Exception Handling Service

Summary

It aims to enable suitable action according to Exception by allowing the specific logic (called post-processing logic) per Exception to flow accurately, that is, Exception process when developing system based on e-government standard framework foundation.
With help of AOP, it was separated from business POJO and defined as an after throwing advice.
For contents related to AOP, refer to AOP module.

We'll talk about Exception.
When Exception occurs, the Exception generation class information and Exception type are important.
Exception generation class information and Exception type are used to determine whether they are target of post-processing logic.

```
        publicCategoryVOselectCategory(CategoryVOvo) throws Exception {
                CategoryVOresultVO = categoryDAO.selectCategory(vo);
                try {
    ....

    // If transferredresultVOis null,EgovBizExceptionoccurs (result.nodata.msg corresponds to message key)
                if (resultVO == null)
                        throwprocessException("result.nodata.msg");
                    // or throw processException("result.nodata.msg", generated Exception );
                returnresultVO;
        }
```

Description

We'll explain the type of executing post-processing logic (leavaTrace), not Exception, and post processing type of Exception mentioned earlier.
In brief,
Exception post-processing type is executed in the order of**AOP(pointCut⇒ after-throw) ⇒ExceptionTransfer.transfer() ⇒ExceptionHandlerService⇒ Handler**.

LeavaTraceis not the structure of using AOP, nor generates Exception. It simply aims to execute post-processing logic.
The order of execution is in the order of **LeavaTrace⇒TraceHandlerService⇒ Handler**.

First of all, let's examine Exception Handling.

AopConfig, ExceptionTransfer Setting and Description

Bean Setting

2 xml files are used in the sample for Exception post processing and leaveaTrace setting.(context-aspect.xml, context-common.xml)

First, we'll examine the area for Exception post-processing.
AOP setting for Exception Handling is as follows.
Since package structure changes during business development, Pointcut can be applied with modification ofegov.sample.service.*Impl.*(..)).

exceptionHandlerService that exists as property of ExceptionTransfer is designed to register several HandleManager.
Here, defaultExceptionHandleManager is registered.

context-aspect.xml

```
...
	<aop:config>
		<aop:pointcut id="serviceMethod"
			expression="execution(* egov.sample.service.*Impl.*(..))" />

		<aop:aspect ref="exceptionTransfer">
			<aop:after-throwing throwing="exception"
				pointcut-ref="serviceMethod" method="transfer" />
		</aop:aspect>
	</aop:config>

	<bean id="exceptionTransfer"
class="egovframework.rte.fdl.cmmn.aspect.ExceptionTransfer">
		<property name="exceptionHandlerService">
			<list>
				<ref bean="defaultExceptionHandleManager" />
			</list>
		</property>
	</bean>

	<bean id="defaultExceptionHandleManager"

	class="egovframework.rte.fdl.cmmn.exception.manager.DefaultExceptionHandleManager">
		<property name="patterns">
			<list>
				<value>**service.*Impl</value>
			</list>
		</property>
		<property name="handlers">
			<list>
				<ref bean="egovHandler" />
			</list>
		</property>
	</bean>

	<bean id="egovHandler"

	class="egovframework.rte.fdl.cmmn.exception.handler.EgovServiceExceptionHandler" />
...
```

defaultExceptionHandleManager hassetPatters() , setHandlers() method. Compare with Exception generation class using the pattern information registered as above, and if it is true, execute the handler registered in the handlers.
pathMatcherthat is used at pattern examination, usesAntPathMatcher .

It is the structure that can register in the patterns after making specific pattern group, define and register post-processing logic corresponding to this.

Handler Implements

First of all, understanding on class is required. While it was explained briefly earlier, it can be summarized as follows:
If Exception occurs, AOP pointcut “After-throwing” hangs and transfer of ExceptionTransfer class is executed.
transfermethod executes the run method of ExceptionHandlerManager. Following is the example of implementation and DefaultExceptionHandleManagercode.
(Requirement at the time of implementation) Upper class is AbsExceptionHandleManagerand interface is ExceptionHandlerService.
It is confirmed that implemented method is run(Exception exception).

DefaultExceptionHandleManager.java

```
public class DefaultExceptionHandleManager extends AbsExceptionHandleManager implements
ExceptionHandlerService {

	@Override
	publicboolean run(Exception exception) throws Exception {

		log.debug(" DefaultExceptionHandleManager.run() ");

		// If matching condition is false
		if (!enableMatcher())
			return false;

		for (String pattern : patterns) {
			log.debug("pattern = " + pattern + ", thisPackageName = " +
thisPackageName);
			log.debug("pm.match(pattern, thisPackageName) =" + pm.match(pattern,
thisPackageName));
			if (pm.match(pattern, thisPackageName)) {
				for (ExceptionHandler eh : handlers) {
					eh.occur(exception, getPackageName());
				}
				break;
			}
		}

		return true;
	}

}
```

Register Customizable Handler

Scenario: create CustomizableHandler class and execute CustomizableHandler at the exception of Helloworld class in the sample package.

First of all,make the CustomHandlerclass as follows.
ExceptionHandleManagerexecutes occur method.
Handler implementation absolutely(required) has ExceptionHandler Interface.

CustomizableHandler.java

```
public class CustomizableHandler implements ExceptionHandler {

	protected Log log = LogFactory.getLog(this.getClass());

	public void occur(Exception ex, StringpackageName) {

		log.debug(" CustomHandler run.....");
		try {
			log.debug(" Run CustomHandler   ...   ");
		} catch (Exception e) {
			e.printStackTrace();
		}
	}
}
```

Let’s register CustomizableHandler.
What should be noted here is to designate Helloworldclass in the sample package to the patterns.

```
	<bean id="exceptionTransfer"
class="egovframework.rte.fdl.cmmn.aspect.ExceptionTransfer">
		<property name="exceptionHandlerService">
			<list>
				<ref bean="customizableExceptionHandleManager" />
			</list>
		</property>
	</bean>

	<bean id="customizableExceptionHandleManager"

	class="egovframework.rte.fdl.cmmn.exception.manager.DefaultExceptionHandleManager">
		<property name="patterns">
			<list>
				<value>**sample.Helloworld</value>
			</list>
		</property>
		<property name="handlers">
			<list>
				<ref bean="customizableHandler1" />
				<ref bean="customizableHandler2" />
				<ref bean="customizableHandler3" />
			</list>
		</property>
	</bean>

	<bean id="customizableHandler1" class="sample.CustomizableHandler" />
	<bean id="customizableHandler2" class="sample.CustomizableHandler" />
	<bean id="customizableHandler3" class="sample.CustomizableHandler" />
```

This way, several Handlers can be registered.

leaveaTrace Setting and Description

Used at the time of Exception or to execute Trace post-processing logic if it is not Exception.
Basic structure of setting is same as the type of Exception post-processing. Setting file is context-common.xml.
It is set in the form of registering TraceHandler in DefaultTraceHandleManager.

Bean Setting

```
...
	<bean id="leaveaTrace" class="egovframework.rte.fdl.cmmn.trace.LeaveaTrace">
		<property name="traceHandlerServices">
			<list>
				<ref bean="traceHandlerService" />
			</list>
		</property>
	</bean>

	<bean id="traceHandlerService"
	class="egovframework.rte.fdl.cmmn.trace.manager.DefaultTraceHandleManager">
		<property name="patterns">
			<list>
				<value>*</value>
			</list>
		</property>
		<property name="handlers">
			<list>
				<ref bean="defaultTraceHandler" />
			</list>
```

```
                </property>
        </bean>

        <bean id="antPathMater" class="org.springframework.util.AntPathMatcher" />

        <bean id="defaultTraceHandler"
                class="egovframework.rte.fdl.cmmn.trace.handler.DefaultTraceHandler" />
...
```

Sample of TraceHandler Expansion Development

Implement Interface TraceHandler as shown below.

```
packageegovframework.rte.fdl.cmmn.trace.handler;

importorg.apache.commons.logging.Log;
importorg.apache.commons.logging.LogFactory;


public class DefaultTraceHandler implements TraceHandler {


public void todo(Class clazz, String message) {
        //Area to put the processing logic to perform...
        System.out.println(" log ==>DefaultTraceHandler run.....");
    }

}
```

Sample of Generation in leaveaTrace Code

Method of use can be recalled as shown below.
Execute Handler by transferring the message information using message key(message.trace.msg).

```
        publicCategoryVOselectCategory(CategoryVOvo) throws Exception {
                CategoryVOresultVO = categoryDAO.selectCategory(vo);
                try {
                  //ArithmeticException forcefully generated
                        inti = 1 / 0;
                } catch (ArithmeticExceptionathex) {
                //Run post-processing logic without generating Exception.
                        leaveaTrace("message.trace.msg");
                }

                returnresultVO;
        }
```

Reference

- exception-handling-framework-for-j2ee
- Effective Java (Joshua Bloch) : Chapter 8 Exception Processing