****

**广州宏天软件BPMX3系统**

**消息处理指导说明**

文档控制页

广州宏天软件有限公司

2012年12月

**版本记录**

|  |  |  |  |
| --- | --- | --- | --- |
| **版本号** | **版本描述** | **责任人** | **修订日期** |
| V0.1 | 草稿 | 庄晓晖 | 2012-12-03 |
| V0.2 | 草稿-增加部署方法 | 庄晓晖 | 2012-12-04 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**目 录**

[1.问题提出 1](#_Toc342316691)

[2.解决方案 2](#_Toc342316692)

[3.具体实现 3](#_Toc342316693)

[3.1 配置文件配置 3](#_Toc342316694)

[3.2 接口实现 4](#_Toc342316695)

问题提出

之前对消息队列处理是硬编码的，对遇到类型的消息就进行处理。需要修改方法，对原来的代码具有依赖性和侵略性。

原来的处理方式如下：

**public** **void** sendMessage(Object model) **throws** Exception {

MessageEngine mSender = (MessageEngine) AppUtil.getBean(MessageEngine.**class**);

//发送邮件

**if** (model **instanceof** MailModel) {

//处理发送邮件消息实现

}

//短消息

**else** **if** (model **instanceof** SmsMobile) {

//处理发送短消息实现

}

//内部消息

**else** **if** (model **instanceof** InnerMessage) {

//处理内部消息实现

}

//如果其它类型的消息又要进行处理 。 这样具有依赖和侵略性

}

解决方案

如果按照上面说的方式进行处理多种消息会对原来的代码进行修改，现在就只需要留出接口，自己对多种消息可以通过配置、接口进行实现。

现在处理方式：

// 处理消息

**private** Map<String, IJmsHandler> handlers = **new** HashMap<String, IJmsHandler>();

**public** **void** setHandlers(Map<String, IJmsHandler> handlers) {

**this**.handlers = handlers;

}

/\*\*

\* 发送消息

\*

\* **@param** model

\* 发送的对象

\* **@throws** Exception

\*/

**public** **void** sendMessage(Object model) **throws** Exception {

**if** (BeanUtils.*isNotEmpty*(handlers) && BeanUtils.*isNotEmpty*(model)) {

handlers.get(model.getClass().getName()).handMessage(model);

} **else** {

**throw** **new** Exception("Object:[" + model + "] is not entity Object ");

}

}

具体实现

* 1. 配置文件配置

/resources/app-jms.xml 该配置文件配置说明：

<!-- 消息消费者 -->

<bean name="messageConsumer" class="com.hotent.core.jms.MessageConsumer">

<property name="handlers">

<map>

<entry key="com.hotent.core.model.MailModel">

<bean class="com.hotent.platform.service.jms.impl.MailHandler"></bean>

</entry>

<entry key="com.hotent.core.model.SmsMobile">

<bean class="com.hotent.platform.service.jms.impl.SmsHandler"></bean>

</entry>

<entry key="com.hotent.core.model.InnerMessage">

<bean class="com.hotent.platform.service.jms.impl.InnerHandler"></bean>

</entry>

<!—可以实现多种类型的发送消息-->

</map>

</property>

</bean>

找到 messageConsumer 这个bean 在<map>下配置 要实现的发送消息类及发送消息的Model。发送消息实现类需要实现IJmsHandler接口。

entry 的key 配置 发送消息的Model

bean 的class配置 发送消息的实现类(该类需要实现IJmsHandler接口)

* 1. 接口实现

在实现IJmsHandler接口的类的方法进行发送消息的相关。

比如实现发送邮件消息如下：

/\*\*

\* 发送邮件消息的实现类

\* **@author** zxh

\*

\*/

**public** **class** MailHandler **implements** IJmsHandler {

**private** **final** Log logger = LogFactory.getLog(MailHandler.**class**);

@Resource

**private** MessageEngine messageEngine;

@Override

**public** **void** handMessage(Object model) {

MailModel mailModel = (MailModel) model;

//具体发送邮件的方法

SimpleMailMessage message = **new** SimpleMailMessage();

message.setSubject(mailModel.getSubject());

message.setTo(mailModel.getTo());

message.setCc(mailModel.getCc());

message.setBcc(mailModel.getBcc());

message.setText(mailModel.getContent());

message.setSentDate(mailModel.getSendDate());

messageEngine.sendMail(message);

logger.debug("MailModel");

}

}