# HSWF框架集成：

## 配置文件spring-security.xml升级：

这个文件SVN上只在webcore工程下有，安装到环境中会同步到每个应用下

SVN路径： /WEBCORE/WebContent/WEB-INF/conf/spring-security.xml

环境路径：/opt/webserver/webapps\_ana/cem/WEB-INF/conf

修改点：

* + - 1. **头信息修改**

http://www.springframework.org/schema/security/spring-security-3.1.xsd

修改为：

http://www.springframework.org/schema/security/spring-security-4.0.xsd

* + - 1. **http配置修改**

<http entry-point-ref="ecm-sm-ProcessingFilterEntryPoint" access-denied-page="/pages/common/403.jsp">

<custom-filter after="FORM\_LOGIN\_FILTER" ref="ecm-sm-ProcessingFilter" />

<custom-filter before="FILTER\_SECURITY\_INTERCEPTOR" ref="ecm-sm-AccessFilter" />

</http>

**修改为：**

<http entry-point-ref="ecm-sm-ProcessingFilterEntryPoint">

<custom-filter after="FORM\_LOGIN\_FILTER" ref="ecm-sm-ProcessingFilter" />

<custom-filter before="FILTER\_SECURITY\_INTERCEPTOR" ref="ecm-sm-AccessFilter" />

<access-denied-handler error-page="/pages/common/403.jsp"/>

<csrf disabled="true"/>这个属性是禁止csrf验证的，

<headers><frame-options policy="SAMEORIGIN"/></headers>

</http>

## jar包升级

### spring的安全jar包升级到4.0

spring-security-acl-3.1.4.RELEASE.jar

spring-security-aspects-3.1.4.RELEASE.jar

spring-security-cas-3.1.4.RELEASE.jar

spring-security-config-3.1.4.RELEASE.jar

spring-security-core-3.1.4.RELEASE.jar

spring-security-crypto-3.1.4.RELEASE.jar

spring-security-ldap-3.1.4.RELEASE.jar

spring-security-openid-3.1.4.RELEASE.jar

spring-security-remoting-3.1.4.RELEASE.jar

spring-security-taglibs-3.1.4.RELEASE.jar

spring-security-web-3.1.4.RELEASE.jar

**升级为：**

spring-security-acl-4.0.1.RELEASE.jar

spring-security-aspects-4.0.1.RELEASE.jar

spring-security-cas-4.0.1.RELEASE.jar

spring-security-config-4.0.1.RELEASE.jar

spring-security-core-4.0.1.RELEASE.jar

spring-security-crypto-4.0.1.RELEASE.jar

spring-security-ldap-4.0.1.RELEASE.jar

spring-security-openid-4.0.1.RELEASE.jar

spring-security-remoting-4.0.1.RELEASE.jar

spring-security-taglibs-4.0.1.RELEASE.jar

spring-security-web-4.0.1.RELEASE.jar

### 平台jar包升级

com.ecm.frm.jar

com.ecm.log-i.jar

com.ecm.sec-i.jar

### 增加HWSF组件（版本号可能会变）

huawei-secure-core-1.0.10.jar

huawei-secure-token-1.0.1.jar

huawei-secure-validator-2.0.1.jar

# 防CSRF

## 在发送的请求的头文件中加入token信息，防止CSRF攻击

## 用sweetui的场景：

通过weet.Ajax.request(config)发送请求

**第一步，在jsp页面header标签中添加**

<meta name="\_csrf" content="${\_csrf.token}"/>

<!-- default header name is X-CSRF-TOKEN -->

<meta name="\_csrf\_header" content="${\_csrf.headerName}"/>

**第二部，在发送json前添加这个属性：**

function request(url, callback, params, loadMaskBool){

var contentType = "application/x-www-form-urlencoded";

var token = $("meta[name='\_csrf']").attr("content");

var header = $("meta[name='\_csrf\_header']").attr("content");

if($.isNull(loadMaskBool)){

loadMaskBool = false;

}

var config = {

url: url,

async: true,

loadMask : loadMaskBool,

contentType: contentType,

beforeRequest : function(xhttp, req){

req.setRequestHeader(header, token);

},

success: function(result){

//返回的数据用于回调函数

if($.isNotNull(result)){

result = result.data;

if(callback && $.isFunction(callback)){

callback(result);

}

}

},

error: function(e){

if (e.readyState === 4 || e.status === 200){

callback();

}

}

};

//传给后台的参数

if(!$.isNull(params)){

config.data = params;

}

//发送请求

Sweet.Ajax.request(config);

};

## 直接用jquery的场景：

**第一步，在jsp页面header标签中添加**

<meta name="\_csrf" content="${\_csrf.token}"/>

<!-- default header name is X-CSRF-TOKEN -->

<meta name="\_csrf\_header" content="${\_csrf.headerName}"/>

**第二步，在发送json前添加这个属性：**

var tz = new Date().getTimezoneOffset()/60;

var offsetSec = tz \* 3600 \* 1000;

var timeZone = me.config.timeZone;

var addSec = timeZone \* 3600 \* 1000;

var dfmt = 'Y-m-d';

if(true == me.config.showHour || "true" == me.config.showHour){

dfmt = 'Y-m-d H:i';

}

var token = $("meta[name='\_csrf']").attr("content");

var key = $("meta[name='\_csrf\_header']").attr("content");

date = new Date(parseInt(value) \* 1000 + offsetSec + addSec).format(dfmt);

var res;

$.ajax({

type: "POST",

url:basePath + '/userQueryWebserverTimeOperate/utcToDate.action',

dataType: "json",

async: false,

timeout:60000,

data: {

utc : value

},

beforeSend : function(req){

req.setRequestHeader(key, token);

},

success: function (result) {

if(result && result.resultDate){

res = result.resultDate;

if(res.toUpperCase().indexOf('(DST)') > -1){

res = res.substring(0,res.indexOf('(DST)'));

}

}

},

error: function (result) {

mvc.log('translate Date fails!');

}

});

if(res){

date = res;

}

## Flex的场景：

**每个业务修改：**

**第一步：在jsp页面header标签中添加**

<meta name="\_csrf" content="${\_csrf.token}"/>

<!-- default header name is X-CSRF-TOKEN -->

<meta name="\_csrf\_header" content="${\_csrf.headerName}"/>

**第二步：在jsp页面**

添加引用：<script type="text/javascript" src="sweet/lib/jquery/jquery-1.9.1.min.js"></script>

添加方法： <script type="text/javascript" >

function getToken()

{

var token = $("meta[name='\_csrf']").attr("content");

var header = $("meta[name='\_csrf\_header']").attr("content");

var obj = {};

obj.key = header;

obj.value = token;

return obj;

}

</script>

**公共修改(修改完后要重新编译应用下所有swf)：**

RSL\src\Controller\UIController.as

1：sendMessage方法添加getToken()方法调用

2：sendHTTPService方法中添加代码把token信息方法请求头里：

3: 添加headerName，tokenValue 属性和getToken()方法：

代码：

**1. sendMessage方法添加getToken()**

public static function sendMessage(reqData:XML, projectName:String=""):void

{

//保存当前的请求

var command:String = reqData['command'].toString();

var commandData:Object = new Object();

commandData.reqData = reqData;

commandData.projectName = projectName;

messageCash[command] = commandData;

getToken();

if(false == \_reqSendFlag)

{

\_reqSendFlag = true;

getConfigParam();

getControllerAction();

getErrorCode();

}

//配置文件和错误码文件以及anction文件已加载后调用

if(\_configReqFlag && \_errorCodeFlag && \_controllerActionFlag)

{

getCasheMessage();

}

}

**2. sendHTTPService方法中添加代码把token信息方法请求头里：**

private static function sendHTTPService(reqData:XML, projectName:String=""):void

{

startTimer();

try

{

var command:String = reqData['command'].toString();

var sendObj:Object = new Object();

if('0' == \_isDebugPerformance)

{

reqData.reqSendTime = (new Date).time;

}

sendObj.req = reqData;

//避免浏览器缓存，产生数字

sendObj.rdm =new Date();

var http:HTTPService;

if(!\_https[command])

{

http = new HTTPService();

//http.url = \_url;

if("" != projectName)

{

var \_cemDrillXdrPageURL:String = getDrillXdrHttpInfo(projectName);

http.url = \_cemDrillXdrPageURL +"/"+ \_action;

}

else

{

http.url = \_url;

}

var controller:String = \_controllerAction[command];

if (controller)

{

http.url = http.url.replace("/" + \_action, controller);

}

if(\_useTestData == 'true')

{

http.url = Test\_GetURL(command);

}

http.method = 'POST';

http.resultFormat = "e4x";

if(headerName && "" != headerName && tokenValue && "" != tokenValue)

{

PrintManager.printErrorMessage(headerName+" : "+tokenValue);

var headObj:Object = new Object();

headObj[headerName] = tokenValue;

http.headers = headObj;

}

http.addEventListener(ResultEvent.RESULT, resultHandler);

http.addEventListener(FaultEvent.FAULT,faultHandler);

\_https[command] = http;

}

else

{

http = \_https[command];

}

//保存每个command最后一次发送的token

var token:AsyncToken = http.send(sendObj);

token.command = command;

token.param = sendObj;

//所有key都使用小写

\_asyncTokens[command.toLocaleLowerCase()] = token;

var req:RequestObject = \_reqObjs[command.toLocaleLowerCase()];

//判断当前界面是不是监控界面

var isShowLoadingPanel:Boolean = checkAppType();

if(isShowLoadingPanel && req.isModal)

{

LoadingManager.setLoadingPanel(command.toLocaleLowerCase(), false,\_isDebug);

}

if(\_isDebug == Enum.DeBugAll)

{

PrintManager.printErrorMessage(ResourceManager.getInstance().getString('languages', 'sendsRequest\_UIC')+':\r\n'+reqData.toString());

}

}

catch(e:Error)

{

var errMsg:String = ResourceManager.getInstance().getString('languages', 'sendsRequestFailed\_UIC');

if(\_isShowAlert == 'true')

{

SCAlert.show(errMsg+': '+e.message);

}

if(\_isDebug != Enum.NoDebug)

{

PrintManager.printErrorMessage(errMsg+': \n'+e.message);

}

}

}

**3: 添加headerName，tokenValue 属性和getToken()方法：**

private static var headerName:String;

private static var tokenValue:String;

private static function getToken():void

{

var param:Object = ExternalInterface.call("getToken");

if(param)

{

headerName = param.key;

tokenValue = param.value;

}

}

Flex添加token参数的效果



