ExecutorService exec;

public AssetsStatisticsServiceImpl() {

exec = Executors.newFixedThreadPool(8); //threadNum线程数

}

/\*\*

\* 多线程专用函数

\* \*\*/

private void assetThreadMore(List<String> sqlList){

try {

if (null != sqlList && sqlList.size() > 0) {

List<Callable<Integer>> tasks = new ArrayList<>();

Callable<Integer> task = null;

for (String string : sqlList) {

final String str = string;

task = new Callable<Integer>() {

@Override

public Integer call() throws Exception {

switchIndex(str);

return 1;

}

};

tasks.add(task);

}

exec.invokeAll(tasks);

//用于返回数据

}

}catch(Exception e){

log.error("--auditThreadMore-Exception-");

}

}

package com.vrv.icssas.assets.service.impl;

import com.vrv.icssas.admin.mapper.dict.DictAssetMapper;

import com.vrv.icssas.admin.model.dict.DictAssetType;

import com.vrv.icssas.assets.mapper.AssetInfoMapper;

import com.vrv.icssas.assets.model.AssetDeptidEvent;

import com.vrv.icssas.common.model.AssetInfo;

import com.vrv.icssas.assets.model.assetsWeb.AssetsStatistic;

import com.vrv.icssas.assets.model.assetsWeb.AssetsStatisticHandle;

import com.vrv.icssas.assets.model.assetsWeb.DeptidAssetCounts;

import com.vrv.icssas.assets.service.AssetsStatisticsService;

import com.vrv.icssas.assets.utils.AssetsOperUtils;

import com.vrv.icssas.common.mapper.DeptMapper;

import com.vrv.icssas.common.util.AssetTypeGroup;

import com.vrv.icssas.common.util.DateUtils;

import com.vrv.icssas.common.util.StringUtils;

import com.vrv.icssas.common.web.WebContent;

import com.vrv.icssas.common.web.model.Dept;

import com.vrv.icssas.event.mapper.EventMapper;

import com.vrv.icssas.event.model.IndexEvent;

import com.vrv.icssas.vulnerability.mapper.PatchMapper;

import com.vrv.icssas.vulnerability.model.IndexPatchEvent;

import com.vrv.icssas.vulnerability.model.PatchDeptidEvent;

import com.vrv.icssas.vulnerability.model.PatchMadeInChina;

import lombok.extern.slf4j.Slf4j;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.List;

import java.util.Map;

import java.util.concurrent.Callable;

import java.util.concurrent.ExecutorService;

import java.util.concurrent.Executors;

import static com.vrv.icssas.common.util.StringUtils.isNotEmpty;

import static com.vrv.icssas.common.web.WebConfigConstant.EVENT\_TABLE\_NAME\_PREFIX;

/\*\*

\* @Classname: icssas

\* @Date: 2021/8/19 10:29

\* @Author: zl

\* @Description:

\*/

@Service

@Slf4j

public class AssetsStatisticsServiceImpl implements AssetsStatisticsService {

ExecutorService exec;

public AssetsStatisticsServiceImpl() {

exec = Executors.newFixedThreadPool(8); //threadNum线程数

}

@Autowired

private DeptMapper deptMapper;

@Autowired

private AssetInfoMapper assetInfoMapper;

@Autowired

private EventMapper eventMapper;

@Autowired

private PatchMapper patchMapper;

@Autowired

private DictAssetMapper dictAssetMapper;

private List<Dept> deptList = null;

private Integer deptId = null;//当前组织id，null时表示未登录

private String deptIdMyAndSon = null;//首页的第一屏组织机构 type:1

private String deptIdSon = null;//尾部的表格 type:2

private String deptIdMyAll = null;//该账户下的所有组织id，多用于获取当前账户下的所有数据 type:3

private String currentDeptIdMyAll = null;

private String eventTodayTable = null;//事件今日表名

private String eventYesterdayTable = null;

private List<String> eventListTable = null;//事件所有表名

private List<AssetTypeGroup> typeGroupList = null;//组织分组类型

private AssetsStatisticHandle threadIndexHandle = new AssetsStatisticHandle();//线程处理结果

@Override

public AssetsStatistic assetsStatistic(String type, Integer id) {

deptList = deptMapper.getDeptList(new Dept());//获取所有组织机构

if (null != WebContent.INSTACE.getDeptId()) {

deptId = WebContent.INSTACE.getDeptId();//登录后

} else {

deptId = 0;//未登录

}

deptIdMyAndSon = StringUtils.deptidMore(deptList, deptId, 1);

deptIdSon = StringUtils.deptidMore(deptList, deptId, 2);

deptIdMyAll = StringUtils.deptidMore(deptList, deptId, 3);

if(null!=id){

currentDeptIdMyAll = StringUtils.deptidMore(deptList, id, 3);

}else{

currentDeptIdMyAll = deptIdMyAll;

}

String monthStr = DateUtils.getDateByMonth();

eventListTable = eventMapper.queryTableName(EVENT\_TABLE\_NAME\_PREFIX + monthStr);//当月所有事件表

if(null==eventListTable || eventListTable.size()==0){

eventListTable = null;

}

String todayDate = StringUtils.replaceSplit( DateUtils.getDateByString(),"-");

String tdEventTable = EVENT\_TABLE\_NAME\_PREFIX + todayDate;//今日事件表名

List<String> tdEvent = eventMapper.queryTableName(tdEventTable);

if(null!=tdEvent && tdEvent.size()>0){

eventTodayTable = tdEventTable;

}

String yesterdayDate = StringUtils.replaceSplit( DateUtils.getBeforeDay(),"-");

String ysEventTable = EVENT\_TABLE\_NAME\_PREFIX + yesterdayDate;//昨日事件表名

List<String> yesterdayEventTables = eventMapper.queryTableName(eventYesterdayTable);

if( null!=yesterdayEventTables && yesterdayEventTables.size()>0 ){

eventYesterdayTable = ysEventTable;

}

//资产分组类型

typeGroupList = StringUtils.typeGroupListOper();

//匹配底层sql请求列

List<String> sqlList = StringUtils.threadAssetTotal(type);

//开始多线程数据处理

assetThreadMore(sqlList);

//最终统待处理

AssetsStatistic ret = new AssetsStatistic();

if(StringUtils.isContain(type,"1") ) {//组织资产统计

List<Dept> deptList = threadIndexHandle.getDeptList();

List<AssetDeptidEvent> assetInfoList = threadIndexHandle.getAssetInfoList();

List<PatchDeptidEvent> assetInfoPatchList = threadIndexHandle.getAssetInfoPatchList();

Map<String, Object> depthList = AssetsOperUtils.depthList(deptList, assetInfoList, assetInfoPatchList);

ret.setDepthList(depthList);

}

if(StringUtils.isContain(type,"2") ) {//资产事件统计

List<DeptidAssetCounts> eventMonthTotal = threadIndexHandle.getEventDayTotal();

Map<String, Object> assetEventEchart = AssetsOperUtils.assetEventEchart(eventMonthTotal);

ret.setAssetEventEchart(assetEventEchart);

}

if(StringUtils.isContain(type,"3") ) {//资产类型统计

List<AssetInfo> assetTypeTotal = threadIndexHandle.getAssetTypeTotal();

Map<String, Object> assetTypeTotalMap = AssetsOperUtils.assetTypeTotal(assetTypeTotal);

ret.setAsseTypeStatistics(assetTypeTotalMap);

}

if(StringUtils.isContain(type,"4") ) {//系统分类资产统计

List<IndexPatchEvent> assetSysTotal = threadIndexHandle.getAssetSysTotal();

Map<String, Object> assetTypeTotalMap = AssetsOperUtils.sysTypeAssetStatistics(assetSysTotal);

ret.setSysTypeAssetStatistics(assetTypeTotalMap);

}

if(StringUtils.isContain(type,"5") ) {//厂商分布TOP10

List<AssetInfo> manufacturerTop10 = threadIndexHandle.getManufacturerTop10();

Map<String, Object> manufacturerTop10Map = AssetsOperUtils.manufacturerTop10Map(manufacturerTop10);

ret.setManufacturerTop10Map(manufacturerTop10Map);

}

if(StringUtils.isContain(type,"6") ) {//国产化分布

List<DictAssetType> madeInChinaStatisticsYes = threadIndexHandle.getMadeInChinaStatisticsYes();

List<DictAssetType> madeInChinaStatisticsNo = threadIndexHandle.getMadeInChinaStatisticsNo();

Map<String, Object> madeInChinaMap = AssetsOperUtils.madeInChinaMap(madeInChinaStatisticsYes, madeInChinaStatisticsNo);

ret.setMadeInChinaStatistics(madeInChinaMap);

}

if(StringUtils.isContain(type,"7") ) {//资产等级评分统计

List<AssetInfo> eventLevelDistributionList = threadIndexHandle.getEventLevelDistributionList();

Map<String, Object> madeInChinaMap = AssetsOperUtils.riskLevelMap(eventLevelDistributionList);

ret.setAssetLevelScore(madeInChinaMap);

}

if(StringUtils.isContain(type,"8") ) {//漏洞资产占比

List<PatchDeptidEvent> patchConts = threadIndexHandle.getPatchConts();

List<AssetDeptidEvent> assetInfoList = threadIndexHandle.getAssetInfoList();

Map<String, Object> patchAssetPer = AssetsOperUtils.patchAssetPer(patchConts, assetInfoList);

ret.setPatchAssetPer(patchAssetPer);

}

if(StringUtils.isContain(type,"9") ) {//资产系统漏洞分布

List<IndexPatchEvent> patchSecurityConts = threadIndexHandle.getPatchSecurityConts();

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

resultMap.put("msg", "资产系统漏洞分布");

resultMap.put("data",patchSecurityConts);

ret.setPatchAssetStatistics(resultMap);

}

if(StringUtils.isContain(type,"11") ) {//漏洞资产国产化分布

List<PatchMadeInChina> pathMadeInChinaYes = threadIndexHandle.getPathMadeInChinaYes();

List<PatchMadeInChina> pathMadeInChinaNo = threadIndexHandle.getPathMadeInChinaNo();

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

resultMap = AssetsOperUtils.madeInChinaEcharts(pathMadeInChinaYes, pathMadeInChinaNo);

ret.setPatchMadeInChinaStatistics(resultMap);

}

if(StringUtils.isContain(type,"11") ) {//漏洞操作系统统计

List<IndexPatchEvent> pathOs = threadIndexHandle.getPathOs();

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

resultMap.put("msg", "漏洞操作系统统计");

resultMap.put("data",pathOs);

ret.setPathOs(resultMap);

}

if(StringUtils.isContain(type,"13") ) {//资产系统统计

List<IndexPatchEvent> assetSys = threadIndexHandle.getAssetSys();

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

resultMap.put("msg", "资产系统统计");

resultMap.put("data",assetSys);

ret.setAssetSys(resultMap);

}

return ret;

}

/\*\*

\* 多线程专用函数

\* \*\*/

private void assetThreadMore(List<String> sqlList){

try {

if (null != sqlList && sqlList.size() > 0) {

List<Callable<Integer>> tasks = new ArrayList<>();

Callable<Integer> task = null;

for (String string : sqlList) {

final String str = string;

task = new Callable<Integer>() {

@Override

public Integer call() throws Exception {

switchIndex(str);

return 1;

}

};

tasks.add(task);

}

exec.invokeAll(tasks);

//用于返回数据

}

}catch(Exception e){

log.error("--auditThreadMore-Exception-");

}

}

private void switchIndex(String type){

switch(type){

//组织区域--主体

case "1\_01"://当前账户子级组织

List<Dept> ret1\_01 = new ArrayList<Dept>();

try {

if(isNotEmpty(deptIdSon)){

ret1\_01 = deptMapper.deptList(deptIdSon);

}

}catch(Exception e){

log.error("deptMapper.deptList---error");

}

threadIndexHandle.setDeptList(ret1\_01);

break;

case "1\_02"://组织对应资产统计

List<AssetDeptidEvent> ret1\_02 = new ArrayList<AssetDeptidEvent>();

try {

if(isNotEmpty(deptIdMyAll)){

ret1\_02 = assetInfoMapper.assetsConts(deptIdMyAll);

}

}catch(Exception e){

log.error("assetInfoMapper.assetsConts---error");

}

threadIndexHandle.setAssetInfoList(ret1\_02);

break;

case "1\_03":///组织对应漏洞资产统计

List<PatchDeptidEvent> ret1\_03 = new ArrayList<PatchDeptidEvent>();

try {

if(isNotEmpty(deptIdMyAll)) {

ret1\_03 = patchMapper.patchAssetConts(deptIdMyAll);

}

}catch(Exception e){

log.error("patchMapper.patchConts---error");

}

threadIndexHandle.setAssetInfoPatchList(ret1\_03);

break;

case "2\_01"://事件月份统计(当月)

List<DeptidAssetCounts> ret2\_02 = new ArrayList<DeptidAssetCounts>();

try {

if(isNotEmpty(currentDeptIdMyAll) && null!=eventListTable && eventListTable.size()>0 ) {

ret2\_02 = assetInfoMapper.eventMonthStatistics(currentDeptIdMyAll, eventListTable);//事件表只统计当月的

}

}catch(Exception e){

log.error("assetInfoMapper.eventMonthStatistics---error");

}

threadIndexHandle.setEventDayTotal(ret2\_02);

break;

case "3\_01"://设备类型

List<AssetInfo> ret3\_01 = new ArrayList<AssetInfo>();

try {

if(isNotEmpty(deptIdMyAll)) {

ret3\_01 = assetInfoMapper.listAssets(deptIdMyAll);

}

}catch(Exception e){

log.error("assetInfoMapper.listAssets---error");

}

threadIndexHandle.setAssetTypeTotal(ret3\_01);

break;

case "4\_01"://系统分类资产统计

List<IndexPatchEvent> ret4\_01 = new ArrayList<IndexPatchEvent>();

try {

if(isNotEmpty(deptIdMyAll)) {

ret4\_01 = patchMapper.pathInfoSystem(deptIdMyAll);

}

}catch(Exception e){

log.error("patchMapper.pathInfoSystem---error");

}

threadIndexHandle.setAssetSysTotal(ret4\_01);

break;

case "5\_01"://厂商分布TOP10

List<AssetInfo> ret5\_01 = new ArrayList<AssetInfo>();

try {

if(isNotEmpty(deptIdMyAll)) {

ret5\_01 = assetInfoMapper.manufacturerTop10All(deptIdMyAll);

}

}catch(Exception e){

log.error("assetInfoMapper.manufacturerTop10All---error");

}

threadIndexHandle.setManufacturerTop10(ret5\_01);

break;

case "6\_01"://设备国产化-国产

List<DictAssetType> ret6\_01 = new ArrayList<DictAssetType>();

try {

if(isNotEmpty(deptIdMyAll)) {

ret6\_01 = dictAssetMapper.madeInChinaStatistics(deptIdMyAll, 1);

}

}catch(Exception e){

log.error("dictAssetMapper.madeInChinaStatistics---error");

}

threadIndexHandle.setMadeInChinaStatisticsYes(ret6\_01);

break;

case "6\_02"://设备国产化-进口

List<DictAssetType> ret6\_02 = new ArrayList<DictAssetType>();

try {

if(isNotEmpty(deptIdMyAll)) {

ret6\_02 = dictAssetMapper.madeInChinaStatistics(deptIdMyAll, 0);

}

}catch(Exception e){

log.error("dictAssetMapper.madeInChinaStatistics---error");

}

threadIndexHandle.setMadeInChinaStatisticsNo(ret6\_02);

break;

case "7\_01"://资产等级评分

List<AssetInfo> ret7\_01 = new ArrayList<AssetInfo>();

try {

if(isNotEmpty(deptIdMyAll)) {

ret7\_01 = assetInfoMapper.assetsRiskLevel(deptIdMyAll);

}

}catch(Exception e){

log.error("assetInfoMapper.assetsRiskLevel---error");

}

threadIndexHandle.setEventLevelDistributionList(ret7\_01);

break;

case "8\_01"://资产数量

//对应1\_02

break;

case "8\_02"://漏洞资产总量

List<PatchDeptidEvent> ret8\_02 = new ArrayList<PatchDeptidEvent>();

try {

if(isNotEmpty(deptIdMyAll)) {

ret8\_02 = patchMapper.patchAssetConts(deptIdMyAll);

}

}catch(Exception e){

log.error("patchMapper.patchConts---error");

}

threadIndexHandle.setPatchConts(ret8\_02);

break;

case "9\_01"://资产系统漏洞分布

List<IndexPatchEvent> ret9\_01 = new ArrayList<IndexPatchEvent>();

try {

if(isNotEmpty(deptIdMyAll)) {

ret9\_01 = patchMapper.pathInfoSystem(deptIdMyAll);

}

}catch(Exception e){

log.error("patchMapper.pathInfoSystem---error");

}

threadIndexHandle.setPatchSecurityConts(ret9\_01);

break;

case "11\_01"://漏洞资产国产化分布-国产

List<PatchMadeInChina> ret11\_01 = new ArrayList<PatchMadeInChina>();

try {

if(isNotEmpty(deptIdMyAll)) {

ret11\_01 = patchMapper.pathMadeInChina(deptIdMyAll, 1);

}

}catch(Exception e){

log.error("patchMapper.pathMadeInChina---error");

}

threadIndexHandle.setPathMadeInChinaYes(ret11\_01);

break;

case "11\_02"://漏洞资产国产化分布-非国产

List<PatchMadeInChina> ret11\_02 = new ArrayList<PatchMadeInChina>();

try {

if(isNotEmpty(deptIdMyAll)) {

ret11\_02 = patchMapper.pathMadeInChina(deptIdMyAll, 0);

}

}catch(Exception e){

log.error("patchMapper.pathMadeInChina---error");

}

threadIndexHandle.setPathMadeInChinaNo(ret11\_02);

break;

case "12\_01"://漏洞操作系统统计

List<IndexPatchEvent> ret12\_01 = new ArrayList<IndexPatchEvent>();

try {

if(isNotEmpty(deptIdMyAll)) {

ret12\_01 = patchMapper.pathOs(deptIdMyAll);

}

}catch(Exception e){

log.error("patchMapper.pathOs---error");

}

threadIndexHandle.setPathOs(ret12\_01);

break;

case "13\_01"://资产系统统计

List<IndexPatchEvent> ret13 = new ArrayList<IndexPatchEvent>();

try {

if(isNotEmpty(deptIdMyAll)) {

ret13 = patchMapper.getAssetSys(deptIdMyAll);

}

}catch(Exception e){

log.error("patchMapper.getAssetSys---error");

}

threadIndexHandle.setAssetSys(ret13);

break;

default:

break;

}

}

}