## cread系统畅读卡同步action使用spring单例作用域引起的问题

## 接口功能描述

1. 在cread系统中提供了如下action功能类：

|  |
| --- |
| com.jd.mobile.read.web.action.card.SynchroReadCardAction |

1. 该类的主要应用场景是通过调用该类的action对应的方法进行畅读卡绑定操作，这个类目中目前只有一个execute方法，通过如下的url调用，执行该类中的execute方法，完成对应场景的畅读卡绑定操作：

|  |
| --- |
| http://cread.e.jd.com/syncReadCard.action?orderId=1234&skuId=1234&pin=test&cardType=5&uuid=xxxxxxx&paramKey=xxxxx |

在SynchroReadCardAction类中获取参数的方式是通过定义全局变量，通过struts2的get和set方法自动完成参数的获取。

1. 目前SynchroReadCardAction在cread系统中的配置方式如下：

首先在spring-config-struts.xml配置文件中对该类进行了配置，将该类的对象托管给spring进行管理：

|  |
| --- |
| *<!-- http同步畅读卡的action，注入访问key-->* <**bean class="com.jd.mobile.read.web.action.card.SynchroReadCardAction"**>  <**property name="sychroReadCardKey" value="${mobile-cread.sync.httpinterface.key}"**/> </**bean**> |

在struts-read.xml配置文件中映射关系的配置如下：

|  |
| --- |
| <**package name="syReadcard" extends="action-json-default"**>  <**action name="syncReadCard" class="com.jd.mobile.read.web.action.card.SynchroReadCardAction"**>  <**result type="json" name="success"**>  <**param name="excludeNullProperties"**>true</**param**>  <**param name="enableGZIP"**>false</**param**>  <**param name="root"**>resultMap</**param**>  <**param name="defaultEncoding"**>utf-8</**param**>  </**result**>  </**action**> </**package**> |

## 应用场景

|  |
| --- |
| http://cread.e.jd.com/syncReadCard.action?orderId=1234&skuId=1234&pin=test&cardType=5&uuid=xxxxxxx&paramKey=xxxxx |

上述url目前调用情况主要是有几个场景：

1. 用户下单直接购买畅读卡，在order系统生成订单信息之后会调用该url进行畅读卡绑定：

|  |
| --- |
| *//直接生成畅读或畅听卡* NameValuePair orderid = **new** NameValuePair(**"orderId"**, String.*valueOf*(mallOrderDetailDO.getOrderId())); NameValuePair pin = **new** NameValuePair(**"pin"**, mallOrderDetailDO.getPin()); NameValuePair skuid = **new** NameValuePair(**"skuId"**, mallOrderDetailDO.getItemId() + **""**); NameValuePair apptype = **new** NameValuePair(**"appType"**, appTypeEnum.getCode() + **""**); NameValuePair uuid = **new** NameValuePair(**"uuid"**, mallOrderDetailDO.getOrderId() + **"\_"** + mallOrderDetailDO.getId()); NameValuePair key = **new** NameValuePair(**"paramKey"**, **syncCardParamKey**); NameValuePair[] nameValuePairs = **new** NameValuePair[]{orderid, pin, skuid, apptype, uuid, key}; String returnJson = HttpClientUtil.*sendHttpRequestByParams*(syncUrl, nameValuePairs, 3000, **"GBK"**); |

1. 用户购买会员PLUS之后，orderman系统会收到mq订单消息，判断用户购买了会员PLUS会直接给用户绑定畅读卡年卡。
2. 用户购买纸书送畅读卡，会调用该url完成畅读卡绑定。
3. 其他一些活动场景，比如预约华为手机送畅读卡，预约小米手机送畅读卡等场景。

上述2、3、4等场景下的调用方式都是通过调用order系统的BookUserSendMsgServiceImpl类中的createCard方法来完成的，当然该方法中的实现原理还是调用上述url向cread系统发送http请求完成畅读卡绑定的。

## 问题现象及问题原因排查

### 问题现象：

目前在做畅读卡结算中发现结算金额出现偏差，通过排查数据发现，有很多用户下单购买的畅读卡没有参与到结算中，原因是因为数据不符合结算条件，参与结算的畅读卡是会根据畅读卡类型进行过滤的，其中只有以下三类畅读卡会进行结算：1、个人购卡；2、企业购卡；4包月卡。也就是cardType为1、2、4的畅读卡会参与结算，目前查询数据库数据发现有很多畅读卡本身是用户购买的畅读卡，cardType类型应该是1，但是数据库中记录的信息是22（购买PLUS会员赠送畅读卡），排查order系统http请求接口的方式没有发现cardType类型设置错误的地方。

### 问题定位：

后来通过反复查看order系统和cread系统中的实现方式，发现了一个问题，在用户下单购买畅读卡时，order系统在http请求调用cread系统url时是没有指定cardType类型这个参数的，在cread系统中进行了判断，如果cardType参数为空，则指定cardType值为1。而其他的几种调用，比如：购买会员PLUS绑定畅读卡、预约小米手机送畅读卡，在http请求中显示的指定了cardType参数的值，这个地方是一个明显的区别。

接下来再查看SynchroReadCardAction类spring-config-struts.xml的配置方式时发现了问题，这个类在托管给spring时scope属性采用默认值，也就是singleton（单例模式），如果采用这种方式，在系统全局中SynchroReadCardAction是个单例类，而通过定义全局变量，通过struts2的get和set方法获取http请求中的参数时会出现参数不一致的问题，比如：通过会员PLUS绑定畅读卡明确指定了cardType为22，但是下单购买畅读卡没有明确指定cardType参数，这时该值就会复用上一次请求中的参数值（因为定义的是全局变量），这种情况下就会导致参数出现错乱。

### 代码验证

##### 测试场景一：

1. 对于上述推测内容进行了测试，测试代码如下：

|  |
| --- |
| **public class** SynchroReadCardAction **extends** EbookBaseAction {  **//定义全局变量**  **private** String **pin**;   **private** Integer **cardType**;  **public** String execute() {  Map<String, Object> resultMap = **new** HashMap<String, Object>();  ValueStack valueStack = ActionContext.*getContext*().getValueStack();  *logger*.error(**"出参=======pin==="** + **pin** + **"===cardType======="** + **cardType**);  resultMap.put(**"cardType"**, **cardType**);  resultMap.put(**"pin"**, **pin**);  valueStack.set(**"resultMap"**, resultMap);  **return *SUCCESS***;  }  ……定义全局变量get和set方法  } |

上述方法实现很简单，就是将http请求的参数放入valueStack返回，查看请求的参数和返回的参数是否一致，测试过程如下：

1. 在spring-confi-struts.xml中配置中配置SynchroReadCardAction类，并且不指定scope参数，采用默认值（singleton），测试结果如下：

|  |
| --- |
| //1、http请求信息  <http://cread.e.jd.com/syncReadCard.action?pin=dzllikelsw&cardType=2>  返回结果：  {   * "pin": "dzllikelsw", * "cardType": 2   }  //2、http请求信息  <http://cread.e.jd.com/syncReadCard.action?pin=lisi>  返回结果：  {   * "pin": "lisi", * **"cardType": 2**   } |

上述测试可以很明显发现问题，在第二个请求中没有指定cardType参数内容，但是结果中依然有cardType值为2，错误很明显。

1. 在spring-confi-struts.xml中配置中配置SynchroReadCardAction类，并且指定scope参数，设置为prototype，测试结果如下：

|  |
| --- |
| //1、http请求信息  <http://cread.e.jd.com/syncReadCard.action?pin=dzllikelsw&cardType=2>  返回结果：  {   * "pin": "dzllikelsw", * "cardType": 2   }  //2、http请求信息  <http://cread.e.jd.com/syncReadCard.action?pin=lisi>  返回结果：  {   * "pin": "lisi"   } |

通过上述测试，我们可以发现，这次的请求返回结果和我们预期的内容是一致的。

##### 测试场景二：

1. 修改测试代码内容如下，在spring-config-struts.xml中将SynchroReadCardAction配置还原为默认配置（singleton）：

|  |
| --- |
| **public class** SynchroReadCardAction **extends** EbookBaseAction {  **//定义全局变量**  **private** String **pin**;   **private** Integer **cardType**;  **public** String execute() {  Map<String, Object> resultMap = **new** HashMap<String, Object>();  ValueStack valueStack = ActionContext.*getContext*().getValueStack();  **try {  Thread.sleep(15000);  } catch (Exception e) {  }**  *logger*.error(**"出参=======pin==="** + **pin** + **"===cardType======="** + **cardType**);  resultMap.put(**"cardType"**, **cardType**);  resultMap.put(**"pin"**, **pin**);  valueStack.set(**"resultMap"**, resultMap);  **return *SUCCESS***;  }  ……定义全局变量get和set方法  } |

|  |
| --- |
| //1、http请求信息  http://cread.e.jd.com/syncReadCard.action?pin=lisi&cardType=7  返回结果：  {   * "pin": "lisi", * **"cardType": 6**   }  //2、http请求信息  <http://cread.e.jd.com/syncReadCard.action?pin=lisi>&cardType=6  返回结果：  {   * "pin": "lisi", * "cardType": 6   } |

上述代码中我们通过让线程睡眠15s的方式来模拟一下出现并发时的场景，出现并发时可能会出现多个请求同时进入代码块执行方法体，通过上述测试，我们首先发起一个请求，让后又发起另外一个请求，请求参数不同，由于存在先后顺序，**导致后面的请求会将前面的请求参数覆盖掉**，通过结果展示，的确出现了参数返回值错误的情况。

如果将SynchroReadCardAction类配置为多例（prototype）模式，我们发现上述情况是不会出现的，返回值是正常。

##### 测试场景三：

修改测试代码内容如下，在spring中将SynchroReadCardAction还还原为默认配置（singleton）：

|  |
| --- |
| **public class** SynchroReadCardAction **extends** EbookBaseAction {  **//定义全局变量**  **private** String **pin**;   **private** Integer **cardType**;  **public** String execute() {  Map<String, Object> resultMap = **new** HashMap<String, Object>();  ValueStack valueStack = ActionContext.*getContext*().getValueStack();  **String pin = request.getParameter("pin");  String cardType = request.getParameter("cardType");**  *logger*.error(**"出参=======pin==="** + **pin** + **"===cardType======="** + **cardType**);  resultMap.put(**"cardType"**, **cardType**);  resultMap.put(**"pin"**, **pin**);  valueStack.set(**"resultMap"**, resultMap);  **return *SUCCESS***;  }  ……定义全局变量get和set方法  } |

上述测试代码中我们将获取参数的方式修改为通过request.getParameter参数的方式，测试结果如下：

|  |
| --- |
| //1、http请求信息  <http://cread.e.jd.com/syncReadCard.action?pin=dzllikelsw&cardType=2>  返回结果：  {   * "pin": "dzllikelsw", * "cardType": 2   }  //2、http请求信息  <http://cread.e.jd.com/syncReadCard.action?pin=lisi>  返回结果：  {   * "pin": "lisi"   } |

通过上述测试，我们可以确定，通过使用request.getParameter的方式来获取参数，由于不再使用全局变量，即使设置为单例作用域也是不会出现参数错误问题的。

### 最终解决方式

1. 将spring-confi-struts.xml配置文件中SynchroReadCardAction类的作用域修改为prototype类型。**（目前线上采用的修改方式，改动很小即可解决问题）**
2. 修改程序中获取参数的方式，修改为使用request.getParameter方式获取参数，不使用全局变量并通过get、set方法获取参数，这种情况下SynchroReadCardAction的配置还可以继续采用默认配置（singleton）。