文档修订历史

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **版本** | **作者** | **工作描述** | **修订历史** | **修改日期** |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

目录

[文档修订历史 1](#_Toc522698973)

[【文档标题名称】 4](#_Toc522698974)

[1 范围 **错误!未定义书签。**](#_Toc522698975)

[1.1 内容范围 **错误!未定义书签。**](#_Toc522698976)

[1.2 适用范围 **错误!未定义书签。**](#_Toc522698977)

[2 业务描述 **错误!未定义书签。**](#_Toc522698978)

[3 角色说明 **错误!未定义书签。**](#_Toc522698979)

[3.1 分销代理商 **错误!未定义书签。**](#_Toc522698980)

[4 业务需求 **错误!未定义书签。**](#_Toc522698981)

[4.1 注册登录 **错误!未定义书签。**](#_Toc522698982)

[4.2 查看及修改账户信息 **错误!未定义书签。**](#_Toc522698983)

[4.3 创建客户 **错误!未定义书签。**](#_Toc522698984)

[4.4 商品信息 **错误!未定义书签。**](#_Toc522698985)

[4.5 修改客户商品价格 **错误!未定义书签。**](#_Toc522698986)

[4.6 推送价格 **错误!未定义书签。**](#_Toc522698987)

[4.7 查看客户信息 **错误!未定义书签。**](#_Toc522698988)

[4.8 代理商利润计算公式 **错误!未定义书签。**](#_Toc522698989)

[4.9 代理商利润及打款 **错误!未定义书签。**](#_Toc522698990)

[4.10 操作界面语言选项 **错误!未定义书签。**](#_Toc522698991)

[5 外围系统关系 **错误!未定义书签。**](#_Toc522698992)

[6 其它需求 **错误!未定义书签。**](#_Toc522698993)

[6.1 部署需求 **错误!未定义书签。**](#_Toc522698994)

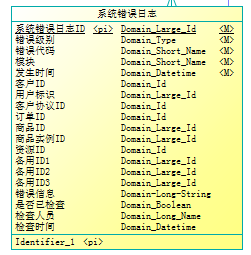
[6.2 环境要求 **错误!未定义书签。**](#_Toc522698995)

[6.3 运维要求 **错误!未定义书签。**](#_Toc522698996)

自身异常处理编程

主要是根据自身的情况来处理异常，并且把异常信息持久化到库表中。

# 表结构说明



## 系统错误日志表（System\_Error）

|  |  |  |  |
| --- | --- | --- | --- |
| Code | Name | Domain | Comment |
| SYSTEM\_ERROR\_ID | 系统错误日志ID | ID域-中-UID |  |
| ERROR\_LEVEL | 错误级别 | 类型域-S6 | WARN  ERROR |
| ERROR\_CODE | 错误代码 | [Domain\_Short\_Name](类型定义.docx#_名称域-短-S30（Domain_Short_Name）) | 每个module独立的错误代码  是不带空格的简短数值或字母编码 |
| MODULE | 模块 | [Domain\_Short\_Name](类型定义.docx#_名称域-短-S30（Domain_Short_Name）) | RATING  POST\_ADD\_LOG |
| ERROR\_TIME | 发生时间 | [Domain\_Datetime](类型定义.docx#_时间域-日期时间-DT（Domain_Datetime）) | 错误发生时间 |
| CUST\_ID | 客户ID | ID域-短-UID |  |
| USER\_ID | 用户标识 | ID域-中-UID | 来源于参与人PARTY\_ID |
| AGREEMENT\_ID | 客户协议ID | ID域-短-UID |  |
| ORDER\_ID | 订单ID | ID域-短-UID | 订单的唯一标识。 |
| OFFER\_ID | 商品ID | ID域-中-UID |  |
| OFFER\_INSTANCE\_ID | 商品实例ID | ID域-中-UID |  |
| RESOURCE\_ID | 资源ID | ID域-短-UID |  |
| SPARE\_ID1 | 备用ID1 | ID域-中-UID |  |
| SPARE\_ID2 | 备用ID2 | ID域-中-UID |  |
| SPARE\_ID3 | 备用ID3 | ID域-中-UID |  |
| ERROR\_MSG | 错误信息 | 字符串域-超长-TEXT |  |
| CHECKED | 是否已检查 | [Domain\_Boolean](类型定义.docx#_布尔域-C1（Domain_Boolean）) | 是否已检查:T是，F否 |
| CHECKER | 检查人员 | [Domain\_Long\_Name](类型定义.docx#_名称域-长-S50（Domain_Long_Name）) | 检查人员 |
| CHECK\_TIME | 检查时间 | [Domain\_Datetime](类型定义.docx#_时间域-日期时间-DT（Domain_Datetime）) | 检查时间 |

# 程序代码

## RateException

|  |
| --- |
| @SuppressWarnings("serial")  public class RateException extends ServiceBizException {  public RateException(Throwable cause, String errorCode, String devMsg) {  super(devMsg, errorCode, null, cause);  }  public RateException(String errorCode, String devMsg, String spareInfo) {  super(devMsg, errorCode, new Object[]{spareInfo});  }  public RateException(String errorCode, String devMsg) {  super(devMsg, errorCode, null);  }  public static AppException makeAppException(Throwable e) {  if (e instanceof AppException) {  return (AppException) e;  } else {  return new RateException(e, formatCode(e), formatMsg(e));  }  }  public void printErrAndStack() {  Throwable cause = getCause();  if (cause == null) {  printStackTrace();  } else {  cause.printStackTrace();  }  }  public static String stackString(Throwable e) {  StringWriter sw = new StringWriter();  PrintWriter pw = new PrintWriter(sw);  e.printStackTrace(pw);  return sw.toString();  }  public static String formatCode(Throwable e) {  String errCode;  if (e instanceof PricingException) {  errCode = ((PricingException) e).getErrorCode();  } else if (e instanceof AppException) {  errCode = ((AppException) e).getCode();  } else {  errCode = "4999";  }  return errCode;  }  public static String formatMsg(Throwable e) {  String errMsg;  if (e instanceof RateException) {  RateException e1 = (RateException) e;  String errorCode = e1.getCode();  String devMsg = e1.getCodeDesc();  errMsg = "错误码=" + errorCode + ",错误信息=" + devMsg;  //4999是被遗漏的异常，需要堆栈信息才能定位  if (errorCode.contains("4999")) {  Throwable e2 = e.getCause();  if (e2 != null) {  errMsg += "。Exception=" + e2.getClass().getSimpleName() + ":" + e2.getMessage();  errMsg += "。StackTrace=" + stackString(e2);  }  }  } else if (e instanceof AppException) {  errMsg = ((AppException) e).getCodeDesc();  if (errMsg == null) {  errMsg = e.getMessage();  }  return errMsg;  } else if (e instanceof FeignException) {  errMsg = "微服务调用网络异常:" + e.getMessage();  } else {  errMsg = "Exception=" + e.getClass().getSimpleName() + ":" + e.getMessage();  errMsg += "。StackTrace=" + stackString(e);  }  return errMsg;  }  // 获取额外信息，主要是对RateException  public static String formatSpareInfo(Throwable e) {  String spareInfo = "";  if (e instanceof RateException) {  Object[] objects = ((RateException) e).getCodeParams();  if (ValidationUtil.isNotEmpty(objects[0])) {  spareInfo = objects[0].toString();  }  }  return spareInfo;  }  } |

## ServiceBizException

|  |
| --- |
| public class ServiceBizException extends AppException {  protected String code;  protected String codeDesc;  protected Object[] codeParams;  public ServiceBizException(String codeDesc, String code, Object[] codeParams, Throwable cause) {  super(cause);  this.code = code;  this.codeParams = codeParams;  this.codeDesc = codeDesc;  }  public ServiceBizException(String codeDesc, String code, Object[] codeParams) {  this(codeDesc, code, codeParams, (Throwable)null);  }  public String getCode() {  return this.code;  }  public Object[] getCodeParams() {  return this.codeParams;  }  public String getCodeDesc() {  return this.codeDesc;  }  } |

# 应用说明

## 订单批价

|  |
| --- |
| /\*\*  \* Description: 网单订单处理，协调租费状态、批价、余额、销账等各个模块流程和事务  \* Created by qiupq at 2018/3/10 22:09  \*/  public class OrderRating extends TwoPhaseNestService<EvaluateResultDto, GoodsInstanceLog> {  private static boolean first = true;  public OrderRating() {  }  public static EvaluateResultDto evaluateService(GoodsInstanceLogDto dto) {  OrderRating orderRating = new OrderRating();  GoodsInstanceLog log = GoodsInstanceLogConverter.fromGoodsInstanceLogDto(dto);  log.setBusiPhase(BusiPhase.EVA.name());  log.setOrderId(IdUtil.uuid32());  EvaluateResultDto result = new EvaluateResultDto();  try {  result = orderRating.evaluate0(log);  }catch (Exception e){  //写日志  logSystemError(log, result,e);  //抛异常  throw RateException.makeAppException(e);  } |

# 附录

## 附录一：@SuppressWarnings("serial")在Java中有什么作用？

* 第一种说明：

@SuppressWarnings可以抑制一些能通过编译但是存在有可能运行异常的代码会发出警告，你确定代码运行时不会出现警告提示的情况下，可以使用这个注释。

("serial") 是序列化警告，当实现了序列化接口的类上缺少serialVersionUID属性的定义时，会出现黄色警告。可以使用@SuppressWarnings将警告关闭

* 第二种说明：

比如有个类实现了java.io.Serialize接口：

package com.onede4.test;

public class TestSerial implements java.io.Serializable {

}

如果代码仅仅写上这样，放在eclipse里会给你提示一个黄色的警告，为啥呢，就是因为，你实现了Serializable 接口，但是没有给出

private static final long serialVersionUID = 5952689219411916553L;

类似这样的私有字段（这个字段是实现了Serializable接口需要的，但不必须）

只要加上类似这样的一个字段，黄色警告就会消失，但是你如果就是不想加这个字段，同时还不想看到那个黄色的警告，没问题，那就要在类上加一个注解（@SuppressWarnings("serial")）

package com.onede4.test;

@SuppressWarnings("serial")

public class TestSerial implements java.io.Serializable {

}

这样eclipse就不会有那个黄色的警告了

不过我们还是推荐给出private static final long serialVersionUID = 5952689219411916553L;

当你的类进行序列化和反序列化操作的时候如果没有这个字段可能会出问题，所以尽量加上这个字段，不要用@SuppressWarnings("serial")注解