## Implementation Environment AOP Guideline

### Summary

The implementation environment of eGov uses XML-schema based AOP method and applied handling of exception and transaction. XML-schema based AOP method is easier to understand the configuration on horizontal references than the @AspectJ Annotation based methods.

### Description

### Handling of exceptions

The implementation environment can be processed in service column with an exception of DAO. The exception provided additionally in the execution environment is as following.

- EgovBizException: An exception commonly used in checked exception. When the developer wants to send specific message by throwing the specific error, processException() should be used.
- ExceptionTransfer: When an exception has occured in the ServiceImpl using the AOP function (in case of after-throwing), it is processed at trace() method. It is thrown by classifying EgovBizException or RuntimeException(ex.DataAccessException) internally. For the patterns declared by the DefaultExceptionHandleManager ExceptionTransfer internally, it works through the handler.

### Declaring aspect

The aspect class is defined as a bean within the Spring configuration file (resources/egovframework.spring/context-aspect.xml) for handling exceptions and declare pointcut and advice for the aspect.

```
<bean id="exceptionTransfer" class="egovframework.rte.fdl.cmmn.aspect.ExceptionTransfer">
...
</bean>

<aop:config>
	<aop:pointcut id="serviceMethod"
		expression="execution(* egovframework.rte.sample.service..*Impl.*(..))" />
	<aop:aspect ref="exceptionTransfer">
		<aop:after-throwing throwing="exception"
			pointcut-ref="serviceMethod" method="transfer" />
	</aop:aspect>
</aop:config>
...
</beans>
```

ExceptionTransfer plays a role of processing exceptions occured in executing methods of the class which the name ends with Impl within the egovframework.rte.sample.service package.

### Declaring advice

The ExceptionTransfer class declared as advice is included in the implementation environment source code. ExceptionTransfer class works to call the ExceptionHandler that is declared in the exception processing configuration file internally. Next is a part of the code in the ExceptionTransfer class.

```
public class ExceptionTransfer {
    ...
	public void transfer(JoinPoint thisJoinPoint, Exception exception) throws Exception {
		log.debug("execute ExceptionTransfer.transfer ");

		Class clazz = thisJoinPoint.getTarget().getClass();
		Locale locale = LocaleContextHolder.getLocale();
```

```
                // For BizException, the message is processed. Only the log is written.
                if (exception instanceof EgovBizException) {
                        log.debug("Exception case :: EgovBizException ");

                        EgovBizException be = (EgovBizException) exception;
                        getLog(clazz).error(be.getMessage(), be.getCause());

                        // Configuration of package and exception occured in the exception handler.
                        processHandling(clazz, exception, pm, exceptionHandlerServices, false);

                        throw be;

                } else if (exception instanceof RuntimeException) {
                        log.debug("RuntimeException case :: RuntimeException ");

                        RuntimeException be = (RuntimeException) exception;
                        getLog(clazz).error(be.getMessage(), be.getCause());

                        // Configuration of package and exception occured in the exception handler.
                        processHandling(clazz, exception, pm, exceptionHandlerServices, true);

                        if (be instanceof DataAccessException) {
                                log.debug("RuntimeException case :: DataAccessException ");
                                DataAccessException sqlEx = (DataAccessException) be;
                                // throw processException(clazz, "fail.data.sql",
                                // new String[] {
                                // Integer.toString(((SQLException) sqlEx
                                // .getCause()).getErrorCode()),
                                // ((SQLException) sqlEx.getCause())
                                // .getLocalizedMessage() }, sqlEx, locale);
                                throw sqlEx;
                        }

                        throw be;

                } else if (exception instanceof FdlException) {
                        log.debug("FdlException case :: FdlException ");

                        FdlException fe = (FdlException) exception;
                        getLog(clazz).error(fe.getMessage(), fe.getCause());

                        throw fe;

                } else {

                        log.debug("case :: Exception ");

                        getLog(clazz).error(exception.getMessage(), exception.getCause());

                        throw processException(clazz, "fail.common.msg", new String[] {},
exception, locale);

                }
        }
}
```

**Handling of transaction**

Refer to “resources/egovframework.spring/context-transaction.xml” file for the transaction configuration in the implementation environment. Next is a part of context-transaction.xml configuration file.

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
 ...
          http://www.springframework.org/schema/tx
         http://www.springframework.org/schema/tx/spring-tx-2.5.xsd
          http://www.springframework.org/schema/aop
         http://www.springframework.org/schema/aop/spring-aop-2.0.xsd">

         <!—Configure transaction manager.   -->
          <bean id="txManager"
class="org.springframework.jdbc.datasource.DataSourceTransactionManager">
            <property name="dataSource" ref="dataSource"/>
          </bean>

         <!-- Configure transaction advice. -->
          <tx:advice id="txAdvice" transaction-manager="txManager">
                   <tx:attributes>
                            <tx:method name="*" rollback-for="Exception"/>
                   </tx:attributes>
          </tx:advice>

         <!-- Configure transaction pointcut. --->
          <aop:config>
                   <aop:pointcut id="requiredTx"
                            expression="execution(* egovframework.rte.sample..impl.*Impl.*(..))"/>
                   <aop:advisor advice-ref="txAdvice"
                            pointcut-ref="requiredTx" />
          </aop:config>

</beans>
```

- txAdvice implements the transaction roll back in exception.
- requiredTx designates methods of any class from the sub impl package from the egovframework.rte.sample package as the pointcut.

**Reference**

- Exception Handling Service of Implementation Environment
- Transaction Service of Implementation Environment