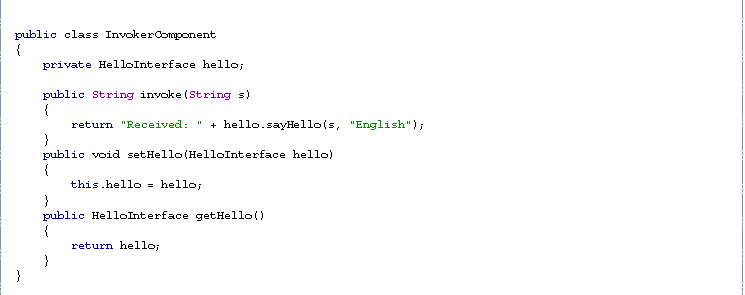
组件在执行期间可以利用绑定去调用外部服务。这种绑定用于一个Java组件绑定一个Java接口,或单一接口方法，到一个出站端点。被调用的外部服务应该实现相同的接口,并且该组件应该封装一个引用接口，通过Mule配置构建器在引导阶段被初始化。引用将使用反射代理类被初始化。



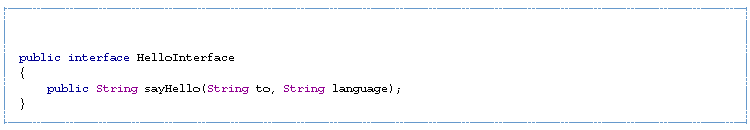
与组件的绑定,您可以配置多个接口或一个接口和一个端点绑定到一个或多个Mule端点。

Mule目前支持Java组件(默认的组件在Mule中)和脚本组件的组件绑定,如Groovy和JRuby。这个页描述了如何配置。

可以通过组件绑定调用外部服务。绑定接口作为一个字段像普通的bean的getter和setter方法那样添加到组件。在绑定中配置的组件,您可以将口连同一个接口里的方法一起绑定到一个mule端点。当该方法被调用时,将调用参数越过Mule端点都被送到另一个服务,这可能是本地也可能是远程的。结果可能从服务中返回并使用返回的参数的方法传递回组件。这个模型非常类似于传统的RPC调用。举例说明：



在这个例子中,该组件InvokerComponent有一个字段“hello”,是HelloInterface类型 getter和setter方法是他的方法， invoke方法将被服务调用同时调用hello.sayhello()方法。 这个调用会导致另一个服务调用。HelloInterface用一个方法sayHello是非常简单的。



在Mule 3.x 中,你可以设置接口返回类型到MuleMessage，目的是给你的组件访问完整的消息,而不只是有效负载。

现在,您只需配置您的组件去绑定sayHello方法到端点去调用另一个服务。



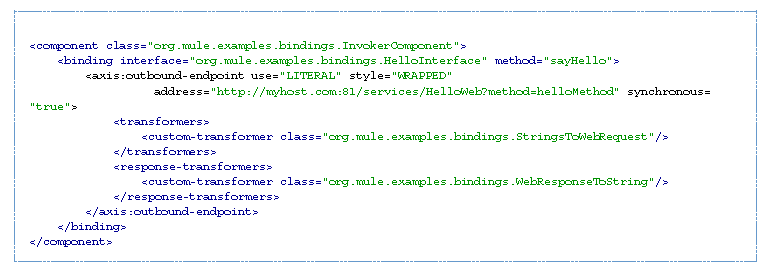
外部web服务的调用是同步的,因为你想要返回一个结果和锁定,直到调用完成。

注意,组件绑定不会使用Java代理对象,除非代理明确处理绑定接口方法。如果使用Spring组件,确保您使用的是CGLib代理。更多信息在spring-AOP代理行为,参见http://static.springsource.org/spring/docs/3.0.x/spring-framework-reference/html/aop.html # aop-introduction-proxies。 如果你使用注释处理器,比如事务,您将指定以下:



处理数据类型

当在端点上使用正常的转换配置去调用和接收结果时，你可以做数据转换处理。在上面的示例中,假设web服务在等一个org. mule examples.bindings.WebHelloRequest对象,并返回一个org mule. examples.bindings WebHelloResponse对象。 你不希望你的组件知道这些外部数据类型,所以，当需要调用的时候。您可以配置转换器去转换做转换



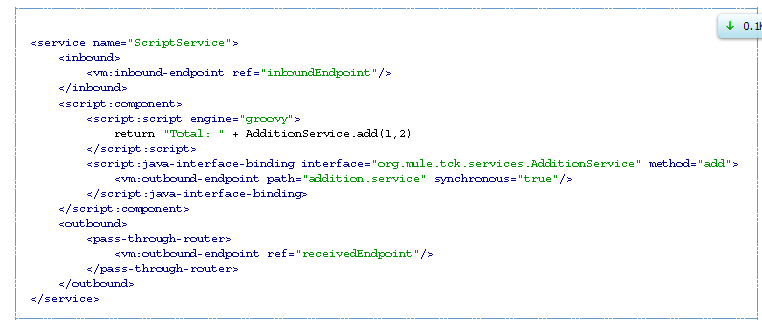
异常

如果远程服务调用触发一个异常的故障,这个异常会序列化的回到本地服务调用并且抛出。如果你的服务想要处理这个异常,您必须在绑定方法处添加异常处理(或java朗例外)并且想往常一样使用try catch块。

目前,invoke()方法的org.mule.routing.binding.BindingInvocationHandler 当有一个异常,甚至当绑定方法/接口抛出一个异常的声明时，总有一个UndeclaredThrowableException抛出。 在Mule 3。 X中, 如果原因是已知的，并且类型匹配一个异常声明是在给定的方法的“抛出”条款中。声明的异常将会返回

脚本组件绑定

类似于Java组件绑定,脚本绑定为你的脚本组件启用相同的行为。当使用一个脚本组件,绑定是绑定到脚本环境文，通过使用绑定接口的类名进行访问。



这个组件的实现是包含在< script:script >元素:



我们提到的绑定接口使用短类名AdditionService和调用add方法,它将调用本地添加服务。

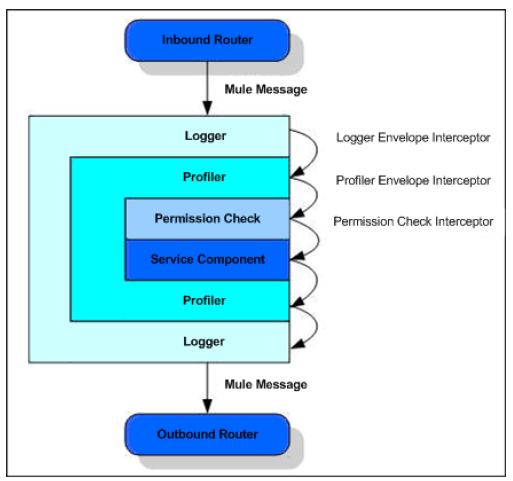
使用拦截器

Mule ESB拦截器是用于将行为附加到多个服务组件上。拦截器模式通常被称为实用AOP(面向方面编程),因为它允许开发人员在一个对象去拦截处理过程，并且尽可能的去改变处理过程和结果。(参见Spring AOP)。 拦截器对附加行为非常有用的，如服务组件的分析，权限和安全检查。

AbstractEnvelopeInterceptor是一个信封过滤器,将在组件调用前后被执行。 适合日志记录和分析。

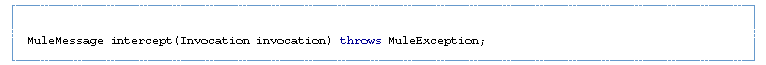
拦截事件流

下面显示了一个示例拦截器堆栈和事件流。



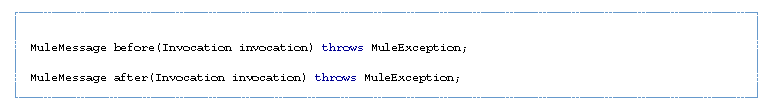
定义拦截器

如果你想在入站消息流上拦截消息流到一个组件,你应该实现拦截器接口。它有一个单一的方法:



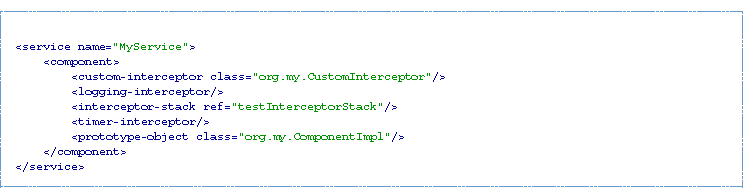
调用参数包含当前的信息和目标组件的服务对象。开发人员在需要的时候可以从消息和操作中提取当前MuleMessage。这个拦截方法必须返回一个MuleMessage传递给组件(或到下一个拦截器在链)。

这个EnvelopeInterceptor以同样的方式工作,除了它所公开了两个方法--得到调用之前和之后的事件处理:



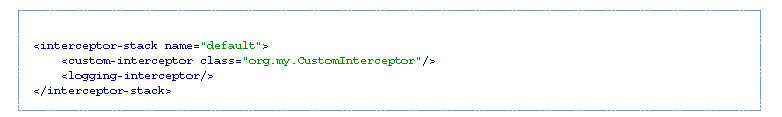
配置拦截器

拦截器可以配置到您的组件，如下所示:



当您配置拦截器,您必须明确的指定的对象工厂(在这个示例中,< prototype-object>)取代 <component class>快捷方式。

您还可以定义拦截器堆栈,这是一个或多个拦截器,可以使用一个逻辑名称引用。使用拦截器堆栈,您必须先将其配置到Mule XML配置文件的全局区域 (上图<model>元素):



您可以配置多个<interceptor>元素在你的组件,您可以混合使用内置的拦截器、自定义拦截器和拦截器堆栈引用。

拦截器配置引用

缓存:意外的程序错误: java.lang.NullPointerException



日志记录拦截器(从1. x移植)。

属性



子元素

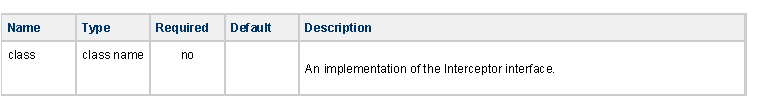


缓存:意外的程序错误:java.lang.NullPointerException



一个用户实现拦截器。

属性



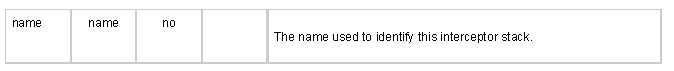
子元素



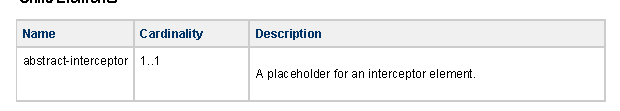
缓存:意外的程序错误:java.lang.NullPointerException



属性

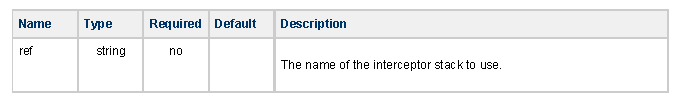
子元素





一堆全局定义的intereceptors的引用。

属性



子元素



缓存:意外的程序错误:java.lang.NullPointerException



计时器录拦截器(从1. x移植)。

属性



子元素



利用传输连接

一个传输负责承载消息，从应用程序到Mule ESB的框架上应用程序。在EIP条款,传输用于实现消息通道并且提供一致的方式到一个底层数据源或消息通道的连接。例如,HTTP传输处理通过HTTP协议发送的消息,而文件传输的读和写文件放在服务器的文件系统。

一个传输在一个特定的通道里还提供了一组功能处理消息。这包括特定的转换器像<body-to-parameter-map-transformer>在HTTP传输和其他特定的元素(如各种文件名解析器的文件传输)

一个传输的核心是连接器。连接器维护传输的配置和声明。当收到消息,一个连接器使用一个消息接收器,它将读取的数据,包它作为消息,并在一个流程或服务组件的入站路由器中将其传递到第一个消息处理器。当发送消息,连接器使用消息调度程序,它在一个流程或服务组件的出站路由器中将接收消息和路由指令从先前的消息处理器和将消息发送到下一个服务或外部服务。

传输是通过在Mule配置端点。您可以查看配置端点，解释了这是如何实现的。

一系列可用的传输和引用主题包括为每个特定的运输捆绑在Mule中的示例配置，可以在传输配置参考中找到。



配置传输器

你可以用以下的方式去配置传输

在Mule的XML配置文件中使用< connector >元素定义一个连接器配置。

在端点设置传输特性来为为单个端点实例定制传输行为。

使用一个端点的URL,它定义了方案和连接信息的传输协议(如tcp:/ / localhost:12345。获取更多信息科参照Mule端点uri。URI包括协议，其次是特定的传输信息,然后零或更多的参数设置为属性的连接器。

这个页面描述了所有传输的普通属性。对于每个传输类型的实际的配置参数为每一个传输单独定义。要看一个特定传输的细节,请参阅传输参考。

一般的连接器属性

所有连接器需要设置唯一的名称属性。此外,他们都包括以下常见的属性。

|  |  |  |  |
| --- | --- | --- | --- |
| 属性 | 描述 | 系统默认值 | 必需 |
| default-connector-exception-strategy | 异常策略用于连接器发生错误时 |  | 不 |
| receiver-threading-profile | 从连接器接收时间时使用该线程属性和WorkManager | 默认的接收着线程设置在Mule配置上 | 是 |
| dispatcher-threading-profile | 从连接器进行事件分发时使用线程属性和WorkManager | 默认调度程序线程设置在Mule配置上 | 是 |
| connection-strategy | 自Mule 2.0起不支持， 用重试策略代替 |  | 不 |
| service-overrides | 服务配置值的范围,对于传输可用于覆盖默认的配置。 |  | 不 |
| createMultipleTransactedReceivers | 是否为这个连接器创建多个并发接收器。这个属性被用于传输支持事务,特别是接收 ,TransactedPollingMessageReceiver扩展,提供了更好的吞吐量。 | false | 不 |
| numberOfConcurrentTransactedReceivers | 如果createMultipleTransactedReceivers被设置为true,大量的并发接收器,将会启动。 |  | 不 |
| dynamicNotification | 是否启用动态通知。 | False | 不 |
| validateConnections | 使用前mule来验证连接。请注意,这只是一个配置提示;传输的实现可能会也可能不会额外来验证连接。 | True | 不 |

你也可以在连接器上设置Spring 属性。当您正在使用一个自定义连接器时这是有用的。

重试策略

重试策略是用于配置当连接失败时连接器如何运转。对于完整的信息,请参见配置重新连接策略。

创建您自己的传输

关于Mule ESB创建一个自定义运输,请参照创建传输。

详细配置信息

* 线程配置文件
* 接收器线程配置文件
* 调度程序线程配置文件

服务覆盖

缓存:意外的程序错误: java.lang.NullPointerException

接收器线程配置文件

线程配置文件时要使用一个连接器接收消息。

***<receiver-threading-profile...> 属性***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 名称 | 类型 | 必需 | 系统默认值 | 描述 |
| maxThreadsActive | 整型 | 不 |  | 可以使用的线程的最大数。 |
| maxThreadsIdle | 整型 | 不 |  | 被毁前线程池中闲置或非活动线程的最大数量。 |
| threadTTL | 整型 | 不 |  | 再被丢弃之前，一个非活动线程多久保存在线程池的时间 |
| poolExhaustedAction | WAIT/DISCARD/DISCARD\_OLDEST/ABORT/RUN | 不 |  | 当最大线程池大小或队列的大小是固定的时候,这个值决定如何处理即将传入的任务。可能的值是:WAIT(等到一个线程可用时, 这个最低的线程数量是零时不要使用,这种情况下,一个线程可能永远不会变成可用),DISCARD(扔掉当前请求，返回),DISCARD\_OLDEST(扔掉最古老的请求,返回),ABORT(抛出一个RuntimeException),RUN(默认;该线程使执行请求运行任务本身,这有助于防止锁定)。 |
| threadWaitTimeout | 整型 | 不 |  | 当线程池的耗尽状态是WAIT时，表示要等多少毫秒。如果该值为负,它会无限期等待。 |
| doThreading | 布尔 | 不 | True | 是否应该使用线程(缺省为true)。 |
| maxBufferSize | 整型 | 不 |  | 当池是在最大的使用容量和池耗尽状态是WAIT时，决定有多少请求排队。缓冲区用作溢出。 |

缓存:意外的程序错误: java.lang.NullPointerException

调度程序线程配置文件

一个连接器调度消息是线程配置文件需要使用。

***<dispatcher-threading-profile...> 属性***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 名称 | 类型 | 必需 | 系统默认值 | 描述 |
| maxThreadsActive | 整型 | 不 |  | 可以使用的线程的最大数。 |
| maxThreadsIdle | 整型 | 不 |  | 被毁前线程池中闲置或非活动线程的最大数量。 |
| threadTTL | 整型 | 不 |  | 再被丢弃之前，一个非活动线程多久保存在线程池的时间 |
| poolExhaustedAction | WAIT/DISCARD/DISCARD\_OLDEST/ABORT/RUN | 不 |  | 当最大线程池大小或队列的大小是固定的时候,这个值决定如何处理即将传入的任务。可能的值是:WAIT(等到一个线程可用时, 这个最低的线程数量是零时不要使用,这种情况下,一个线程可能永远不会变成可用),DISCARD(扔掉当前请求，返回),DISCARD\_OLDEST(扔掉最古老的请求,返回),ABORT(抛出一个RuntimeException),RUN(默认;该线程使执行请求运行任务本身,这有助于防止锁定)。 |
| threadWaitTimeout | 整型 | 不 |  | 当线程池的耗尽状态是WAIT时，表示要等多少毫秒。如果该值为负,它会无限期等待。 |
| doThreading | 布尔 | 不 | True | 是否应该使用线程(缺省为true)。 |
| maxBufferSize | 整型 | 不 |  | 当池是在最大的使用容量和池耗尽状态是WAIT时，决定有多少请求排队。缓冲区用作溢出。 |

缓存:意外的程序错误: java.lang.NullPointerException

服务覆盖

服务覆盖允许连接器通过更改部分传输实现去进一步配置/定制, ,例如, 使用消息接收器或调度程序实现,或者消息适配器。

***<service-overrides...>属性***

