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

import java.util.Date;

public class BizReligionDetailInfo {

private String religionName,url,title,terminal,visitTime,srcIP,category,domain,DNS,snapshot,

terminalDetail,tarIP,srcPort,protocol,MAC,times\_date;

private Integer visitTimes=0;

private Date date;

public String getTimes\_date() {

return times\_date;

}

public void setTimes\_date(String times\_date) {

this.times\_date = times\_date;

}

public Date getDate() {

return date;

}

public void setDate(Date date) {

this.date = date;

}

public String getReligionName() {

return religionName;

}

public void setReligionName(String religionName) {

this.religionName = religionName;

}

public String getUrl() {

return url;

}

public void setUrl(String url) {

this.url = url;

}

public String getTitle() {

return title;

}

public void setTitle(String title) {

this.title = title;

}

public String getTerminal() {

return terminal;

}

public void setTerminal(String terminal) {

this.terminal = terminal;

}

public String getVisitTime() {

return visitTime;

}

public void setVisitTime(String visitTime) {

this.visitTime = visitTime;

}

public String getSrcIP() {

return srcIP;

}

public void setSrcIP(String srcIP) {

this.srcIP = srcIP;

}

public String getCategory() {

return category;

}

public void setCategory(String category) {

this.category = category;

}

public String getDomain() {

return domain;

}

public void setDomain(String domain) {

this.domain = domain;

}

public String getDNS() {

return DNS;

}

public void setDNS(String DNS) {

this.DNS = DNS;

}

public String getSnapshot() {

return snapshot;

}

public void setSnapshot(String snapshot) {

this.snapshot = snapshot;

}

public String getTerminalDetail() {

return terminalDetail;

}

public void setTerminalDetail(String terminalDetail) {

this.terminalDetail = terminalDetail;

}

public String getTarIP() {

return tarIP;

}

public void setTarIP(String tarIP) {

this.tarIP = tarIP;

}

public String getSrcPort() {

return srcPort;

}

public void setSrcPort(String srcPort) {

this.srcPort = srcPort;

}

public String getProtocol() {

return protocol;

}

public void setProtocol(String protocol) {

this.protocol = protocol;

}

public String getMAC() {

return MAC;

}

public void setMAC(String MAC) {

this.MAC = MAC;

}

public Integer getVisitTimes() {

return visitTimes;

}

public void setVisitTimes(Integer visitTimes) {

this.visitTimes = visitTimes;

}

}

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

import java.util.Date;

public class BizReligionPercent {

private String religionName;

private Integer visitTime=0;

private Double percentage=0D;

private Date timesDate;

private String dateStr;

private String visite\_time;

private String url;

private String title;

private String terminal;

public String getTitle() {

return title;

}

public void setTitle(String title) {

this.title = title;

}

public String getTerminal() {

return terminal;

}

public void setTerminal(String terminal) {

this.terminal = terminal;

}

public String getUrl() {

return url;

}

public void setUrl(String url) {

this.url = url;

}

public String getVisite\_time() {

return visite\_time;

}

public void setVisite\_time(String visite\_time) {

this.visite\_time = visite\_time;

}

public String getDateStr() {

return dateStr;

}

public void setDateStr(String dateStr) {

this.dateStr = dateStr;

}

public Date getTimesDate() {

return timesDate;

}

public void setTimesDate(Date timesDate) {

this.timesDate = timesDate;

}

public String getReligionName() {

return religionName;

}

public void setReligionName(String religionName) {

this.religionName = religionName;

}

public Integer getVisitTime() {

return visitTime;

}

public void setVisitTime(Integer visitTime) {

this.visitTime = visitTime;

}

public Double getPercentage() {

return percentage;

}

public void setPercentage(Double percentage) {

this.percentage = percentage;

}

}

package com.yingchong.hive;

import java.util.ArrayList;

import org.apache.hadoop.hive.ql.exec.UDFArgumentException;

import org.apache.hadoop.hive.ql.exec.UDFArgumentLengthException;

import org.apache.hadoop.hive.ql.metadata.HiveException;

import org.apache.hadoop.hive.ql.udf.generic.GenericUDTF;

import org.apache.hadoop.hive.serde2.objectinspector.ObjectInspector;

import org.apache.hadoop.hive.serde2.objectinspector.ObjectInspectorFactory;

import org.apache.hadoop.hive.serde2.objectinspector.StructObjectInspector;

import org.apache.hadoop.hive.serde2.objectinspector.primitive.PrimitiveObjectInspectorFactory;

public class HiveUDTF extends GenericUDTF{

@Override

public StructObjectInspector initialize(ObjectInspector [] args) throws UDFArgumentException {

if (args.length != 1) {

throw new UDFArgumentLengthException("ExplodeMap takes only one argument");

}

if (args[0].getCategory() != ObjectInspector.Category.PRIMITIVE) {

throw new UDFArgumentException("ExplodeData takes string as a parameter");

}

ArrayList<String> fieldNames = new ArrayList<String>();

ArrayList<ObjectInspector> fieldOIs = new ArrayList<ObjectInspector>();

fieldNames.add("type");

fieldOIs.add(PrimitiveObjectInspectorFactory.javaStringObjectInspector);

fieldNames.add("template\_id");

fieldOIs.add(PrimitiveObjectInspectorFactory.javaStringObjectInspector);

fieldNames.add("num");

fieldOIs.add(PrimitiveObjectInspectorFactory.javaStringObjectInspector);

return ObjectInspectorFactory.getStandardStructObjectInspector(fieldNames, fieldOIs);

}

@Override

public void process(Object[] args) throws HiveException {

String input = args[0].toString();

input = input.substring(2, input.length() - 2).replaceAll("\"", "");

String[] test = input.split(",");

for (int i = 0; i < test.length; i++) {

try {

String[] result = test[i].split(";");

forward(result);

} catch (Exception e) {

continue;

}

}

}

@Override

public void close() throws HiveException {

}

}

package com.yingchong.util;

import java.util.ArrayList;

import java.util.Collection;

import java.util.Properties;

import org.apache.kafka.clients.consumer.ConsumerRecords;

import org.apache.kafka.clients.consumer.KafkaConsumer;

public class KafkaUtil {

public static ConsumerRecords consumer (String bootstrap,String groupId,String topic){

Properties properties = new Properties();

properties.put("bootstrap.servers", bootstrap);

properties.put("key.deserializer", "org.apache.kafka.common.serialization.IntegerDeserializer");

properties.put("value.deserializer", "org.apache.kafka.common.serialization.StringDeserializer");

properties.put("group.id", groupId);

properties.put("auto.offset.reset", "earliest");

properties.put("enable.auto.commit", "true");

@SuppressWarnings("resource")

KafkaConsumer<Integer, String> consumer = new KafkaConsumer<>(properties);

Collection collection = new ArrayList<>();

collection.add(topic);

consumer.subscribe(collection);

ConsumerRecords<Integer,String> records = consumer.poll(Integer.MAX\_VALUE);

return records;

}

}

package com.yingchong.hbase;

import java.io.IOException;

import java.util.ArrayList;

import java.util.List;

import org.apache.hadoop.hbase.Cell;

import org.apache.hadoop.hbase.CellUtil;

import org.apache.hadoop.hbase.HColumnDescriptor;

import org.apache.hadoop.hbase.HTableDescriptor;

import org.apache.hadoop.hbase.TableName;

import org.apache.hadoop.hbase.client.Admin;

import org.apache.hadoop.hbase.client.Connection;

import org.apache.hadoop.hbase.client.Get;

import org.apache.hadoop.hbase.client.HTable;

import org.apache.hadoop.hbase.client.Put;

import org.apache.hadoop.hbase.client.Result;

import org.apache.hadoop.hbase.client.ResultScanner;

import org.apache.hadoop.hbase.client.Scan;

import org.apache.hadoop.hbase.client.Table;

import org.apache.hadoop.hbase.io.compress.Compression.Algorithm;

import org.apache.hadoop.hbase.util.Bytes;

import com.jh.util.HBaseUtil;

import com.jh.util.TimeUtil;

public class HBaseTest {

public static void main(String[] args) {

HBaseUtil.getConnection("hadoop102:2181,hadoop103:2181,hadoop104:2181");

HBaseTest hBaseTest = new HBaseTest();

hBaseTest.createTable("hadoop102:2181,hadoop103:2181,hadoop104:2181",

"mate:hello", "info");

hBaseTest.deleteTable("hadoop102:2181,hadoop103:2181,hadoop104:2181",

"hello");

// hBaseTest.put("hadoop102:2181,hadoop103:2181,hadoop104:2181",

// "mate:hello", "1001\_"+TimeUtil.getTime(), "info", "name", "small seven");

//hBaseTest.get("hadoop102:2181,hadoop103:2181,hadoop104:2181", "mate:hello", "1001");

}

// 创建表

public void createTable(String zks, String tableName, String family) {

// 获取连接

Connection con = HBaseUtil.getConnection(zks);

try {

// 获取admin对象，用于管理表

Admin admin = con.getAdmin();

// 获取表描述对象，可以给它设置一些属性

HTableDescriptor descriptor = new HTableDescriptor(TableName.valueOf(tableName));

// 获取列簇描述对象

HColumnDescriptor colum = new HColumnDescriptor(family);

// 设置snappy压缩

colum.setCompressionType(Algorithm.SNAPPY);

// 把列簇描述器添加到表描述器里

descriptor.addFamily(colum);

// 创建表

// admin.createTable(descriptor);

// 创建预分区表

// admin.createTable(descriptor, new byte[][] { Bytes.toBytes("1"),

// Bytes.toBytes("2") });

admin.createTable(descriptor, "1".getBytes(), "5".getBytes(), 6);

} catch (IOException e) {

e.printStackTrace();

}

}

// 删除表

public void deleteTable(String zks, String tableName) {

Connection con = HBaseUtil.getConnection(zks);

try {

Admin admin = con.getAdmin();

TableName tName = TableName.valueOf(tableName);

admin.disableTable(tName);

admin.deleteTable(tName);

} catch (IOException e) {

e.printStackTrace();

}

}

// 插入数据

public void put(String zks, String tableName, String rowkey, String columFamily, String qualifier, String value) {

Connection con = HBaseUtil.getConnection(zks);

Put put = new Put(Bytes.toBytes(rowkey));

put.addColumn(Bytes.toBytes(columFamily), Bytes.toBytes(qualifier), Bytes.toBytes(value));

try {

HTable table = (HTable) con.getTable(TableName.valueOf(tableName));

table.put(put);

} catch (IOException e) {

e.printStackTrace();

}

}

// 查询数据

public void get(String zks, String tableName, String rowkey) {

HTable table = HBaseUtil.getHtable(zks, tableName);

Get get = new Get(Bytes.toBytes(rowkey));

try {

Result result = table.get(get);

List<Cell> list = result.listCells();

for (Cell cell : list) {

StringBuilder builder = new StringBuilder();

builder.append(Bytes.toString(CellUtil.cloneRow(cell))).append("\t")

.append(Bytes.toString(CellUtil.cloneFamily(cell))).append("\t")

.append(Bytes.toString(CellUtil.cloneQualifier(cell))).append("\t")

.append(Bytes.toString(CellUtil.cloneValue(cell)));

System.out.println(builder.toString());

}

} catch (IOException e) {

e.printStackTrace();

}

}

// scan查询

public List<User> scanTable(String zks, String tableName) {

// list保存返回的用户的信息

List<User> list = new ArrayList<User>();

// 获取table对象

Table table = HBaseUtil.getHtable(zks, tableName);

try {

// 创建一个scan对象

Scan scan = new Scan();

// 指定列簇

// scan.addFamily(Bytes.toBytes("info"));

// 指定列，结果只返回这个列

// scan.addColumn(Bytes.toBytes("info"), Bytes.toBytes("name"));

// 指定开始和结束的rowkey，结果返回rowkey区间的数据，提高效率

// scan.setStartRow(Bytes.toBytes("1000"));

// scan.setStopRow(Bytes.toBytes("1003"));

// 指定返回结果的时间戳范围

// scan.setTimeRange(0, 1545813502516L);

// 设置返回的cell的版本数量

// scan.setMaxVersions(2);

// 行过滤器

// RowFilter rowFilter = new RowFilter(CompareOp.EQUAL, new

// Binaryingchongomparator(Bytes.toBytes("1001")));

// scan.setFilter(rowFilter);

// 列过滤器

// Filter qualifierFilter = new QualifierFilter(CompareOp.EQUAL, new

// Binaryingchongomparator(Bytes.toBytes("name")));

// scan.setFilter(qualifierFilter);

// 值过滤器，只返回对应的值的那一列

// Filter valueFilter = new ValueFilter(CompareOp.GREATER, new

// Binaryingchongomparator(Bytes.toBytes("bao")));

// scan.setFilter(valueFilter);

// //单列值过滤器，返回整行

// SingleColumnValueFilter filter = new

// SingleColumnValueFilter(Bytes.toBytes("info"),

// Bytes.toBytes("name"), CompareOp.EQUAL, new

// Binaryingchongomparator(Bytes.toBytes("xiaotao")));

// //如果为true，匹配没有的字段，那么这一行数据就是无效

// filter.setFilterIfMissing(true);

// //scan.setFilter(filter);

//

// SingleColumnValueFilter filter1 = new

// SingleColumnValueFilter(Bytes.toBytes("info"),

// Bytes.toBytes("age"), CompareOp.EQUAL, new

// Binaryingchongomparator(Bytes.toBytes("12")));

// filter1.setFilterIfMissing(true);

//

// FilterList filterList = new FilterList();

// filterList.addFilter(filter);

// filterList.addFilter(filter1);

// scan.setFilter(filterList);

// //前缀过滤器，判断rowkey的前缀

// PrefixFilter filter = new PrefixFilter(Bytes.toBytes("1001"));

// scan.setFilter(filter);

// //时间戳过滤器，返回指定时间戳的数据

// List<Long> list1 = new ArrayList<>();

// list1.add(1545719108363L);

// TimestampsFilter filter = new TimestampsFilter(list1);

// scan.setFilter(filter);

// ResultScanner里保存着所有行的数据

ResultScanner scanner = table.getScanner(scan);

// 遍历ResultScanner,result为每一行

for (Result result : scanner) {

// 创建user，用于保存结果数据

User user = new User();

// 获取所有的单元格，单元格里存放的是对应的列的值

List<Cell> cells = result.listCells();

for (Cell cell : cells) {

// 获取列名称

String colum = Bytes.toString(CellUtil.cloneQualifier(cell));

// 获取单元格的值

String value = Bytes.toString(CellUtil.cloneValue(cell));

// 通过判断，给user设置对应属性的值

switch (colum) {

case "name":

user.setName(value);

break;

case "age":

user.setAge(Integer.parseInt(value));

break;

case "sex":

user.setSex(value);

break;

}

}

// 把user添加到list

list.add(user);

}

} catch (IOException e) {

e.printStackTrace();

}

return list;

}

}

package com.yingchong.access;

import java.io.OutputStream;

import javax.servlet.http.Cookie;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import org.apache.commons.lang3.StringUtils;

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

import org.springframework.stereotype.Service;

import org.springframework.web.method.HandlerMethod;

import org.springframework.web.servlet.handler.HandlerInterceptorAdapter;

import com.alibaba.fastjson.JSON;

import com.yingchongredis.AccessKey;

import com.yingchongredis.RedisService;

import com.yingchongresult.CodeMsg;

import com.yingchongresult.Result;

import com.yingchongservice.MiaoshaUserService;

@Service

public class AccessInterceptor extends HandlerInterceptorAdapter{

@Autowired

MiaoshaUserService userService;

@Autowired

RedisService redisService;

@Override

public boolean preHandle(HttpServletRequest request, HttpServletResponse response, Object handler)

throws Exception {

if(handler instanceof HandlerMethod) {

MiaoshaUser user = getUser(request, response);

UserContext.setUser(user);

HandlerMethod hm = (HandlerMethod)handler;

AccessLimit accessLimit = hm.getMethodAnnotation(AccessLimit.class);

if(accessLimit == null) {

return true;

}

int seconds = accessLimit.seconds();

int maxCount = accessLimit.maxCount();

boolean needLogin = accessLimit.needLogin();

String key = request.getRequestURI();

if(needLogin) {

if(user == null) {

render(response, CodeMsg.SESSION\_ERROR);

return false;

}

key += "\_" + user.getId();

}else {

//do nothing

}

AccessKey ak = AccessKey.withExpire(seconds);

Integer count = redisService.get(ak, key, Integer.class);

if(count == null) {

redisService.set(ak, key, 1);

}else if(count < maxCount) {

redisService.incr(ak, key);

}else {

render(response, CodeMsg.ACCESS\_LIMIT\_REACHED);

return false;

}

}

return true;

}

private void render(HttpServletResponse response, CodeMsg cm)throws Exception {

response.setContentType("application/json;charset=UTF-8");

OutputStream out = response.getOutputStream();

String str = JSON.toJSONString(Result.error(cm));

out.write(str.getBytes("UTF-8"));

out.flush();

out.close();

}

private MiaoshaUser getUser(HttpServletRequest request, HttpServletResponse response) {

String paramToken = request.getParameter(MiaoshaUserService.COOKI\_NAME\_TOKEN);

String cookieToken = getCookieValue(request, MiaoshaUserService.COOKI\_NAME\_TOKEN);

if(StringUtils.isEmpty(cookieToken) && StringUtils.isEmpty(paramToken)) {

return null;

}

String token = StringUtils.isEmpty(paramToken)?cookieToken:paramToken;

return userService.getByToken(response, token);

}

private String getCookieValue(HttpServletRequest request, String cookiName) {

Cookie[] cookies = request.getCookies();

if(cookies == null || cookies.length <= 0){

return null;

}

for(Cookie cookie : cookies) {

if(cookie.getName().equals(cookiName)) {

return cookie.getValue();

}

}

return null;

}

}

package com.yingchong.action;

import java.util.List;

import com.opensymphony.xwork2.ActionContext;

import com.opensymphony.xwork2.ActionSupport;

import com.yingchong.dao.EmployeeDAO;

import com.yingchong.domain.Employee;

import com.yingchong.test.TestUtil;

public class EmployeeAction extends ActionSupport {

private String employeeId;

private String name;

private int currentPage;

private int totalPage;

private int recordNumber;

/\*业务层对象\*/

EmployeeDAO employeeDAO = new EmployeeDAO();

public String getEmployeeId() {

return employeeId;

}

public void setEmployeeId(String employeeId) {

this.employeeId = employeeId;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public int getCurrentPage() {

return currentPage;

}

public void setCurrentPage(int currentPage) {

this.currentPage = currentPage;

}

public int getTotalPage() {

return totalPage;

}

public void setTotalPage(int totalPage) {

this.totalPage = totalPage;

}

public int getRecordNumber() {

return recordNumber;

}

public void setRecordNumber(int recordNumber) {

this.recordNumber = recordNumber;

}

private Employee employee;

public String AddView() {

return "add\_view";

}

@SuppressWarnings("deprecation")

public String AddEmployee() {

ActionContext ctx = ActionContext.getContext();

String employeeId = employee.getEmployeeId();

Employee employee\_db = employeeDAO.GetEmployeeById(employeeId);

if(null != employee\_db) {

ctx.put("error", java.net.URLEncoder.encode("已经存在!"));

return "error";

}

try {

employee.setEditTime(TestUtil.getDateString());

employeeDAO.AddEmployee(employee);

ctx.put("message", java.net.URLEncoder.encode("成功!"));

return "add\_success";

} catch (Exception e) {

e.printStackTrace();

ctx.put("error", java.net.URLEncoder.encode("失败!"));

return "error";

}

}

public String QueryEmployee() {

if(employeeId == null) employeeId = "";

if(null == name) name = "";

if(currentPage == 0) currentPage = 1;

List<Employee> employeeList = employeeDAO.QueryEmployeeInfo(employeeId, name, currentPage);

employeeDAO.CalculateTotalPageAndRecordNumber(employeeId, name); //计算总的页数和总的记录数

totalPage = employeeDAO.getTotalPage(); //获取到总的页码数目

recordNumber = employeeDAO.getRecordNumber(); //当前查询条件下总记录数

ActionContext ctx = ActionContext.getContext();

ctx.put("employeeList", employeeList);

ctx.put("totalPage", totalPage);

ctx.put("recordNumber", recordNumber);

ctx.put("currentPage", currentPage);

ctx.put("employeeId", employeeId);

ctx.put("name", name);

return "query\_view";

}

public String ModifyEmployeeQuery() {

Employee employee= employeeDAO.GetEmployeeById(employeeId);

ActionContext ctx = ActionContext.getContext();

ctx.put("employee", employee);

return "modify\_view";

}

public String ModifyEmployee() {

ActionContext ctx = ActionContext.getContext();

try {

employee.setEditTime(TestUtil.getDateString());

employeeDAO.UpdateEmployee(employee);

ctx.put("message", java.net.URLEncoder.encode("更新成功!"));

return "add\_success";

} catch (Exception e) {

e.printStackTrace();

ctx.put("error", java.net.URLEncoder.encode("更新失败!"));

return "error";

}

}

public String DeleteEmployee() {

ActionContext ctx = ActionContext.getContext();

try {

employeeDAO.DeleteEmployee(employeeId);

ctx.put("message", java.net.URLEncoder.encode("删除成功!"));

return "add\_success";

} catch (Exception e) {

e.printStackTrace();

ctx.put("error", java.net.URLEncoder.encode("删除失败!"));

return "error";

}

}

public Employee getEmployee() {

return employee;

}

public void setEmployee(Employee employee) {

this.employee = employee;

}

}

package com.yingchong.utils;

import org.hibernate.HibernateException;

import org.hibernate.Session;

import org.hibernate.cfg.Configuration;

/\*\*

\* Configures and provides access to Hibernate sessions, tied to the

\* current thread of execution. Follows the Thread Local Session

\* pattern, see {@link http://hibernate.org/42.html }.

\*/

public class HibernateUtil {

/\*\*

\* Location of hibernate.cfg.xml file.

\* Location should be on the classpath as Hibernate uses

\* #resourceAsStream style lookup for its configuration file.

\* The default classpath location of the hibernate config file is

\* in the default package. Use #setConfigFile() to update

\* the location of the configuration file for the current session.

\*/

private static String CONFIG\_FILE\_LOCATION = "/hibernate.cfg.xml";

private static final ThreadLocal<Session> threadLocal = new ThreadLocal<Session>();

private static Configuration configuration = new Configuration();

private static org.hibernate.SessionFactory sessionFactory;

private static String configFile = CONFIG\_FILE\_LOCATION;

static {

try {

configuration.configure(configFile);

sessionFactory = configuration.buildSessionFactory();

} catch (Exception e) {

System.err

.println("%%%% Error Creating SessionFactory %%%%");

e.printStackTrace();

}

}

private HibernateUtil() {

}

/\*\*

\* Returns the ThreadLocal Session instance. Lazy initialize

\* the <code>SessionFactory</code> if needed.

\*

\* @return Session

\* @throws HibernateException

\*/

public static Session getSession() throws HibernateException {

Session session = (Session) threadLocal.get();

if (session == null || !session.isOpen()) {

if (sessionFactory == null) {

rebuildSessionFactory();

}

session = (sessionFactory != null) ? sessionFactory.openSession()

: null;

threadLocal.set(session);

}

return session;

}

/\*\*

\* Rebuild hibernate session factory

\*

\*/

public static void rebuildSessionFactory() {

try {

configuration.configure(configFile);

sessionFactory = configuration.buildSessionFactory();

} catch (Exception e) {

System.err

.println("%%%% Error Creating SessionFactory %%%%");

e.printStackTrace();

}

}

/\*\*

\* Close the single hibernate session instance.

\*

\* @throws HibernateException

\*/

public static void closeSession() throws HibernateException {

Session session = (Session) threadLocal.get();

threadLocal.set(null);

if (session != null) {

session.close();

}

}

/\*\*

\* return session factory

\*

\*/

public static org.hibernate.SessionFactory getSessionFactory() {

return sessionFactory;

}

/\*\*

\* return session factory

\*

\* session factory will be rebuilded in the next call

\*/

public static void setConfigFile(String configFile) {

HibernateUtil.configFile = configFile;

sessionFactory = null;

}

/\*\*

\* return hibernate configuration

\*

\*/

public static Configuration getConfiguration() {

return configuration;

}

}

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

import java.util.Date;

public class BizReligionDetailInfo {

private String religionName,url,title,terminal,visitTime,srcIP,category,domain,DNS,snapshot,

terminalDetail,tarIP,srcPort,protocol,MAC,times\_date;

private Integer visitTimes=0;

private Date date;

public String getTimes\_date() {

return times\_date;

}

public void setTimes\_date(String times\_date) {

this.times\_date = times\_date;

}

public Date getDate() {

return date;

}

public void setDate(Date date) {

this.date = date;

}

public String getReligionName() {

return religionName;

}

public void setReligionName(String religionName) {

this.religionName = religionName;

}

public String getUrl() {

return url;

}

public void setUrl(String url) {

this.url = url;

}

public String getTitle() {

return title;

}

public void setTitle(String title) {

this.title = title;

}

public String getTerminal() {

return terminal;

}

public void setTerminal(String terminal) {

this.terminal = terminal;

}

public String getVisitTime() {

return visitTime;

}

public void setVisitTime(String visitTime) {

this.visitTime = visitTime;

}

public String getSrcIP() {

return srcIP;

}

public void setSrcIP(String srcIP) {

this.srcIP = srcIP;

}

public String getCategory() {

return category;

}

public void setCategory(String category) {

this.category = category;

}

public String getDomain() {

return domain;

}

public void setDomain(String domain) {

this.domain = domain;

}

public String getDNS() {

return DNS;

}

public void setDNS(String DNS) {

this.DNS = DNS;

}

public String getSnapshot() {

return snapshot;

}

public void setSnapshot(String snapshot) {

this.snapshot = snapshot;

}

public String getTerminalDetail() {

return terminalDetail;

}

public void setTerminalDetail(String terminalDetail) {

this.terminalDetail = terminalDetail;

}

public String getTarIP() {

return tarIP;

}

public void setTarIP(String tarIP) {

this.tarIP = tarIP;

}

public String getSrcPort() {

return srcPort;

}

public void setSrcPort(String srcPort) {

this.srcPort = srcPort;

}

public String getProtocol() {

return protocol;

}

public void setProtocol(String protocol) {

this.protocol = protocol;

}

public String getMAC() {

return MAC;

}

public void setMAC(String MAC) {

this.MAC = MAC;

}

public Integer getVisitTimes() {

return visitTimes;

}

public void setVisitTimes(Integer visitTimes) {

this.visitTimes = visitTimes;

}

}

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

import java.util.Date;

public class BizReligionPercent {

private String religionName;

private Integer visitTime=0;

private Double percentage=0D;

private Date timesDate;

private String dateStr;

private String visite\_time;

private String url;

private String title;

private String terminal;

public String getTitle() {

return title;

}

public void setTitle(String title) {

this.title = title;

}

public String getTerminal() {

return terminal;

}

public void setTerminal(String terminal) {

this.terminal = terminal;

}

public String getUrl() {

return url;

}

public void setUrl(String url) {

this.url = url;

}

public String getVisite\_time() {

return visite\_time;

}

public void setVisite\_time(String visite\_time) {

this.visite\_time = visite\_time;

}

public String getDateStr() {

return dateStr;

}

public void setDateStr(String dateStr) {

this.dateStr = dateStr;

}

public Date getTimesDate() {

return timesDate;

}

public void setTimesDate(Date timesDate) {

this.timesDate = timesDate;

}

public String getReligionName() {

return religionName;

}

public void setReligionName(String religionName) {

this.religionName = religionName;

}

public Integer getVisitTime() {

return visitTime;

}

public void setVisitTime(Integer visitTime) {

this.visitTime = visitTime;

}

public Double getPercentage() {

return percentage;

}

public void setPercentage(Double percentage) {

this.percentage = percentage;

}

}

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

import java.util.List;

public class BizReligionTrend {

private List<BizReligionPercent> fojiaoList;

private List<BizReligionPercent> daojiaoList;

private List<BizReligionPercent> tianzhujiaoList;

private List<BizReligionPercent> yisilanList;

private List<BizReligionPercent> jiduList;

public List<BizReligionPercent> getFojiaoList() {

return fojiaoList;

}

public void setFojiaoList(List<BizReligionPercent> fojiaoList) {

this.fojiaoList = fojiaoList;

}

public List<BizReligionPercent> getDaojiaoList() {

return daojiaoList;

}

public void setDaojiaoList(List<BizReligionPercent> daojiaoList) {

this.daojiaoList = daojiaoList;

}

public List<BizReligionPercent> getTianzhujiaoList() {

return tianzhujiaoList;

}

public void setTianzhujiaoList(List<BizReligionPercent> tianzhujiaoList) {

this.tianzhujiaoList = tianzhujiaoList;

}

public List<BizReligionPercent> getYisilanList() {

return yisilanList;

}

public void setYisilanList(List<BizReligionPercent> yisilanList) {

this.yisilanList = yisilanList;

}

public List<BizReligionPercent> getJiduList() {

return jiduList;

}

public void setJiduList(List<BizReligionPercent> jiduList) {

this.jiduList = jiduList;

}

}

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.utils;

import com.yingchong.service.data\_service.service.ReligionService;

import org.jdom.Document;

import org.jdom.Element;

import org.jdom.JDOMException;

import org.jdom.Text;

import org.jdom.input.SAXBuilder;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import java.io.ByteArrayInputStream;

import java.io.IOException;

import java.io.InputStream;

import java.nio.charset.StandardCharsets;

import java.util.HashMap;

import java.util.List;

import java.util.Map;

public class JdomUtils {

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

/\*\*

\* 将 XML 转化为 map

\*

\* @param strxml

\* @return

\* @throws JDOMException

\* @throws IOException

\*/

public static Map<String,String> transferXmlToMap(String strxml) {

//strxml = strxml.replaceFirst("encoding=\".\*\"", "encoding=\"UTF-8\"");

if ("".equals(strxml)) {

return null;

}

Map<String,String> m = new HashMap<>();

SAXBuilder builder = new SAXBuilder();

try {

InputStream in = new ByteArrayInputStream(strxml.getBytes(StandardCharsets.UTF\_8));

Document doc = builder.build(in);

// 解析 DOM

Element root = doc.getRootElement();

List list = root.getChildren();

for (Object o : list) {

Element e = (Element) o;

if(e.getContentSize()==0) continue;

Text t = (Text) e.getContent().get(0);

m.put(e.getAttribute("n").getValue(), t.getValue());

}

//关闭流

in.close();

} catch (Exception e) {

logger.error("",e);

//throw new IOException(e.getMessage()); // 统一转化为 IO 异常输出

}finally {

builder = null;

}

return m;

}

public static void main(String[] args) throws IOException {

long s1 = System.currentTimeMillis();

String str = "<?xml version=\"1.0\" encoding=\"utf-8\"?>\n" +

"<\_d>\n" +

" <\_f n=\"private\_type\">jb</\_f>\n" +

" <\_f n=\"is\_webapp\">henda</\_f>\n" +

" <\_f n=\"line\_no\">hhhhh</\_f>\n" +

" <\_f n=\"dealed\_line\_no\">aaaaa</\_f>\n" +

" <\_f n=\"urllog\_type\">0</\_f>\n" +

" <\_f n=\"mac\">64-a0-e7-41-64-41</\_f>\n" +

" <\_f n=\"termtype\">ç§»åŠ¨ç»ˆç«¯(Androidç³»ç»Ÿç§»åŠ¨ç»ˆç«¯)</\_f>\n" +

" <\_f n=\"nProtocol\">6</\_f>\n" +

" <\_f n=\"host\">m.sogo.com</\_f>\n" +

" <\_f n=\"trace\_t\">web\_url</\_f>\n" +

" <\_f n=\"urltype\">3858868786</\_f>\n" +

" <\_f n=\"urldata\">m.sogo.com/</\_f>\n" +

" <\_f n=\"url\">m.sogo.com/</\_f>\n" +

" <\_f n=\"usr\_name\">10.30.102.80</\_f>\n" +

" <\_f n=\"DNS\">get.shouji.sogou.com</\_f>\n" +

"</\_d>";

// for (int i = 0; i < 10000; i++) {

//

// Map<String,String> map = transferXmlToMap(string);

// logger.info("i={}",i);

// }

long s2 = System.currentTimeMillis();

//Map<String,String> map = transferXmlToMap(str);

logger.info("耗时:{}",s2-s1);

// for (Object o : map.keySet()) {

// String key = o.toString(); // 拿到键

// String val = String.valueOf(map.get(key)); // 拿到值

// System.out.println(key + "=====" + val);

// }

}

}

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

import java.text.DateFormat;

import java.text.ParseException;

import java.text.SimpleDateFormat;

import java.time.Instant;

import java.util.\*;

/\*\*

\* 时间工具类

\*/

public class DateUtil {

/\*\*\*

\*

\* @return 返回 yyyy-MM-dd HH:mm:ss 格式的字符串时间戳

\*/

public static String getNowFullDate() {

SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");

Date date = new Date();

String now = sdf.format(date);

return now;

}

/\*\*

\* 转换字符串为Date类型

\*

\* @param dateStr

\* @return

\*/

public static Date formatStringToDate(String dateStr) {

if (dateStr == null)

return null;

SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");

try {

return sdf.parse(dateStr);

} catch (ParseException e) {

e.printStackTrace();

return null;

}

}

/\*\*

\* 转换特定格式的字符串为Date类型数据

\*

\* @param dateStr 字符串

\* @param type 格式

\* @return

\*/

public static Date formatStringToDate(String dateStr, String type) {

if (type == null) {

return formatStringToDate(dateStr);

} else {

SimpleDateFormat sdf = new SimpleDateFormat(type);

try {

return sdf.parse(dateStr);

} catch (ParseException e) {

e.printStackTrace();

return null;

}

}

}

/\*\*

\* Java long类型毫秒时间戳转换为字符串格式时间

\*

\* @param longTime

\* @return

\*/

public static String formatLongToStr(Long longTime) {

SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");

Date date = new Date(longTime);

String now = sdf.format(date);

return now;

}

/\*\*

\* long毫秒时间戳转换为指定格式的字符串时间戳

\*

\* @param longTime

\* @param type

\* @return

\*/

public static String formatLongToStr(Long longTime, String type) {

if (type == null) {

return formatLongToStr(longTime);

} else {

SimpleDateFormat sdf = new SimpleDateFormat(type);

Date date = new Date(longTime);

return sdf.format(date);

}

}

/\*\*

\* 不同格式的时间互转

\*

\* @param date

\* @param fromType

\* @param toType

\* @return

\*/

public static String formatStrToStr(String date, String fromType, String toType) {

Date date1 = formatStringToDate(date, fromType);

SimpleDateFormat sdf = new SimpleDateFormat(toType);

return sdf.format(date1);

}

/\*\*

\* @param date

\* @return

\*/

public static String formatDateStrToString(String date) {

return formatStrToStr(date, "yyyy-MM-dd HH:mm:ss", "yyyy-MM-dd");

}

/\*\*

\* 时间字符串转换为时间戳 精确度到秒

\*

\* @param time

\* @return

\*/

public static String gitTimes(String time) throws ParseException {

SimpleDateFormat format = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");

Date date = format.parse(time);

return String.valueOf(date.getTime() / 1000);

}

/\*\*

\* 获取几分钟前的时间

\*

\* @param minute

\* @return

\*/

public static String getTimeByMinute(int minute) {

Calendar calendar = Calendar.getInstance();

calendar.add(Calendar.MINUTE, minute);

return new SimpleDateFormat("yyyy-MM-dd HH:mm:ss").format(calendar.getTime());

}

public static String dateStrFormat(String dateStr) {

SimpleDateFormat formats = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");

String reaultStr = null;

try {

Date date = formats.parse(dateStr);

reaultStr = formats.format(date);

} catch (Exception e) {

e.printStackTrace();

}

return reaultStr;

}

/\*\*

\* 获取SimpleDateFormat

\*

\* @param parttern 日期格式

\* @return SimpleDateFormat对象

\* @throws RuntimeException 异常：非法日期格式

\*/

private static SimpleDateFormat getDateFormat(String parttern) throws RuntimeException {

return new SimpleDateFormat(parttern);

}

/\*\*

\* 获取日期中的某数值。如获取月份

\*

\* @param date 日期

\* @param dateType 日期格式

\* @return 数值

\*/

private static int getInteger(Date date, int dateType) {

Calendar calendar = Calendar.getInstance();

calendar.setTime(date);

return calendar.get(dateType);

}

/\*\*

\* 增加日期中某类型的某数值。如增加日期

\*

\* @param date 日期

\* @param dateType 类型

\* @param amount 数值

\* @return 计算后日期

\*/

private static Date addInteger(Date date, int dateType, int amount) {

Date myDate = null;

if (date != null) {

Calendar calendar = Calendar.getInstance();

calendar.setTime(date);

calendar.add(dateType, amount);

myDate = calendar.getTime();

}

return myDate;

}

/\*\*

\* 获取精确的日期

\*

\* @param timestamps 时间long集合

\* @return 日期

\*/

private static Date getAccurateDate(List<Long> timestamps) {

Date date = null;

long timestamp = 0;

Map<Long, long[]> map = new HashMap<Long, long[]>();

List<Long> absoluteValues = new ArrayList<Long>();

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

if (timestamps.size() > 1) {

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

for (int j = i + 1; j < timestamps.size(); j++) {

long absoluteValue = Math.abs(timestamps.get(i) - timestamps.get(j));

absoluteValues.add(absoluteValue);

long[] timestampTmp = {timestamps.get(i), timestamps.get(j)};

map.put(absoluteValue, timestampTmp);

}

}

// 有可能有相等的情况。如2012-11和2012-11-01。时间戳是相等的

long minAbsoluteValue = -1;

if (!absoluteValues.isEmpty()) {

// 如果timestamps的size为2，这是差值只有一个，因此要给默认值

minAbsoluteValue = absoluteValues.get(0);

}

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

for (int j = i + 1; j < absoluteValues.size(); j++) {

if (absoluteValues.get(i) > absoluteValues.get(j)) {

minAbsoluteValue = absoluteValues.get(j);

} else {

minAbsoluteValue = absoluteValues.get(i);

}

}

}

if (minAbsoluteValue != -1) {

long[] timestampsLastTmp = map.get(minAbsoluteValue);

if (absoluteValues.size() > 1) {

timestamp = Math.max(timestampsLastTmp[0], timestampsLastTmp[1]);

} else if (absoluteValues.size() == 1) {

// 当timestamps的size为2，需要与当前时间作为参照

long dateOne = timestampsLastTmp[0];

long dateTwo = timestampsLastTmp[1];

if ((Math.abs(dateOne - dateTwo)) < 100000000000L) {

timestamp = Math.max(timestampsLastTmp[0], timestampsLastTmp[1]);

} else {

long now = new Date().getTime();

if (Math.abs(dateOne - now) <= Math.abs(dateTwo - now)) {

timestamp = dateOne;

} else {

timestamp = dateTwo;

}

}

}

}

} else {

timestamp = timestamps.get(0);

}

}

if (timestamp != 0) {

date = new Date(timestamp);

}

return date;

}

/\*\*

\* 判断字符串是否为日期字符串

\*

\* @param date 日期字符串

\* @return true or false

\*/

public static boolean isDate(String date) {

boolean isDate = false;

if (date != null) {

if (StringToDate(date) != null) {

isDate = true;

}

}

return isDate;

}

/\*\*

\* 将日期字符串转化为日期。失败返回null。

\*

\* @param date 日期字符串

\* @param parttern 日期格式

\* @return 日期

\*/

public static Date StringToDate(String date, String parttern) {

Date myDate = null;

if (date != null) {

try {

myDate = getDateFormat(parttern).parse(date);

} catch (Exception e) {

}

}

return myDate;

}

public static Date StringToDate(String date) {

Date myDate = null;

if (date != null) {

try {

myDate = getDateFormat("yyyy-MM-dd HH:mm:ss").parse(date);

} catch (Exception e) {

}

}

return myDate;

}

/\*\*

\* 将日期转化为日期字符串。失败返回null。

\*

\* @param date 日期

\* @param parttern 日期格式

\* @return 日期字符串

\*/

public static String DateToString(Date date, String parttern) {

String dateString = null;

if (date != null) {

try {

dateString = getDateFormat(parttern).format(date);

} catch (Exception e) {

}

}

return dateString;

}

/\*\*

\* 增加日期的年份。失败返回null。

\*

\* @param date 日期

\* @param yearAmount 增加数量。可为负数

\* @return 增加年份后的日期

\*/

public static Date addYear(Date date, int yearAmount) {

return addInteger(date, Calendar.YEAR, yearAmount);

}

/\*\*

\* 增加日期的月份。失败返回null。

\*

\* @param date 日期

\* @param yearAmount 增加数量。可为负数

\* @return 增加月份后的日期

\*/

public static Date addMonth(Date date, int yearAmount) {

return addInteger(date, Calendar.MONTH, yearAmount);

}

/\*\*

\* 增加日期的天数。失败返回null。

\*

\* @param date 日期

\* @param dayAmount 增加数量。可为负数

\* @return 增加天数后的日期

\*/

public static Date addDay(Date date, int dayAmount) {

return addInteger(date, Calendar.DATE, dayAmount);

}

/\*\*

\* 增加日期的小时。失败返回null。

\*

\* @param date 日期

\* @param hourAmount 增加数量。可为负数

\* @return 增加小时后的日期

\*/

public static Date addHour(Date date, int hourAmount) {

return addInteger(date, Calendar.HOUR\_OF\_DAY, hourAmount);

}

/\*\*

\* 增加日期的分钟。失败返回null。

\*

\* @param date 日期

\* @param hourAmount 增加数量。可为负数

\* @return 增加分钟后的日期

\*/

public static Date addMinute(Date date, int hourAmount) {

return addInteger(date, Calendar.MINUTE, hourAmount);

}

/\*\*

\* 增加日期的秒钟。失败返回null。

\*

\* @param date 日期

\* @param hourAmount 增加数量。可为负数

\* @return 增加秒钟后的日期

\*/

public static Date addSecond(Date date, int hourAmount) {

return addInteger(date, Calendar.SECOND, hourAmount);

}

/\*\*

\* 获取日期的年份。失败返回0。

\*

\* @param date 日期字符串

\* @return 年份

\*/

public static int getYear(String date) {

return getYear(StringToDate(date));

}

/\*\*

\* 获取日期的年份。失败返回0。

\*

\* @param date 日期

\* @return 年份

\*/

public static int getYear(Date date) {

return getInteger(date, Calendar.YEAR);

}

/\*\*

\* 获取日期的月份。失败返回0。

\*

\* @param date 日期字符串

\* @return 月份

\*/

public static int getMonth(String date) {

return getMonth(StringToDate(date));

}

/\*\*

\* 获取日期的月份。失败返回0。

\*

\* @param date 日期

\* @return 月份

\*/

public static int getMonth(Date date) {

return getInteger(date, Calendar.MONTH);

}

/\*\*

\* 获取日期的天数。失败返回0。

\*

\* @param date 日期字符串

\* @return 天

\*/

public static int getDay(String date) {

return getDay(StringToDate(date));

}

/\*\*

\* 获取日期的天数。失败返回0。

\*

\* @param date 日期

\* @return 天

\*/

public static int getDay(Date date) {

return getInteger(date, Calendar.DATE);

}

/\*\*

\* 获取日期的小时。失败返回0。

\*

\* @param date 日期字符串

\* @return 小时

\*/

public static int getHour(String date) {

return getHour(StringToDate(date));

}

/\*\*

\* 获取日期的小时。失败返回0。

\*

\* @param date 日期

\* @return 小时

\*/

public static int getHour(Date date) {

return getInteger(date, Calendar.HOUR\_OF\_DAY);

}

/\*\*

\* 获取日期的分钟。失败返回0。

\*

\* @param date 日期字符串

\* @return 分钟

\*/

public static int getMinute(String date) {

return getMinute(StringToDate(date));

}

/\*\*

\* 获取日期的分钟。失败返回0。

\*

\* @param date 日期

\* @return 分钟

\*/

public static int getMinute(Date date) {

return getInteger(date, Calendar.MINUTE);

}

/\*\*

\* 获取日期的秒钟。失败返回0。

\*

\* @param date 日期字符串

\* @return 秒钟

\*/

public static int getSecond(String date) {

return getSecond(StringToDate(date));

}

/\*\*

\* 获取日期的秒钟。失败返回0。

\*

\* @param date 日期

\* @return 秒钟

\*/

public static int getSecond(Date date) {

return getInteger(date, Calendar.SECOND);

}

/\*\*

\* unix时间戳转换为dateFormat

\*

\* @param beginDate

\* @return

\*/

public static Date timestampToDate(Integer beginDate) {

Date date = Date.from(Instant.ofEpochSecond(beginDate));

return date;

}

/\*\*

\* 根据生日获取当前年龄

\*

\* @param brithday

\* @return

\*/

public static int getCurrentAgeByBirthdate(String brithday) {

int age = 0;

Calendar born = Calendar.getInstance();

Calendar now = Calendar.getInstance();

if (brithday != null) {

now.setTime(new Date());

born.setTime(formatStringToDate(brithday));

if (born.after(now)) {

throw new IllegalArgumentException("年龄不能超过当前日期");

}

age = now.get(Calendar.YEAR) - born.get(Calendar.YEAR);

int nowDayOfYear = now.get(Calendar.DAY\_OF\_YEAR);

int bornDayOfYear = born.get(Calendar.DAY\_OF\_YEAR);

System.out.println("nowDayOfYear:" + nowDayOfYear + " bornDayOfYear:" + bornDayOfYear);

if (nowDayOfYear < bornDayOfYear) {

age -= 1;

}

}

return age;

}

/\*\*

\* 格式化日期 输出格式按formatType走

\*/

public static String formatDateToStr(Date d,String formatType) {

SimpleDateFormat dateFormat = new SimpleDateFormat(formatType) ; //使用了默认的格式创建了一个日期格式化对象。

String s = dateFormat.format(d);

return s;

}

/\*\*

\* date2比date1多的天数

\* @param date1

\* @param date2

\* @return

\*/

public static int differentDays(Date date1,Date date2)

{

Calendar cal1 = Calendar.getInstance();

cal1.setTime(date1);

Calendar cal2 = Calendar.getInstance();

cal2.setTime(date2);

int day1= cal1.get(Calendar.DAY\_OF\_YEAR);

int day2 = cal2.get(Calendar.DAY\_OF\_YEAR);

int year1 = cal1.get(Calendar.YEAR);

int year2 = cal2.get(Calendar.YEAR);

if(year1 != year2) //同一年

{

int timeDistance = 0 ;

for(int i = year1 ; i < year2 ; i ++)

{

if(i%4==0 && i%100!=0 || i%400==0) //闰年

{

timeDistance += 366;

}

else //不是闰年

{

timeDistance += 365;

}

}

return timeDistance + (day2-day1) ;

}

else //不同年

{

//System.out.println("判断day2 - day1 : " + (day2-day1));

return day2-day1;

}

}

public static void main(String[] args) throws ParseException {

DateFormat dateFormat1 = new SimpleDateFormat("yyyy-MM-dd");

Date d1 = dateFormat1.parse("2019-05-01");

Date d2 = dateFormat1.parse("2019-05-15");

//System.out.println(formatDateToStr(d,"yyyy-M-d"));

//System.out.println(formatDateToStr(d,"yyyy/M/d"));

System.out.println(differentDays(d1,d2));

}

}

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

import com.github.pagehelper.PageHelper;

import com.github.pagehelper.PageInfo;

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

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

import com.yingchong.service.data\_service.BizBean.biz\_religion.BizReligionDetailInfo;

import com.yingchong.service.data\_service.BizBean.biz\_religion.BizReligionPercent;

import com.yingchong.service.data\_service.BizBean.biz\_religion.BizReligionTrend;

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

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

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

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

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

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

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

import com.yingchong.service.data\_service.service.thread.CompareThread;

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

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

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

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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

import org.springframework.stereotype.Service;

import java.nio.charset.StandardCharsets;

import java.time.LocalDate;

import java.time.format.DateTimeFormatter;

import java.util.ArrayList;

import java.util.Date;

import java.util.List;

import java.util.Map;

import java.util.concurrent.ExecutorService;

@Service

public class ReligionService {

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

//private Integer start = 0;

private Integer step = 10000;

private String trace\_t = "trace\_t";

private String web\_url = "web\_url";

private String url = "url";

@Autowired

private MyActionMapper myActionMapper;

@Autowired

private FeatureUrlMapper featureUrlMapper;

@Autowired

private FeatureKeyMapper featureKeyMapper;

@Autowired

private FeatureKeyOppositeMapper featureKeyOppositeMapper;

@Autowired

private ReligionTimesMapper religionTimesMapper;

@Autowired

private MyReligionTimeMapper myReligionTimeMapper;

/\*\*\*

\* @param user

\* @param startDate

\* @param endDate

\* @param page

\* @param pageSize

\* @return

\*/

//@JsonInclude(JsonInclude.Include.NON\_NULL)

public ResponseBean<PageInfo<BizReligionDetailInfo>> peopleVisitTimesDetail(

String user,

String startDate,String endDate,

Integer page,Integer pageSize

) {

PageHelper.startPage(page, pageSize);

List<BizReligionDetailInfo> bizReligionDetailInfos = myReligionTimeMapper.peopleVisitTimesDetail(user, startDate, endDate);

for (BizReligionDetailInfo bizReligionDetailInfo : bizReligionDetailInfos) {

convertStrColumn(bizReligionDetailInfo);

String visitTime = bizReligionDetailInfo.getVisitTime();

Date date = bizReligionDetailInfo.getDate();

long t = DateUtil.formatStringToDate(visitTime).getTime() + (date ==null? 0 :date.getTime());

bizReligionDetailInfo.setVisitTime(DateUtil.formatLongToStr(t,"yyyy-MM-dd HH:mm:ss"));

}

PageInfo<BizReligionDetailInfo> data = new PageInfo<>(bizReligionDetailInfos);

return new ResponseBean<>(data);

}

/\*\*

\* @param startDate

\* @param endDate

\* @param page

\* @param pageSize

\* @return

\*/

public ResponseBean<PageInfo<BizReligionDetailInfo>> peopleVisitTimes(

String user,

String startDate,String endDate,

Integer page,Integer pageSize

) {

PageHelper.startPage(page, pageSize);

List<BizReligionDetailInfo> bizReligionDetailInfos = myReligionTimeMapper.selectPeopleVisitTimes(user,startDate, endDate);

for (BizReligionDetailInfo bizReligionDetailInfo : bizReligionDetailInfos) {

convertStrColumn(bizReligionDetailInfo);

}

PageInfo<BizReligionDetailInfo> data = new PageInfo<>(bizReligionDetailInfos);

return new ResponseBean<>(data);

}

private void convertStrColumn(BizReligionDetailInfo bizReligionDetailInfo) {

bizReligionDetailInfo.setUrl(CodeUtils.convertCharset(bizReligionDetailInfo.getUrl()));

bizReligionDetailInfo.setTitle(CodeUtils.convertCharset(bizReligionDetailInfo.getTitle()));

bizReligionDetailInfo.setTerminalDetail(CodeUtils.convertCharset(bizReligionDetailInfo.getTerminalDetail()));

bizReligionDetailInfo.setTerminal(CodeUtils.convertCharset(bizReligionDetailInfo.getTerminal()));

bizReligionDetailInfo.setReligionName(CodeUtils.convertCharset(bizReligionDetailInfo.getReligionName()));

}

/\*\*

\* @param startDate

\* @param endDate

\* @return ResponseBean<List<BizReligionDetailInfo>>

\*/

public ResponseBean<PageInfo<BizReligionDetailInfo>> religionRank(

String startDate,String endDate,

Integer page,Integer pageSize) {

PageHelper.startPage(page, pageSize);

List<BizReligionDetailInfo> bizReligionDetailInfos = myReligionTimeMapper.selectReligionUrlRank(startDate, endDate);

for (BizReligionDetailInfo bizReligionDetailInfo : bizReligionDetailInfos) {

bizReligionDetailInfo.setTitle(CodeUtils.convertCharset(bizReligionDetailInfo.getTitle()));

bizReligionDetailInfo.setUrl(CodeUtils.convertCharset(bizReligionDetailInfo.getUrl()));

bizReligionDetailInfo.setTerminal(CodeUtils.convertCharset(bizReligionDetailInfo.getTerminal()));

bizReligionDetailInfo.setTerminalDetail(CodeUtils.convertCharset(bizReligionDetailInfo.getTerminalDetail()));

String data = bizReligionDetailInfo.getTimes\_date();

String time = bizReligionDetailInfo.getVisitTime();

time = time.substring(10,time.length()-1);

bizReligionDetailInfo.setVisitTime(data + time);

}

PageInfo<BizReligionDetailInfo> data = new PageInfo<>(bizReligionDetailInfos);

return new ResponseBean<>(data);

}

/\*\*

\* @param religionName

\* @param startDate

\* @param endDate

\* @return ResponseBean<List < BizReligionDetailInfo>>

\*/

public ResponseBean<PageInfo<BizReligionDetailInfo>> religionDetail(

String religionName,String startDate,String endDate,

Integer page,Integer pageSize) {

PageHelper.startPage(page, pageSize);

//List<BizReligionDetailInfo> bizReligionDetailInfos = myReligionTimeMapper.selectReligionDetail(CodeUtils.convertCharset88591(religionName),startDate,endDate);

List<BizReligionDetailInfo> bizReligionDetailInfos = myReligionTimeMapper.selectReligionDetail(new String(religionName.getBytes(StandardCharsets.UTF\_8), StandardCharsets.ISO\_8859\_1),startDate,endDate);

for (BizReligionDetailInfo bizReligionDetailInfo : bizReligionDetailInfos) {

bizReligionDetailInfo.setVisitTime(bizReligionDetailInfo.getDate() == null?"":DateUtil.formatDateToStr(bizReligionDetailInfo.getDate(),"yyyy-MM-dd HH:mm:ss"));

bizReligionDetailInfo.setReligionName(CodeUtils.convertCharset(bizReligionDetailInfo.getReligionName()));

bizReligionDetailInfo.setUrl(CodeUtils.convertCharset(bizReligionDetailInfo.getUrl()));

bizReligionDetailInfo.setTitle(CodeUtils.convertCharset(bizReligionDetailInfo.getTitle()));

bizReligionDetailInfo.setTerminal(CodeUtils.convertCharset(bizReligionDetailInfo.getTerminal()));

bizReligionDetailInfo.setTerminalDetail(CodeUtils.convertCharset(bizReligionDetailInfo.getTerminalDetail()));

}

PageInfo<BizReligionDetailInfo> data = new PageInfo<>(bizReligionDetailInfos);

return new ResponseBean<>(data);

}

/\*\*

\*

\* @param startDate

\* @param endDate

\* @return ResponseBean

\*/

public ResponseBean<BizReligionTrend> religionTrend(String startDate, String endDate) {

BizReligionTrend bizReligionTrend = new BizReligionTrend();

List<BizReligionPercent> bizReligionPercents;

for (String religionName : religionNames) {

String religionName1 = new String(religionName.getBytes(StandardCharsets.UTF\_8), StandardCharsets.ISO\_8859\_1);

bizReligionPercents = myReligionTimeMapper.selectReligionTrend(religionName1,startDate, endDate);

{

for (BizReligionPercent bizReligionPercent : bizReligionPercents) {

bizReligionPercent.setReligionName;

bizReligionPercent.setDateStr(DateUtil.formatDateToStr(bizReligionPercent.getTimesDate(),AppTypeService.dateParttern));

}

bizReligionTrend.setFojiaoList(bizReligionPercents);

}

if (religionName.equals {

for (BizReligionPercent bizReligionPercent : bizReligionPercents) {

bizReligionPercent.setReligionName;

bizReligionPercent.setDateStr(DateUtil.formatDateToStr(bizReligionPercent.getTimesDate(),AppTypeService.dateParttern));

}

bizReligionTrend.setDaojiaoList(bizReligionPercents);

}

if (religionName.equals {

for (BizReligionPercent bizReligionPercent : bizReligionPercents) {

bizReligionPercent.setReligionName;

bizReligionPercent.setDateStr(DateUtil.formatDateToStr(bizReligionPercent.getTimesDate(),AppTypeService.dateParttern));

}

bizReligionTrend.setJiduList(bizReligionPercents);

}

if (religionName.equals {

for (BizReligionPercent bizReligionPercent : bizReligionPercents) {

bizReligionPercent.setReligionName;

bizReligionPercent.setDateStr(DateUtil.formatDateToStr(bizReligionPercent.getTimesDate(),AppTypeService.dateParttern));

}

bizReligionTrend.setYisilanList(bizReligionPercents);

}

if (religionName.equals {

for (BizReligionPercent bizReligionPercent : bizReligionPercents) {

bizReligionPercent.setReligionName;

bizReligionPercent.setDateStr(DateUtil.formatDateToStr(bizReligionPercent.getTimesDate(),AppTypeService.dateParttern));

}

bizReligionTrend.setTianzhujiaoList(bizReligionPercents);

}

}

checkData(bizReligionTrend,startDate,endDate);

return new ResponseBean<>(bizReligionTrend);

}

private void checkData(BizReligionTrend bizReligionTrend,String startDate,String endDate) {

List<BizReligionPercent> daojiaoList = bizReligionTrend.getDaojiaoList();

List<BizReligionPercent> fojiaoList = bizReligionTrend.getFojiaoList();

List<BizReligionPercent> jiduList = bizReligionTrend.getJiduList();

List<BizReligionPercent> tianzhujiaoList = bizReligionTrend.getTianzhujiaoList();

List<BizReligionPercent> yisilanList = bizReligionTrend.getYisilanList();

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

LocalDate end = LocalDate.parse(endDate, DateTimeFormatter.ofPattern(AppTypeService.dateParttern)).plusDays(1);

int i = 0;

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

{

logger.info("date ={}",date);

checkListData(fojiaoList, start, i, date);

checkListData(daojiaoList, start, i, date);

checkListData(jiduList, start, i, date);

checkListData(tianzhujiaoList, start, i, date);

checkListData(yisilanList, start, i, date);

i++;

}

}

private void checkListData(List<BizReligionPercent> jiaoList, LocalDate start, int i, LocalDate date,String religionName) {

if(jiaoList.size()-1 >= i){//循环不超过 size

BizReligionPercent bizReligionPercent = jiaoList.get(i);

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

bizReligionPercent = new BizReligionPercent();

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

bizReligionPercent.setReligionName(religionName);

bizReligionPercent.setVisitTime(0);

jiaoList.add(i,bizReligionPercent);

}

}else {//超过,超过分为数组为空和不为空的时候

BizReligionPercent bizReligionPercent = new BizReligionPercent();

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

bizReligionPercent.setReligionName(religionName);

bizReligionPercent.setVisitTime(0);

jiaoList.add(bizReligionPercent);

}

}

/\*\*

\* @param startDate

\* @param endDate

\* @return ResponseBean

\*/

public ResponseBean<List<BizReligionPercent>> religionCategory(String startDate, String endDate) {

List<BizReligionPercent> bizReligionPercents = myReligionTimeMapper.selectReligionPercent(startDate, endDate);

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

BizReligionPercent other = new BizReligionPercent();

other.setReligionName("其他");

double p = 0;

for (BizReligionPercent bizReligionPercent : bizReligionPercents) {

String religionName = (CodeUtils.convertCharset(bizReligionPercent.getReligionName()));

if (religionName.equals

|| religionName.equals

|| religionName.equals

|| religionName.equals

|| religionName.equals{

bizReligionPercent.setReligionName(religionName);

String date = DateUtil.DateToString(bizReligionPercent.getTimesDate(),"yyyy-MM-dd HH:mm:ss");

String time = bizReligionPercent.getVisite\_time();

time = time.substring(10,time.length() - 1);

bizReligionPercent.setVisite\_time(date + time);

result.add(bizReligionPercent);

p += bizReligionPercent.getPercentage();

} else {

other.setVisitTime(other.getVisitTime() + bizReligionPercent.getVisitTime());

}

}

other.setPercentage(1 - p);

result.add(other);

return new ResponseBean<>(result);

}

/\*\*

\*

\* @param date

\* @return

\*/

public boolean insertReligionTimes(String date,Integer type) {

ReligionTimesExample example = new ReligionTimesExample();

example.createCriteria().andTimesDateEqualTo(DateUtil.StringToDate(date,AppTypeService.dateParttern));

long l = religionTimesMapper.countByExample(example);

if (l > 0) {

logger.info("已经导入过数据:{}",l);

return true;

}

ExecutorService pool = CompareThread.newCachedThreadPool();

long ss1 = System.currentTimeMillis();

//logger.info("s1=========={}",System.currentTimeMillis());

String tableName = date.replaceAll("-", "") + "\_action";

List<FeatureUrl> featureUrls = featureUrlMapper.selectByExample(new FeatureUrlExample());

List<FeatureKey> featureKeys = featureKeyMapper.selectByExample(new FeatureKeyExample());

Integer totalCount = myActionMapper.selectCountAction(tableName);

//logger.info("s2=========={}",System.currentTimeMillis());

int times = totalCount / step + 1;

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

int s1 = (i \* step);

batchSQL(pool,tableName, featureUrls,featureKeys, s1,date,type);

//logger.info("s4=========={}",System.currentTimeMillis());

logger.info("i={},s1={}", i, s1);

}

long ss2 = System.currentTimeMillis();

pool.shutdown();

//logger.info("s5=========={}",System.currentTimeMillis());

oppositeFilter();

logger.info("接口耗时:{}毫秒", ss2 - ss1);

return true;

}

private void executeJob(ExecutorService pool, String tableName, List<FeatureUrl> featureUrls, int s1,String date) {

Runnable runnable = new Runnable() {

@Override

public void run() {

//batchSQL(tableName, featureUrls, s1,date);

}

};

pool.execute(runnable);

}

private void executeJob1(ExecutorService pool, List<FeatureUrl> featureUrls,List<FeatureKey> featureKeys, BizActionBean bizActionBean,String date) {

Runnable runnable = new Runnable() {

@Override

public void run() {

compareUrlAndKey(featureUrls,featureKeys,date,bizActionBean);

}

};

pool.execute(runnable);

}

private void batchSQL(ExecutorService pool,String tableName, List<FeatureUrl> featureUrls,List<FeatureKey> featureKeys, int s1,String date,Integer type) {

logger.info("启动线程{}查询:", s1);

List<BizActionBean> bizActionBeans = myActionMapper.selectActionById(tableName, s1, step);

long s3 = System.currentTimeMillis();

//logger.info("s3=========={}",System.currentTimeMillis());

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

for (BizActionBean bizActionBean : bizActionBeans) {

if(type==1){//单线程

compareUrlAndKey(featureUrls,featureKeys, date, bizActionBean);

}else{//多线程

executeJob1(pool,featureUrls,featureKeys,bizActionBean,date);

}

}

}

long s4 = System.currentTimeMillis();

logger.info("对比耗时:{}",s4-s3);

bizActionBeans = null;

}

private void compareUrlAndKey(List<FeatureUrl> featureUrls, List<FeatureKey> featureKeys,String date, BizActionBean bizActionBean) {

Map<String, String> resultMap = null;

try {

resultMap = JdomUtils.transferXmlToMap(bizActionBean.getResult());

} catch (Exception e) {

e.printStackTrace();

}

if (resultMap != null) {

String s = resultMap.get(trace\_t);

if (web\_url.equals(s)) {

String userVisitUrl = resultMap.get(url);

String userVisitTitle = resultMap.get("title");

for (FeatureUrl featureUrl : featureUrls) {

compareUrl(bizActionBean, resultMap, userVisitUrl, featureUrl,date);

}

for (FeatureKey featureKey : featureKeys) {

if (userVisitUrl.contains(featureKey.getKeyWord())

||(userVisitTitle!=null && userVisitTitle.contains(featureKey.getKeyWord()))

) {

this.insertReligionTimesData(bizActionBean,resultMap,userVisitUrl,featureKey.getReligionName(),date);

}

}

}

s = null;

}

resultMap = null;

}

private void compareUrl(BizActionBean bizActionBean, Map<String, String> resultMap, String userVisitUrl, FeatureUrl featureUrl,String date) {

//logger.info("userVisitUrl={}, featureUrl={}",userVisitUrl,featureUrl.getUrl());

//if(userVisitUrl.contains("baidu.com"))

if (userVisitUrl.contains(featureUrl.getUrl())) {

insertReligionTimesData(bizActionBean, resultMap, userVisitUrl,featureUrl.getReligionName() , date);

}

}

private void insertReligionTimesData(BizActionBean bizActionBean, Map<String, String> resultMap, String userVisitUrl, String religionName, String date) {

ReligionTimes rt = new ReligionTimes();

rt.setReligionName(religionName);

if(userVisitUrl.length()>1000) userVisitUrl = userVisitUrl.substring(0,1000);

rt.setUrl(userVisitUrl);

rt.setWebName(resultMap.get("title"));

rt.setWebTitle(resultMap.get("title"));

rt.setHostIp(bizActionBean.getHostIp());

rt.setDetIp(bizActionBean.getDstIp());

rt.setHostPort(bizActionBean.getSrcPort());

rt.setTerminalDetail(resultMap.get("termtype"));

rt.setDns(resultMap.get("DNS"));

if(resultMap.get("urldata").length()>1000){

rt.setDomainName(resultMap.get("urldata").substring(0,1000));

}else {

rt.setDomainName(resultMap.get("urldata"));

}

rt.setMacAddress(resultMap.get("mac"));

rt.setProtocol(resultMap.get("nProtocol"));

rt.setVisiteTime(bizActionBean.getRecordTime());

rt.setTerminalType(resultMap.get("termtype"));

rt.setCreateTime(new Date());

rt.setUpdateTime(new Date());

rt.setTimesDate(DateUtil.StringToDate(date, AppTypeService.dateParttern));

int insert = religionTimesMapper.insert(rt);

logger.info(":{}, insert={}", userVisitUrl, insert);

}

public void oppositeFilter() {

List<FeatureKeyOpposite> featureKeyOpposites = featureKeyOppositeMapper.selectByExample(new FeatureKeyOppositeExample());

List<ReligionTimes> religionTimes = religionTimesMapper.selectByExample(new ReligionTimesExample());

for (ReligionTimes religionTime : religionTimes) {

for (FeatureKeyOpposite featureKeyOpposite : featureKeyOpposites) {

//if(religionTime.getUrl().contains(featureKeyOpposite.getKeyword()))

//logger.info("1id={},1url={},1key={},result={}",religionTime.getId(),religionTime.getUrl(),featureKeyOpposite.getKeyword(),religionTime.getUrl().contains(featureKeyOpposite.getKeyword()));

if ((religionTime.getWebTitle()!=null && religionTime.getWebTitle().contains(featureKeyOpposite.getKeyword()))

|| (religionTime.getUrl() !=null && religionTime.getUrl().contains(featureKeyOpposite.getKeyword()))

) {

logger.info("id={}, fanxiang key = {}",featureKeyOpposite.getKeyword(),religionTime.getId());

religionTimesMapper.deleteByPrimaryKey(religionTime.getId());

}

}

}

}

}

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

import com.yingchong.service.data\_service.service.thread.CompareThread;

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.util.concurrent.ExecutorService;

@Service

public class CalculateDataService {

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

@Autowired

private IndexService indexService;

@Autowired

private AppTypeService appTypeService;

@Autowired

private ReligionService religionService;

public void calculIndexDate() {

LocalDate today = LocalDate.now();

LocalDate endDate = LocalDate.of(2018, 10, 3);

ExecutorService pool = CompareThread.newCachedThreadPool();

for (LocalDate date = today.minusDays(1); date.isAfter(endDate); date = date.minusDays(1))

{

logger.info("计算:{}的数据",date);

//this.TimeTask(date.toString());

this.executeJob(pool,date.toString());

}

pool.shutdown();

}

private void executeJob(ExecutorService pool, String date) {

Runnable runnable = () -> TimeTaskIndex(date);

pool.execute(runnable);

}

public void calculReligionDate() {

LocalDate today = LocalDate.now();

LocalDate endDate = LocalDate.of(2018, 10, 3);

for (LocalDate date = today.minusDays(1); date.isAfter(endDate); date = date.minusDays(1))

{

logger.info("计算:{}的数据",date);

try {

religionService.insertReligionTimes(date.toString(),1);

} catch (Exception e) {

logger.error("每日数据",e);

}

}

}

public void calculReligionDateByDate(String startDateStr,String endDateStr) {

LocalDate startDate = LocalDate.parse(startDateStr);

LocalDate endDate = LocalDate.parse(endDateStr);

for (LocalDate date = startDate; date.isAfter(endDate); date = date.minusDays(1))

{

logger.info("计算:{}的数据",date);

try {

religionService.insertReligionTimes(date.toString(),1);

} catch (Exception e) {

logger.error("每日数据",e);

}

}

}

public void TimeTask(String date) {

indexData(date);

try {

religionService.insertReligionTimes(date,1);

} catch (Exception e) {

logger.error("每日数据",e);

}

}

public void TimeTaskIndex(String date) {

indexData(date);

}

private void indexData(String date) {

try {

indexService.insertFluxResult(date);

} catch (Exception e) {

logger.error("插入数据异常", e);

}

try {

indexService.insertOnlineTime(date);

} catch (Exception e) {

logger.error("同步数据", e);

}

try {

indexService.insertAppFluxSort(date);

} catch (Exception e) {

logger.error("同步数据", e);

}

try {

appTypeService.insertActionType(date);

} catch (Exception e) {

logger.error("每日数据", e);

}

}

}