token==="+token);   if (!newtoken.equals(token)) {   resp.sendRedirect("[http://www.baidu.com"](http://www.baidu.com/));   // token验证未通过返回地址   }   // 验证结束  // resp.sendRedirect("[http://ip:端口/WebReport/ReportServer?op=touchpf&\_\_redirect\_\_=false&username="+user+"&fr\_password="+user);](http://ip:%E7%AB%AF%E5%8F%A3/WebReport/ReportServer?op=touchpf&__redirect__=false&username=)  // System.out.println("＃＃＃＃user:＃＃＃＃"+user);   // session.setAttribute("fsusername", username);   // session.setAttribute("fspassword", username);  // 开始报表认证  try {   User U = UserControl.getInstance().getByUserName(username);   if (U == null) {  // resp.getWriter().write("<script>window.location='www.baidu.com';</script>");   resp.sendRedirect("[http://www.baidu.com"](http://www.baidu.com/)); //用户名错误，跳转地址   System.out   .println("---------------username-------------------");   System.out.println(username);   System.out   .println("---------------loginid---------------------");   System.out.println(username);   }   System.out.println(U.getId());   FSAuthentication authentication = new DaoFSAuthentication(   new UserInfo(U.getId(), username, username));   // HttpSession session = request.getSession(true);   long userid = authentication.getUserInfo().getId();   System.out.println(userid);  PrivilegeInfoSessionMananger.login(new FServicePrivilegeLoader(   username, UserControl.getInstance()   .getAllSRoleNames(userid), UserControl   .getInstance().getUserDP(userid)), session, resp);   session.setAttribute(FSConstants.P\_KEYS.PRIVILEGE\_AUTHENCATION\_KEY,   authentication);   UserControl.getInstance().login(userid);   resp.sendRedirect("[http://ip:端口/WebReport/ReportServer?op=fs"](http://ip:%E7%AB%AF%E5%8F%A3/WebReport/ReportServer?op=fs));  //认证成功跳转决策平台   // System.out.println("+++++++++++++++++++username+++++++++++++++++++");   // System.out.println(username);   System.out   .println("+++++++++++++++++++loginid++++++++++++++++++++++++");   // System.out.println(username);   } catch (Exception e) {   // TODO Auto-generated catch block   e.printStackTrace();   } // 通过用户名获取用户表  filterChain.doFilter(req, res);  }  public void init(FilterConfig filterconfig) throws ServletException {   }  public void destroy() {   }  public String hexSHA1(String value) {   try {   MessageDigest md = MessageDigest.getInstance("SHA-1");   md.update(value.getBytes("utf-8"));   byte[] digest = md.digest();   return byteToHexString(digest);   } catch (Exception ex) {   throw new RuntimeException(ex);   }   }  public String byteToHexString(byte[] bytes) {   return String.valueOf(Hex.encodeHex(bytes));   }  } |

**2、filter编译需引入的jar包**



**3、集成效果**



注：集成后默认的显示H5效果（需要安装H5插件，否则和PC端显示一样的效果）



视觉帮做的圆角图标，可上传到客户app做报表模块图标

**PS ：**

概述：EMobile单点登录到第三方web页面认证方式详解（可对照filter代码）

接口：直接在EMobile服务端后台配置要登录的第三方服务地址即可，无需配置参数， 系统会自动生成

说明：代码中的loginid、stamp、token分别是参数值，第三方直接获取即可，secrect  是后台取的链接密钥值，token要在第三方进行验证

token生成方式示例代码：

|  |
| --- |
| token  =  CodeUtil.hexSHA1(secrect+loginid+stamp); |

其中的CodeUtil类如下：

|  |
| --- |
| package io.rong.util;  import java.security.MessageDigest;  import org.apache.commons.codec.binary.Hex;  public class CodeUtil {  public static String hexSHA1(String value) {  try {  MessageDigest md = MessageDigest.getInstance("SHA-1");  md.update(value.getBytes("utf-8"));  byte[] digest = md.digest();  return byteToHexString(digest);  } catch (Exception ex) {  throw new RuntimeException(ex);  }  }  public static String byteToHexString(byte[] bytes) {  return String.valueOf(Hex.encodeHex(bytes));  }  } |