Headstrong

2008

Flex

Exception Handling in BlazeDS Environment

Arunkumar K

JP Techno Park, #3/1 Millers Road, Bangalore - 560052

Contents

[1. Exception Handling with BlazeDS and Flex 2](#_Toc210632679)

Ref: <http://www.flexpasta.com/index.php/2008/05/16/exception-handling-with-blazeds-and-flex/>

# Exception Handling with BlazeDS and Flex

A common hurdle a developer may face is dealing with exceptions in BlazeDS. When an exception is thrown in Java, how do we handle this in flex? Here is a simple and flexible approach inspired by Scott Morgan.

**1. Create a Java Class that extends RuntimeException.**  
  
package com.flexpasta.exception;  
public class FlexException extends RuntimeException  
{  
public FlexException(String message)  
{  
super(message);  
}  
}

**2. Create a matching flex class called FlexException**, and put the usual RemoteObject tag in their so BlazeDS will know what to look for. Also remember in actionscript, each object must be instantiated at least once, so be sure to do that somewhere with FlexException.

**3. In Java**, whenever there is a known exception, or some error where the end user needs to be shown a message, **throw a FlexException and pass in the message the user should see** in the constructor. Let’s say the user just entered an invalid password. throw new FlexException(”Wrong Password Entered, Please Try Again.”);

**4. Flex receives the Exception in the form of a fault.** In the fault handler, you call this method:  
  
/\*\*  
\* Parse the incoming fault for an FlexException class. If the exception class is found and it has a matching actionscript class,  
\* show an alert to the user with the correspoding messages. If we don't recognize the Exception type, show a generic error.  
\*  
\* @param fault  
\*  
\*/  
public static function handleException(fault:Fault): void  
{  
var msg : String = fault.faultString;  
var clazz : Class = getExceptionClass(fault);  
var instance : Error = null;  
if (clazz != null)  
{  
Alert.show(msg,'Error');  
}  
else  
{  
Alert.show('Error! Please try again. If this issue persists, contact the system administrator');  
}  
}  
/\*\*  
\*  
\* Return the Class corresponding to the exception by parsing the fault string. If a corresponding actionscript error class is found,  
\* return it. Otherwise, return null.  
\*  
\* @return  
\*  
\*/  
public static function getExceptionClass(fault:Fault) : Class  
{  
var clazz:Class = null;  
var index:int = myFault.faultString.indexOf("FlexException");  
if (index != -1)  
{  
var cname:String = myFault.faultString.substr(0,index-1);  
try  
{  
clazz = getDefinitionByName("com.flexpasta.FlexException") as Class;  
}  
catch (e:ReferenceError)  
{}  
}  
return clazz;  
}  
  
When we see a Flex Exception, we show the message to the end user. If for some reason we have an unplanned error in Java, like NullPointerException, a message is displayed to the end user: ‘Error! Please try again. If this issue persists, contact the system administrator’. A generic message to handle all unplanned errors.

**5. What if the app needs to DO something special with certain Java Exceptions?** Say the user has just entered 3 password attempts, and I want to be able to handle locking their account through an exception.

 Create a new class in Java called UserLockedException that extends FlexException

 Create a matching actionscript UserLockedException.

 Add code in the flex fault handler to look for a UserLockedException, when the fault handler sees it, it can still display a message to the end user, but now flex knows to take a different code path to complete the process.

**Conclusion**

It is pretty simple with BlazeDS to create a base exception handling system. Depending on project needs, this approach can easily be modified or enhanced. In flex, it is important to setup a strong exception handling that accounts for both planned and unplanned exceptions.