package com.yingchong.service.data\_service.mapper;

import org.apache.ibatis.annotations.Param;

public class MyReligionTimeProvider {

public String selectPeopleVisitTimes(

@Param("user") String user,

@Param("startDate") String startDate,

@Param("endDate") String endDate

) {

if(user != null && !user.isEmpty())

return "select religion\_name religionName,url url,web\_title title,terminal\_type terminal,visite\_time visitTime,host\_ip srcIP,domain\_name domain,dns DNS, terminal\_detail terminalDetail,\n" +

"det\_ip tarIP,host\_port srcPort ,protocol protocol,mac\_address MAC,count(\*) visitTimes from religion\_times\n" +

"where times\_date >= #{startDate} and times\_date <= #{endDate} and host\_ip like concat('%',#{user},'%') " +

"GROUP BY host\_ip order by visitTimes desc ";

else

return "select religion\_name religionName,url url,web\_title title,terminal\_type terminal,visite\_time visitTime,host\_ip srcIP,domain\_name domain,dns DNS, terminal\_detail terminalDetail,\n" +

"det\_ip tarIP,host\_port srcPort ,protocol protocol,mac\_address MAC,count(\*) visitTimes from religion\_times\n" +

"where times\_date >= #{startDate} and times\_date <= #{endDate} " +

"GROUP BY host\_ip order by visitTimes desc ";

}

}

package com.yingchong.service.data\_service.service;

import com.yingchong.service.data\_service.BizBean.ResponseBean;

import com.yingchong.service.data\_service.BizBean.biz\_app.BizAppBean;

import com.yingchong.service.data\_service.BizBean.biz\_flux.BizDataBean;

import com.yingchong.service.data\_service.BizBean.biz\_interTime.BizInterBean;

import com.yingchong.service.data\_service.mapper.MyAppMapper;

import com.yingchong.service.data\_service.mapper.MyFluxMapper;

import com.yingchong.service.data\_service.mapper.MyInterMapper;

import com.yingchong.service.data\_service.mybatis.mapper.AppFluxSortMapper;

import com.yingchong.service.data\_service.mybatis.mapper.FluxResultMapper;

import com.yingchong.service.data\_service.mybatis.mapper.OnlineTimeMapper;

import com.yingchong.service.data\_service.mybatis.model.\*;

import com.yingchong.service.data\_service.utils.CodeUtils;

import com.yingchong.service.data\_service.utils.DateUtil;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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

import org.springframework.stereotype.Service;

import java.text.DecimalFormat;

import java.util.ArrayList;

import java.util.Date;

import java.util.List;

@Service

public class IndexService {

private static final Logger logger = LoggerFactory.getLogger(IndexService.class);

@Autowired

private MyFluxMapper myFluxMapper;

@Autowired

private MyInterMapper myInterMapper;

@Autowired

private MyAppMapper myAppMapper;

@Autowired

private FluxResultMapper fluxResultMapper;

@Autowired

private OnlineTimeMapper onlineTimeMapper;

@Autowired

private AppFluxSortMapper appFluxSortMapper;

/\*\*

\* @param startDate 开始时间 2019-05-01

\* @param endData 结束时间 2019-05-15

\* @return

\*/

public ResponseBean<List<BizDataBean>> Flux\_1(String startDate, String endData) {

int days = DateUtil.differentDays(DateUtil.StringToDate(startDate, "yyyy-MM-dd"), DateUtil.StringToDate(endData, "yyyy-MM-dd"));

List<BizDataBean> dataList = new ArrayList<>();

for (int i = 0; i <= days; i++) {

//String startDate1 = startDate.replaceAll("-", "");

Date date = DateUtil.addDay(DateUtil.StringToDate(startDate, "yyyy-MM-dd"), i);

String startDate1 = DateUtil.formatDateStrToString(DateUtil.formatDateToStr(date, "yyyy-MM-dd HH:mm:ss"));

String param = startDate1.replaceAll("-", "");

List<BizDataBean> bizDataBeans = myFluxMapper.selectFlux(param + "\_flux");

BizDataBean bizDataBean = bizDataBeans.get(0);

bizDataBean.setDate(startDate1);

dataList.add(bizDataBean);

}

return new ResponseBean<>(dataList);

}

/\*\*

\* 查询指定日期的原始数据

\*

\* @param date 日期 2019-05-17

\* @return 1

\*/

public List<BizDataBean> Flux(String date) {

String param = date.replaceAll("-", "");

List<BizDataBean> bizDataBeans = myFluxMapper.selectFlux(param + "\_flux");

for (BizDataBean bizDataBean : bizDataBeans) {

bizDataBean.setDate(date);

}

return bizDataBeans;

}

/\*\*

\* 每日同步数据

\*

\* @param date

\* @return

\*/

public boolean insertFluxResult(String date) {

FluxResultExample example = new FluxResultExample();

example.createCriteria().andFluxDateEqualTo(DateUtil.StringToDate(date, "yyyy-MM-dd"));

List<FluxResult> fluxResults = fluxResultMapper.selectByExample(example);

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

logger.info("数据已经插入,不再重复插入");

return true;

}

List<BizDataBean> flux = this.Flux(date);

try {

for (BizDataBean bizDataBean : flux) {

FluxResult fr = new FluxResult();

fr.setUpload(Double.parseDouble(bizDataBean.getUploadFlux()));

fr.setDownload(Double.parseDouble(bizDataBean.getDownFlux()));

fr.setFluxDate(DateUtil.formatStringToDate(bizDataBean.getDate(), "yyyy-MM-dd"));

Date nowDate = new Date();

fr.setCreateTime(nowDate);

fr.setUpdateTime(nowDate);

int insert = fluxResultMapper.insert(fr);

}

} catch (Exception e) {

logger.error("insertFluxResult error:", e);

return false;

}

return true;

}

/\*\*

\* 查询结果集,返回给前端数据

\*

\* @param startDate 开始时间 2019-05-01

\* @param endData 结束时间 2019-05-15

\* @return

\*/

public ResponseBean<List<BizDataBean>> Flux(String startDate, String endData) {

FluxResultExample example = new FluxResultExample();

example.createCriteria().andFluxDateBetween(DateUtil.StringToDate(startDate, "yyyy-MM-dd"), DateUtil.StringToDate(endData, "yyyy-MM-dd"));

example.setOrderByClause("flux\_date");

List<FluxResult> fluxResults = fluxResultMapper.selectByExample(example);

List<BizDataBean> resultData = new ArrayList<>();

for (FluxResult fluxResult : fluxResults) {

BizDataBean dataBean = new BizDataBean();

dataBean.setDate(DateUtil.formatDateToStr(fluxResult.getFluxDate(), "yyyy-MM-dd"));

dataBean.setDownFlux(fluxResult.getDownload().toString());

dataBean.setUploadFlux(fluxResult.getUpload().toString());

resultData.add(dataBean);

}

return new ResponseBean<>(resultData);

}

//上网时长,查询指定日期的原始数据

public List<BizInterBean> Inter(String date) {

String param = date.replaceAll("-", "");

List<BizInterBean> bizInterBeans = myInterMapper.selectInter(param + "\_time\_count");

for (BizInterBean bizInterBean : bizInterBeans) {

bizInterBean.setDate(date);

}

return bizInterBeans;

}

//上网时长,每日同步数据

public boolean insertOnlineTime(String date) {

OnlineTimeExample example = new OnlineTimeExample();

example.createCriteria().andResultDateEqualTo(DateUtil.StringToDate(date, "yyyy-MM-dd"));

List<OnlineTime> onlineTimes = onlineTimeMapper.selectByExample(example);

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

logger.info("数据已经插入,不再重复插入");

return true;

}

List<BizInterBean> inter = this.Inter(date);

try {

for (BizInterBean bizInterBean : inter) {

OnlineTime ot = new OnlineTime();

ot.setOnlineTime(bizInterBean.getAvgTime());

ot.setResultDate(DateUtil.formatStringToDate(bizInterBean.getDate(), "yyyy-MM-dd"));

Date nowDate = new Date();

ot.setCreateTime(nowDate);

ot.setUpdateTime(nowDate);

int insert = onlineTimeMapper.insert(ot);

}

} catch (Exception e) {

logger.error("insertOnlineTime error:", e);

return false;

}

return true;

}

//上网时长,查询结果集,返回给前端数据

public ResponseBean<List<BizInterBean>> Inter(String startDate, String endData) {

OnlineTimeExample example = new OnlineTimeExample();

example.createCriteria().andResultDateBetween(DateUtil.StringToDate(startDate, "yyyy-MM-dd"), DateUtil.StringToDate(endData, "yyyy-MM-dd"));

example.setOrderByClause("result\_date");

List<OnlineTime> onlineTimes = onlineTimeMapper.selectByExample(example);

List<BizInterBean> resultInter = new ArrayList<>();

for (OnlineTime onlineTime : onlineTimes) {

BizInterBean interBean = new BizInterBean();

interBean.setDate(DateUtil.formatDateToStr(onlineTime.getResultDate(), "yyyy-MM-dd"));

interBean.setAvgTime(onlineTime.getOnlineTime());

resultInter.add(interBean);

}

return new ResponseBean<>(resultInter);

}

//应用流量，查询指定日期的原始数据

public List<BizAppBean> App(String date) {

String param = date.replaceAll("-", "");

List<BizAppBean> bizAppBeans = myAppMapper.selectApp(param + "\_flux");

for (BizAppBean bizAppBean : bizAppBeans) {

bizAppBean.setDate(date);

}

return bizAppBeans;

}

//应用流量，每日同步数据

public boolean insertAppFluxSort(String date) {

AppFluxSortExample example = new AppFluxSortExample();

example.createCriteria().andFluxDateEqualTo(date);

List<AppFluxSort> appFluxSorts = appFluxSortMapper.selectByExample(example);

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

logger.info("数据已经插入,不再重复插入");

return true;

}

List<BizAppBean> app = this.App(date);

try {

for (BizAppBean bizAppBean : app) {

AppFluxSort afs = new AppFluxSort();

afs.setAppName(bizAppBean.getAppName());

afs.setFlux(bizAppBean.getFlux());

afs.setFluxPercentage(bizAppBean.getFluxPercentage());

afs.setFluxDate(date);

Date nowDate = new Date();

afs.setCreateTime(nowDate);

afs.setUpdateTime(nowDate);

int insert = appFluxSortMapper.insert(afs);

}

} catch (Exception e) {

logger.error("insertAppFluxSort error:", e);

return false;

}

return true;

}

//应用流量,查询结果集,返回给前端数据

public ResponseBean<List<BizAppBean>> App(String startdate,String enddate) {

List<BizAppBean> bizAppBeans = myAppMapper.selectappSort(startdate,enddate);

List<BizAppBean> result = new ArrayList<>();

BizAppBean other = new BizAppBean();

other.setAppName("其他");

double p = 0,b = 0,c = 0,d = 0;

DecimalFormat df = new DecimalFormat("#.00");

for(BizAppBean bizAppBean : bizAppBeans){

String appName = CodeUtils.convertCharset(bizAppBean.getAppName());

if (appName.equals("访问网站")

|| appName.equals("Web流媒体")

|| appName.equals("P2P")

|| appName.equals("P2P流媒体")

|| appName.equals("移动终端应用")) {

bizAppBean.setAppName(appName);

String fp = bizAppBean.getFluxPercentage();

bizAppBean.setFluxPercentage(df.format(Double.parseDouble(fp)));

result.add(bizAppBean);

p += Double.parseDouble(bizAppBean.getFluxPercentage());

d += bizAppBean.getFlux();

b = bizAppBean.getFlux();

c = Double.parseDouble(bizAppBean.getFluxPercentage());

}

}

if (c == 0) {

other.setFluxPercentage("0");

other.setFlux(0D);

result.add(other);

return new ResponseBean<>(result);

}

double a = 100 - p;

String str = df.format(a);

double e = b/c\*100 - d;

String st = df.format(e);

other.setFluxPercentage(str);

other.setFlux(Double.parseDouble(st));

result.add(other);

return new ResponseBean<>(result);

}

}

package com.yingchong.service.data\_service.service;

import com.yingchong.service.data\_service.BizBean.ResponseBean;

import com.yingchong.service.data\_service.BizBean.biz\_app.BizAppRelationBean;

import com.yingchong.service.data\_service.BizBean.biz\_app.BizAppTreadBean;

import com.yingchong.service.data\_service.BizBean.biz\_app.BizAppTypeBean;

import com.yingchong.service.data\_service.mapper.MyAppTypeMapper;

import com.yingchong.service.data\_service.mybatis.mapper.ActionListMapper;

import com.yingchong.service.data\_service.mybatis.mapper.ActionTypeMapper;

import com.yingchong.service.data\_service.mybatis.model.ActionType;

import com.yingchong.service.data\_service.mybatis.model.ActionTypeExample;

import com.yingchong.service.data\_service.utils.CodeUtils;

import com.yingchong.service.data\_service.utils.DateUtil;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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

import org.springframework.stereotype.Service;

import java.time.LocalDate;

import java.time.format.DateTimeFormatter;

import java.util.\*;

@Service

public class AppTypeService {

private static final Logger logger = LoggerFactory.getLogger(AppTypeService.class);

@Autowired

private MyAppTypeMapper myAppTypeMapper;

@Autowired

private ActionTypeMapper actionTypeMapper;

@Autowired

private ActionListMapper actionListMapper;

public static final String dateParttern = "yyyy-MM-dd";

public boolean insertActionType(String date) {

try {

ActionTypeExample example = new ActionTypeExample();

example.createCriteria().andActionDateEqualTo(DateUtil.StringToDate(date,dateParttern));

List<ActionType> actionTypes = actionTypeMapper.selectByExample(example);

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

logger.info("数据已经插入,不再重复插入");

return true;

}

List<BizAppTypeBean> bizAppBeans = myAppTypeMapper.selectApp(date.replaceAll("-", "") + "\_action");

for (BizAppTypeBean type : bizAppBeans) {

ActionType at = new ActionType();

at.setActionCount(type.getNum());

String app = type.getApp();

//logger.info("utf8: {}",new String(app.getBytes(StandardCharsets.ISO\_8859\_1), StandardCharsets.UTF\_8));

at.setActionName(app);

at.setActionDate(DateUtil.StringToDate(date, dateParttern));

Date date1 = new Date();

at.setCreateTime(date1);

at.setUpdateTime(date1);

actionTypeMapper.insert(at);

}

} catch (Exception e) {

logger.error("插入action\_type 数据异常:",e);

return false;

}

return true;

}

/\*\*

\* 网络意识形态分类

\* @param startDate

\* @param endDate

\* @return

\*/

public ResponseBean<List<BizAppTypeBean>> actionTypeList(String startDate, String endDate) {

List<BizAppTypeBean> bizAppTypeBeans = myAppTypeMapper.selectAppTypeResult(startDate, endDate);

for (BizAppTypeBean bizAppTypeBean : bizAppTypeBeans) {//中文转码

bizAppTypeBean.setApp(CodeUtils.convertCharset(bizAppTypeBean.getApp()));

}

//List<ActionList> actionLists = actionListMapper.selectByExample(new ActionListExample());

List<BizAppRelationBean> bizAppRelationBeans = myAppTypeMapper.selectAppTypeRelation();

for (BizAppRelationBean bizAppRelationBean : bizAppRelationBeans) {//中文转码

//bizAppRelationBean.setActionName(CodeUtils.convertCharset(bizAppRelationBean.getActionName()));

bizAppRelationBean.setActionName(CodeUtils.convertUtf8ToLatin1(bizAppRelationBean.getActionName()));

bizAppRelationBean.setRelationItemName(CodeUtils.convertUtf8ToLatin1(bizAppRelationBean.getRelationItemName()));

}

Map<String, Integer> map = new HashMap<>();

for (BizAppTypeBean bizAppTypeBean : bizAppTypeBeans) {

String actionTypeName = bizAppTypeBean.getApp();

for (BizAppRelationBean bizAppRelationBean : bizAppRelationBeans) {

String relationItemName = bizAppRelationBean.getRelationItemName();

if(actionTypeName!=null && actionTypeName.equals(relationItemName)){

String actionName = bizAppRelationBean.getActionName();

Integer integer = map.get(actionName);

if(integer==null) integer = 0;

map.put(actionName, integer + bizAppTypeBean.getNum());

break;

}

}

}

Iterator<String> iterator = map.keySet().iterator();

List<BizAppTypeBean> res = new ArrayList<>();

while (iterator.hasNext()) {

BizAppTypeBean bean = new BizAppTypeBean();

String key = iterator.next();

Integer num = map.get(key);

bean.setApp(key);

bean.setNum(num);

res.add(bean);

}

//orderByNum(res);

res = orderByOrder(res,bizAppRelationBeans);

return new ResponseBean<>(res);

}

/\*\*

\* 通过数量倒序排序

\* @param res

\*/

private void orderByNum(List<BizAppTypeBean> res) {

Comparator<BizAppTypeBean> comparator = (o1, o2) -> o2.getNum() - o1.getNum();

res.sort(comparator);

}

/\*\*

\* 通过指定 order 排序

\* @param res

\*/

private List<BizAppTypeBean> orderByOrder(List<BizAppTypeBean> res,List<BizAppRelationBean> bizAppRelationBeans) {

Map<Integer, BizAppTypeBean> map = new TreeMap<>();

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

for (BizAppRelationBean bizAppRelationBean : bizAppRelationBeans) {

if(bizAppRelationBean.getActionName().equals(res.get(i).getApp()))

map.put(bizAppRelationBean.getOrderNum(),res.get(i));

}

}

res = new ArrayList<>();

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

if (map.get(i) != null) {

res.add(map.get(i));

}

}

return res;

}

/\*\*

\* 趋势

\* @param startDate 开始时间

\* @param endDate 结束时间

\* @return

\*/

public ResponseBean<BizAppTreadBean> actionTypeTrend(String startDate, String endDate) {

BizAppTreadBean bizAppTreadBean = new BizAppTreadBean();

List<BizAppTypeBean> top3 = myAppTypeMapper.selectAppTypeResult(startDate, endDate);

if (top3 == null || top3.size()==0) {

return new ResponseBean<>(bizAppTreadBean);

}

List<BizAppTypeBean> bizAppTypeBeans1 = null;

List<BizAppTypeBean> bizAppTypeBeans2 = null;

List<BizAppTypeBean> bizAppTypeBeans3 = null;

for (int i=0;i<3;i++) {

BizAppTypeBean bizAppTypeBean = top3.get(i);

List<BizAppTypeBean> bizAppTypeBeans = myAppTypeMapper.selectAppTypeTreadResultByAppName(startDate, endDate, CodeUtils.convertCharset88591(bizAppTypeBean.getApp()));

if(i==0){

bizAppTypeBeans1 = bizAppTypeBeans;

for (BizAppTypeBean appTypeBean : bizAppTypeBeans1) {

appTypeBean.setApp(CodeUtils.convertCharset(appTypeBean.getApp()));

appTypeBean.setDateStr(DateUtil.formatDateToStr(appTypeBean.getDate(),AppTypeService.dateParttern));

}

bizAppTreadBean.setList1(bizAppTypeBeans1);

}

if(i==1){

bizAppTypeBeans2 = bizAppTypeBeans;

for (BizAppTypeBean appTypeBean : bizAppTypeBeans2) {

appTypeBean.setApp(CodeUtils.convertCharset(appTypeBean.getApp()));

appTypeBean.setDateStr(DateUtil.formatDateToStr(appTypeBean.getDate(),AppTypeService.dateParttern));

}

bizAppTreadBean.setList2(bizAppTypeBeans2);

}

if(i==2) {

bizAppTypeBeans3 = bizAppTypeBeans;

for (BizAppTypeBean appTypeBean : bizAppTypeBeans3) {

appTypeBean.setApp(CodeUtils.convertCharset(appTypeBean.getApp()));

appTypeBean.setDateStr(DateUtil.formatDateToStr(appTypeBean.getDate(),AppTypeService.dateParttern));

}

bizAppTreadBean.setList3(bizAppTypeBeans3);

}

}

checkTrendData(bizAppTreadBean, startDate, endDate);

/\*List<BizAppTypeBean> bizAppTypeBeans = myAppTypeMapper.selectAppTypeTreadResult(startDate, endDate);

for (BizAppTypeBean bizAppTypeBean : bizAppTypeBeans) {

//bizAppTypeBean.setApp(new String(bizAppTypeBean.getApp().getBytes(StandardCharsets.ISO\_8859\_1), StandardCharsets.UTF\_8));

bizAppTypeBean.setApp(CodeUtils.convertCharset(bizAppTypeBean.getApp()));

bizAppTypeBean.setDateStr(DateUtil.formatDateToStr(bizAppTypeBean.getDate(),"yyyy-MM-dd"));

}\*/

return new ResponseBean<>(bizAppTreadBean);

}

private void checkTrendData(BizAppTreadBean bizAppTreadBean,String startDate, String endDate) {

List<BizAppTypeBean> list1 = bizAppTreadBean.getList1();

List<BizAppTypeBean> list2 = bizAppTreadBean.getList2();

List<BizAppTypeBean> list3 = bizAppTreadBean.getList3();

LocalDate start = LocalDate.parse(startDate, DateTimeFormatter.ofPattern(AppTypeService.dateParttern));

LocalDate end = LocalDate.parse(endDate, DateTimeFormatter.ofPattern(AppTypeService.dateParttern));

int i = 0;

for (LocalDate date = start; date.isBefore(end); date = date.plusDays(1))

{

checkListData(list1, start, i, date);

checkListData(list2, start, i, date);

checkListData(list3, start, i, date);

i++;

}

}

private void checkListData(List<BizAppTypeBean> list,LocalDate start, int i, LocalDate date) {

String app = list.get(0).getApp();

if(list.size()-1 > i){

BizAppTypeBean appTypeBean = list.get(i);

if(appTypeBean==null || !date.toString().equals(appTypeBean.getDateStr())){

appTypeBean = new BizAppTypeBean();

appTypeBean.setDateStr(start.plusDays(i).toString());

appTypeBean.setApp(app);

appTypeBean.setNum(0);

list.add(i,appTypeBean);

}

}else {

BizAppTypeBean appTypeBean = new BizAppTypeBean();

appTypeBean.setDateStr(start.plusDays(i+1).toString());

appTypeBean.setApp(app);

appTypeBean.setNum(0);

list.add(appTypeBean);

}

}

}

package com.yingchong.service.data\_service.BizBean.biz\_action;

import java.util.Date;

public class BizActionBean {

private Integer recordId;

private Integer devId;

private String group;

private String user;

private String hostIp,hostIpBin,dstIp,dstIpBin,ipVersion,site,tmType,serv,app;

private Integer srcPort,servPort;

private String netAction;

private Date recordTime;

private String result;

public Integer getRecordId() {

return recordId;

}

public void setRecordId(Integer recordId) {

this.recordId = recordId;

}

public Integer getDevId() {

return devId;

}

public void setDevId(Integer devId) {

this.devId = devId;

}

public String getGroup() {

return group;

}

public void setGroup(String group) {

this.group = group;

}

public String getUser() {

return user;

}

public void setUser(String user) {

this.user = user;

}

public String getHostIp() {

return hostIp;

}

public void setHostIp(String hostIp) {

this.hostIp = hostIp;

}

public String getHostIpBin() {

return hostIpBin;

}

public void setHostIpBin(String hostIpBin) {

this.hostIpBin = hostIpBin;

}

public String getDstIp() {

return dstIp;

}

public void setDstIp(String dstIp) {

this.dstIp = dstIp;

}

public String getDstIpBin() {

return dstIpBin;

}

public void setDstIpBin(String dstIpBin) {

this.dstIpBin = dstIpBin;

}

public String getIpVersion() {

return ipVersion;

}

public void setIpVersion(String ipVersion) {

this.ipVersion = ipVersion;

}

public String getSite() {

return site;

}

public void setSite(String site) {

this.site = site;

}

public String getTmType() {

return tmType;

}

public void setTmType(String tmType) {

this.tmType = tmType;

}

public String getServ() {

return serv;

}

public void setServ(String serv) {

this.serv = serv;

}

public String getApp() {

return app;

}

public void setApp(String app) {

this.app = app;

}

public Integer getSrcPort() {

return srcPort;

}

public void setSrcPort(Integer srcPort) {

this.srcPort = srcPort;

}

public Integer getServPort() {

return servPort;

}

public void setServPort(Integer servPort) {

this.servPort = servPort;

}

public String getNetAction() {

return netAction;

}

public void setNetAction(String netAction) {

this.netAction = netAction;

}

public Date getRecordTime() {

return recordTime;

}

public void setRecordTime(Date recordTime) {

this.recordTime = recordTime;

}

public String getResult() {

return result;

}

public void setResult(String result) {

this.result = result;

}

}

package com.yingchong.service.data\_service.BizBean.biz\_app;

public class BizAppBean {

private String date,appName,fluxPercentage;

private Double flux;

public String getDate() {

return date;

}

public void setDate(String date) {

this.date = date;

}

public String getAppName() {

return appName;

}

public void setAppName(String appName) {

this.appName = appName;

}

public String getFluxPercentage() {

return fluxPercentage;

}

public void setFluxPercentage(String fluxPercentage) {

this.fluxPercentage = fluxPercentage;

}

public Double getFlux() {

return flux;

}

public void setFlux(Double flux) {

this.flux = flux;

}

}

package com.yingchong.service.data\_service.BizBean.biz\_app;

import java.util.Date;

public class BizAppTypeBean {

private String app;

private Integer num;

private Date date;

private String dateStr;

public String getApp() {

return app;

}

public void setApp(String app) {

this.app = app;

}

public Integer getNum() {

return num;

}

public void setNum(Integer num) {

this.num = num;

}

public Date getDate() {

return date;

}

public void setDate(Date date) {

this.date = date;

}

public String getDateStr() {

return dateStr;

}

public void setDateStr(String dateStr) {

this.dateStr = dateStr;

}

}

package com.yingchong.service.data\_service.BizBean.biz\_app;

import java.util.List;

public class BizAppTreadBean {

private List<BizAppTypeBean> list1,list2,list3;

public List<BizAppTypeBean> getList1() {

return list1;

}

public void setList1(List<BizAppTypeBean> list1) {

this.list1 = list1;

}

public List<BizAppTypeBean> getList2() {

return list2;

}

public void setList2(List<BizAppTypeBean> list2) {

this.list2 = list2;

}

public List<BizAppTypeBean> getList3() {

return list3;

}

public void setList3(List<BizAppTypeBean> list3) {

this.list3 = list3;

}

}

package com.yingchong.service.data\_service.BizBean.biz\_flux;

public class BizDataBean {

private String date, uploadFlux, downFlux;

public String getDate() {

return date;

}

public void setDate(String date) {

this.date = date;

}

public String getUploadFlux() {

return uploadFlux;

}

public void setUploadFlux(String uploadFlux) {

this.uploadFlux = uploadFlux;

}

public String getDownFlux() {

return downFlux;

}

public void setDownFlux(String downFlux) {

this.downFlux = downFlux;

}

}

package com.yingchong.service.data\_service.BizBean;

import com.yingchong.service.data\_service.utils.DataErrorCode;

import io.swagger.annotations.ApiModelProperty;

public class ResponseBean<T>{

@ApiModelProperty(value = "返回码")

private String retCode=DataErrorCode.SUCCESS.getCode();//返回码

@ApiModelProperty(value = "返回消息")

private String retMsg=DataErrorCode.SUCCESS.getMsg();//返回消息

@ApiModelProperty(value = "返回数据")

private T data;//返回数据

public ResponseBean() {

}

public ResponseBean(String retCode, String retMsg) {

this.retCode = retCode;

this.retMsg = retMsg;

}

public ResponseBean(Integer retCode, String retMsg) {

this.retCode = String.valueOf(retCode);

this.retMsg = retMsg;

}

public ResponseBean(T t) {

this.setData(t);

}

public String getRetCode() {

return retCode;

}

public void setRetCode(String retCode) {

this.retCode = retCode;

}

public String getRetMsg() {

return retMsg;

}

public void setRetMsg(String retMsg) {

this.retMsg = retMsg;

}

public T getData() {

return data;

}

public void setData(T data) {

this.data = data;

}

public void setCodeAndMsg(String retCode, String retMsg) {

this.retCode = retCode;

this.retMsg = retMsg;

}

public void setCodeAndMsg(Integer retCode, String retMsg) {

this.retCode = String.valueOf(retCode);

this.retMsg = retMsg;

}

@Override

public String toString() {

return "ResponseBean [retCode=" + retCode + ", retMsg=" + retMsg + ", data=" + data + "]";

}

}

package com.yingchong.service.data\_service.config;

import org.springframework.boot.bind.RelaxedPropertyResolver;

import org.springframework.context.EnvironmentAware;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.core.env.Environment;

import org.springframework.web.filter.CharacterEncodingFilter;

import javax.servlet.Filter;

import java.util.HashMap;

import java.util.Map;

@Configuration

public class WebConfig implements EnvironmentAware {

// 解析application.yml

private RelaxedPropertyResolver propResolver;

/\*\*

\* 解决中文内容编码问题，统一用UTF-8编码

\*

\* @return

\*/

@Bean

public Filter characterEncodingFilter() {

CharacterEncodingFilter characterEncodingFilter = new CharacterEncodingFilter();

characterEncodingFilter.setEncoding("UTF-8");

characterEncodingFilter.setForceEncoding(true);

return characterEncodingFilter;

}

@Override

public void setEnvironment(Environment environment) {

// TODO Auto-generated method stub

propResolver = new RelaxedPropertyResolver(environment);

}

public Map<String,String> staticResource(){

Map<String,String> commonStaticResource = null;

try {

commonStaticResource = new HashMap<String,String>();

commonStaticResource.put("hostUrl", propResolver.getProperty("business.hostUrl"));//医疗特殊分类“家庭医生” 前台判断业态使用

} catch (Exception e) {

e.printStackTrace();

}

return commonStaticResource;

}

}

package com.yingchong.service.data\_service.config;

import com.google.common.base.Predicates;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import springfox.documentation.builders.ApiInfoBuilder;

import springfox.documentation.builders.ParameterBuilder;

import springfox.documentation.builders.PathSelectors;

import springfox.documentation.builders.RequestHandlerSelectors;

import springfox.documentation.schema.ModelRef;

import springfox.documentation.service.ApiInfo;

import springfox.documentation.service.Parameter;

import springfox.documentation.spi.DocumentationType;

import springfox.documentation.spring.web.plugins.Docket;

import springfox.documentation.swagger2.annotations.EnableSwagger2;

import java.util.ArrayList;

import java.util.List;

/\*\*

\* Created by zhoujia on 2019/05/15.

\*/

@Configuration

@EnableSwagger2

public class Swagger2 {

@Bean

public Docket createRestApi() {

ParameterBuilder tokenPar = new ParameterBuilder();

List<Parameter> pars = new ArrayList<>();

tokenPar.name("token").description("令牌").modelRef(new ModelRef("string")).parameterType("header").required(false).build();

pars.add(tokenPar.build());

return new Docket(DocumentationType.SWAGGER\_2)

.apiInfo(apiInfo())

.select()

.apis(RequestHandlerSelectors.basePackage("com.yingchong.service.data\_service")) //指定api接口的package

.paths(PathSelectors.any())

.paths(Predicates.not(PathSelectors.regex("/error.\*")))// 错误路径不监控

.build().globalOperationParameters(pars);

}

private ApiInfo apiInfo() {

return new ApiInfoBuilder()

.title("data-service")

.description("数据操作接口")

//.termsOfServiceUrl("http://blog.didispace.com/")

.contact("zhoujia")

.version("1.0")

.build();

}

}

package com.yingchong.service.data\_service.mapper;

import com.yingchong.service.data\_service.BizBean.biz\_action.BizActionBean;

import org.apache.ibatis.annotations.Param;

import org.apache.ibatis.annotations.Select;

import org.springframework.stereotype.Repository;

import java.util.List;

@Repository

public interface MyActionMapper {

@Select(" SELECT record\_id recordId,`group` `group`,`user` `user`,site site,serv serv,app app, " +

" result result,host\_ip hostIp,dst\_ip dstIp,ip\_version ipVersion, " +

" tm\_type tmType ,src\_port srcPort,serv\_port servPort,net\_action netAction,record\_time recordTime " +

" from ${tableName} LIMIT #{start},#{step}")

List<BizActionBean> selectAction(

@Param("tableName") String tableName,

@Param("start") Integer start,

@Param("step") Integer step

);

@Select(" SELECT record\_id recordId, `group` `group`, `user` `user`, site site, serv serv, app app, result result, host\_ip hostIp, dst\_ip dstIp, ip\_version ipVersion, tm\_type tmType, src\_port srcPort, serv\_port servPort, net\_action netAction, record\_time recordTime " +

" from ${tableName} "+

" where result LIKE \"%web\_url%\" and result not like \"%baidu.com\" and result not LIKE \"%qq.com\" and result not LIKE \"sogou.com\" and result not like \"163.com\" " +

" LIMIT #{start},#{step}")

List<BizActionBean> selectActionByWhere(

@Param("tableName") String tableName,

@Param("start") Integer start,

@Param("step") Integer step

);

@Select(" SELECT record\_id recordId,`group` `group`,`user` `user`,site site,serv serv,app app, result result,host\_ip hostIp,dst\_ip dstIp,ip\_version ipVersion, tm\_type tmType ,src\_port srcPort,serv\_port servPort,net\_action netAction,record\_time recordTime \n" +

" from ${tableName} " +

" where record\_id > #{recordId} " +

" LIMIT #{step}")

List<BizActionBean> selectActionById(

@Param("tableName") String tableName,

@Param("recordId") Integer recordId,

@Param("step") Integer step

);

@Select(" SELECT count(record\_id) from ${tableName}")

Integer selectCountAction(@Param("tableName") String tableName);

}

package com.yingchong.service.data\_service.mapper;

import com.yingchong.service.data\_service.BizBean.biz\_app.BizAppBean;

import org.apache.ibatis.annotations.Param;

import org.apache.ibatis.annotations.Select;

import org.springframework.stereotype.Repository;

import java.util.List;

@Repository

public interface MyAppMapper {

@Select("select a.serv appName,a.flux flux,FORMAT(a.flux/b.flux1\*100,2) fluxPercentage from (select serv,sum(up\_flux+down\_flux)/1024/1024/1024 flux from ${tableName} GROUP BY serv ORDER BY flux ) a join (select sum(up\_flux+down\_flux)/1024/1024/1024 flux1 from ${tableName}) b")

List<BizAppBean> selectApp(@Param("tableName") String tableName);

//@Select("select app\_name appName,flux Flux,flux\_percentage FluxPercentage from app\_flux\_sort where flux\_date >= #{startTime} and flux\_date <= #{endTime}")

@Select("select a.flux flux,a.app\_name appName,a.flux/(select SUM(flux) from app\_flux\_sort where flux\_date >= #{startTime} and flux\_date <= #{endTime})\*100 fluxPercentage from (select SUM(flux) flux,app\_name app\_name from app\_flux\_sort where flux\_date >= #{startTime} and flux\_date <= #{endTime} GROUP BY app\_name) a\n")

List<BizAppBean> selectappSort(

@Param("startTime") String startTime,

@Param("endTime") String endTime

);

}

package com.yingchong.service.data\_service.mapper;

import com.yingchong.service.data\_service.BizBean.biz\_app.BizAppRelationBean;

import com.yingchong.service.data\_service.BizBean.biz\_app.BizAppTypeBean;

import org.apache.ibatis.annotations.Param;

import org.apache.ibatis.annotations.Select;

import org.springframework.stereotype.Repository;

import java.util.List;

@Repository

public interface MyAppTypeMapper {

@Select("select count(\*) num,app from ${tableName} group by app order by num desc")

List<BizAppTypeBean> selectApp(@Param("tableName") String tableName);

@Select("select sum(action\_count) num,action\_name app from action\_type where action\_date >= #{startTime} and action\_date <= #{endTime} group by app order by num desc ")

List<BizAppTypeBean> selectAppTypeResult(

@Param("startTime") String startTime,

@Param("endTime") String endTime

);

@Select("select sum(action\_count) num,action\_name app,action\_date date from action\_type where action\_date >= #{startTime} and action\_date <= #{endTime} group by action\_date,app ORDER BY num desc ")

List<BizAppTypeBean> selectAppTypeTreadResult(

@Param("startTime") String startTime,

@Param("endTime") String endTime

);

@Select("select sum(action\_count) num,action\_name app,action\_date date \n" +

"from action\_type \n" +

"where action\_date >= #{startTime} and action\_date <= #{endTime} and action\_name = #{appName} \n" +

"GROUP BY action\_date ORDER BY action\_date")

List<BizAppTypeBean> selectAppTypeTreadResultByAppName(

@Param("startTime") String startTime,

@Param("endTime") String endTime,

@Param("appName") String appName

);

@Select("SELECT al.type\_name actionName,ar.action\_name relationItemName,al.order\_num orderNum from action\_relation ar left join action\_list al on al.id = ar.relation\_id " +

"where al.type\_name is not null")

List<BizAppRelationBean> selectAppTypeRelation();

}

package com.yingchong.service.data\_service.mapper;

import java.util.Map;

public class MyUserProvider {

public String queryUser(Map<String, Object> param) {

String whereCondition = this.getWhereCondition(param);

return "SELECT u.id id,u.user\_name userName,u.password password, u.nick\_name nickName,u.description description, " +

" u.login\_time loginTime, create\_time createTime " +

"from user u " +

"where " + whereCondition;

}

private String getWhereCondition(Map<String,Object> param) {

StringBuilder condition = new StringBuilder(" 1=1 ");

if (param.get("userName") != null && !String.valueOf(param.get("userName")).equals("")) {

condition.append(" and u.user\_name like concat('%',#{userName},'%') ");

}

if (param.get("startDate") != null ) {

condition.append(" and u.create\_time >= #{startDate} ");

}

if (param.get("endDate") != null ) {

condition.append(" and u.create\_time <= #{endDate} ");

}

if (param.get("description") != null && !String.valueOf(param.get("description")).equals("")) {

condition.append(" and u.description like concat('%',#{description},'%') ");

}

return condition.append(" order by u.create\_time ").toString();

}

}

package com.yingchong.service.data\_service.mybatis.model;

import java.util.Date;

public class ActionList {

private Integer id;

private String typeName;

private Integer orderNum;

private Date createTime;

private Date updateTime;

public Integer getId() {

return id;

}

public void setId(Integer id) {

this.id = id;

}

public String getTypeName() {

return typeName;

}

public void setTypeName(String typeName) {

this.typeName = typeName;

}

public Integer getOrderNum() {

return orderNum;

}

public void setOrderNum(Integer orderNum) {

this.orderNum = orderNum;

}

public Date getCreateTime() {

return createTime;

}

public void setCreateTime(Date createTime) {

this.createTime = createTime;

}

public Date getUpdateTime() {

return updateTime;

}

public void setUpdateTime(Date updateTime) {

this.updateTime = updateTime;

}

}

package com.yingchong.service.data\_service.mybatis.model;

import java.util.ArrayList;

import java.util.Date;

import java.util.List;

public class ActionListExample {

protected String orderByClause;

protected boolean distinct;

protected List<Criteria> oredCriteria;

public ActionListExample() {

oredCriteria = new ArrayList<Criteria>();

}

public void setOrderByClause(String orderByClause) {

this.orderByClause = orderByClause;

}

public String getOrderByClause() {

return orderByClause;

}

public void setDistinct(boolean distinct) {

this.distinct = distinct;

}

public boolean isDistinct() {

return distinct;

}

public List<Criteria> getOredCriteria() {

return oredCriteria;

}

public void or(Criteria criteria) {

oredCriteria.add(criteria);

}

public Criteria or() {

Criteria criteria = createCriteriaInternal();

oredCriteria.add(criteria);

return criteria;

}

public Criteria createCriteria() {

Criteria criteria = createCriteriaInternal();

if (oredCriteria.size() == 0) {

oredCriteria.add(criteria);

}

return criteria;

}

protected Criteria createCriteriaInternal() {

Criteria criteria = new Criteria();

return criteria;

}

public void clear() {

oredCriteria.clear();

orderByClause = null;

distinct = false;

}

protected abstract static class GeneratedCriteria {

protected List<Criterion> criteria;

protected GeneratedCriteria() {

super();

criteria = new ArrayList<Criterion>();

}

public boolean isValid() {

return criteria.size() > 0;

}

public List<Criterion> getAllCriteria() {

return criteria;

}

public List<Criterion> getCriteria() {

return criteria;

}

protected void addCriterion(String condition) {

if (condition == null) {

throw new RuntimeException("Value for condition cannot be null");

}

criteria.add(new Criterion(condition));

}

protected void addCriterion(String condition, Object value, String property) {

if (value == null) {

throw new RuntimeException("Value for " + property + " cannot be null");

}

criteria.add(new Criterion(condition, value));

}

protected void addCriterion(String condition, Object value1, Object value2, String property) {

if (value1 == null || value2 == null) {

throw new RuntimeException("Between values for " + property + " cannot be null");

}

criteria.add(new Criterion(condition, value1, value2));

}

public Criteria andIdIsNull() {

addCriterion("id is null");

return (Criteria) this;

}

public Criteria andIdIsNotNull() {

addCriterion("id is not null");

return (Criteria) this;

}

public Criteria andIdEqualTo(Integer value) {

addCriterion("id =", value, "id");

return (Criteria) this;

}

public Criteria andIdNotEqualTo(Integer value) {

addCriterion("id <>", value, "id");

return (Criteria) this;

}

public Criteria andIdGreaterThan(Integer value) {

addCriterion("id >", value, "id");

return (Criteria) this;

}

public Criteria andIdGreaterThanOrEqualTo(Integer value) {

addCriterion("id >=", value, "id");

return (Criteria) this;

}

public Criteria andIdLessThan(Integer value) {

addCriterion("id <", value, "id");

return (Criteria) this;

}

public Criteria andIdLessThanOrEqualTo(Integer value) {

addCriterion("id <=", value, "id");

return (Criteria) this;

}

public Criteria andIdIn(List<Integer> values) {

addCriterion("id in", values, "id");

return (Criteria) this;

}

public Criteria andIdNotIn(List<Integer> values) {

addCriterion("id not in", values, "id");

return (Criteria) this;

}

public Criteria andIdBetween(Integer value1, Integer value2) {

addCriterion("id between", value1, value2, "id");

return (Criteria) this;

}

public Criteria andIdNotBetween(Integer value1, Integer value2) {

addCriterion("id not between", value1, value2, "id");

return (Criteria) this;

}

public Criteria andTypeNameIsNull() {

addCriterion("type\_name is null");

return (Criteria) this;

}

public Criteria andTypeNameIsNotNull() {

addCriterion("type\_name is not null");

return (Criteria) this;

}

public Criteria andTypeNameEqualTo(String value) {

addCriterion("type\_name =", value, "typeName");

return (Criteria) this;

}

public Criteria andTypeNameNotEqualTo(String value) {

addCriterion("type\_name <>", value, "typeName");

return (Criteria) this;

}

public Criteria andTypeNameGreaterThan(String value) {

addCriterion("type\_name >", value, "typeName");

return (Criteria) this;

}

public Criteria andTypeNameGreaterThanOrEqualTo(String value) {

addCriterion("type\_name >=", value, "typeName");

return (Criteria) this;

}

public Criteria andTypeNameLessThan(String value) {

addCriterion("type\_name <", value, "typeName");

return (Criteria) this;

}

public Criteria andTypeNameLessThanOrEqualTo(String value) {

addCriterion("type\_name <=", value, "typeName");

return (Criteria) this;

}

public Criteria andTypeNameLike(String value) {

addCriterion("type\_name like", value, "typeName");

return (Criteria) this;

}

public Criteria andTypeNameNotLike(String value) {

addCriterion("type\_name not like", value, "typeName");

return (Criteria) this;

}

public Criteria andTypeNameIn(List<String> values) {

addCriterion("type\_name in", values, "typeName");

return (Criteria) this;

}

public Criteria andTypeNameNotIn(List<String> values) {

addCriterion("type\_name not in", values, "typeName");

return (Criteria) this;

}

public Criteria andTypeNameBetween(String value1, String value2) {

addCriterion("type\_name between", value1, value2, "typeName");

return (Criteria) this;

}

public Criteria andTypeNameNotBetween(String value1, String value2) {

addCriterion("type\_name not between", value1, value2, "typeName");

return (Criteria) this;

}

public Criteria andOrderNumIsNull() {

addCriterion("order\_num is null");

return (Criteria) this;

}

public Criteria andOrderNumIsNotNull() {

addCriterion("order\_num is not null");

return (Criteria) this;

}

public Criteria andOrderNumEqualTo(Integer value) {

addCriterion("order\_num =", value, "orderNum");

return (Criteria) this;

}

public Criteria andOrderNumNotEqualTo(Integer value) {

addCriterion("order\_num <>", value, "orderNum");

return (Criteria) this;

}

public Criteria andOrderNumGreaterThan(Integer value) {

addCriterion("order\_num >", value, "orderNum");

return (Criteria) this;

}

public Criteria andOrderNumGreaterThanOrEqualTo(Integer value) {

addCriterion("order\_num >=", value, "orderNum");

return (Criteria) this;

}

public Criteria andOrderNumLessThan(Integer value) {

addCriterion("order\_num <", value, "orderNum");

return (Criteria) this;

}

public Criteria andOrderNumLessThanOrEqualTo(Integer value) {

addCriterion("order\_num <=", value, "orderNum");

return (Criteria) this;

}

public Criteria andOrderNumIn(List<Integer> values) {

addCriterion("order\_num in", values, "orderNum");

return (Criteria) this;

}

public Criteria andOrderNumNotIn(List<Integer> values) {

addCriterion("order\_num not in", values, "orderNum");

return (Criteria) this;

}

public Criteria andOrderNumBetween(Integer value1, Integer value2) {

addCriterion("order\_num between", value1, value2, "orderNum");

return (Criteria) this;

}

public Criteria andOrderNumNotBetween(Integer value1, Integer value2) {

addCriterion("order\_num not between", value1, value2, "orderNum");

return (Criteria) this;

}

public Criteria andCreateTimeIsNull() {

addCriterion("create\_time is null");

return (Criteria) this;

}

public Criteria andCreateTimeIsNotNull() {

addCriterion("create\_time is not null");

return (Criteria) this;

}

public Criteria andCreateTimeEqualTo(Date value) {

addCriterion("create\_time =", value, "createTime");

return (Criteria) this;

}

public Criteria andCreateTimeNotEqualTo(Date value) {

addCriterion("create\_time <>", value, "createTime");

return (Criteria) this;

}

public Criteria andCreateTimeGreaterThan(Date value) {

addCriterion("create\_time >", value, "createTime");

return (Criteria) this;

}

public Criteria andCreateTimeGreaterThanOrEqualTo(Date value) {

addCriterion("create\_time >=", value, "createTime");

return (Criteria) this;

}

public Criteria andCreateTimeLessThan(Date value) {

addCriterion("create\_time <", value, "createTime");

return (Criteria) this;

}

public Criteria andCreateTimeLessThanOrEqualTo(Date value) {

addCriterion("create\_time <=", value, "createTime");

return (Criteria) this;

}

public Criteria andCreateTimeIn(List<Date> values) {

addCriterion("create\_time in", values, "createTime");

return (Criteria) this;

}

public Criteria andCreateTimeNotIn(List<Date> values) {

addCriterion("create\_time not in", values, "createTime");

return (Criteria) this;

}

public Criteria andCreateTimeBetween(Date value1, Date value2) {

addCriterion("create\_time between", value1, value2, "createTime");

return (Criteria) this;

}

public Criteria andCreateTimeNotBetween(Date value1, Date value2) {

addCriterion("create\_time not between", value1, value2, "createTime");

return (Criteria) this;

}

public Criteria andUpdateTimeIsNull() {

addCriterion("update\_time is null");

return (Criteria) this;

}

public Criteria andUpdateTimeIsNotNull() {

addCriterion("update\_time is not null");

return (Criteria) this;

}

public Criteria andUpdateTimeEqualTo(Date value) {

addCriterion("update\_time =", value, "updateTime");

return (Criteria) this;

}

public Criteria andUpdateTimeNotEqualTo(Date value) {

addCriterion("update\_time <>", value, "updateTime");

return (Criteria) this;

}

public Criteria andUpdateTimeGreaterThan(Date value) {

addCriterion("update\_time >", value, "updateTime");

return (Criteria) this;

}

public Criteria andUpdateTimeGreaterThanOrEqualTo(Date value) {

addCriterion("update\_time >=", value, "updateTime");

return (Criteria) this;

}

public Criteria andUpdateTimeLessThan(Date value) {

addCriterion("update\_time <", value, "updateTime");

return (Criteria) this;

}

public Criteria andUpdateTimeLessThanOrEqualTo(Date value) {

addCriterion("update\_time <=", value, "updateTime");

return (Criteria) this;

}

public Criteria andUpdateTimeIn(List<Date> values) {

addCriterion("update\_time in", values, "updateTime");

return (Criteria) this;

}

public Criteria andUpdateTimeNotIn(List<Date> values) {

addCriterion("update\_time not in", values, "updateTime");

return (Criteria) this;

}

public Criteria andUpdateTimeBetween(Date value1, Date value2) {

addCriterion("update\_time between", value1, value2, "updateTime");

return (Criteria) this;

}

public Criteria andUpdateTimeNotBetween(Date value1, Date value2) {

addCriterion("update\_time not between", value1, value2, "updateTime");

return (Criteria) this;

}

}

public static class Criteria extends GeneratedCriteria {

protected Criteria() {

super();

}

}

public static class Criterion {

private String condition;

private Object value;

private Object secondValue;

private boolean noValue;

private boolean singleValue;

private boolean betweenValue;

private boolean listValue;

private String typeHandler;

public String getCondition() {

return condition;

}

public Object getValue() {

return value;

}

public Object getSecondValue() {

return secondValue;

}

public boolean isNoValue() {

return noValue;

}

public boolean isSingleValue() {

return singleValue;

}

public boolean isBetweenValue() {

return betweenValue;

}

public boolean isListValue() {

return listValue;

}

public String getTypeHandler() {

return typeHandler;

}

protected Criterion(String condition) {

super();

this.condition = condition;

this.typeHandler = null;

this.noValue = true;

}

protected Criterion(String condition, Object value, String typeHandler) {

super();

this.condition = condition;

this.value = value;

this.typeHandler = typeHandler;

if (value instanceof List<?>) {

this.listValue = true;

} else {

this.singleValue = true;

}

}

protected Criterion(String condition, Object value) {

this(condition, value, null);

}

protected Criterion(String condition, Object value, Object secondValue, String typeHandler) {

super();

this.condition = condition;

this.value = value;

this.secondValue = secondValue;

this.typeHandler = typeHandler;

this.betweenValue = true;

}

protected Criterion(String condition, Object value, Object secondValue) {

this(condition, value, secondValue, null);

}

}

}

package com.yingchong.service.data\_service.mybatis.model;

import java.util.Date;

public class FluxResult {

private Integer id;

private Double upload;

private Double download;

private Date fluxDate;

private Date createTime;

private Date updateTime;

public Integer getId() {

return id;

}

public void setId(Integer id) {

this.id = id;

}

public Double getUpload() {

return upload;

}

public void setUpload(Double upload) {

this.upload = upload;

}

public Double getDownload() {

return download;

}

public void setDownload(Double download) {

this.download = download;

}

public Date getFluxDate() {

return fluxDate;

}

public void setFluxDate(Date fluxDate) {

this.fluxDate = fluxDate;

}

public Date getCreateTime() {

return createTime;

}

public void setCreateTime(Date createTime) {

this.createTime = createTime;

}

public Date getUpdateTime() {

return updateTime;

}

public void setUpdateTime(Date updateTime) {

this.updateTime = updateTime;

}

}

package com.yingchong.service.data\_service.mybatis.model;

import java.util.ArrayList;

import java.util.Date;

import java.util.Iterator;

import java.util.List;

public class FluxResultExample {

protected String orderByClause;

protected boolean distinct;

protected List<Criteria> oredCriteria;

public FluxResultExample() {

oredCriteria = new ArrayList<Criteria>();

}

public void setOrderByClause(String orderByClause) {

this.orderByClause = orderByClause;

}

public String getOrderByClause() {

return orderByClause;

}

public void setDistinct(boolean distinct) {

this.distinct = distinct;

}

public boolean isDistinct() {

return distinct;

}

public List<Criteria> getOredCriteria() {

return oredCriteria;

}

public void or(Criteria criteria) {

oredCriteria.add(criteria);

}

public Criteria or() {

Criteria criteria = createCriteriaInternal();

oredCriteria.add(criteria);

return criteria;

}

public Criteria createCriteria() {

Criteria criteria = createCriteriaInternal();

if (oredCriteria.size() == 0) {

oredCriteria.add(criteria);

}

return criteria;

}

protected Criteria createCriteriaInternal() {

Criteria criteria = new Criteria();

return criteria;

}

public void clear() {

oredCriteria.clear();

orderByClause = null;

distinct = false;

}

protected abstract static class GeneratedCriteria {

protected List<Criterion> criteria;

protected GeneratedCriteria() {

super();

criteria = new ArrayList<Criterion>();

}

public boolean isValid() {

return criteria.size() > 0;

}

public List<Criterion> getAllCriteria() {

return criteria;

}

public List<Criterion> getCriteria() {

return criteria;

}

protected void addCriterion(String condition) {

if (condition == null) {

throw new RuntimeException("Value for condition cannot be null");

}

criteria.add(new Criterion(condition));

}

protected void addCriterion(String condition, Object value, String property) {

if (value == null) {

throw new RuntimeException("Value for " + property + " cannot be null");

}

criteria.add(new Criterion(condition, value));

}

protected void addCriterion(String condition, Object value1, Object value2, String property) {

if (value1 == null || value2 == null) {

throw new RuntimeException("Between values for " + property + " cannot be null");

}

criteria.add(new Criterion(condition, value1, value2));

}

protected void addCriterionForJDBCDate(String condition, Date value, String property) {

if (value == null) {

throw new RuntimeException("Value for " + property + " cannot be null");

}

addCriterion(condition, new java.sql.Date(value.getTime()), property);

}

protected void addCriterionForJDBCDate(String condition, List<Date> values, String property) {

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

throw new RuntimeException("Value list for " + property + " cannot be null or empty");

}

List<java.sql.Date> dateList = new ArrayList<java.sql.Date>();

Iterator<Date> iter = values.iterator();

while (iter.hasNext()) {

dateList.add(new java.sql.Date(iter.next().getTime()));

}

addCriterion(condition, dateList, property);

}

protected void addCriterionForJDBCDate(String condition, Date value1, Date value2, String property) {

if (value1 == null || value2 == null) {

throw new RuntimeException("Between values for " + property + " cannot be null");

}

addCriterion(condition, new java.sql.Date(value1.getTime()), new java.sql.Date(value2.getTime()), property);

}

public Criteria andIdIsNull() {

addCriterion("id is null");

return (Criteria) this;

}

public Criteria andIdIsNotNull() {

addCriterion("id is not null");

return (Criteria) this;

}

public Criteria andIdEqualTo(Integer value) {

addCriterion("id =", value, "id");

return (Criteria) this;

}

public Criteria andIdNotEqualTo(Integer value) {

addCriterion("id <>", value, "id");

return (Criteria) this;

}

public Criteria andIdGreaterThan(Integer value) {

addCriterion("id >", value, "id");

return (Criteria) this;

}

public Criteria andIdGreaterThanOrEqualTo(Integer value) {

addCriterion("id >=", value, "id");

return (Criteria) this;

}

public Criteria andIdLessThan(Integer value) {

addCriterion("id <", value, "id");

return (Criteria) this;

}

public Criteria andIdLessThanOrEqualTo(Integer value) {

addCriterion("id <=", value, "id");

return (Criteria) this;

}

public Criteria andIdIn(List<Integer> values) {

addCriterion("id in", values, "id");

return (Criteria) this;

}

public Criteria andIdNotIn(List<Integer> values) {

addCriterion("id not in", values, "id");

return (Criteria) this;

}

public Criteria andIdBetween(Integer value1, Integer value2) {

addCriterion("id between", value1, value2, "id");

return (Criteria) this;

}

public Criteria andIdNotBetween(Integer value1, Integer value2) {

addCriterion("id not between", value1, value2, "id");

return (Criteria) this;

}

public Criteria andUploadIsNull() {

addCriterion("upload is null");

return (Criteria) this;

}

public Criteria andUploadIsNotNull() {

addCriterion("upload is not null");

return (Criteria) this;

}

public Criteria andUploadEqualTo(Double value) {

addCriterion("upload =", value, "upload");

return (Criteria) this;

}

public Criteria andUploadNotEqualTo(Double value) {

addCriterion("upload <>", value, "upload");

return (Criteria) this;

}

public Criteria andUploadGreaterThan(Double value) {

addCriterion("upload >", value, "upload");

return (Criteria) this;

}

public Criteria andUploadGreaterThanOrEqualTo(Double value) {

addCriterion("upload >=", value, "upload");

return (Criteria) this;

}

public Criteria andUploadLessThan(Double value) {

addCriterion("upload <", value, "upload");

return (Criteria) this;

}

public Criteria andUploadLessThanOrEqualTo(Double value) {

addCriterion("upload <=", value, "upload");

return (Criteria) this;

}

public Criteria andUploadIn(List<Double> values) {

addCriterion("upload in", values, "upload");

return (Criteria) this;

}

public Criteria andUploadNotIn(List<Double> values) {

addCriterion("upload not in", values, "upload");

return (Criteria) this;

}

public Criteria andUploadBetween(Double value1, Double value2) {

addCriterion("upload between", value1, value2, "upload");

return (Criteria) this;

}

public Criteria andUploadNotBetween(Double value1, Double value2) {

addCriterion("upload not between", value1, value2, "upload");

return (Criteria) this;

}

public Criteria andDownloadIsNull() {

addCriterion("download is null");

return (Criteria) this;

}

public Criteria andDownloadIsNotNull() {

addCriterion("download is not null");

return (Criteria) this;

}

public Criteria andDownloadEqualTo(Double value) {

addCriterion("download =", value, "download");

return (Criteria) this;

}

public Criteria andDownloadNotEqualTo(Double value) {

addCriterion("download <>", value, "download");

return (Criteria) this;

}

public Criteria andDownloadGreaterThan(Double value) {

addCriterion("download >", value, "download");

return (Criteria) this;

}

public Criteria andDownloadGreaterThanOrEqualTo(Double value) {

addCriterion("download >=", value, "download");

return (Criteria) this;

}

public Criteria andDownloadLessThan(Double value) {

addCriterion("download <", value, "download");

return (Criteria) this;

}

public Criteria andDownloadLessThanOrEqualTo(Double value) {

addCriterion("download <=", value, "download");

return (Criteria) this;

}

public Criteria andDownloadIn(List<Double> values) {

addCriterion("download in", values, "download");

return (Criteria) this;

}

public Criteria andDownloadNotIn(List<Double> values) {

addCriterion("download not in", values, "download");

return (Criteria) this;

}

public Criteria andDownloadBetween(Double value1, Double value2) {

addCriterion("download between", value1, value2, "download");

return (Criteria) this;

}

public Criteria andDownloadNotBetween(Double value1, Double value2) {

addCriterion("download not between", value1, value2, "download");

return (Criteria) this;

}

public Criteria andFluxDateIsNull() {

addCriterion("flux\_date is null");

return (Criteria) this;

}

public Criteria andFluxDateIsNotNull() {

addCriterion("flux\_date is not null");

return (Criteria) this;

}

public Criteria andFluxDateEqualTo(Date value) {

addCriterionForJDBCDate("flux\_date =", value, "fluxDate");

return (Criteria) this;

}

public Criteria andFluxDateNotEqualTo(Date value) {

addCriterionForJDBCDate("flux\_date <>", value, "fluxDate");

return (Criteria) this;

}

public Criteria andFluxDateGreaterThan(Date value) {

addCriterionForJDBCDate("flux\_date >", value, "fluxDate");

return (Criteria) this;

}

public Criteria andFluxDateGreaterThanOrEqualTo(Date value) {

addCriterionForJDBCDate("flux\_date >=", value, "fluxDate");

return (Criteria) this;

}

public Criteria andFluxDateLessThan(Date value) {

addCriterionForJDBCDate("flux\_date <", value, "fluxDate");

return (Criteria) this;

}

public Criteria andFluxDateLessThanOrEqualTo(Date value) {

addCriterionForJDBCDate("flux\_date <=", value, "fluxDate");

return (Criteria) this;

}

public Criteria andFluxDateIn(List<Date> values) {

addCriterionForJDBCDate("flux\_date in", values, "fluxDate");

return (Criteria) this;

}

public Criteria andFluxDateNotIn(List<Date> values) {

addCriterionForJDBCDate("flux\_date not in", values, "fluxDate");

return (Criteria) this;

}

public Criteria andFluxDateBetween(Date value1, Date value2) {

addCriterionForJDBCDate("flux\_date between", value1, value2, "fluxDate");

return (Criteria) this;

}

public Criteria andFluxDateNotBetween(Date value1, Date value2) {

addCriterionForJDBCDate("flux\_date not between", value1, value2, "fluxDate");

return (Criteria) this;

}

public Criteria andCreateTimeIsNull() {

addCriterion("create\_time is null");

return (Criteria) this;

}

public Criteria andCreateTimeIsNotNull() {

addCriterion("create\_time is not null");

return (Criteria) this;

}

public Criteria andCreateTimeEqualTo(Date value) {

addCriterion("create\_time =", value, "createTime");

return (Criteria) this;

}

public Criteria andCreateTimeNotEqualTo(Date value) {

addCriterion("create\_time <>", value, "createTime");

return (Criteria) this;

}

public Criteria andCreateTimeGreaterThan(Date value) {

addCriterion("create\_time >", value, "createTime");

return (Criteria) this;

}

public Criteria andCreateTimeGreaterThanOrEqualTo(Date value) {

addCriterion("create\_time >=", value, "createTime");

return (Criteria) this;

}

public Criteria andCreateTimeLessThan(Date value) {

addCriterion("create\_time <", value, "createTime");

return (Criteria) this;

}

public Criteria andCreateTimeLessThanOrEqualTo(Date value) {

addCriterion("create\_time <=", value, "createTime");

return (Criteria) this;

}

public Criteria andCreateTimeIn(List<Date> values) {

addCriterion("create\_time in", values, "createTime");

return (Criteria) this;

}

public Criteria andCreateTimeNotIn(List<Date> values) {

addCriterion("create\_time not in", values, "createTime");

return (Criteria) this;

}

public Criteria andCreateTimeBetween(Date value1, Date value2) {

addCriterion("create\_time between", value1, value2, "createTime");

return (Criteria) this;

}

public Criteria andCreateTimeNotBetween(Date value1, Date value2) {

addCriterion("create\_time not between", value1, value2, "createTime");

return (Criteria) this;

}

public Criteria andUpdateTimeIsNull() {

addCriterion("update\_time is null");

return (Criteria) this;

}

public Criteria andUpdateTimeIsNotNull() {

addCriterion("update\_time is not null");

return (Criteria) this;

}

public Criteria andUpdateTimeEqualTo(Date value) {

addCriterion("update\_time =", value, "updateTime");

return (Criteria) this;

}

public Criteria andUpdateTimeNotEqualTo(Date value) {

addCriterion("update\_time <>", value, "updateTime");

return (Criteria) this;

}

public Criteria andUpdateTimeGreaterThan(Date value) {

addCriterion("update\_time >", value, "updateTime");

return (Criteria) this;

}

public Criteria andUpdateTimeGreaterThanOrEqualTo(Date value) {

addCriterion("update\_time >=", value, "updateTime");

return (Criteria) this;

}

public Criteria andUpdateTimeLessThan(Date value) {

addCriterion("update\_time <", value, "updateTime");

return (Criteria) this;

}

public Criteria andUpdateTimeLessThanOrEqualTo(Date value) {

addCriterion("update\_time <=", value, "updateTime");

return (Criteria) this;

}

public Criteria andUpdateTimeIn(List<Date> values) {

addCriterion("update\_time in", values, "updateTime");

return (Criteria) this;

}

public Criteria andUpdateTimeNotIn(List<Date> values) {

addCriterion("update\_time not in", values, "updateTime");

return (Criteria) this;

}

public Criteria andUpdateTimeBetween(Date value1, Date value2) {

addCriterion("update\_time between", value1, value2, "updateTime");

return (Criteria) this;

}

public Criteria andUpdateTimeNotBetween(Date value1, Date value2) {

addCriterion("update\_time not between", value1, value2, "updateTime");

return (Criteria) this;

}

}

public static class Criteria extends GeneratedCriteria {

protected Criteria() {

super();

}

}

public static class Criterion {

private String condition;

private Object value;

private Object secondValue;

private boolean noValue;

private boolean singleValue;

private boolean betweenValue;

private boolean listValue;

private String typeHandler;

public String getCondition() {

return condition;

}

public Object getValue() {

return value;

}

public Object getSecondValue() {

return secondValue;

}

public boolean isNoValue() {

return noValue;

}

public boolean isSingleValue() {

return singleValue;

}

public boolean isBetweenValue() {

return betweenValue;

}

public boolean isListValue() {

return listValue;

}

public String getTypeHandler() {

return typeHandler;

}

protected Criterion(String condition) {

super();

this.condition = condition;

this.typeHandler = null;

this.noValue = true;

}

protected Criterion(String condition, Object value, String typeHandler) {

super();

this.condition = condition;

this.value = value;

this.typeHandler = typeHandler;

if (value instanceof List<?>) {

this.listValue = true;

} else {

this.singleValue = true;

}

}

protected Criterion(String condition, Object value) {

this(condition, value, null);

}

protected Criterion(String condition, Object value, Object secondValue, String typeHandler) {

super();

this.condition = condition;

this.value = value;

this.secondValue = secondValue;

this.typeHandler = typeHandler;

this.betweenValue = true;

}

protected Criterion(String condition, Object value, Object secondValue) {

this(condition, value, secondValue, null);

}

}

}

package com.yingchong.service.data\_service.mybatis.model;

import java.util.Date;

public class AppFluxSort {

private Integer id;

private String appName;

private Double flux;

private String fluxPercentage;

private String fluxDate;

private Date createTime;

private Date updateTime;

public Integer getId() {

return id;

}

public void setId(Integer id) {

this.id = id;

}

public String getAppName() {

return appName;

}

public void setAppName(String appName) {

this.appName = appName;

}

public Double getFlux() {

return flux;

}

public void setFlux(Double flux) {

this.flux = flux;

}

public String getFluxPercentage() {

return fluxPercentage;

}

public void setFluxPercentage(String fluxPercentage) {

this.fluxPercentage = fluxPercentage;

}

public String getFluxDate() {

return fluxDate;

}

public void setFluxDate(String fluxDate) {

this.fluxDate = fluxDate;

}

public Date getCreateTime() {

return createTime;

}

public void setCreateTime(Date createTime) {

this.createTime = createTime;

}

public Date getUpdateTime() {

return updateTime;

}

public void setUpdateTime(Date updateTime) {

this.updateTime = updateTime;

}

}

package com.yingchong.service.data\_service.mybatis.model;

import java.util.ArrayList;

import java.util.Date;

import java.util.List;

public class AppFluxSortExample {

protected String orderByClause;

protected boolean distinct;

protected List<Criteria> oredCriteria;

public AppFluxSortExample() {

oredCriteria = new ArrayList<Criteria>();

}

public void setOrderByClause(String orderByClause) {

this.orderByClause = orderByClause;

}

public String getOrderByClause() {

return orderByClause;

}

public void setDistinct(boolean distinct) {

this.distinct = distinct;

}

public boolean isDistinct() {

return distinct;

}

public List<Criteria> getOredCriteria() {

return oredCriteria;

}

public void or(Criteria criteria) {

oredCriteria.add(criteria);

}

public Criteria or() {

Criteria criteria = createCriteriaInternal();

oredCriteria.add(criteria);

return criteria;

}

public Criteria createCriteria() {

Criteria criteria = createCriteriaInternal();

if (oredCriteria.size() == 0) {

oredCriteria.add(criteria);

}

return criteria;

}

protected Criteria createCriteriaInternal() {

Criteria criteria = new Criteria();

return criteria;

}

public void clear() {

oredCriteria.clear();

orderByClause = null;

distinct = false;

}

protected abstract static class GeneratedCriteria {

protected List<Criterion> criteria;

protected GeneratedCriteria() {

super();

criteria = new ArrayList<Criterion>();

}

public boolean isValid() {

return criteria.size() > 0;

}

public List<Criterion> getAllCriteria() {

return criteria;

}

public List<Criterion> getCriteria() {

return criteria;

}

protected void addCriterion(String condition) {

if (condition == null) {

throw new RuntimeException("Value for condition cannot be null");

}

criteria.add(new Criterion(condition));

}

protected void addCriterion(String condition, Object value, String property) {

if (value == null) {

throw new RuntimeException("Value for " + property + " cannot be null");

}

criteria.add(new Criterion(condition, value));

}

protected void addCriterion(String condition, Object value1, Object value2, String property) {

if (value1 == null || value2 == null) {

throw new RuntimeException("Between values for " + property + " cannot be null");

}

criteria.add(new Criterion(condition, value1, value2));

}

public Criteria andIdIsNull() {

addCriterion("id is null");

return (Criteria) this;

}

public Criteria andIdIsNotNull() {

addCriterion("id is not null");

return (Criteria) this;

}

public Criteria andIdEqualTo(Integer value) {

addCriterion("id =", value, "id");

return (Criteria) this;

}

public Criteria andIdNotEqualTo(Integer value) {

addCriterion("id <>", value, "id");

return (Criteria) this;

}

public Criteria andIdGreaterThan(Integer value) {

addCriterion("id >", value, "id");

return (Criteria) this;

}

public Criteria andIdGreaterThanOrEqualTo(Integer value) {

addCriterion("id >=", value, "id");

return (Criteria) this;

}

public Criteria andIdLessThan(Integer value) {

addCriterion("id <", value, "id");

return (Criteria) this;

}

public Criteria andIdLessThanOrEqualTo(Integer value) {

addCriterion("id <=", value, "id");

return (Criteria) this;

}

public Criteria andIdIn(List<Integer> values) {

addCriterion("id in", values, "id");

return (Criteria) this;

}

public Criteria andIdNotIn(List<Integer> values) {

addCriterion("id not in", values, "id");

return (Criteria) this;

}

public Criteria andIdBetween(Integer value1, Integer value2) {

addCriterion("id between", value1, value2, "id");

return (Criteria) this;

}

public Criteria andIdNotBetween(Integer value1, Integer value2) {

addCriterion("id not between", value1, value2, "id");

return (Criteria) this;

}

public Criteria andAppNameIsNull() {

addCriterion("app\_name is null");

return (Criteria) this;

}

public Criteria andAppNameIsNotNull() {

addCriterion("app\_name is not null");

return (Criteria) this;

}

public Criteria andAppNameEqualTo(String value) {

addCriterion("app\_name =", value, "appName");

return (Criteria) this;

}

public Criteria andAppNameNotEqualTo(String value) {

addCriterion("app\_name <>", value, "appName");

return (Criteria) this;

}

public Criteria andAppNameGreaterThan(String value) {

addCriterion("app\_name >", value, "appName");

return (Criteria) this;

}

public Criteria andAppNameGreaterThanOrEqualTo(String value) {

addCriterion("app\_name >=", value, "appName");

return (Criteria) this;

}

public Criteria andAppNameLessThan(String value) {

addCriterion("app\_name <", value, "appName");

return (Criteria) this;

}

public Criteria andAppNameLessThanOrEqualTo(String value) {

addCriterion("app\_name <=", value, "appName");

return (Criteria) this;

}

public Criteria andAppNameLike(String value) {

addCriterion("app\_name like", value, "appName");

return (Criteria) this;

}

public Criteria andAppNameNotLike(String value) {

addCriterion("app\_name not like", value, "appName");

return (Criteria) this;

}

public Criteria andAppNameIn(List<String> values) {

addCriterion("app\_name in", values, "appName");

return (Criteria) this;

}

public Criteria andAppNameNotIn(List<String> values) {

addCriterion("app\_name not in", values, "appName");

return (Criteria) this;

}

public Criteria andAppNameBetween(String value1, String value2) {

addCriterion("app\_name between", value1, value2, "appName");

return (Criteria) this;

}

public Criteria andAppNameNotBetween(String value1, String value2) {

addCriterion("app\_name not between", value1, value2, "appName");

return (Criteria) this;

}

public Criteria andFluxIsNull() {

addCriterion("flux is null");

return (Criteria) this;

}

public Criteria andFluxIsNotNull() {

addCriterion("flux is not null");

return (Criteria) this;

}

public Criteria andFluxEqualTo(Double value) {

addCriterion("flux =", value, "flux");

return (Criteria) this;

}

public Criteria andFluxNotEqualTo(Double value) {

addCriterion("flux <>", value, "flux");

return (Criteria) this;

}

public Criteria andFluxGreaterThan(Double value) {

addCriterion("flux >", value, "flux");

return (Criteria) this;

}

public Criteria andFluxGreaterThanOrEqualTo(Double value) {

addCriterion("flux >=", value, "flux");

return (Criteria) this;

}

public Criteria andFluxLessThan(Double value) {

addCriterion("flux <", value, "flux");

return (Criteria) this;

}

public Criteria andFluxLessThanOrEqualTo(Double value) {

addCriterion("flux <=", value, "flux");

return (Criteria) this;

}

public Criteria andFluxIn(List<Double> values) {

addCriterion("flux in", values, "flux");

return (Criteria) this;

}

public Criteria andFluxNotIn(List<Double> values) {

addCriterion("flux not in", values, "flux");

return (Criteria) this;

}

public Criteria andFluxBetween(Double value1, Double value2) {

addCriterion("flux between", value1, value2, "flux");

return (Criteria) this;

}

public Criteria andFluxNotBetween(Double value1, Double value2) {

addCriterion("flux not between", value1, value2, "flux");

return (Criteria) this;

}

public Criteria andFluxPercentageIsNull() {

addCriterion("flux\_percentage is null");

return (Criteria) this;

}

public Criteria andFluxPercentageIsNotNull() {

addCriterion("flux\_percentage is not null");

return (Criteria) this;

}

public Criteria andFluxPercentageEqualTo(String value) {

addCriterion("flux\_percentage =", value, "fluxPercentage");

return (Criteria) this;

}

public Criteria andFluxPercentageNotEqualTo(String value) {

addCriterion("flux\_percentage <>", value, "fluxPercentage");

return (Criteria) this;

}

public Criteria andFluxPercentageGreaterThan(String value) {

addCriterion("flux\_percentage >", value, "fluxPercentage");

return (Criteria) this;

}

public Criteria andFluxPercentageGreaterThanOrEqualTo(String value) {

addCriterion("flux\_percentage >=", value, "fluxPercentage");

return (Criteria) this;

}

public Criteria andFluxPercentageLessThan(String value) {

addCriterion("flux\_percentage <", value, "fluxPercentage");

return (Criteria) this;

}

public Criteria andFluxPercentageLessThanOrEqualTo(String value) {

addCriterion("flux\_percentage <=", value, "fluxPercentage");

return (Criteria) this;

}

public Criteria andFluxPercentageLike(String value) {

addCriterion("flux\_percentage like", value, "fluxPercentage");

return (Criteria) this;

}

public Criteria andFluxPercentageNotLike(String value) {

addCriterion("flux\_percentage not like", value, "fluxPercentage");

return (Criteria) this;

}

public Criteria andFluxPercentageIn(List<String> values) {

addCriterion("flux\_percentage in", values, "fluxPercentage");

return (Criteria) this;

}

public Criteria andFluxPercentageNotIn(List<String> values) {

addCriterion("flux\_percentage not in", values, "fluxPercentage");

return (Criteria) this;

}

public Criteria andFluxPercentageBetween(String value1, String value2) {

addCriterion("flux\_percentage between", value1, value2, "fluxPercentage");

return (Criteria) this;

}

public Criteria andFluxPercentageNotBetween(String value1, String value2) {

addCriterion("flux\_percentage not between", value1, value2, "fluxPercentage");

return (Criteria) this;

}

public Criteria andFluxDateIsNull() {

addCriterion("flux\_date is null");

return (Criteria) this;

}

public Criteria andFluxDateIsNotNull() {

addCriterion("flux\_date is not null");

return (Criteria) this;

}

public Criteria andFluxDateEqualTo(String value) {

addCriterion("flux\_date =", value, "fluxDate");

return (Criteria) this;

}

public Criteria andFluxDateNotEqualTo(String value) {

addCriterion("flux\_date <>", value, "fluxDate");

return (Criteria) this;

}

public Criteria andFluxDateGreaterThan(String value) {

addCriterion("flux\_date >", value, "fluxDate");

return (Criteria) this;

}

public Criteria andFluxDateGreaterThanOrEqualTo(String value) {

addCriterion("flux\_date >=", value, "fluxDate");

return (Criteria) this;

}

public Criteria andFluxDateLessThan(String value) {

addCriterion("flux\_date <", value, "fluxDate");

return (Criteria) this;

}

public Criteria andFluxDateLessThanOrEqualTo(String value) {

addCriterion("flux\_date <=", value, "fluxDate");

return (Criteria) this;

}

public Criteria andFluxDateLike(String value) {

addCriterion("flux\_date like", value, "fluxDate");

return (Criteria) this;

}

public Criteria andFluxDateNotLike(String value) {

addCriterion("flux\_date not like", value, "fluxDate");

return (Criteria) this;

}

public Criteria andFluxDateIn(List<String> values) {

addCriterion("flux\_date in", values, "fluxDate");

return (Criteria) this;

}

public Criteria andFluxDateNotIn(List<String> values) {

addCriterion("flux\_date not in", values, "fluxDate");

return (Criteria) this;

}

public Criteria andFluxDateBetween(String value1, String value2) {

addCriterion("flux\_date between", value1, value2, "fluxDate");

return (Criteria) this;

}

public Criteria andFluxDateNotBetween(String value1, String value2) {

addCriterion("flux\_date not between", value1, value2, "fluxDate");

return (Criteria) this;

}

public Criteria andCreateTimeIsNull() {

addCriterion("create\_time is null");

return (Criteria) this;

}

public Criteria andCreateTimeIsNotNull() {

addCriterion("create\_time is not null");

return (Criteria) this;

}

public Criteria andCreateTimeEqualTo(Date value) {

addCriterion("create\_time =", value, "createTime");

return (Criteria) this;

}

public Criteria andCreateTimeNotEqualTo(Date value) {

addCriterion("create\_time <>", value, "createTime");

return (Criteria) this;

}

public Criteria andCreateTimeGreaterThan(Date value) {

addCriterion("create\_time >", value, "createTime");

return (Criteria) this;

}

public Criteria andCreateTimeGreaterThanOrEqualTo(Date value) {

addCriterion("create\_time >=", value, "createTime");

return (Criteria) this;

}

public Criteria andCreateTimeLessThan(Date value) {

addCriterion("create\_time <", value, "createTime");

return (Criteria) this;

}

public Criteria andCreateTimeLessThanOrEqualTo(Date value) {

addCriterion("create\_time <=", value, "createTime");

return (Criteria) this;

}

public Criteria andCreateTimeIn(List<Date> values) {

addCriterion("create\_time in", values, "createTime");

return (Criteria) this;

}

public Criteria andCreateTimeNotIn(List<Date> values) {

addCriterion("create\_time not in", values, "createTime");

return (Criteria) this;

}

public Criteria andCreateTimeBetween(Date value1, Date value2) {

addCriterion("create\_time between", value1, value2, "createTime");

return (Criteria) this;

}

public Criteria andCreateTimeNotBetween(Date value1, Date value2) {

addCriterion("create\_time not between", value1, value2, "createTime");

return (Criteria) this;

}

public Criteria andUpdateTimeIsNull() {

addCriterion("update\_time is null");

return (Criteria) this;

}

public Criteria andUpdateTimeIsNotNull() {

addCriterion("update\_time is not null");

return (Criteria) this;

}

public Criteria andUpdateTimeEqualTo(Date value) {

addCriterion("update\_time =", value, "updateTime");

return (Criteria) this;

}

public Criteria andUpdateTimeNotEqualTo(Date value) {

addCriterion("update\_time <>", value, "updateTime");

return (Criteria) this;

}

public Criteria andUpdateTimeGreaterThan(Date value) {

addCriterion("update\_time >", value, "updateTime");

return (Criteria) this;

}

public Criteria andUpdateTimeGreaterThanOrEqualTo(Date value) {

addCriterion("update\_time >=", value, "updateTime");

return (Criteria) this;

}

public Criteria andUpdateTimeLessThan(Date value) {

addCriterion("update\_time <", value, "updateTime");

return (Criteria) this;

}

public Criteria andUpdateTimeLessThanOrEqualTo(Date value) {

addCriterion("update\_time <=", value, "updateTime");

return (Criteria) this;

}

public Criteria andUpdateTimeIn(List<Date> values) {

addCriterion("update\_time in", values, "updateTime");

return (Criteria) this;

}

public Criteria andUpdateTimeNotIn(List<Date> values) {

addCriterion("update\_time not in", values, "updateTime");

return (Criteria) this;

}

public Criteria andUpdateTimeBetween(Date value1, Date value2) {

addCriterion("update\_time between", value1, value2, "updateTime");

return (Criteria) this;

}

public Criteria andUpdateTimeNotBetween(Date value1, Date value2) {

addCriterion("update\_time not between", value1, value2, "updateTime");

return (Criteria) this;

}

}

public static class Criteria extends GeneratedCriteria {

protected Criteria() {

super();

}

}

public static class Criterion {

private String condition;

private Object value;

private Object secondValue;

private boolean noValue;

private boolean singleValue;

private boolean betweenValue;

private boolean listValue;

private String typeHandler;

public String getCondition() {

return condition;

}

public Object getValue() {

return value;

}

public Object getSecondValue() {

return secondValue;

}

public boolean isNoValue() {

return noValue;

}

public boolean isSingleValue() {

return singleValue;

}

public boolean isBetweenValue() {

return betweenValue;

}

public boolean isListValue() {

return listValue;

}

public String getTypeHandler() {

return typeHandler;

}

protected Criterion(String condition) {

super();

this.condition = condition;

this.typeHandler = null;

this.noValue = true;

}

protected Criterion(String condition, Object value, String typeHandler) {

super();

this.condition = condition;

this.value = value;

this.typeHandler = typeHandler;

if (value instanceof List<?>) {

this.listValue = true;

} else {

this.singleValue = true;

}

}

protected Criterion(String condition, Object value) {

this(condition, value, null);

}

protected Criterion(String condition, Object value, Object secondValue, String typeHandler) {

super();

this.condition = condition;

this.value = value;

this.secondValue = secondValue;

this.typeHandler = typeHandler;

this.betweenValue = true;

}

protected Criterion(String condition, Object value, Object secondValue) {

this(condition, value, secondValue, null);

}

}

}

package com.yingchong.service.data\_service.mybatis.mapper;

import com.yingchong.service.data\_service.mybatis.model.FluxResult;

import com.yingchong.service.data\_service.mybatis.model.FluxResultExample;

import org.apache.ibatis.annotations.\*;

import org.apache.ibatis.type.JdbcType;

import org.springframework.stereotype.Repository;

import java.util.List;

@Repository

public interface FluxResultMapper {

@SelectProvider(type=FluxResultSqlProvider.class, method="countByExample")

long countByExample(FluxResultExample example);

@DeleteProvider(type=FluxResultSqlProvider.class, method="deleteByExample")

int deleteByExample(FluxResultExample example);

@Delete({

"delete from flux\_result",

"where id = #{id,jdbcType=INTEGER}"

})

int deleteByPrimaryKey(Integer id);

@Insert({

"insert into flux\_result (id, upload, ",

"download, flux\_date, ",

"create\_time, update\_time)",

"values (#{id,jdbcType=INTEGER}, #{upload,jdbcType=DOUBLE}, ",

"#{download,jdbcType=DOUBLE}, #{fluxDate,jdbcType=DATE}, ",

"#{createTime,jdbcType=TIMESTAMP}, #{updateTime,jdbcType=TIMESTAMP})"

})

@Options(useGeneratedKeys = true, keyProperty = "id", keyColumn = "id")

int insert(FluxResult record);

@InsertProvider(type=FluxResultSqlProvider.class, method="insertSelective")

int insertSelective(FluxResult record);

@SelectProvider(type=FluxResultSqlProvider.class, method="selectByExample")

@Results({

@Result(column="id", property="id", jdbcType=JdbcType.INTEGER, id=true),

@Result(column="upload", property="upload", jdbcType=JdbcType.DOUBLE),

@Result(column="download", property="download", jdbcType=JdbcType.DOUBLE),

@Result(column="flux\_date", property="fluxDate", jdbcType=JdbcType.DATE),

@Result(column="create\_time", property="createTime", jdbcType=JdbcType.TIMESTAMP),

@Result(column="update\_time", property="updateTime", jdbcType=JdbcType.TIMESTAMP)

})

List<FluxResult> selectByExample(FluxResultExample example);

@Select({

"select",

"id, upload, download, flux\_date, create\_time, update\_time",

"from flux\_result",

"where id = #{id,jdbcType=INTEGER}"

})

@Results({

@Result(column="id", property="id", jdbcType=JdbcType.INTEGER, id=true),

@Result(column="upload", property="upload", jdbcType=JdbcType.DOUBLE),

@Result(column="download", property="download", jdbcType=JdbcType.DOUBLE),

@Result(column="flux\_date", property="fluxDate", jdbcType=JdbcType.DATE),

@Result(column="create\_time", property="createTime", jdbcType=JdbcType.TIMESTAMP),

@Result(column="update\_time", property="updateTime", jdbcType=JdbcType.TIMESTAMP)

})

FluxResult selectByPrimaryKey(Integer id);

@UpdateProvider(type=FluxResultSqlProvider.class, method="updateByExampleSelective")

int updateByExampleSelective(@Param("record") FluxResult record, @Param("example") FluxResultExample example);

@UpdateProvider(type=FluxResultSqlProvider.class, method="updateByExample")

int updateByExample(@Param("record") FluxResult record, @Param("example") FluxResultExample example);

@UpdateProvider(type=FluxResultSqlProvider.class, method="updateByPrimaryKeySelective")

int updateByPrimaryKeySelective(FluxResult record);

@Update({

"update flux\_result",

"set upload = #{upload,jdbcType=DOUBLE},",

"download = #{download,jdbcType=DOUBLE},",

"flux\_date = #{fluxDate,jdbcType=DATE},",

"create\_time = #{createTime,jdbcType=TIMESTAMP},",

"update\_time = #{updateTime,jdbcType=TIMESTAMP}",

"where id = #{id,jdbcType=INTEGER}"

})

int updateByPrimaryKey(FluxResult record);

}

package com.yingchong.service.data\_service.mybatis.mapper;

import com.yingchong.service.data\_service.mybatis.model.AppFluxSort;

import com.yingchong.service.data\_service.mybatis.model.AppFluxSortExample;

import com.yingchong.service.data\_service.mybatis.model.AppFluxSortExample.Criteria;

import com.yingchong.service.data\_service.mybatis.model.AppFluxSortExample.Criterion;

import org.apache.ibatis.jdbc.SQL;

import java.util.List;

import java.util.Map;

public class AppFluxSortSqlProvider {

public String countByExample(AppFluxSortExample example) {

SQL sql = new SQL();

sql.SELECT("count(\*)").FROM("app\_flux\_sort");

applyWhere(sql, example, false);

return sql.toString();

}

public String deleteByExample(AppFluxSortExample example) {

SQL sql = new SQL();

sql.DELETE\_FROM("app\_flux\_sort");

applyWhere(sql, example, false);

return sql.toString();

}

public String insertSelective(AppFluxSort record) {

SQL sql = new SQL();

sql.INSERT\_INTO("app\_flux\_sort");

if (record.getId() != null) {

sql.VALUES("id", "#{id,jdbcType=INTEGER}");

}

if (record.getAppName() != null) {

sql.VALUES("app\_name", "#{appName,jdbcType=VARCHAR}");

}

if (record.getFlux() != null) {

sql.VALUES("flux", "#{flux,jdbcType=DOUBLE}");

}

if (record.getFluxPercentage() != null) {

sql.VALUES("flux\_percentage", "#{fluxPercentage,jdbcType=VARCHAR}");

}

if (record.getFluxDate() != null) {

sql.VALUES("flux\_date", "#{fluxDate,jdbcType=VARCHAR}");

}

if (record.getCreateTime() != null) {

sql.VALUES("create\_time", "#{createTime,jdbcType=TIMESTAMP}");

}

if (record.getUpdateTime() != null) {

sql.VALUES("update\_time", "#{updateTime,jdbcType=TIMESTAMP}");

}

return sql.toString();

}

public String selectByExample(AppFluxSortExample example) {

SQL sql = new SQL();

if (example != null && example.isDistinct()) {

sql.SELECT\_DISTINCT("id");

} else {

sql.SELECT("id");

}

sql.SELECT("app\_name");

sql.SELECT("flux");

sql.SELECT("flux\_percentage");

sql.SELECT("flux\_date");

sql.SELECT("create\_time");

sql.SELECT("update\_time");

sql.FROM("app\_flux\_sort");

applyWhere(sql, example, false);

if (example != null && example.getOrderByClause() != null) {

sql.ORDER\_BY(example.getOrderByClause());

}

return sql.toString();

}

public String updateByExampleSelective(Map<String, Object> parameter) {

AppFluxSort record = (AppFluxSort) parameter.get("record");

AppFluxSortExample example = (AppFluxSortExample) parameter.get("example");

SQL sql = new SQL();

sql.UPDATE("app\_flux\_sort");

if (record.getId() != null) {

sql.SET("id = #{record.id,jdbcType=INTEGER}");

}

if (record.getAppName() != null) {

sql.SET("app\_name = #{record.appName,jdbcType=VARCHAR}");

}

if (record.getFlux() != null) {

sql.SET("flux = #{record.flux,jdbcType=DOUBLE}");

}

if (record.getFluxPercentage() != null) {

sql.SET("flux\_percentage = #{record.fluxPercentage,jdbcType=VARCHAR}");

}

if (record.getFluxDate() != null) {

sql.SET("flux\_date = #{record.fluxDate,jdbcType=VARCHAR}");

}

if (record.getCreateTime() != null) {

sql.SET("create\_time = #{record.createTime,jdbcType=TIMESTAMP}");

}

if (record.getUpdateTime() != null) {

sql.SET("update\_time = #{record.updateTime,jdbcType=TIMESTAMP}");

}

applyWhere(sql, example, true);

return sql.toString();

}

public String updateByExample(Map<String, Object> parameter) {

SQL sql = new SQL();

sql.UPDATE("app\_flux\_sort");

sql.SET("id = #{record.id,jdbcType=INTEGER}");

sql.SET("app\_name = #{record.appName,jdbcType=VARCHAR}");

sql.SET("flux = #{record.flux,jdbcType=DOUBLE}");

sql.SET("flux\_percentage = #{record.fluxPercentage,jdbcType=VARCHAR}");

sql.SET("flux\_date = #{record.fluxDate,jdbcType=VARCHAR}");

sql.SET("create\_time = #{record.createTime,jdbcType=TIMESTAMP}");

sql.SET("update\_time = #{record.updateTime,jdbcType=TIMESTAMP}");

AppFluxSortExample example = (AppFluxSortExample) parameter.get("example");

applyWhere(sql, example, true);

return sql.toString();

}

public String updateByPrimaryKeySelective(AppFluxSort record) {

SQL sql = new SQL();

sql.UPDATE("app\_flux\_sort");

if (record.getAppName() != null) {

sql.SET("app\_name = #{appName,jdbcType=VARCHAR}");

}

if (record.getFlux() != null) {

sql.SET("flux = #{flux,jdbcType=DOUBLE}");

}

if (record.getFluxPercentage() != null) {

sql.SET("flux\_percentage = #{fluxPercentage,jdbcType=VARCHAR}");

}

if (record.getFluxDate() != null) {

sql.SET("flux\_date = #{fluxDate,jdbcType=VARCHAR}");

}

if (record.getCreateTime() != null) {

sql.SET("create\_time = #{createTime,jdbcType=TIMESTAMP}");

}

if (record.getUpdateTime() != null) {

sql.SET("update\_time = #{updateTime,jdbcType=TIMESTAMP}");

}

sql.WHERE("id = #{id,jdbcType=INTEGER}");

return sql.toString();

}

protected void applyWhere(SQL sql, AppFluxSortExample example, boolean includeExamplePhrase) {

if (example == null) {

return;

}

String parmPhrase1;

String parmPhrase1\_th;

String parmPhrase2;

String parmPhrase2\_th;

String parmPhrase3;

String parmPhrase3\_th;

if (includeExamplePhrase) {

parmPhrase1 = "%s #{example.oredCriteria[%d].allCriteria[%d].value}";

parmPhrase1\_th = "%s #{example.oredCriteria[%d].allCriteria[%d].value,typeHandler=%s}";

parmPhrase2 = "%s #{example.oredCriteria[%d].allCriteria[%d].value} and #{example.oredCriteria[%d].criteria[%d].secondValue}";

parmPhrase2\_th = "%s #{example.oredCriteria[%d].allCriteria[%d].value,typeHandler=%s} and #{example.oredCriteria[%d].criteria[%d].secondValue,typeHandler=%s}";

parmPhrase3 = "#{example.oredCriteria[%d].allCriteria[%d].value[%d]}";

parmPhrase3\_th = "#{example.oredCriteria[%d].allCriteria[%d].value[%d],typeHandler=%s}";

} else {

parmPhrase1 = "%s #{oredCriteria[%d].allCriteria[%d].value}";

parmPhrase1\_th = "%s #{oredCriteria[%d].allCriteria[%d].value,typeHandler=%s}";

parmPhrase2 = "%s #{oredCriteria[%d].allCriteria[%d].value} and #{oredCriteria[%d].criteria[%d].secondValue}";

parmPhrase2\_th = "%s #{oredCriteria[%d].allCriteria[%d].value,typeHandler=%s} and #{oredCriteria[%d].criteria[%d].secondValue,typeHandler=%s}";

parmPhrase3 = "#{oredCriteria[%d].allCriteria[%d].value[%d]}";

parmPhrase3\_th = "#{oredCriteria[%d].allCriteria[%d].value[%d],typeHandler=%s}";

}

StringBuilder sb = new StringBuilder();

List<Criteria> oredCriteria = example.getOredCriteria();

boolean firstCriteria = true;

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

Criteria criteria = oredCriteria.get(i);

if (criteria.isValid()) {

if (firstCriteria) {

firstCriteria = false;

} else {

sb.append(" or ");

}

sb.append('(');

List<Criterion> criterions = criteria.getAllCriteria();

boolean firstCriterion = true;

for (int j = 0; j < criterions.size(); j++) {

Criterion criterion = criterions.get(j);

if (firstCriterion) {

firstCriterion = false;

} else {

sb.append(" and ");

}

if (criterion.isNoValue()) {

sb.append(criterion.getCondition());

} else if (criterion.isSingleValue()) {

if (criterion.getTypeHandler() == null) {

sb.append(String.format(parmPhrase1, criterion.getCondition(), i, j));

} else {

sb.append(String.format(parmPhrase1\_th, criterion.getCondition(), i, j,criterion.getTypeHandler()));

}

} else if (criterion.isBetweenValue()) {

if (criterion.getTypeHandler() == null) {

sb.append(String.format(parmPhrase2, criterion.getCondition(), i, j, i, j));

} else {

sb.append(String.format(parmPhrase2\_th, criterion.getCondition(), i, j, criterion.getTypeHandler(), i, j, criterion.getTypeHandler()));

}

} else if (criterion.isListValue()) {

sb.append(criterion.getCondition());

sb.append(" (");

List<?> listItems = (List<?>) criterion.getValue();

boolean comma = false;

for (int k = 0; k < listItems.size(); k++) {

if (comma) {

sb.append(", ");

} else {

comma = true;

}

if (criterion.getTypeHandler() == null) {

sb.append(String.format(parmPhrase3, i, j, k));

} else {

sb.append(String.format(parmPhrase3\_th, i, j, k, criterion.getTypeHandler()));

}

}

sb.append(')');

}

}

sb.append(')');

}

}

if (sb.length() > 0) {

sql.WHERE(sb.toString());

}

}

}

package com.yingchong.service.data\_service.mybatis.mapper;

import com.yingchong.service.data\_service.mybatis.model.ActionType;

import com.yingchong.service.data\_service.mybatis.model.ActionTypeExample.Criteria;

import com.yingchong.service.data\_service.mybatis.model.ActionTypeExample.Criterion;

import com.yingchong.service.data\_service.mybatis.model.ActionTypeExample;

import java.util.List;

import java.util.Map;

import org.apache.ibatis.jdbc.SQL;

public class ActionTypeSqlProvider {

public String countByExample(ActionTypeExample example) {

SQL sql = new SQL();

sql.SELECT("count(\*)").FROM("action\_type");

applyWhere(sql, example, false);

return sql.toString();

}

public String deleteByExample(ActionTypeExample example) {

SQL sql = new SQL();

sql.DELETE\_FROM("action\_type");

applyWhere(sql, example, false);

return sql.toString();

}

public String insertSelective(ActionType record) {

SQL sql = new SQL();

sql.INSERT\_INTO("action\_type");

if (record.getId() != null) {

sql.VALUES("id", "#{id,jdbcType=INTEGER}");

}

if (record.getActionName() != null) {

sql.VALUES("action\_name", "#{actionName,jdbcType=VARCHAR}");

}

if (record.getActionCount() != null) {

sql.VALUES("action\_count", "#{actionCount,jdbcType=INTEGER}");

}

if (record.getActionDate() != null) {

sql.VALUES("action\_date", "#{actionDate,jdbcType=DATE}");

}

if (record.getCreateTime() != null) {

sql.VALUES("create\_time", "#{createTime,jdbcType=TIMESTAMP}");

}

if (record.getUpdateTime() != null) {

sql.VALUES("update\_time", "#{updateTime,jdbcType=TIMESTAMP}");

}

return sql.toString();

}

public String selectByExample(ActionTypeExample example) {

SQL sql = new SQL();

if (example != null && example.isDistinct()) {

sql.SELECT\_DISTINCT("id");

} else {

sql.SELECT("id");

}

sql.SELECT("action\_name");

sql.SELECT("action\_count");

sql.SELECT("action\_date");

sql.SELECT("create\_time");

sql.SELECT("update\_time");

sql.FROM("action\_type");

applyWhere(sql, example, false);

if (example != null && example.getOrderByClause() != null) {

sql.ORDER\_BY(example.getOrderByClause());

}

return sql.toString();

}

public String updateByExampleSelective(Map<String, Object> parameter) {

ActionType record = (ActionType) parameter.get("record");

ActionTypeExample example = (ActionTypeExample) parameter.get("example");

SQL sql = new SQL();

sql.UPDATE("action\_type");

if (record.getId() != null) {

sql.SET("id = #{record.id,jdbcType=INTEGER}");

}

if (record.getActionName() != null) {

sql.SET("action\_name = #{record.actionName,jdbcType=VARCHAR}");

}

if (record.getActionCount() != null) {

sql.SET("action\_count = #{record.actionCount,jdbcType=INTEGER}");

}

if (record.getActionDate() != null) {

sql.SET("action\_date = #{record.actionDate,jdbcType=DATE}");

}

if (record.getCreateTime() != null) {

sql.SET("create\_time = #{record.createTime,jdbcType=TIMESTAMP}");

}

if (record.getUpdateTime() != null) {

sql.SET("update\_time = #{record.updateTime,jdbcType=TIMESTAMP}");

}

applyWhere(sql, example, true);

return sql.toString();

}

public String updateByExample(Map<String, Object> parameter) {

SQL sql = new SQL();

sql.UPDATE("action\_type");

sql.SET("id = #{record.id,jdbcType=INTEGER}");

sql.SET("action\_name = #{record.actionName,jdbcType=VARCHAR}");

sql.SET("action\_count = #{record.actionCount,jdbcType=INTEGER}");

sql.SET("action\_date = #{record.actionDate,jdbcType=DATE}");

sql.SET("create\_time = #{record.createTime,jdbcType=TIMESTAMP}");

sql.SET("update\_time = #{record.updateTime,jdbcType=TIMESTAMP}");

ActionTypeExample example = (ActionTypeExample) parameter.get("example");

applyWhere(sql, example, true);

return sql.toString();

}

public String updateByPrimaryKeySelective(ActionType record) {

SQL sql = new SQL();

sql.UPDATE("action\_type");

if (record.getActionName() != null) {

sql.SET("action\_name = #{actionName,jdbcType=VARCHAR}");

}

if (record.getActionCount() != null) {

sql.SET("action\_count = #{actionCount,jdbcType=INTEGER}");

}

if (record.getActionDate() != null) {

sql.SET("action\_date = #{actionDate,jdbcType=DATE}");

}

if (record.getCreateTime() != null) {

sql.SET("create\_time = #{createTime,jdbcType=TIMESTAMP}");

}

if (record.getUpdateTime() != null) {

sql.SET("update\_time = #{updateTime,jdbcType=TIMESTAMP}");

}

sql.WHERE("id = #{id,jdbcType=INTEGER}");

return sql.toString();

}

protected void applyWhere(SQL sql, ActionTypeExample example, boolean includeExamplePhrase) {

if (example == null) {

return;

}

String parmPhrase1;

String parmPhrase1\_th;

String parmPhrase2;

String parmPhrase2\_th;

String parmPhrase3;

String parmPhrase3\_th;

if (includeExamplePhrase) {

parmPhrase1 = "%s #{example.oredCriteria[%d].allCriteria[%d].value}";

parmPhrase1\_th = "%s #{example.oredCriteria[%d].allCriteria[%d].value,typeHandler=%s}";

parmPhrase2 = "%s #{example.oredCriteria[%d].allCriteria[%d].value} and #{example.oredCriteria[%d].criteria[%d].secondValue}";

parmPhrase2\_th = "%s #{example.oredCriteria[%d].allCriteria[%d].value,typeHandler=%s} and #{example.oredCriteria[%d].criteria[%d].secondValue,typeHandler=%s}";

parmPhrase3 = "#{example.oredCriteria[%d].allCriteria[%d].value[%d]}";

parmPhrase3\_th = "#{example.oredCriteria[%d].allCriteria[%d].value[%d],typeHandler=%s}";

} else {

parmPhrase1 = "%s #{oredCriteria[%d].allCriteria[%d].value}";

parmPhrase1\_th = "%s #{oredCriteria[%d].allCriteria[%d].value,typeHandler=%s}";

parmPhrase2 = "%s #{oredCriteria[%d].allCriteria[%d].value} and #{oredCriteria[%d].criteria[%d].secondValue}";

parmPhrase2\_th = "%s #{oredCriteria[%d].allCriteria[%d].value,typeHandler=%s} and #{oredCriteria[%d].criteria[%d].secondValue,typeHandler=%s}";

parmPhrase3 = "#{oredCriteria[%d].allCriteria[%d].value[%d]}";

parmPhrase3\_th = "#{oredCriteria[%d].allCriteria[%d].value[%d],typeHandler=%s}";

}

StringBuilder sb = new StringBuilder();

List<Criteria> oredCriteria = example.getOredCriteria();

boolean firstCriteria = true;

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

Criteria criteria = oredCriteria.get(i);

if (criteria.isValid()) {

if (firstCriteria) {

firstCriteria = false;

} else {

sb.append(" or ");

}

sb.append('(');

List<Criterion> criterions = criteria.getAllCriteria();

boolean firstCriterion = true;

for (int j = 0; j < criterions.size(); j++) {

Criterion criterion = criterions.get(j);

if (firstCriterion) {

firstCriterion = false;

} else {

sb.append(" and ");

}

if (criterion.isNoValue()) {

sb.append(criterion.getCondition());

} else if (criterion.isSingleValue()) {

if (criterion.getTypeHandler() == null) {

sb.append(String.format(parmPhrase1, criterion.getCondition(), i, j));

} else {

sb.append(String.format(parmPhrase1\_th, criterion.getCondition(), i, j,criterion.getTypeHandler()));

}

} else if (criterion.isBetweenValue()) {

if (criterion.getTypeHandler() == null) {

sb.append(String.format(parmPhrase2, criterion.getCondition(), i, j, i, j));

} else {

sb.append(String.format(parmPhrase2\_th, criterion.getCondition(), i, j, criterion.getTypeHandler(), i, j, criterion.getTypeHandler()));

}

} else if (criterion.isListValue()) {

sb.append(criterion.getCondition());

sb.append(" (");

List<?> listItems = (List<?>) criterion.getValue();

boolean comma = false;

for (int k = 0; k < listItems.size(); k++) {

if (comma) {

sb.append(", ");

} else {

comma = true;

}

if (criterion.getTypeHandler() == null) {

sb.append(String.format(parmPhrase3, i, j, k));

} else {

sb.append(String.format(parmPhrase3\_th, i, j, k, criterion.getTypeHandler()));

}

}

sb.append(')');

}

}

sb.append(')');

}

}

if (sb.length() > 0) {

sql.WHERE(sb.toString());

}

}

}

package com.yingchong.service.data\_service.mybatis.mapper;

import com.yingchong.service.data\_service.mybatis.model.OnlineTime;

import com.yingchong.service.data\_service.mybatis.model.OnlineTimeExample.Criteria;

import com.yingchong.service.data\_service.mybatis.model.OnlineTimeExample.Criterion;

import com.yingchong.service.data\_service.mybatis.model.OnlineTimeExample;

import java.util.List;

import java.util.Map;

import org.apache.ibatis.jdbc.SQL;

public class OnlineTimeSqlProvider {

public String countByExample(OnlineTimeExample example) {

SQL sql = new SQL();

sql.SELECT("count(\*)").FROM("online\_time");

applyWhere(sql, example, false);

return sql.toString();

}

public String deleteByExample(OnlineTimeExample example) {

SQL sql = new SQL();

sql.DELETE\_FROM("online\_time");

applyWhere(sql, example, false);

return sql.toString();

}

public String insertSelective(OnlineTime record) {

SQL sql = new SQL();

sql.INSERT\_INTO("online\_time");

if (record.getId() != null) {

sql.VALUES("id", "#{id,jdbcType=INTEGER}");

}

if (record.getOnlineTime() != null) {

sql.VALUES("online\_time", "#{onlineTime,jdbcType=BIGINT}");

}

if (record.getResultDate() != null) {

sql.VALUES("result\_date", "#{resultDate,jdbcType=DATE}");

}

if (record.getCreateTime() != null) {

sql.VALUES("create\_time", "#{createTime,jdbcType=TIMESTAMP}");

}

if (record.getUpdateTime() != null) {

sql.VALUES("update\_time", "#{updateTime,jdbcType=TIMESTAMP}");

}

return sql.toString();

}

public String selectByExample(OnlineTimeExample example) {

SQL sql = new SQL();

if (example != null && example.isDistinct()) {

sql.SELECT\_DISTINCT("id");

} else {

sql.SELECT("id");

}

sql.SELECT("online\_time");

sql.SELECT("result\_date");

sql.SELECT("create\_time");

sql.SELECT("update\_time");

sql.FROM("online\_time");

applyWhere(sql, example, false);

if (example != null && example.getOrderByClause() != null) {

sql.ORDER\_BY(example.getOrderByClause());

}

return sql.toString();

}

public String updateByExampleSelective(Map<String, Object> parameter) {

OnlineTime record = (OnlineTime) parameter.get("record");

OnlineTimeExample example = (OnlineTimeExample) parameter.get("example");

SQL sql = new SQL();

sql.UPDATE("online\_time");

if (record.getId() != null) {

sql.SET("id = #{record.id,jdbcType=INTEGER}");

}

if (record.getOnlineTime() != null) {

sql.SET("online\_time = #{record.onlineTime,jdbcType=BIGINT}");

}

if (record.getResultDate() != null) {

sql.SET("result\_date = #{record.resultDate,jdbcType=DATE}");

}

if (record.getCreateTime() != null) {

sql.SET("create\_time = #{record.createTime,jdbcType=TIMESTAMP}");

}

if (record.getUpdateTime() != null) {

sql.SET("update\_time = #{record.updateTime,jdbcType=TIMESTAMP}");

}

applyWhere(sql, example, true);

return sql.toString();

}

public String updateByExample(Map<String, Object> parameter) {

SQL sql = new SQL();

sql.UPDATE("online\_time");

sql.SET("id = #{record.id,jdbcType=INTEGER}");

sql.SET("online\_time = #{record.onlineTime,jdbcType=BIGINT}");

sql.SET("result\_date = #{record.resultDate,jdbcType=DATE}");

sql.SET("create\_time = #{record.createTime,jdbcType=TIMESTAMP}");

sql.SET("update\_time = #{record.updateTime,jdbcType=TIMESTAMP}");

OnlineTimeExample example = (OnlineTimeExample) parameter.get("example");

applyWhere(sql, example, true);

return sql.toString();

}

public String updateByPrimaryKeySelective(OnlineTime record) {

SQL sql = new SQL();

sql.UPDATE("online\_time");

if (record.getOnlineTime() != null) {

sql.SET("online\_time = #{onlineTime,jdbcType=BIGINT}");

}

if (record.getResultDate() != null) {

sql.SET("result\_date = #{resultDate,jdbcType=DATE}");

}

if (record.getCreateTime() != null) {

sql.SET("create\_time = #{createTime,jdbcType=TIMESTAMP}");

}

if (record.getUpdateTime() != null) {

sql.SET("update\_time = #{updateTime,jdbcType=TIMESTAMP}");

}

sql.WHERE("id = #{id,jdbcType=INTEGER}");

return sql.toString();

}

protected void applyWhere(SQL sql, OnlineTimeExample example, boolean includeExamplePhrase) {

if (example == null) {

return;

}

String parmPhrase1;

String parmPhrase1\_th;

String parmPhrase2;

String parmPhrase2\_th;

String parmPhrase3;

String parmPhrase3\_th;

if (includeExamplePhrase) {

parmPhrase1 = "%s #{example.oredCriteria[%d].allCriteria[%d].value}";

parmPhrase1\_th = "%s #{example.oredCriteria[%d].allCriteria[%d].value,typeHandler=%s}";

parmPhrase2 = "%s #{example.oredCriteria[%d].allCriteria[%d].value} and #{example.oredCriteria[%d].criteria[%d].secondValue}";

parmPhrase2\_th = "%s #{example.oredCriteria[%d].allCriteria[%d].value,typeHandler=%s} and #{example.oredCriteria[%d].criteria[%d].secondValue,typeHandler=%s}";

parmPhrase3 = "#{example.oredCriteria[%d].allCriteria[%d].value[%d]}";

parmPhrase3\_th = "#{example.oredCriteria[%d].allCriteria[%d].value[%d],typeHandler=%s}";

} else {

parmPhrase1 = "%s #{oredCriteria[%d].allCriteria[%d].value}";

parmPhrase1\_th = "%s #{oredCriteria[%d].allCriteria[%d].value,typeHandler=%s}";

parmPhrase2 = "%s #{oredCriteria[%d].allCriteria[%d].value} and #{oredCriteria[%d].criteria[%d].secondValue}";

parmPhrase2\_th = "%s #{oredCriteria[%d].allCriteria[%d].value,typeHandler=%s} and #{oredCriteria[%d].criteria[%d].secondValue,typeHandler=%s}";

parmPhrase3 = "#{oredCriteria[%d].allCriteria[%d].value[%d]}";

parmPhrase3\_th = "#{oredCriteria[%d].allCriteria[%d].value[%d],typeHandler=%s}";

}

StringBuilder sb = new StringBuilder();

List<Criteria> oredCriteria = example.getOredCriteria();

boolean firstCriteria = true;

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

Criteria criteria = oredCriteria.get(i);

if (criteria.isValid()) {

if (firstCriteria) {

firstCriteria = false;

} else {

sb.append(" or ");

}

sb.append('(');

List<Criterion> criterions = criteria.getAllCriteria();

boolean firstCriterion = true;

for (int j = 0; j < criterions.size(); j++) {

Criterion criterion = criterions.get(j);

if (firstCriterion) {

firstCriterion = false;

} else {

sb.append(" and ");

}

if (criterion.isNoValue()) {

sb.append(criterion.getCondition());

} else if (criterion.isSingleValue()) {

if (criterion.getTypeHandler() == null) {

sb.append(String.format(parmPhrase1, criterion.getCondition(), i, j));

} else {

sb.append(String.format(parmPhrase1\_th, criterion.getCondition(), i, j,criterion.getTypeHandler()));

}

} else if (criterion.isBetweenValue()) {

if (criterion.getTypeHandler() == null) {

sb.append(String.format(parmPhrase2, criterion.getCondition(), i, j, i, j));

} else {

sb.append(String.format(parmPhrase2\_th, criterion.getCondition(), i, j, criterion.getTypeHandler(), i, j, criterion.getTypeHandler()));

}

} else if (criterion.isListValue()) {

sb.append(criterion.getCondition());

sb.append(" (");

List<?> listItems = (List<?>) criterion.getValue();

boolean comma = false;

for (int k = 0; k < listItems.size(); k++) {

if (comma) {

sb.append(", ");

} else {

comma = true;

}

if (criterion.getTypeHandler() == null) {

sb.append(String.format(parmPhrase3, i, j, k));

} else {

sb.append(String.format(parmPhrase3\_th, i, j, k, criterion.getTypeHandler()));

}

}

sb.append(')');

}

}

sb.append(')');

}

}

if (sb.length() > 0) {

sql.WHERE(sb.toString());

}

}

}