此文章已于 11:18:58 2018/3/7 发布到 HealerJean梦想博客

1、项目常用但是遗忘

类别 [选择一个类别或键入一个新类别] ; [选择一个类别或键入一个新类别] ; [选择一个类别或键入一个新类别] ; DDKJ ; [选择一个类别或键入一个新类别]

# 1、对比之前的项目容易忘记的

# 1、前后台数据类型不匹配问题

### 1、使用angularJS，model如果是从后台取得的对象，全部放到了model中，时间也变成了时间戳，如果直接传的话，很明显会因为后台，date不能转化，而失败，而如果是正常的yyyy-MM-dd是可以通过inbind转化的，所以在js接收到全部数据后要进行筛选要接收的数据，不一定都用。或者可以直接选出来日期设置为null

$scope.pictureCardBag.cdate = undefined;  
$scope.pictureCardBag.udate = undefined;

### （1、如果是从列表中通过item中炸出来的item copy复制出来的，date可能会不存在，但是通过id找的有时候是存在的）

### 2、之前大连理工的项目成功是因为，后台的对象直接铺到了前台，变成了name的value，从而通过name传值，就不会包含其他的时间戳了。而且date根本也没有使用inbind，而是直接使用的DTO对象，String的日期

# 2、前后台数据参数的传递的问题

## 1、根据id修改数据的时候，使用angularJs 这种方式是通过js之前传递id路由参数的，然后到了另外一个js中再发起请求，

## 2、这个和我之前开发的项目是有不同的来之前的项目是直接通话id查找，再讲页面铺到前台。

## 1、针对map类型的后台model put，前台是不能够解析得 。在针对数据字典的时候，每个js直接主动发起请求去获取数据

## 2、之前的项目中的数据自定是因为编写勒thyme方言和thyme的自己的东西实现的

# 3、针对随机链接，其实无疑就是一个路径后面加了一个id，而针对链接失效，其实就是加一个status状态码，和一个过期时间(可以同时设置)，第一次过期请求的时候，先查看status状态码是否为过期1，如果不是则比较时间，如果过期了，然后将status设置为过期字段就可以喽（对于不需要登录可以访问，按照我的理解，可以这样对于shiro通过路径设置不需要登录即可（没有实际操作过），对于cas则有自己得方式）

# 4、修改list中某一属性，不需要重新list.add

@Override  
public Page<SkinAppInfoData> findSkinAppInfoList(Pageable pageable, Object... param) throws AppException {  
 Map data = MyBatisHelper.mergeParameterMap(pageable,param);  
 List<SkinAppInfoData> dataList = skinsMapper.findSkinList(data);  
 for(SkinAppInfoData skinAppInfoData :dataList){  
 if(skinAppInfoData.getChannelJson()!=null){  
 skinAppInfoData.setChannelJsonList((List<ChannelJson>)JSONArray.toCollection(JSONArray.fromObject(skinAppInfoData.getChannelJson())));  
 }  
 }  
 Long count = skinsMapper.countSkinList(data);  
 return new PageImpl<SkinAppInfoData>(dataList,pageable,count);  
}

# 5、针对用户恶意访问链接

## 1、对于大连理工大学很容易造成，因为访问链接，然后直接向前面thyme中铺设数据，但是对于多点的angularj而言。则通过js控制页面展示，很容易将它扼杀在摇篮中，连js都没有又怎么会调用URL地址呢

## 2、具有权限url，进行拦截器验证更加不会的了

# 6、数据库库唯一性设计

## 再高高并发访问的时候，注意使用唯一约束，防止再高并发状态下重复导入数据

# 7、大量数据的处理.

## 7.2、如果历史数据是固定的

### 在设计一个统计需求的时候，我们可以将统计结果放入一张表中，直接查询单表就可以，然后设置一个定时器让他半夜的时候执行历史全部的数据，执行完历史全部数据之后，再每天半夜可以设置头一天的历史数据。

## 7.3、删除字段问题，我们要设置好利用status删除字段查询的问题

### 查找可用的，以及直接通过关键信息查找，即使以及被删除了，也要保证能查到，根据不同场景使用不同的数据，所以下面第一个写的就是废话。明白了吧

CouponAdzone findByAdzonePidAndStatus(String adzonePid,Integer status);  
  
CouponAdzone findByAdzonePid(String adzonePid);

### 1、正常情况下不应该从数据库表中直接删除，而是应该添加字段，让它变成不可见。（这种不可见需要再查询的时候特别 注意）

#### 1、删除代码，

@Override  
public void delContact(Long ContactId) {  
 Contact Contact = ContactDAO.findOne(ContactId);  
 Contact.setIsVisible(Contact.***DISABLE***);  
 ContactDAO.save(Contact);  
}

#### 2、对应实体类字段

@Entity  
@Table(name = "crm\_contact\_record")  
public class Contact implements Serializable {  
  
 public static final int ***ENABLE*** = 1; //可见  
 public static final int ***DISABLE*** = 0; //不可见  
  
 @Id  
 @GeneratedValue(strategy = GenerationType.***IDENTITY***)  
 private Long id;  
 private Long adminId;  
 private Long customerId;  
 private String title;  
 private String contactName;  
 private Integer type;  
 private String content;  
 private Integer isVisible;  
 private Date contactTime;  
 private Date cdate;

# 8、不要使用 0 1 2等数据，尽可能使用变量

public static final Integer ***NUMBER\_0*** = 0;  
public static final Integer ***NUMBER\_1*** = 1;  
public static final Integer ***NUMBER\_2*** = 2;  
public static final Integer ***NUMBER\_3*** = 3;

public String getStatusDes() {  
 if(status==null){  
 return "未知";  
 }  
 if(status.compareTo(***NUMBER\_0***)==0){  
 return "禁止";  
 }else if(status.compareTo(***NUMBER\_1***)==0){  
 return "正常";  
 }else if(status.compareTo(***NUMBER\_2***)==0){  
 return "异常";  
 }else {  
 return "未知";  
 }  
}

# 9。添加、删除、更新使用一个接口（status，判断是否删除0删除，1保存，，更新仅仅用于不能重复项跟业务有关，根据场景选择，下面这种情况是我自己设计的）

/\*\*  
 \* 添加商品黑名单  
 \* @return  
 \*/  
@Override  
public CouponItemBlackList addItemBlackList(CouponItemBlackList couponItemBlackList) {  
  
 CouponItemBlackList origin = null ;  
 if(couponItemBlackList.getType().compareTo(CouponItemBlackList.***type\_shop***)==0){  
 origin = couponItemBlackListRepository.findByShopId (couponItemBlackList.getShopId());  
 }else if(couponItemBlackList.getType().compareTo(CouponItemBlackList.***type\_item***)==0) {  
 origin = couponItemBlackListRepository.findByItemId(couponItemBlackList.getItemId());  
 }  
 if(origin!=null){ //新建对象  
 couponItemBlackList.setId(origin.getId());  
 }  
 couponItemBlackListRepository.save(couponItemBlackList);  
 return couponItemBlackList;  
}

# 10、工具类中提供给工具类自己使用的使用private

/\*\*  
 \* @param resString  
 \* @return String  
 \* @Description Json 格式化到控制台打印  
 \*  
 \*/  
public static String responseFormat(String resString){  
  
 StringBuffer jsonForMatStr = new StringBuffer();  
 int level = 0;  
 for(int index=0;index<resString.length();index++)//将字符串中的字符逐个按行输出  
 {  
 //获取s中的每个字符  
 char c = resString.charAt(index);  
  
 //level大于0并且jsonForMatStr中的最后一个字符为\n,jsonForMatStr加入\t  
 if (level > 0 && '\n' == jsonForMatStr.charAt(jsonForMatStr.length() - 1)) {  
 jsonForMatStr.append(getLevelStr(level));  
 }  
 //遇到"{"和"["要增加空格和换行，遇到"]"和"}"要减少空格，以对应，遇到","要换行  
 switch (c) {  
 case '{':  
 case '[':  
 jsonForMatStr.append(c + "\n");  
 level++;  
 break;  
 case ',':  
 jsonForMatStr.append(c + "\n");  
 break;  
 case ']':  
 case '}':  
 jsonForMatStr.append("\n");  
 level--;  
 jsonForMatStr.append(getLevelStr(level));  
 jsonForMatStr.append(c);  
 break;  
 default:  
 jsonForMatStr.append(c);  
 break;  
 }  
 }  
 return jsonForMatStr.toString();  
}  
 /\*\*  
   
 \* 不对外开放，只给本类使用，所以使用private

\*/  
 private static String getLevelStr(int level) {  
 StringBuffer levelStr = new StringBuffer();  
 for (int levelI = 0; levelI < level; levelI++) {  
 levelStr.append("\t");  
 }  
 return levelStr.toString();  
 }