










































































java--

ONLYXX[]-()-*

com.fr.fs.control.UserControl

- ✓ C  UserControl
 - m   UserControl()
 - m   getInstance(): UserControl
 - m   isRootManagerPasswordMatch(String): boolean
 - m   getUsersByJRoleID(long): Set<Long>
 - m   getSuperManagerID(): long
 - m   getUser(String, String): User
 - m   getByUserName(String): User
 - m   getUser(long): User
 - m   getUserByNameCacheFirst(String): User
 - m   getUserByMobile(String): User
 - m   getUsersByMobile(String): User[]
 - m   getUsersByEmail(String): User[]
 - m   getUser(String): Long
 - m   getOpenDAO(Class<? extends T>): T
 - m   login(long): void
 - m   logout(long): void
 - m   addUser(User): boolean
 - m   addUser(User, boolean): boolean
 - m   deleteUser(long): boolean
 - m   updateUserAuthInfo(User): boolean
 - m   updateLoginUserInfo(String, String): void
 - m   updateUserPersonalInfo(User): boolean
 - m   updatePassword(long, String, String): boolean
 - m   isPass(String, String, String): boolean
 - m   encodePassword(String): String
 - m   encodePassword(String, String): String
 - m   getAllSRoleNames(long): FArray
 - m   findAllUser(): List
 - m   findAllUserWithoutSort(): List
 - m   getAllMailUser(Boolea): JSONArray
 - m   getAllMailUser(): JSONArray
 - m   findAllAuthUser(long): List
 - m   findAllAuthUserIds(long): List<Long>
 - m   findAllRoleUsers(AbstractDepAndCRolePrivile
 - m   getNoRoleUsers(): Set<User>
 - m   getAllUserInfo4D(): JSONArray



UserControl.java

ok

actions

plugin.xmlwebService

Service

```
<extra-core>  
  <WebService class="com.fr.plugin.Actions"/>  
</extra-core>
```

Serviceaction

actionsService

Actions

```
package com.fr.plugin;

import com.fr.plugin.actions.BlueMessageLoginAction;
import com.fr.plugin.actions.GetAction;
import com.fr.plugin.actions.SaveAction;
import com.fr.plugin.actions.UserTransferAction;
import com.fr.plugin.utilUrlUtils;
import com.fr.stable.StringUtils;
import com.fr.stable.fun.Service;
import com.fr.web.core.ActionNoSessionCMD;
import com.fr.web.core.WebActionsDispatcher;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class Actions implements Service {
    //
    private ActionNoSessionCMD[] actions = {
        new LoginAction()
    };

    @Override
    public String actionOP() {
        // op=my_plugin_actions
        return "my_plugin_actions";
    }

    @Override
    public void process(HttpServletRequest httpServletRequest, HttpServletResponse httpServletResponse, String
s, String sl) throws Exception {
        //cmd
        WebActionsDispatcher.dealForActionNoSessionIDCMD(httpServletRequest, httpServletResponse, actions);
    }
}
```

ActionNoSessionCMD ActionCMD

loginAction

```
package com.fr.plugin.actions;

import com.fr.base.TemplateUtils;
import com.fr.cluster.stable.ClusterState;
import com.fr.file.BaseClusterHelper;
import com.fr.file.ClusterConfigManager;
import com.fr.file.ClusterService;
import com.fr.fs.base.entity.User;
import com.fr.fs.base.entity.UserInfo;
import com.fr.fs.control.UserControl;
import com.fr.fs.web.FSConstants;
import com.fr.general.ComparatorUtils;
import com.fr.json.JSONException;
import com.fr.json.JSONObject;
import com.fr.plugin.pojo.BlueMessageUserInfo;
import com.fr.plugin.pojo.UserModel;
import com.fr.plugin.utilHttpUtils;
import com.fr.plugin.utilUrlUtils;
```

```

import com.fr.stable.Constants;
import com.fr.stable.StringUtils;
import com.fr.web.cluster.ClusterManager;
import com.fr.web.core.ActionNoSessionCMD;
import com.fr.web.utils.WebUtils;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import java.io.PrintWriter;
import java.util.HashMap;
import java.util.Map;

public class BlueMessageLoginAction extends ActionNoSessionCMD {
    @Override
    public String getCMD() {
        return "blue_message_login";
    }

    @Override
    public void actionCMD(HttpServletRequest req, HttpServletResponse httpServletResponse) throws Exception {
        PrintWriter writer = WebUtils.createPrintWriter(httpServletResponse);
        String code = req.getParameter("code");
        if (StringUtils.isNotBlank(code)) {
            BlueMessageUserInfo userInfo=null;
            try {
                userInfo = Service.getOAuth2AccessToken(code);//code
            }catch (Exception e){
                writer.write(""+e.getMessage());
                writer.flush();
                writer.close();
                return;
            }
            if (userInfo != null) {
                HttpSession session = req.getSession(true);
                BlueMessageUserInfo newUser = Service.getUserInfo(userInfo);
                UserModel userModel = newUser.getUserModel();
                if (userModel != null) {
                    UserControl userControl = UserControl.getInstance();
                    User userByMobile = userControl.getUserByMobile(userModel.getMobile());//
                    if (userByMobile != null) {
                        String password = userByMobile.getPassword();
                        String username = userByMobile.getUsername();
                        UserInfo ui = new UserInfo(username, password,true);
                        ui.dealBrowserCookies(httpServletResponse, session);
                        boolean isTemplate = ComparatorUtils.equals(true, session.getAttribute("isTemplate"));
                        Object oo = session.getAttribute(isTemplate ? Constants.PF.TEMPLATE_ORIGINAL_URL :
Constants.ORIGINAL_URL);
                        String url = (oo == null) ? getRenderedUrl() : oo.toString()
                            + "&_" + System.currentTimeMillis();
                        addServerID(session);

                        httpServletResponse.sendRedirect(url);
                        return;
                    }else {
                        User user=new User(userModel.getMobile(),"123456",userModel.getName());
                        user.setMobile(userModel.getMobile());
                        user.setEmail(userModel.getEmail());
                        user.setWorkphone(user.getMobile());
                        userControl.addUser(user);
                        UserInfo ui = new UserInfo(userModel.getId(),"123456",true);
                        ui.dealBrowserCookies(httpServletResponse, session);
                        boolean isTemplate = ComparatorUtils.equals(true, session.getAttribute("isTemplate"));
                        Object oo = session.getAttribute(isTemplate ? Constants.PF.TEMPLATE_ORIGINAL_URL :
Constants.ORIGINAL_URL);

                        String url = (oo == null) ? getRenderedUrl() : oo.toString()
                            + "&_" + System.currentTimeMillis();
                        addServerID(session);
                        httpServletResponse.sendRedirect(url);
                        return;
                    }
                }
            }
        }
    }
}

```

```

        }
    }
}
writer.write("~");
writer.flush();
writer.close();
}
private void addServerID(HttpSession session) {
    if (ClusterConfigManager.getInstance().isUseCluster()) {
        ClusterService mainService = ClusterManager.getInstance().getMainService();
        String serviceName = mainService.getServiceName() + "_" + System.currentTimeMillis();
        session.setAttribute(FSConstants.SERVER_ID, serviceName);
    }
}
//
public static String getRenderedUrl() throws Exception {
    Map<String, Object> para = new HashMap<String, Object>();
    if (BaseClusterHelper.getClusterState() == ClusterState.LEADER) {
        para.put("serverURL", "http://" + ClusterConfigManager.getInstance().getPublicURL());
    }

    return TemplateUtils.renderParameter4Tpl("${serverURL}${servletURL}?op=fs", para);
}
}

```

getCMDactionactionspringrequeustMapping

UserControl userControl = UserControl.getInstance();