Exception Management Inside an Adapter Object

Application Designer binds its page processing to adapter objects providing properties and methods - as explained in the previous sections. What happens if an error happens at runtime, e.g. an error occurs in the method of an adapter object that is called after the user pressed a button?

This chapter covers the following topics:

- Normal Exceptions are to be Handled by the Application
- Errors and Runtime Exceptions - The Default Behavior
- Interrupting the Application Designer Request Processing - AdapterNotAvailableError
- Errors and Runtime Exceptions - The Special Behavior

Normal Exceptions are to be Handled by the Application

The first rule is: normal exceptions (i.e. no "Errors", no "Runtime Exceptions") are to be handled by the application itself.

This means: a property is provided by the corresponding set/get methods (or by an equivalent method when using dynamic binding). The methods must not throw any exception, i.e. in their declarations there is no "throws" element.

Example for a correct implementation:

```
public void setFirstName(String value)
{
    ...
    ...
}
public String getFirstName()
{
    ...
    ...
}
```

The following example is an *incorrect* implementation because application exceptions are thrown:

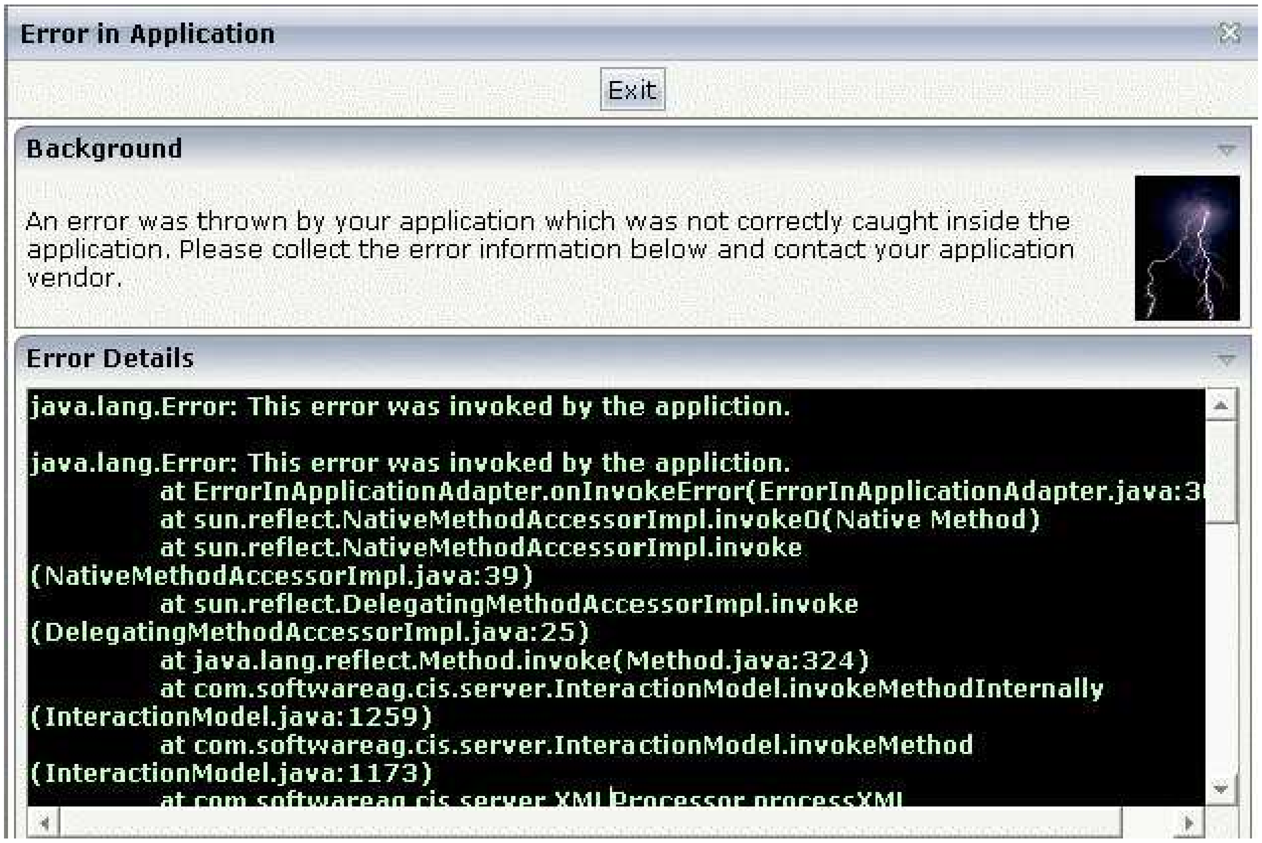
```
public void setFirstName(String value)
    throws ApplException
{
    ...
    ...
}
public String getFirstName()
    throws ApplException
{
    ...
    ...
}
```

Consequence: Application Designer passes values from the browser front end into the adapter object, invokes certain activities inside this object and collects data from the object to pass data changes back to the browser. Application exceptions are not relevant from Application Designer’s point of view - they only affect the application internally.

Errors and Runtime Exceptions - The Default Behavior

Of course your application still can throw "Error" exceptions or "Runtime" exceptions. These are the exceptions that need not be declared inside a method’s code - but that can be thrown any time at any place.

If Application Designer receives an error or runtime exception, Application Designer displays a page by default in which the error information is shown.



An error was thrown by your application which was not correctly caught inside the application. Please collect the error information below and contact your application vendor.

```
java.lang.Error: This error was invoked by the appliction.

java.lang.Error: This error was invoked by the appliction.
        at ErrorInApplicationAdapter.onInvokeError(ErrorInApplicationAdapter.java:3
        at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
        at sun.reflect.NativeMethodAccessorImpl.invoke
(NativeMethodAccessorImpl.java:39)
        at sun.reflect.DelegatingMethodAccessorImpl.invoke
(DelegatingMethodAccessorImpl.java:25)
        at java.lang.reflect.Method.invoke(Method.java:324)
        at com.softwareag.cis.server.InteractionModel.invokeMethodInternally
(InteractionModel.java:1259)
        at com.softwareag.cis.server.InteractionModel.invokeMethod
(InteractionModel.java:1173)
        at com.softwareag.cis.server.XMLProcessor.processXML
```

In addition, Application Designer writes a full stack dump into its runtime log.

Interrupting the Application Designer Request Processing - AdapterNotAvailableError

There may be some situations - within a special environment context - that do not allow to process the page at all. Maybe you have a page that requires a user to be logged on; if the Application Designer request processing starts now, you may decide with the method `reactOnDataCollectionStart` that you do not want to start the request since it does not make sense at all and just causes exceptions.

The only thing you want to do in such a scenario is to "escape" to a page which helps out of the situation. For example, if you miss logon information, you want to "escape" to the logon page and return to your original page afterwards.

The Java error class `AdapterNotAvailableError` is provided for this reason. The following adapter code shows an example on how to handle this error:

```
package com.softwareag.cis.demoapps;

...
...

public class Rescue1Adapter
    extends Adapter
{
    ...
    ...
    /** start of data transfer */
    public void reactOnDataTransferStart()
    {
        super.reactOnDataTransferStart();
        // fetch user and sytem info from session context
        m_user = (String)findSessionContext().lookup("rescueexample/user",false);
        m_system = (String)findSessionContext().lookup("rescueexample/system",false);
        // if not logged on ==> switch to rescue2 page
        if (m_user == null ||
            m_system == null)
        {
            // prepare Rescue2 page
            Rescue2Adapter r2a = (Rescue2Adapter)findAdapter(Rescue2Adapter.class);
            r2a.init("Rescue1.html");
            // throw error in order to interrupt normal processing and switch
            // to Rescue2 page
            throw new AdapterNotAvailableError("Rescue2.html");
        }
    }
    ...
    ...
}
```

Inside the `reactOnDataTransferStart` method, a user and a system variable are read from the session context. If one of them is null, the adapter decides to switch to page *Rescue2.html* and throws an `AdapterNotAvailableError` error. Before, it pre-fetches the page adapter of the page to escape to and initializes the page with certain information (in this example, it passes its own page name).

The error page is opened inside the same subsession as the one throwing the error.

Errors and Runtime Exceptions - The Special Behavior

There is a set of methods available in the adapter with which you can influence the standard error behavior:

- `handleErrorDuringInitPhase()`
- `handleErrorDuringSetPhase()`

- `handleErrorDuringInvokePhase()`
- `handleErrorDuringGetPhase()`

Depending on the method you have the following possibilities:

- **handleErrorDuringInitPhase()**
  This method is called when an error occurs in the `init` method of the adapter.

- **handleErrorDuringSetPhase() and handleErrorDuringGetPhase()**
  These methods are called when an error occurs during the set and the get phase of the adapter request processing.

  You may throw an `AdapterNotAvailableError` to navigate to a page of your own in order to present to the user detailed error information - and maybe also some way to solve the error.

- **handleErrorDuringInvokePhase()**
  This method is called when an error occurs during the invoke phase of the adapter request processing.

  You can decide whether normal Application Designer runtime processing continues, whether you want to navigate to an error handling page (via `PageNotAvailableError`), or whether the standard error processing of Application Designer is done.

See the API documentation (Java Doc) for further details.