wangleifire

wangleifire

mule入门 xmpp tcp 配置 之 xml配置 | 什么是EIP

2009-03-19

ESB学习笔记（Spring Integration实战）

ESB学习笔记（Spring Integration实战）

- 介绍

Spring Integration是Spring公司的一套ESB框架。
前面ESB介绍中我也做了一定了解。我们来看一下它主要做什么的。
Spring Integration is motivated by the following goals:

- Provide a simple model for implementing complex enterprise integration solutions.(暂时相信它吧，谁让它搞个Spring框架，的确给人方便一把。)
- Facilitate asynchronous, message-driven behavior within a Spring-based application.(这个不谈，Spring框架就是它玩的。再说这一点与它竞争只有Mule啦。)
- Promote intuitive, incremental adoption for existing Spring users. (也暂时相信它，别人都只说给用户提升。)

Spring Integration is guided by the following principles:

- Components should be loosely coupled for modularity and testability.（松耦合，好像很早很早就听说过。像做梦一样）
- The framework should enforce separation of concerns between business logic and integration logic.（分开程度要取决业务吧。）
- Extension points should be abstract in nature but within well-defined boundaries to promote reuse and portability.（美妙现实世界产品）

- 源码下载打开它的网页，http://www.springsource.org/spring-integration

主页上也没有东东，但有个下源代码的地方，svn开工啦。

Java代码

```
svn co https://src.springframework.org/svn/spring-integration/trunk  springintegration
```

Java代码

```
svn co https://src.springframework.org/svn/spring-integration/trunk  springintegration
```

下载完后，进入build-spring-integration目录执行ant.完成后，导入到Eclipse中。
导入项目会有很多，先添加时会有报错。这里需要添加一个变量。
IVY_CACHE＝<checkout-dir>/ivy-cache/repository

这里要注意的事，也是我遇到问题。执行ant时，他会去下载Ivy，如果你本身在%ANT_HOME%\lib里有Ivy.jar包，由于我暂时找不到如何处理，我就直接将Ant中的jar删除掉后就没有问题。
另外在ant过程中，测试步骤可能会在file模块中出现问题，可以将相关test类中代码注释掉。

- HelloWorld源码分析在samples项目中，打开helloworld包里面有三个文件。

Java代码

```
package org.springframework.integration.samples.helloworld;

/**
 * @author Mark Fisher
 */
public class HelloService {

    public String sayHello(String name) {
        return "Hello " + name;
    }

```

- FLEX 3D (1)
- 网页游戏 (2)
- TDD (1)
- 项目管理 (11)
- flash game (3)
- iphone app (1)
- c# (28)
- 通信 (4)
- mvvm (0)
- MEF (0)

我的相册

溜真冰

共 14 张

我的留言簿 >>更多留言

- 我们公司正招聘RED5 和 FLEX 工程师，广州工作，希望能邀请到你，我的QQ ...
  -- by kuaisou
- 请问：CruiseControl编译时不能识别VS2008的头文件,我已经在系统变 ...
  -- by deiqiyelu
- 我是一个web项目，里面加了Flex项目进去，我想把Flex项目中的as文件和as ...
  -- by zengxiaoxia123

其他分类

- 我的收藏 (0)
- 我的书籍 (2)
- 我的论坛主题贴 (0)
- 我的所有论坛贴 (0)
- 我的精华良好贴 (0)

最近加入圈子

存档

- 2011-02 (1)
- 2010-11 (1)
- 2010-10 (2)

```
}
```

Java代码

```java
package org.springframework.integration.samples.helloworld;

/**
 * @author Mark Fisher
 */
public class HelloService {

        public String sayHello(String name) {
                return "Hello " + name;
        }

}
```

helloworldDemo.xml

Xml代码

```xml
<?xml version="1.0" encoding="UTF-8"?>
<beans:beans xmlns="http://www.springframework.org/schema/integration"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:beans="http://www.springframework.org/schema/beans"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
            http://www.springframework.org/schema/beans/spring-beans-2.5.xsd
            http://www.springframework.org/schema/integration
            http://www.springframework.org/schema/integration/spring-integration-1.0.xsd">

    <channel id="inputChannel"/>

    <channel id="outputChannel">
        <queue capacity="10"/>
    </channel>

    <service-activator input-channel="inputChannel"
                       output-channel="outputChannel"
                       ref="helloService"
                       method="sayHello"/>

    <beans:bean id="helloService" class="org.springframework.integration.samples.helloworld.HelloService"/>
</beans:beans>
```

Xml代码

```xml
<?xml version="1.0" encoding="UTF-8"?>
<beans:beans xmlns="http://www.springframework.org/schema/integration"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:beans="http://www.springframework.org/schema/beans"
        xsi:schemaLocation="http://www.springframework.org/schema/beans
                        http://www.springframework.org/schema/beans/spring-beans-2.5.xsd
                        http://www.springframework.org/schema/integration
                        http://www.springframework.org/schema/integration/spring-integration-1.0.xsd">

        <channel id="inputChannel"/>

        <channel id="outputChannel">
                <queue capacity="10"/>
        </channel>

        <service-activator input-channel="inputChannel"
                           output-channel="outputChannel"
                           ref="helloService"
                           method="sayHello"/>

```

- 更多存档...

评论排行榜

- Windows环境下配置+运行red5源码+AS3连接 ...
- SQLITE入门至精通
- air写文件保存在安装目录
- WPF 新弹出窗口抢焦点问题
- WPF实现RichTextBox插入图片及调整行距

RSS
Add to Google


```
        <beans:bean id="helloService" class="org.springframework.integration.samples.helloworld.HelloService"/>
</beans:beans>
```

HelloWorldDemo.java

Java代码

```java
package org.springframework.integration.samples.helloworld;

import org.springframework.context.support.AbstractApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
import org.springframework.integration.channel.BeanFactoryChannelResolver;
import org.springframework.integration.channel.ChannelResolver;
import org.springframework.integration.channel.PollableChannel;
import org.springframework.integration.core.MessageChannel;
import org.springframework.integration.message.StringMessage;

/**
 * Demonstrates a basic message endpoint.
 *
 * @author Mark Fisher
 */
public class HelloWorldDemo {

    public static void main(String[] args) {
        AbstractApplicationContext context = new ClassPathXmlApplicationContext("helloWorldDemo.xml", HelloWorldDemo.class);
        ChannelResolver channelResolver = new BeanFactoryChannelResolver(context);
        MessageChannel inputChannel = channelResolver.resolveChannelName("inputChannel");
        PollableChannel outputChannel = (PollableChannel) channelResolver.resolveChannelName("outputChannel");
        inputChannel.send(new StringMessage("World"));
        System.out.println(outputChannel.receive(0).getPayload());
        context.stop();
    }

}
```

Java代码

```java
package org.springframework.integration.samples.helloworld;

import org.springframework.context.support.AbstractApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
import org.springframework.integration.channel.BeanFactoryChannelResolver;
import org.springframework.integration.channel.ChannelResolver;
import org.springframework.integration.channel.PollableChannel;
import org.springframework.integration.core.MessageChannel;
import org.springframework.integration.message.StringMessage;

/**
 * Demonstrates a basic message endpoint.
 *
 * @author Mark Fisher
 */
public class HelloWorldDemo {

        public static void main(String[] args) {
                AbstractApplicationContext context = new ClassPathXmlApplicationContext("helloWorldDemo.xml", HelloWorldDemo.class);
                ChannelResolver channelResolver = new BeanFactoryChannelResolver(context);
                MessageChannel inputChannel = channelResolver.resolveChannelName("inputChannel");
                PollableChannel outputChannel = (PollableChannel) channelResolver.resolveChannelName("outputChannel");
```

```
                    inputChannel.send(new StringMessage("World"));
                    System.out.println(outputChannel.receive(0).getPayload());
                    context.stop();
            }

}
```

- Cafe源码分析 Cafe示例描述的是星巴克的订单处理故事。

其示例描述在：http://www.enterpriseintegrationpatterns.com/ramblings/18_starbucks.html
这里简单描述一下，以免大家看英文太累
文章讲在星巴克喝咖啡时，收银员可能只有一个，而冲咖啡员工会有多个，如何让收银员产生订单异步发送给冲咖啡员工。并且冲咖啡员工可能是竞争上岗的，就当他们是计件工吧。
这里要考虑问题：
1，冲咖啡员工使用不同设备，不同咖啡冲调时间可能不同。
2，冲咖啡员工可能会将相同类型的咖啡同时一起冲调。
星巴克如何处理这个问题？
就当他解决了这个问题，它是如何把每个咖啡又送回给每个客户呢？当然，星巴克采用“标识关系模式”，将每个咖啡杯上标上名称，并通过叫喊方式。
但并不是每天都是美好的，总有出错的时候。例如，收银员无法支付？冲调一杯你不喜欢的咖啡，你要换一杯？冲咖啡的设备坏了，星巴克要退你钱...这些异常情况如何处理。
因此就会有以下三种方式异常处理：
1,关闭交易，什么都不做。
2,重做，重新发起行为。
3,修正行为,相当于退钱这种行为。
因此，这里这篇文章后面讨论一下两阶段提交为什么不适合星巴克，如果你让收银员、冲咖啡员工，买单的人需要在一个“事务”中，交易所有完成后，再进行下一个业务。估计星巴克会马上倒闭啦。因此星巴克采用“Conversation pattern”模式。

好啦，业务了解清楚，我们再来看一下完整XML文件。在这里我没有采用示例详细的xml方式，而没有采用annotation方式。

Xml代码

```xml
<?xml version="1.0" encoding="UTF-8"?>
<beans:beans xmlns="http://www.springframework.org/schema/integration"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:beans="http://www.springframework.org/schema/beans"
    xmlns:stream="http://www.springframework.org/schema/integration/stream"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
            http://www.springframework.org/schema/beans/spring-beans-2.5.xsd
            http://www.springframework.org/schema/integration
            http://www.springframework.org/schema/integration/spring-integration-1.0.xsd
            http://www.springframework.org/schema/integration/stream
            http://www.springframework.org/schema/integration/stream/spring-integration-stream-1.0.xsd">

    <gateway id="cafe" service-interface="org.springframework.integration.samples.cafe.Cafe"/>

    <channel id="orders"/>
    <splitter input-channel="orders" ref="orderSplitter" method="split" output-channel="drinks"/>

    <channel id="drinks"/>
    <router input-channel="drinks" ref="drinkRouter" method="resolveOrderItemChannel"/>

    <channel id="coldDrinks">
        <queue capacity="10"/>
    </channel>
    <service-activator input-channel="coldDrinks" ref="barista"
                       method="prepareColdDrink" output-channel="preparedDrinks"/>

    <channel id="hotDrinks">
        <queue capacity="10"/>
    </channel>
    <service-activator input-channel="hotDrinks" ref="barista"
```

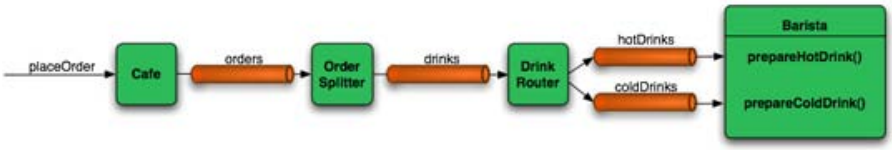
```
                        method="prepareHotDrink" output-channel="preparedDrinks"/>

    <channel id="preparedDrinks"/>
    <aggregator input-channel="preparedDrinks" ref="waiter"
                method="prepareDelivery" output-channel="deliveries"/>

    <stream:stdout-channel-adapter id="deliveries"/>

    <beans:bean id="orderSplitter"
                class="org.springframework.integration.samples.cafe.xml.OrderSplitter"/>

    <beans:bean id="drinkRouter"
                class="org.springframework.integration.samples.cafe.xml.DrinkRouter"/>

    <beans:bean id="barista" class="org.springframework.integration.samples.cafe.xml.Barista"/>

    <beans:bean id="waiter" class="org.springframework.integration.samples.cafe.xml.Waiter"/>

    <poller id="poller" default="true">
        <interval-trigger interval="1000"/>
    </poller>

</beans:beans>
```

Xml代码

```
<?xml version="1.0" encoding="UTF-8"?>
<beans:beans xmlns="http://www.springframework.org/schema/integration"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:beans="http://www.springframework.org/schema/beans"
    xmlns:stream="http://www.springframework.org/schema/integration/stream"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
            http://www.springframework.org/schema/beans/spring-beans-2.5.xsd
            http://www.springframework.org/schema/integration
            http://www.springframework.org/schema/integration/spring-integration-1.0.xsd
            http://www.springframework.org/schema/integration/stream
            http://www.springframework.org/schema/integration/stream/spring-integration-stream-1.0.xsd">

    <gateway id="cafe" service-interface="org.springframework.integration.samples.cafe.Cafe"/>

    <channel id="orders"/>
    <splitter input-channel="orders" ref="orderSplitter" method="split" output-channel="drinks"/>

    <channel id="drinks"/>
    <router input-channel="drinks" ref="drinkRouter" method="resolveOrderItemChannel"/>

    <channel id="coldDrinks">
        <queue capacity="10"/>
    </channel>
    <service-activator input-channel="coldDrinks" ref="barista"
                       method="prepareColdDrink" output-channel="preparedDrinks"/>

    <channel id="hotDrinks">
        <queue capacity="10"/>
    </channel>
    <service-activator input-channel="hotDrinks" ref="barista"
                       method="prepareHotDrink" output-channel="preparedDrinks"/>

    <channel id="preparedDrinks"/>
    <aggregator input-channel="preparedDrinks" ref="waiter"
                method="prepareDelivery" output-channel="deliveries"/>

    <stream:stdout-channel-adapter id="deliveries"/>
```

```

    <beans:bean id="orderSplitter"
                class="org.springframework.integration.samples.cafe.xml.OrderSplitter"/>

    <beans:bean id="drinkRouter"
                class="org.springframework.integration.samples.cafe.xml.DrinkRouter"/>

    <beans:bean id="barista" class="org.springframework.integration.samples.cafe.xml.Barista"/>

    <beans:bean id="waiter" class="org.springframework.integration.samples.cafe.xml.Waiter"/>

    <poller id="poller" default="true">
        <interval-trigger interval="1000"/>
    </poller>

</beans:beans>
```

以下是参考文档中的示例描述图：



CafeDemo代码创建了订单。这家咖啡店有两种饮料，一种是热的，一种是冷的，消息将这订单包装到一个"orders"的channel（频道）。一个endpoint侦听到订单频道并根据订单情况进行分开处理。

完成分开处理后，程序交给DrinksRouter经过drink频道。而DrinkRouter一个职责就是将订单内容中的热咖啡和冷咖啡交给不同的channel处理。

Xml代码

```
<gateway id="cafe" service-interface="org.springframework.integration.samples.cafe.Cafe"/>
```

Xml代码

```
<gateway id="cafe" service-interface="org.springframework.integration.samples.cafe.Cafe"/>
```


这里Gateway主要是根据接口生成代理类。

Java代码

```
Cafe cafe = (Cafe) context.getBean("cafe");
                DrinkOrder order = new DrinkOrder();
                Drink hotDoubleLatte = new Drink(DrinkType.LATTE, 2, false);
                Drink icedTripleMocha = new Drink(DrinkType.MOCHA, 3, true);
                order.addDrink(hotDoubleLatte);
                order.addDrink(icedTripleMocha);
                for (int i = 0; i < 100; i++) {
                        cafe.placeOrder(order);
                }
```

Java代码

```
Cafe cafe = (Cafe) context.getBean("cafe");
                DrinkOrder order = new DrinkOrder();
                Drink hotDoubleLatte = new Drink(DrinkType.LATTE, 2, false);
                Drink icedTripleMocha = new Drink(DrinkType.MOCHA, 3, true);
                order.addDrink(hotDoubleLatte);
                order.addDrink(icedTripleMocha);
                for (int i = 0; i < 100; i++) {
                        cafe.placeOrder(order);
                }
```

Java代码

```java
@MessageEndpoint(input="orders", output="drinks")
public class OrderSplitter {

	@Splitter
	public List<Drink> split(Message<DrinkOrder> orderMessage) {
		return orderMessage.getPayload().getDrinks();
	}

}
```

Java代码

```java
@MessageEndpoint(input="orders", output="drinks")
public class OrderSplitter {

	@Splitter
	public List<Drink> split(Message<DrinkOrder> orderMessage) {
		return orderMessage.getPayload().getDrinks();
	}

}
```


Java代码

```java
@MessageEndpoint(input="drinks")
public class DrinkRouter {

	@Router
	public String resolveDrinkChannel(Drink drink) {
		return (drink.isIced()) ? "coldDrinks" : "hotDrinks";
	}

}
```

Java代码

```java
@MessageEndpoint(input="drinks")
public class DrinkRouter {

	@Router
	public String resolveDrinkChannel(Drink drink) {
		return (drink.isIced()) ? "coldDrinks" : "hotDrinks";
	}

}
```


Xml代码

```xml
<handler-endpoint handler="coldBarista" input-channel="coldDrinks"
	method="prepareColdDrink">
</handler-endpoint>

<handler-endpoint handler="hotBarista" input-channel="hotDrinks"
	method="prepareHotDrink">
</handler-endpoint>
```

Xml代码

```xml
<handler-endpoint handler="coldBarista" input-channel="coldDrinks"
	method="prepareColdDrink">
</handler-endpoint>
```

```
4.
5.  <handler-endpoint handler="hotBarista" input-channel="hotDrinks"
6.          method="prepareHotDrink">
7.  </handler-endpoint>
```

Java代码

```
1.  public void prepareColdDrink(Message<Drink> drinkMessage) {
2.                  Drink drink = drinkMessage.getPayload();
3.                  //no changes to the rest of the code
4.          }
```

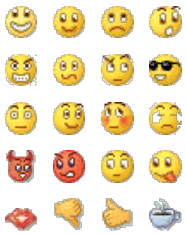
mule入门 xmpp tcp 配置 之 xml配置 | 什么是EIP

20:44 | 浏览 (1678) | 评论 (0) | 分类: camel | 相关推荐

评论

发表评论

表情图标



B I U Quote Code List Img URL Flash Table

字体颜色: 标准 字体大小: 标准 对齐: 标准

提示：选择您需要装饰的文字, 按上列按钮即可添加上相应的标签

您还没有登录，请登录后发表评论(快捷键 Alt+S / Ctrl+Enter) 提交