Java中的异常处理框架

Shubhashish_Mandal

2017 年 5 月 1 日 警察

下载 ExceptionHandler.zip - 16.5 KB

介绍

在 java 中，通过添加 try-catch-finally 块很容易处理异常。对于简单的应用程序或测试目的，它是可以的。但是，当您考虑开发企业应用程序时，您可以理解您需要一些结构良好的代码来以更好、更有效的方式解决这个问题。在本文中，我试图解释如何以更好的管理方式处理异常。

背景

在传统方法中，您在处理企业应用程序时可能会面临很多问题。
Initilay 很难找出问题所在。但是当应用程序的数量增加时，你可以理解这一点。
以下是关键因素：

1. 可维护性：经常可以看到，针对不同类型或相同类型的异常编写了相同类型的处理程序代码。如果需要任何更改，很难跟踪。因此，维护代码本身对于模块化基础应用程序来说是一个很大的挑战。

2.可重用性：如果目的相同，那么为什么我们不应该为不同的异常使用相同的代码。正如第1点所讨论的，这也降低了相同目的的可重用性的机会。

3. 可插拔和灵活：今天hadler 只编写工作来记录异常。以后可能需要发邮件或者需要post到某个jms队列或者需要做一些回滚任务或者可能需要做所有的任务或者可能是任务的组合。在正常方法中，几乎不可能按需更改代码，因为处理代码与异常源紧密耦合。

4.可读性：经常看到，catch块本身包含一些复杂的代码，不仅增加了方法的长度，而且降低了代码的可读性。

5.易于使用：在遗留应用程序中，有时很难追踪异常处理。在 trdationl 方法中，您必须花足够的时间来了解流程。

以上问题很常见，每个人都遇到过一次。

这个框架背后的关键策略是将处理程序逻辑与异常源分开，使关系成为有损耦合并在需要时钩住这个hadler。分离处理程序逻辑并不是什么大不了的事。我们可以创建一个处理程序类并将逻辑保留在其中。但主要关心的是如何挂钩它，以便它也增加可重用性、可维护性、灵活性和简单性。开始了。

让我们举一个简单的例子：

爪哇 复制代码

```
public void  checkLogin() {
     try  {

         //Statement 1 which may throw  new UserNotExistException("User not Exist.");

         //Statement 2 which may throw new InvalidCredentialException("Invalid Credentials.");

         //Statement 3 which may throw new DBException("Failed to retrieve data.");

         //Statement 4 which may throw new Exception("General Exception.");

     } catch (UserNotExistException uex) {
         //To do
     } catch (InvalidCredentialException fex) {
         //To do
     } catch (DBException dex) {
         //To do
     } catch (Exception ex) {
         //To do
     }
 }
```

上面的例子是一个非常简单的验证用户的例子。我们可以看到这个方法可能会抛出 4 种异常（在现实世界中可能会有所不同）。还有一些 catch 块可以捕获这些异常。我们非常熟悉这种类型的场景。

根据要求，如果发生任何异常，应用程序可能会采取以下处理程序动作（单个动作或动作组合）来处理每种类型的异常：（例如，我采取了 4 个处理程序动作）

1. 想记录异常
2. 想要将错误详细信息发送到某个 jms 队列
3. 想发邮件
4. 想做一些默认的工作

现在通过添加上述处理程序来重写 catch 块

爪哇 缩小▲ 复制代码

```
public boolean checkLogin() {
    try  {

        //Statement 1 which may throw  new UserNotExistException("User not Exist.");

        //Statement 2 which may throw new InvalidCredentialException("Invalid Credentials.");

        //Statement 3 which may throw new DBException("Failed to retrieve data.");

        //Statement 4 which may throw new Exception("General Exception.");

    }  catch (UserNotExistException uex) {
        //To do

        // want to log exception
        // want to do some default work

    } catch (InvalidCredentialException fex) {
        //To do

        // want to log exception
        // want to send mail
        // want to do some default work

    } catch (DBException dex) {
        //To do

        // want to log exception
        // want to send error details to some jms queue
        // want to send mail
        // want to do some default work
```

```
    } catch (Exception ex) {
        //To do
        // want to log exception

    }
    return <some_flag>;
}
```

当您在基于模块的应用程序中工作时，您可以理解维护代码库的痛苦。现在看下面的例子，它和上面的例子做同样的事情，但是以一种有效的方式。

爪哇 复制代码

```
@HandleExceptions({
    @HandleException(exceptionType = UserNotExistException.class, handlers = {
         "logExceptionHandler","defaultExceptionHandler"
    }) ,
    @HandleException(exceptionType = InvalidCredentialException.class, handlers = {
        "logExceptionHandler", "sendEmailExceptionHandler", "defaultExceptionHandler"
    }) ,
    @HandleException(exceptionType = DBException.class, handlers = {
        "logExceptionHandler", "JMSExceptionHandler", "sendEmailExceptionHandler",
"defaultExceptionHandler"
    }) ,
    @HandleException(exceptionType = Exception.class, handlers = {
        "logExceptionHandler"
    })
})
public void checkLogin() {
    //Statement 1 which may throw  new UserNotExistException("User not Exist.");

    //Statement 2 which may throw new InvalidCredentialException("Invalid Credentials.");

    //Statement 3 which may throw new DBException("Failed to retrieve data.");

    //Statement 4 which may throw new Exception("General Exception.");
}
```

如果我们忽略注解部分（稍后介绍），很容易理解它比上面的例子非常简单和干净。您可以看到方法主体内没有 catch 块。现在该方法更加清晰易读，并且只关注它打算做什么。
现在开始挖掘代码库以了解框架。我在 Spring 框架之上创建了这个框架。

这个框架背后的关键思想是在抛出异常之后拦截异常，但在它被捕获之前并根据需要执行必要的操作。为了实现这一点，我`@AfterThrowing`在 Spring 应用程序中使用了 Spring AOP建议。（如果您不熟悉 AOP（面向方面的编程），请访问 http://docs.spring.io/spring/docs/current/spring-framework-reference/html/aop.html）。

进入深处

现在是时候查看代码库了。**com.sm.exceptionhandler.annotation 包中**有三个用户定义的注释。

- `Handlers.java`
- `HandleException.java`
- `HandleExceptions.java`

在这个框架中有两种绑定异常处理程序的方法。一种是：

`Handlers.java`： 用于注释自定义异常类以注册默认处理程序。

C++ 复制代码

```
@Handlers({"fileExceptionHandler", "defaultExceptionHandler", "logExceptionHandler",
"JMSExceptionHandler"})
public class UserNotExistException extends BaseException {
```

第二个onr是：

HandleException.java: 可能抛出异常的类必须使用此注释进行注释。

C++ 复制代码

```
@Retention(RetentionPolicy.RUNTIME)
@Target(ElementType.METHOD)
public @interface HandleException {
    static final String DEFAULT_HANDLER = "defaultExceptionHandler";
    public Class<?> exceptionType() default Exception.class;
    public String[] handlers() default {DEFAULT_HANDLER};
}
```

它有两个属性。要响应特定异常，用户需要提及异常类型exceptionType 以及对此异常做出反应的处理程序列表。这里的**处理程序** 是 Spring bean id。可以提及针对特定异常的多个处理程序。

如果有可能获得多种类型的异常并且您想单独处理它们，那么exceptionType param 将帮助您解决这个问题。

您可以使用以下代码来处理多类型异常

C++ 复制代码

```
@HandleExceptions({
     @HandleException(exceptionType = FileNotExistException.class, handlers = {
         "defaultExceptionHandler", "logExceptionHandler",
         "JMSExceptionHandler"}),
     @HandleException(exceptionType = UserNotExistException.class, handlers = {
         "logExceptionHandler", "JMSExceptionHandler"})
})
public Integer testExceptionOne(int type) {
}
```

或者您可以编写最简单的一个来处理每个异常。

C++ 复制代码

```
@HandleException(handlers = {"fileExceptionHandler",
        "defaultExceptionHandler", "logExceptionHandler"})
public void testExceptionTwo() {
    throw new RuntimeException("Io Exception");
}
```

HandleException.java: 包含HandleException.

HandleExceptionAspect.java是在运行时基本上与应用程序编织的方面。使用@AfterThrowing,aspect 只能在抛出给定类型的异常时才能执行通知，并且还可以访问通知正文中抛出的异常。

ExceptionHandlerFramework.java 包含调用和处理相应处理程序的主要逻辑。

其余的 java 类非常简单明了，一目了然。

兴趣点

它非常简单易用

记住

附加的代码库是一个 netbeans 项目，并在 java 1.7 中进行了测试。

执照

分享

关于作者

Shubhashish_Mandal
软件开发人员（高级）
印度
手表该会员
让世界开源

评论和讨论

添加评论或问题 | 电子邮件提醒 | Search Comments

第一 页上一页 下一页

运行测试代码时出错
Devesh Singh 31-Oct-18 22:36

刷新 1

一般 新闻 建议 问题 错误 答案 笑话 赞美 咆哮 管理员

使用Ctrl+Left/Right 切换消息，Ctrl+Up/Down 切换主题，Ctrl+Shift+Left/Right 切换页面。