# 平台监控

## index.jsp

E:\workspace\yh\OpenBridge-monitor\ob-monitor-web\src\main\

resources\META-INF\resources\monitor\architecture\index.jsp

### 获取配置中的数据和节点数据

${ WEB\_APP\_PATH }/monitor/architecture/graph.do

${ WEB\_APP\_PATH }/monitor/architecture/node/status.do

**var** def = $.ajax({

'url' : '${ WEB\_APP\_PATH }/monitor/architecture/graph.do',

'type' : 'POST',

'cache' : **false**,

'dataType' : 'json',

'data' : {

'business\_type' : 'all'

}

});

def['done'](**function**(msg){

**var** data = msg.data;

**var** m = **new** Graph({

'id' : 'graph-area',

'data' : data,

'webapp' : '${ WEB\_APP\_PATH }',

'status\_url' : '${ WEB\_APP\_PATH }/monitor/architecture/node/status.do',

'textClick' : **function**(node,pro,msg){

**var** nodeurl=pro.url;

**var** dialog = common.dialogIframe("节点信息","${ WEB\_APP\_PATH }/monitor/resources/node/info?nodeurl="+nodeurl,1100,300,**function**(){

dialog.close();

});

}

});

});

# MonitorArchitectureController

E:\workspace\yh\OpenBridge-monitor\ob-monitor-web\src\main\java\

com\harmazing\openbridge\monitor\controller\MonitorArchitectureController.java

## /monitor/architecture/graph

获取所有的图形数据

## /monitor/architecture/node/status

@RequestMapping("/node/status")

@ResponseBody

**public** JsonResponse nodeStatus(HttpServletRequest request, HttpServletResponse response, GraphParam param) {

iArchitectureService.getAppAndNodeStatus(param);

}

根据图形数据分别进行httpclient或者tcp socket的判断

E:\workspace\yh\OpenBridge-monitor\ob-monitor-web\src\main\java\

com\harmazing\openbridge\monitor\service\impl\MonitorArchitectureServiceImpl.java

**public** Map<String, Object> getAppAndNodeStatus(GraphParam param) {

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

**if** ("http".equals(param.getProtocol().toLowerCase())) {

httpGet(url);

result.put("msg", "成功");

} **else** **if** ("tcp".equals(param.getProtocol().toLowerCase())) {

socket = **new** Socket(is[0], Integer.*parseInt*(is[1]));

socket.setSoTimeout(1000);

result.put("msg", "成功");

}

**return** result;

}

# MonitorArchitectureServiceImpl

E:\workspace\yh\OpenBridge-monitor\ob-monitor-web\src\main\java\

com\harmazing\openbridge\monitor\service\impl\MonitorArchitectureServiceImpl.java

## parseArchitecture节点

} **else** **if** ("children".equals(nodeName)) {

parseChildren((Element) item, graphVo, document, userId);

}

### parseChildren

**private** **void** parseChildren(Element root, GraphVo result, Document document, String userId) {

} **else** **if** (item.getNodeName().equals(***METHOD***)) {

result.getChildren().addAll(parseMethod((Element) item, document, userId));

}

}

### 使用反射调用获取节点数据

<architecture name=*"应用访问代理"*>

<properties>

<property key=*"protocol"*>tcp</property>

</properties>

<children>

<method class=*"com.harmazing.openbridge.monitor.service.impl.MonitorArchitectureServiceImpl"* method=*"getPaasProxys"*></method>

</children>

</architecture>

<architecture code=*"paasos.resRestfulUrl"* name=*"Paas资源"*>

<properties>

<property key=*"protocol"*>http</property>

</properties>

<children>

<method class=*"com.harmazing.openbridge.monitor.service.impl.MonitorArchitectureServiceImpl"* method=*"getPaasResources"*></method>

</children>

</architecture>

<method class=*"com.harmazing.openbridge.monitor.service.impl.MonitorArchitectureServiceImpl"* method=*"getPaas"*></method>

<architecture code=*"paasos.k8s.test.apiserver"* name=*"Kubernetes测试"*>

<properties>

<property key=*"protocol"*>http</property>

</properties>

<children>

<method class=*"com.harmazing.openbridge.monitor.service.impl.MonitorArchitectureServiceImpl"* method=*"getPaasNodes"*></method>

</children>

</architecture>

<architecture code=*"paasos.k8s.live.apiserver"* name=*"Kubernetes生产"*>

<properties>

<property key=*"protocol"*>http</property>

</properties>

<children>

<method class=*"com.harmazing.openbridge.monitor.service.impl.MonitorArchitectureServiceImpl"* method=*"getPaasNodes"*></method>

</children>

</architecture>

### 更新组件监控

判断domain\_ip是否存在对应的domain\_key中，如果没有则新增。

@[Select](eclipse-javadoc:%E2%98%82=ob-monitor-web/C:%5C/Users%5C/dell%5C/.m2%5C/repository%5C/org%5C/mybatis%5C/mybatis%5C/3.3.0%5C/mybatis-3.3.0.jar%3Corg.apache.ibatis.annotations(Select.class%E2%98%83Select)([value](eclipse-javadoc:%E2%98%82=ob-monitor-web/C:%5C/Users%5C/dell%5C/.m2%5C/repository%5C/org%5C/mybatis%5C/mybatis%5C/3.3.0%5C/mybatis-3.3.0.jar%3Corg.apache.ibatis.annotations(Select.class%E2%98%83Select~value)={"SELECT \* FROM domain\_ip WHERE domain\_key = #{domainKey} "})

**private** GraphVo parseArchitecture(Element root, Document document, String userId) {

Domain domain = **new** Domain();

domain.setDomainName(ConfigUtil.*getConfigString*(code));

domain.setDomainKey(code);

Domain domainOld = domainMapper.findByKey(code);

**if** (domainOld != **null**) {

**if** (!domain.getDomainName().equals(domainOld.getDomainName())) {

domainMapper.updateName(domain);

}

} **else** {

domainMapper.insert(domain);

}

**return** graphVo;

}

#### 异步执行新增组件监控

ExecutorService exec = Executors.*newCachedThreadPool*();

//不论更新或者新增都触发组件监控，此时domain已经生成了id

**if**(triggerFlag){

*logger*.debug("更新或者新增组件监控"+domain);

exec.execute(()->{

**try** {

domainService.triggerSave(domain);

} **catch** (MonitorException e) {

*logger*.error("保存组件监控出错",e);

}

});

}

#### asynTriggerSave

**private** **void** asynTriggerSave(String code, GraphVo graphVo) {

// 不论更新或者新增都触发组件监控，此时domain已经生成了id

*exec*.execute(() -> {

**try** {

ConfigUtil.*loadPropertiesOnStartup*();// 刷新配置

Domain domain = **new** Domain();

domain.setDomainName(ConfigUtil.*getConfigString*(code));

domain.setResume(graphVo.getName());

domain.setDomainKey(code);

Object protocol = graphVo.getProperty().get("protocol");

**if** (!ObjectUtils.*isEmpty*(protocol)) {

domain.setProtocol(protocol.toString());

}

triggerSave(code, domain);

} **catch** (Exception e) {

*logger*.error("保存组件监控出错", e);

}

});

}

#### OS更新配置后刷新monitor中的配置

Os更新配置后刷新monitor中的配置。刷新后，ConfigUtil保存的值是最新的，构建新的Domain

graphVo.getProperty().put("url", ConfigUtil.*getConfigString*(code));

*logger*.debug("获取"+code+"配置");

ConfigUtil.*loadPropertiesOnStartup*();//刷新配置

Domain domain = **new** Domain();

domain.setDomainName(ConfigUtil.*getConfigString*(code));

domain.setDomainKey(code);

Domain domainOld = domainMapper.findByKey(code);

**boolean** triggerFlag = **false**;

**if** (domainOld != **null**) {

**if** (!domain.getDomainName().equals(domainOld.getDomainName())) {

domain.setId(domainOld.getId());

domainMapper.updateName(domain);

triggerFlag=**true**;

}

} **else** {

domainMapper.insert(domain);

triggerFlag=**true**;

}

ExecutorService exec = Executors.*newCachedThreadPool*();

//不论更新或者新增都触发组件监控，此时domain已经生成了id

**if**(triggerFlag){

*logger*.debug("更新或者新增组件监控"+domain);

exec.execute(()->{

**try** {

domainService.triggerSave(domain);

} **catch** (MonitorException e) {

*logger*.error("保存组件监控出错",e);

}

});

}

## “Kubernetes测试”节点获取

### architecture.xml

E:\workspace\yh\OpenBridge-monitor\ob-monitor-web\src\main\resources\META-INF\architecture\architecture.xml

<architecture code=*"paasos.k8s.test.apiserver"* name=*"Kubernetes测试"*>

<properties>

<property key=*"protocol"*>http</property>

</properties>

<children>

<method class=*"com.harmazing.openbridge.monitor.service.impl.MonitorArchitectureServiceImpl"* method=*"getPaasNodes"*></method>

</children>

</architecture>

### parserXml

**private** GraphVo parserXml(String bussiness\_type, String userId) {

GraphVo result = parseArchitecture(root, document, userId);

}

### parseArchitecture

**private** GraphVo parseArchitecture(Element root, Document document, String userId) {

**if** ("properties".equals(nodeName)) {

parseProperties((Element) item, graphVo, document, userId);

} **else** **if** ("children".equals(nodeName)) {

parseChildren((Element) item, graphVo, document, userId);

}

}

### parseChildren

**private** **void** parseChildren(Element root, GraphVo result, Document document, String userId) {

} **else** **if** (item.getNodeName().equals(***METHOD***)) {

result.getChildren().addAll(parseMethod((Element) item, document, userId));

}

### parseMethod

**private** List<GraphVo> parseMethod(Element root, Document document, String userId) {

String classname = root.getAttribute("class");

String method = root.getAttribute(***METHOD***);

**try** {

Class clazz = Class.*forName*(classname);

Object o = SpringUtil.*getBean*(clazz);

Method m = clazz.getMethod(method, **new** Class[] { String.**class** });

Object r = m.invoke(o, **new** Object[] { userId });

**if** (r == **null**) {

**return** **new** ArrayList<GraphVo>();

}

**return** (List<GraphVo>) r;

**return** **new** ArrayList<GraphVo>();

}

## getPaasNodes

@Override

**public** List<GraphVo> getPaasNodes(String userId) {

// io.fabric8.kubernetes.api.model.NodeList nl =

// KubernetesUtil.getNodes();

List<GraphVo> li = **new** ArrayList<GraphVo>();

**for** (io.fabric8.kubernetes.api.model.NodeList nl : KubernetesUtil.*getNodeMaps*().values()) {

**for** (io.fabric8.kubernetes.api.model.Node h : nl.getItems()) {

GraphVo v = **new** GraphVo();

v.setName(h.getMetadata().getName());

List<NodeAddress> addresses = h.getStatus().getAddresses();

String ip = **null**;

**for** (NodeAddress a : addresses) {

ip = a.getAddress();

}

v.getProperty().put("url", ip + ":4194");

v.getProperty().put("protocol", "tcp");

t.add(v);

}

li.addAll(t);

}

**return** li;

}

### 新增组件监控

这里只考虑了新增的情况，修改的情况没有考虑

List<Domain> domains = domainMapper.findByHostPort(ip, "4194");

**if**(CollectionUtils.*isEmpty*(domains)){

**try** {

Domain domain = **new** Domain(***K8S\_NODE***+nodeNum,ip,"4194",***K8S\_NODE***+nodeNum,***K8S\_NODE***+nodeNum,"tcp");

domainMapper.insert(domain);

domainService.save(domain);

nodeNum++;

} **catch** (MonitorException e) {

*logger*.error("保存组件监控出错",e);

}

}

## parseProperties

**if** ("properties".equals(nodeName)) {

parseProperties((Element) item, graphVo, document, userId);

} **else** **if** ("children".equals(nodeName)) {

parseChildren((Element) item, graphVo, document, userId);

}

仅解析property字段，而property字段目前仅包含protocol

E:\workspace\yh\OpenBridge-monitor\ob-monitor-web\src\main\

resources\META-INF\architecture\architecture.xml

<architecture code=*"api.svn.url"* name=*"Subversion"* cluster=*"true"*>

<properties>

<property key=*"protocol"*>http</property>

</properties>

</architecture>

**private** **void** parseProperties(Element root, GraphVo result, Document document, String userId) {

**if** (root == **null** || root.getChildNodes().getLength() == 0) {

**return**;

}

NodeList childNodes = root.getChildNodes();

**for** (**int** i = 0; i < childNodes.getLength(); i++) {

Node item = childNodes.item(i);

**if** (Node.***ELEMENT\_NODE*** != item.getNodeType() || !item.getNodeName().equals("property")) {

**continue**;

}

Element el = (Element) item;

String k = el.getAttribute("key");

String v = el.getAttribute("value");

**if** (StringUtils.*isEmpty*(v)) {

v = el.getChildNodes().item(0).getNodeValue();

}

result.getProperty().put(k, v);

}

}

## parseChildren

### 递归调用parseArchitecture

parseChildren会递归调用parseArchitecture

**if** ("properties".equals(nodeName)) {

parseProperties((Element) item, graphVo, document, userId);

} **else** **if** ("children".equals(nodeName)) {

parseChildren((Element) item, graphVo, document, userId);

}

**private** **void** parseChildren(Element root, GraphVo result, Document document, String userId) {

**if** (item.getNodeName().equals("architecture")) {

result.getChildren().add(parseArchitecture((Element) item, document, userId));

}

}

# SQL

## 修改表结构

ALTER TABLE

falcon\_portal.domain\_ip ADD (protocol VARCHAR(200));

#自增domain\_ip.id

ALTER TABLE

falcon\_portal.domain\_ip MODIFY COLUMN id INT NOT NULL AUTO\_INCREMENT;

ALTER TABLE falcon\_portal.domain\_ip ADD column grp\_id int;

ALTER TABLE falcon\_portal.domain\_ip ADD (grp\_id int);

ALTER TABLE falcon\_portal.domain\_ip drop column grp\_id;

## 清理相关

select \* from grp where grp.grp\_type=200;

select \* from grp\_tpl;

select \* from tpl where tpl.type=200;

select \* from falcon\_portal.grp\_tpl where grp\_tpl.tpl\_id=758;

select \* from grp where grp.id=362;

select \* from grp\_user;

select \* from tpl where tpl.id in (758);

select \* from falcon\_portal.strategy where strategy.tpl\_id=758;

select \* from action where action.id=1780;

select \* from team where team.name='组件Logstash告警组';

select \* from falcon\_portal.rel\_team\_user where rel\_team\_user.tid=1927;

select \* from falcon\_portal.domain\_ip;

delete from grp where grp.grp\_type=200;

delete from tpl where tpl.type=200;

delete from falcon\_portal.domain\_ip;

delete from team where team.type=200;

## 发送邮件相关SQL

找到tpl后，找到对应action.id，然后根据action.uic查找team.id，最后根据team.id查询到rel\_team\_user的user\_id

select \* from grp where grp.grp\_type=200;

select \* from grp\_tpl;

select \* from tpl where tpl.type=200;

select \* from falcon\_portal.grp\_tpl where grp\_tpl.tpl\_id=758;

select \* from grp where grp.id=362;

select \* from grp\_user;

select \* from tpl where tpl.action\_id in (select id from action where action.uic='组件Logstash告警组');

select \* from falcon\_portal.strategy where strategy.tpl\_id=826;

select \* from action where action.id=1780;

select \* from action where action.uic='组件Logstash告警组';

select \* from team where team.name='组件Logstash告警组';

select \* from falcon\_portal.rel\_team\_user where rel\_team\_user.tid=1995;

select \* from falcon\_portal.domain\_ip;

select \* from falcon\_portal.domain\_ip where domain\_ip.domain\_key='192.168.0.197:4194';

#清理相关

delete from grp where grp.grp\_type=200;

delete from tpl where tpl.type=200;

delete from falcon\_portal.domain\_ip;

delete from team where team.type=200;

# index.jsp

E:\workspace\yh\OpenBridge-monitor\ob-monitor-web\src\main\

resources\META-INF\resources\monitor\architecture\index.jsp

从graph中获取数据

https://paas.dev.yihecloud.com/monitor/monitor/architecture/graph.do

构造Graph

**var** def = $.ajax({

'url' : '${ WEB\_APP\_PATH }/monitor/architecture/graph.do',

'type' : 'POST',

'cache' : **false**,

'dataType' : 'json',

'data' : {

'business\_type' : 'all'

}

});

def['done'](**function**(msg){

**var** data = msg.data;

**var** m = **new** Graph({

'id' : 'graph-area',

'data' : data,

'webapp' : '${ WEB\_APP\_PATH }',

'status\_url' : '${ WEB\_APP\_PATH }/monitor/architecture/node/status.do',

'afterChangeTop' : **function**(nodes){

**var** source = nodes[1][0];

source.setTop(source.getChildren()[0].getTop()+(source.getChildren()[source.getChildren().length-1].getTop()-source.getChildren()[0].getTop())/2);

},

}

## 定义errorClick

'errorClick' : **function**(node,pro,msg){

**var** html="<div style='width:300px;height:100px'><table style='width:100%;height:100%'>";

html+="<tr><td>"+"检测类型："+pro.protocol+"协议"+"</td></tr>";

**if**(pro.protocol=="tcp"){

**var** pa = "([0-9]{1,3}\\.[0-9]{1,3}\\.[0-9]{1,3}\\.[0-9]{1,3}:[0-9]{1,6})";

**var** url=pro.url;

**var** ma = url.match(pa);

**if**(ma!=**null**){

url = ma[0];

**var** is = url.split(":");

html+="<tr><td>"+"检测参数："+"url:"+is[0]+";端口:"+is[1]+"</td></tr>";

}

}

**if**(pro.protocol=="http"){

**if**(pro.url!=undefined){

html+="<tr><td>"+"检测参数："+pro.url+"</td></tr>";

}

}

**if**(msg!=**null**&&msg!=""){

html+="<tr><td>"+"检测结果："+msg+"</td></tr>";

}

**if**(pro.msg!=undefined){

html+="<tr><td>"+"检测结果："+pro.msg+"</td></tr>";

}

html+="</table></div>";

/\* common.tooltip(html,$("#"+$(this).attr('id')),{guide: 1,time: 2000,maxWidth:300}); \*/

**if**(pro.msg=="成功"||msg=="成功"){

common.alert(html,1);

}**else**{

common.alert(html);}

},

# architecture.js

E:\workspace\yh\OpenBridge-monitor\ob-monitor-web\src\main\

resources\META-INF\resources\assets\js\architecture.js

## 构造options.data中数据

**var** \_nodes = {};

**var** root = toObject(options.data,1,\_nodes,**null**);

h=0;

toRender(root);

## toObject

//迭代 转换成对象

**function** toObject(root,level,\_\_nodes,\_\_source){

**var** n = **new** Node(root,level,\_\_source);

**if**(\_\_nodes[level]==**null**){

\_\_nodes[level]= [];

}

n.setIndex(\_\_nodes[level].length);

\_\_nodes[level].push(n);

**if**(root.children==**null** || root.children.length ==0){

**return** n;

}

**for**(**var** i=0;i<root.children.length;i++){

**var** \_\_node = toObject(root.children[i],parseInt(level)+1,\_\_nodes,n);

n.addClidren(\_\_node);

}

**return** n;

}

# DomainController.java

E:\workspace\yh\OpenBridge-monitor\ob-monitor-web\src\main\java\

com\harmazing\openbridge\alarm\controller\DomainController.java

## setMemeberParam-Collectors.*joining*

将 list转换为两个字符串

**private** **void** setMemeberParam(HttpServletRequest request) {

String userId = ConfigUtil.*getConfigString*(***PAASOS\_MONITOR\_MAINTAINER***);

**if** (userId != **null**) {

List<SysUser> users = teamService.getUserByIds(userId.split(","));

request.setAttribute("memIds",

users != **null** ? users.stream().map(SysUser::getUserId).collect(Collectors.*joining*(";")) : "");

request.setAttribute("userNames",

users != **null** ? users.stream().map(SysUser::getUserName).collect(Collectors.*joining*(";")) : "");

}

}

# MonitorResourcesServiceImpl

E:\workspace\yh\OpenBridge-monitor\ob-monitor-web\src\main\java\

com\harmazing\openbridge\monitor\service\impl\MonitorResourcesServiceImpl.java

## saveHostPortTemplate监控进程端口

**public** Template saveHostPortTemplate(String userId, String type, String host,

String port, String grpName, String tplName, String teamName){

Group group;

Template template;

**try** {

group = insertHostPortGroup(userId, type, grpName);

saveGroupHostByIp(group.getId(), host);

template = insertHostPortTemplate(userId, type, tplName, port, group);

insertHostPortTeamAndAction(userId, type, teamName, template);

templateMapper.deleteUnusedComponentTpl();

} **catch** (Exception e) {

logger.error("新增HostPortTemplate出错",e);

template = **null**;

}

**return** template;

}

### 清理未使用的策略

templateMapper.deleteUnusedComponentTpl();

查询到未使用的组件监控策略

select \* from falcon\_portal.tpl where tpl.type=200 and tpl\_name not in

(select concat('组件',CONVERT(domain\_ip.resume,char),'策略') from falcon\_portal.domain\_ip where domain\_ip.resume is not null);

DELETE FROM falcon\_portal.tpl where tpl.type=200 and tpl\_name not in (select concat('组件',CONVERT(domain\_ip.resume,char),'策略') from falcon\_portal.domain\_ip where domain\_ip.resume is not null);

## save节点监控保存组员

**public** **void** save(String userId, String tplData, Group group, String users, String tplIds) {

teamMapper.deleteMember(Integer.*toString*(team.getId()));

**if** (StringUtil.*isNotNull*(users) && JSONArray.*parseArray*(users).size() > 0) {

JSONArray u = JSONArray.*parseArray*(users);

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

JSONObject m = u.getJSONObject(i);

**if** (StringUtil.*isNotNull*(m.get(***USER\_ID***).toString())) {

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

params.put(***T\_ID***, team.getId());

params.put(***USER\_ID***, m.get(***USER\_ID***).toString());

teamMapper.addMember(params);

}

}

}

}

## 查询告警日志

<mapper namespace=*"com.harmazing.openbridge.alarm.dao.AlarmLogMapper"*>

<select id=*"countByMetricAndTags"* resultType=*"java.util.HashMap"*>

select endpoint,tags,count(id) as countNum from alarm\_log where metric=#{metric} and tags in

<foreach item=*"item"* index=*"index"* collection=*"portList"* open=*"("*

separator=*","* close=*")"*>

#{item}

</foreach>

and endpoint in

<foreach item=*"item"* index=*"index"* collection=*"hostList"* open=*"("*

separator=*","* close=*")"*>

#{item}

</foreach>

group by tags,endpoint

</select>

</mapper>

# TeamMapper.xml

E:\workspace\yh\OpenBridge-monitor\ob-monitor-web\src\main\java\

com\harmazing\openbridge\alarm\dao\TeamMapper.xml

## getUserByIds

<select id=*"getUserByIds"* parameterType=*"java.util.List"* resultType=*"com.harmazing.openbridge.sys.user.model.SysUser"*>

select \* from sys\_user where user\_id in

<foreach item=*"item"* index=*"index"* collection=*"list"* open=*"("*

separator=*","* close=*")"*>

#{item}

</foreach>

</select>

## batchAddMember

<insert id=*"batchAddMember"* parameterType=*"java.util.List"*>

insert into rel\_team\_user(tid,uid,user\_id) values

<foreach item=*"item"* collection=*"list"* separator=*","*>

(#{item.tId}, #{item.tId}, #{item.userId})

</foreach>

</insert>

# list.jsp

E:\workspace\yh\OpenBridge-monitor\ob-monitor-web\src\main\resources\META-INF\resources\alarm\domain\list.jsp

## selectUser

**function** selectUser(){

**var** params = {};

params.query="all";

params.userIds = $("#memIds").val();

params.userNames = $("#memNames").html();

dialogUser('memIds','memNames',**function**(){

**var** userIds = $("#memIds").val().split(";");

**var** userNames = $("#memNames").html().split(";");

**var** str = '';

$.each(userIds,**function**(i,v){

str += "<option value='"+v+"'>"+userNames[i]+"</option>";

});

$("#memNames").html(str);

**var** url = '${WEB\_APP\_PATH}/domains/updateMaintainer?userId='+userIds;

$.get(url,**function**(json){

**if**(json.code==0){

common.tips("更新组件告警接收人成功！",1);

} **else** {

common.tips("更新组件告警接收人失败！",2);

}

});

},params);

}

选择好接收人后，更新出入框，并同时更新视图中对应的配置。Mysql的视图是支持修改的。

### frameElement

http://localhost:8080/monitor/sys/user/dialog.do?query=all&multiple=true

window.frameElement即为包含本页面的iframe或frame对象

var multiple="true";

var params = frameElement.params;

if(params == null){

params = {};

}

var ids = null;

if(params.userIds==null||$.trim(params.userIds) == ""){

ids = [];

}else{

ids = params.userIds.split(";");

}

var names = null;

if(params.userNames==null||$.trim(params.userNames) == ""){

names = [];

}else{

names = params.userNames.split(";");

}

## dialogUser

top该变更永远指分割窗口最高层次的浏览器窗口

function dialogUser(multiple,valId,textId,cb,params){

var url = WEB\_APP\_PATH+"/sys/user/dialog.do?query="+params.query+"&multiple="+multiple;

var dialog = top.common.dialogIframe("请选择接收人",url,650,600,null,['确定','取消'],{

btn1 :function(index,layero){

var iwin = layero.find("iframe").get(0).contentWindow;

if(iwin.dialogClose && typeof(iwin.dialogClose) === 'function'){

var obj = (iwin.dialogClose());

if(obj!=null&&obj.ids!=null){

var idObj = $("#"+valId);

var textObj = $("#"+textId);

if(multiple){

idObj.val(obj.ids.join(";"));

if(textObj.is(":input")){

textObj.val(obj.texts.join(";"));

}else{

textObj.html(obj.texts.join(";"));

}

}else{

for(var i=0;i<obj.ids.length;i++){

if(i==obj.ids.length-1){

idObj.val(obj.ids[i]);

if(textObj.is(":input")){

textObj.val(obj.texts[i]);

}else{

textObj.html(obj.texts[i]);

}

}

}

}

}

}

if(cb!=null)

cb();

},

btn2 :function(index,layero){

}

});

dialog.getIFrame().params = params;

}

### dialogClose

iwin.dialogClose()

http://localhost:8080/monitor/sys/user/dialog.do?query=all&multiple=true

function dialogClose(){

if(multiple){

var obj = {};

obj.ids = ids;

obj.texts = names;

return obj;

}else{

var obj = {};

obj.ids = [];

obj.texts = [];

var rows = $("#userTable").find("tr");

for(var i=0;i<rows.length;i++){

if(i==0){}

else{

var row = rows[i];

var chb = $(":checked",row);

if(chb && chb.length > 0){

obj.ids.push(chb.val());

obj.texts.push($(row).find(".userName").text());

}

}

}

return obj;

}

}

# common.js

common.dialogIframe = function(\_title,\_url,w,h,cb,buttons,btnCb){

var opt = {

title:\_title,

type: 2,

area: [w+'px', h+'px'],

fix: false, //不固定

maxmin: false,

btn:buttons,

content: \_url

};

$.extend(opt,btnCb);

var dialog = top.common.dialog(opt);

top.document.getElementById("layui-layer-iframe"+dialog.index+"").callback = cb;

top.document.getElementById("layui-layer-iframe"+dialog.index+"").dialog = dialog;

return dialog;

}

# 没有按钮的情况下，关闭iframe刷新父页面

## Layer.js

common.dialogIframeNoClose = **function**(\_title,\_url,w,h,cb,buttons,btnCb){

**var** opt = {

title:\_title,

type: 2,

area: [w+'px', h+'px'],

fix: **false**, //不固定

maxmin: **false**,

btn:buttons,

closeBtn:**false**,//屏蔽右上角的关闭按钮

content: \_url

};

$.extend(opt,btnCb);

**var** dialog = top.common.dialog(opt);

top.document.getElementById("layui-layer-iframe"+dialog.index+"").callback = cb;

top.document.getElementById("layui-layer-iframe"+dialog.index+"").dialog = dialog;

**return** dialog;

}

## alarm\domain\check.jsp

E:\workspace\yh\OpenBridge-monitor\ob-monitor-web\src\main\resources\

META-INF\resources\alarm\domain\check.jsp

**function** closeAndRefresh(){

**var** index = parent.layer.getFrameIndex(window.name); //获取窗口索引

parent.location.reload();

parent.layer.close(index);

}