Generalitat de Catalunya
Centre de Telecomunicacions
i Tecnologies de la Informació

Framework Corporatiu J2EE

Servei d'excepcions

Versió 1.4

Barcelona, 13 / abril / 2007

Històric de modificacions

Data	Autor	Comentaris	Versió
09/12/2005	Atos Origin, sae openTrends	Versió inicial del document	1.0
03/01/2006	Atos Origin, SAE	Versió 1.1 d'OpenFrame	1.1
13/02/2007	Atos Origin, SAE	Versió 1.2 d'OpenFrame	1.2
16/03/2007	Atos Origin, SAE	Versió 1.3 d'OpenFrame	1.3
13/04/2007	Atos Origin, SAE	Versió 1.4 d'OpenFrame	1.4

Llegenda de Marcadors

Índex

1. Introducció

1.1. Propósit

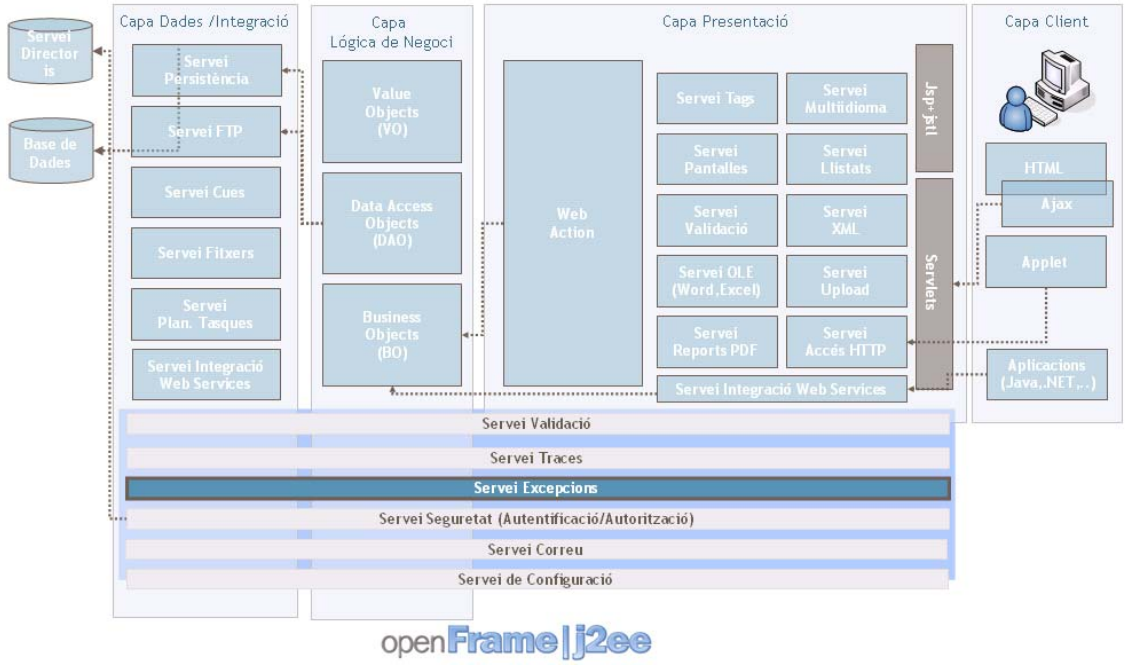
La gestió d'excepcions permet informar que s'ha produït un error al realitzar una petició. Aquest error podrà ser tractat adequadament i en cas necessari informar a l'usuari, llençar una traça, enviar un correu, etc.

La gestió correcta de les excepcions és molt important i crítica, però generalment la forma en la que es generen i getionen les excepcions en les aplicacions és un dels aspectes més ignorats en el disseny de les mateixes. L'ús apropiat de les excepcions fa que els nostres aplicatius siguin més robusts, més fàcils de desenvolupar i mantenir, més lliures d'errors i més fàcils de utilitzar. Per aquest motiu és important que donem el màxim de detall en les excepcions.

Per evitar l'ús innecessari de blocs 'try-catch' dins el codi dels nostres aplicatius openFrame proporciona un mecanisme d'intercepció, pel qual indicarem quines excepcions volem tractar i quins gestors les tractaran sense haver d'incorporar cap referència a cap classe de l'aplicació.

1.2. Context i Escenaris d'Ús

El Servei de Traces es troba dins dels serveis de Propósit General de openFrame.



El seu ús és imprescindible en openFrame, ja que tots els serveis utilitzen la gestió d'excepcions definida per aquest servei.

1.3. Versions i Dependències

En el present apartat es mostren quines són les versions i dependències necessàries per fer ús del Servei.

Dins la llista de dependències es mostren diferenciades:

- Dependències bàsiques. Llibreries necessàries per fer ús del servei
- Dependències adicionals. Aquestes dependències són necessàries per poder fer ús de característiques concretes del servei o per l'ús dels tests unitaris proporcionats amb el servei

1.3.1. Versions

No s'han produït canvis respecte la versió 1.3.

1.3.2. Dependències Bàsiques

Nom	Tipus	Versió	Descripció
aspectwerkz	jar	2.0	Utilització de Aspectwerkz per intercepció de les excepcions
aspectwerkz-core	jar	2.0	Ídem
aspectwerkz-extensions	jar	2.0	Ídem
dom4j	jar	1.6.1	http://dom4j.sourceforge.net
jrexx	jar	1.1.1	
openFrame-services-logging	jar	1.0	Utilitzar també les dependències del Servei de Logging
servletapi	jar	2.4	
trove	jar	1.0.2	
spring	jar	1.2.5	http://www.springframework.org

En cas de utilitzar un servei de openFrame és important que es facin servir també les dependències del servei en qüestió.

1.3.3. Dependències Adicionals

- Proves Unitàries del Servei

Nom	Tipus	Versió	Descripció
junit	jar	3.8.1	

1.4. A qui va dirigit

Aquest document va dirigit als següents perfils:

- Programador. Per conéixer l'ús del servei
- Arquitecte. Per conéixer quins són els components i la configuració del servei
- Administrador. Per conéixer cóm configurar el servei en cadascun dels entorns en cas de necessitat

1.5. Documents i Fonts de Referència

1.6. Glossari

AOP (Aspect Oriented Programming)

AOP permet definir funcionalitats comunes a diferents components d'un aplicatiu de forma única i amb un tractament general i localitzat, mitjançant la intercepció de la funcionalitat própia del mateix. Aquestes funcionalitats comunes es denominen *aspectes*.

Aspectwerkz
AspectWerkz és un framework AOP que ofereix gran simplicitat permetent la definició de aspectes, advices i introduccions amb classes POJOs. A més permet que els aspectes puguin ser definits mitjançant anotacions Java 5, doclets per JDK 1.3 i 1.4 o fins i tot mitjançant un fitxer XML simple.

El codi d'intercepció s'afegeix de forma automàtica al bytecode de la classe original mitjançant una compilació adicional. Aquest 'bytecode' modificat canvia la forma de cridar els nostres mètodes sense canviar la funcionalitat, i afegir així els conceptes importants en la programació orientada a aspectes: joinpoint, advice, etc...

Checked Exception

Tipus d'excepció que hereta de java.lang.Exception. El llenguatge Java obliga a que aquestes excepcions hagin de ser capturades mitjançant un bloc 'try-catch' o bé declarar al mètode que es torna a llençar l'excepció (finalment algú l'haurà de capturar) mitjançant la clàusula 'throws'.

Unchecked Exception

A diferència de les *checked exceptions,* poden ser ignorades i per tant no és necessari capturar-les ni declarar-les, encara que com veurem posteriorment, és una bona pràctica declarar-les en la signatura del mètode.

2. Descripció Detallada

2.1. Arquitectura i Components

openFrame ofereix una arquitectura d'excepcions totalment deslligada de qualsevol implementació. Únicament es basa en les característiques de les excepcions del llenguatge Java.

Els components podem classificar-los en:

- Components per representar diferents tipus d'excepcions
- Components per donar detall de les excepcions
- Components per a la gestió de les excepcions
- Components d'integració AOP (interns arquitectura)

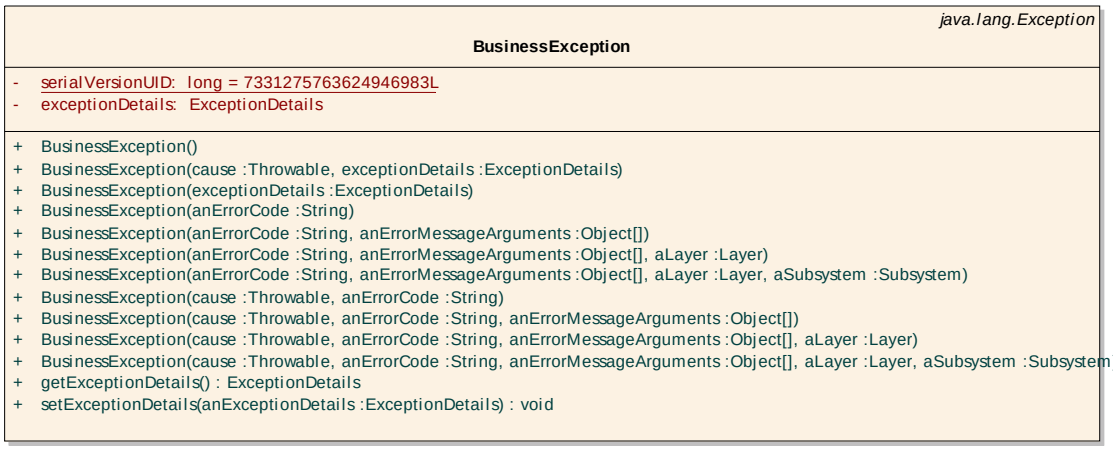
Per a més detall de cadascun dels components veure la documentació disponible al Javadoc.

2.1.1. Tipus d'Excepcions

El Servei d'Excepcions defineix 2 tipus d'excepcions principals que es basen en checked exceptions y en unchecked exceptions (veure glossari).

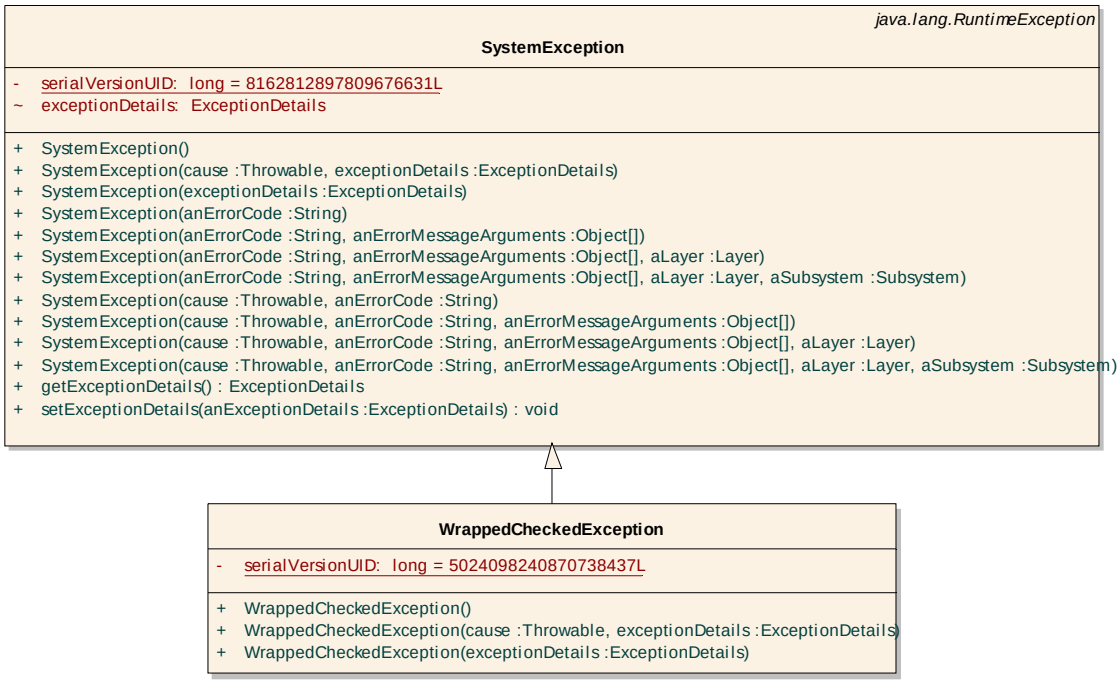
Component	Package	Descripció
BusinessException	net.opentrends.openframe.services.exceptions	Excepció que hereta de java.lang.Exception. Es tracta per tant d'una excepció de tipus checked. Permet afegir informació de detall de l'excepció mitjançant l'ús d'un objecte 'ExceptionDetails'
SystemException	net.opentrends.openframe.services.exceptions	Excepció que hereta de java.lang.RuntimeException. Es tracta per tant d'una excepció de tipus unchecked. Serveix de base per les excepcions definides per openFrame. Aquestes excepcions no han de ser capturades per la lógica de negoci. Permet afegir informació de detall de l'excepció mitjançant l'ús d'un objecte 'ExceptionDetails'
WrappedCheckedException	net.opentrends.openframe.services.exceptions	Excepció de tipus *unchecked* que serveix per encapsular una excepció de tipus *checked*

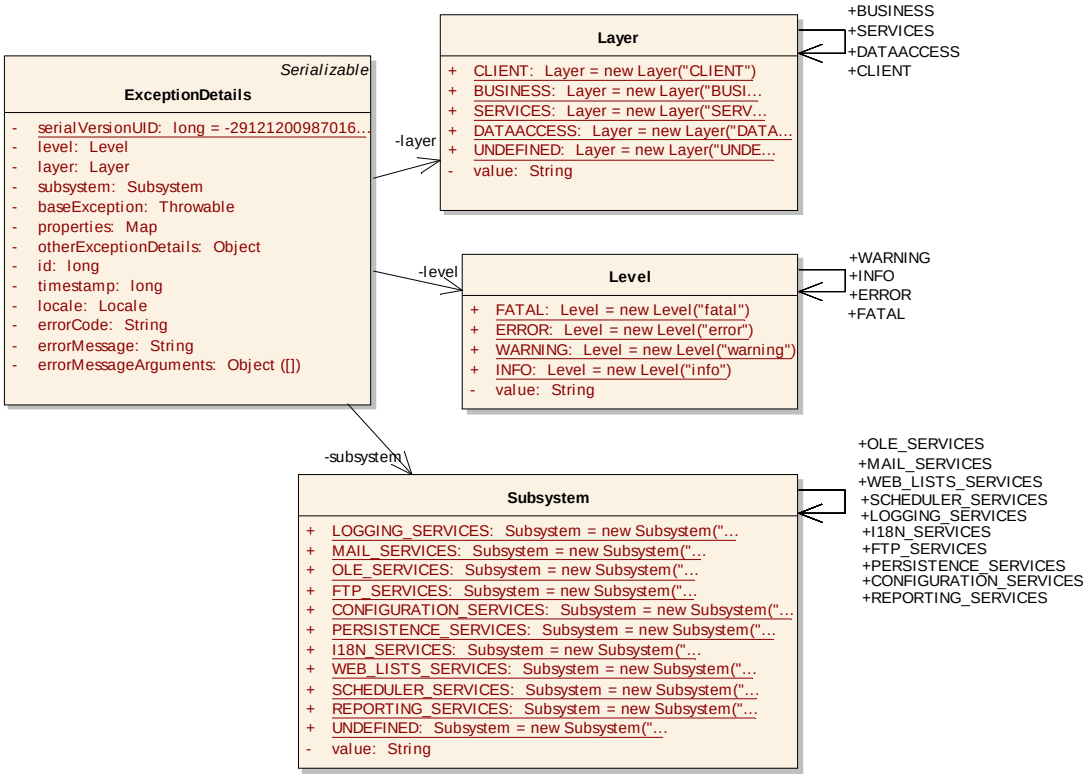
En el següent gràfic es pot veure de forma general la definició de cadascuna d'aquestes classes (per a més referència consultar la seva documentació al Javadoc del servei).



2.1.2. Informació de Detall de les Excepcions

En qualsevol de les excepcions anteriors es poden afegir detalls adicionals de les excepcions.





S'ofereixen els següents components:

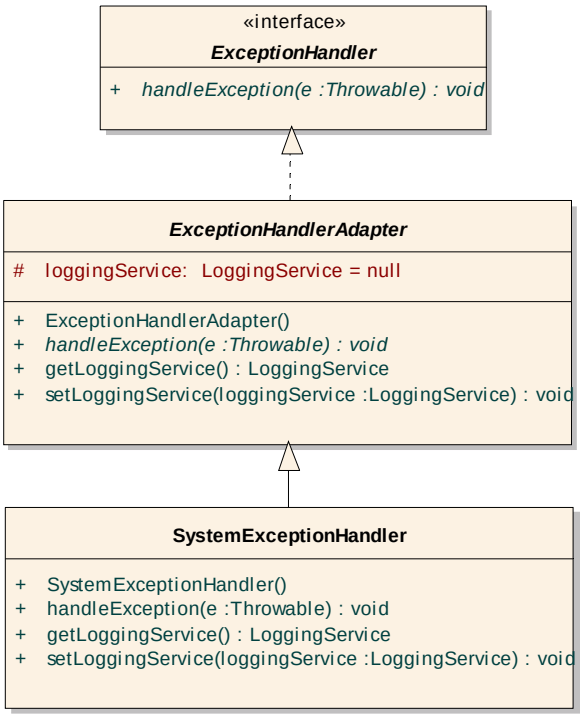
Component	Package	Descripció
ExceptionDetails	net.opentrends.openframe.services.exceptions	Objecte de detalls d'una excepció Permet especificar entre d'altres: • level. Nivell de l'excepció (veure classe Level a continuació) • layer. Capa en la que s'ha generat l'excepció (veure classe Layer a continuació) • subsystem. Subsistema en el que s'ha generat l'excepció • properties. Mapa de propietats específic per indicar més informació a l'excepció • otherExceptionDetails. Objecte per afegir detalls de l'excepció que no siguin un mapa (per exemple incorporar l'objecte de negoci complet) • id. Identificador de l'excepció • locale. Llenguatge de l'excepció

Component	Package	Descripció
		(obsolet) • errorCode. Codi de l'error multiidioma per generar el missatge a l'usuari • errorArguments. Arguments a incorporar al missatge obtingut segons el errorCode • errorMessage (obsolet)
Layer	net.opentrends.openframe.services.exceptions	Permet els següents valors: • BUSINESS • SERVICES • DATAACCESS • CLIENT
Level	net.opentrends.openframe.services.exceptions	Permet els següents valors: • INFO • WARNING • ERROR • FATAL
Subsystem	net.opentrends.openframe.services.exceptions	En l'actualitat permet especificar qualsevol valor dels serveis de openFrame. En cas de necessitat es poden definir nous valors per cada aplicació heretant aquesta classe.

2.1.3. Components per la Gestió de les Excepcions

Mitjançant l'ús de AOP permeten gestionar les excepcions per tal de realitzar procediments comuns (traces, ...).

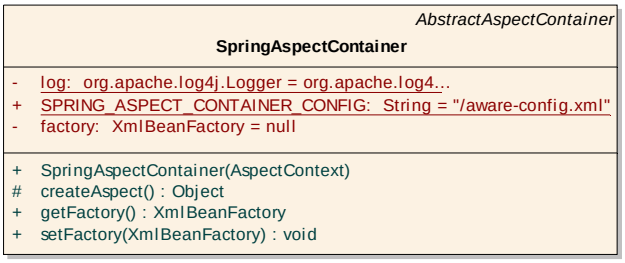
Component	Package	Descripció
ExceptionHandler	net.opentrends.openframe.services.exceptions.handlers	Interfície de gestió d'una excepció
ExceptionHandlerAspect	net.opentrends.openframe.services.exceptions.aop.aspectwerkz	Gestor AOP de diferents tipus d' excepcions. Permet definir una llista de classes excepcions a capturar i definir per cadascuna d'elles quin ExceptionHandler les tracta
SystemExceptionHandler	net.opentrends.openframe.services.exceptions.handlers	Exemple de ExceptionHandler que captura les excepcions de tipus 'SystemException' i realitza una traça mitjançant el Servei de Traces

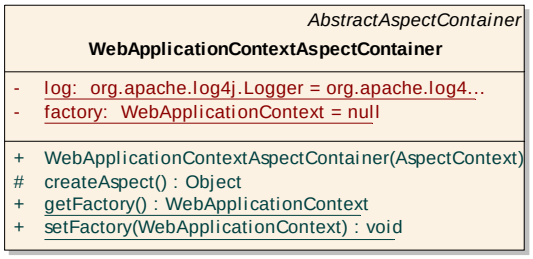


2.1.4. Components d'Integració AspectWerkz i Spring

Aquests components permeten la integració entre AspectWerkz i Spring, de forma que poguem definir de forma externa la injecció d'objectes als nostres gestors d'excepcions (per exemple per tal que aquests puguin fer ús d'altres serveis de openFrame).

Component	Package	Descripció
SpringAspectContainer	net.opentrends.openframe.services.exceptions.aop.aspectwerkz.container	Contenidor de aspectes que usa Spring per gestionar la seva instanciació i configuració.
WebApplicationContextAspectContainer	net.opentrends.openframe.services.exceptions.aop.aspectwerkz.container	Implementació específica per a la integració amb aplicacions Web





2.2. Instal·lació i Configuració

2.2.1. Instal·lació

La instal·lació del servei requereix de la utilització de la llibreria 'openFrame-services-exceptions' i les dependències indicades a l'apartat 'Introducció-Versions i Dependències'.

2.2.2. Configuració

La configuració del Servei d'Excepcions implica 3 pasos:

1) Definir els gestors d'excepcions
2) Definir el mapeig entre tipus d'excepcions i els gestors que les tractaran
3) Definir el mecanisme automàtic d'intercepció (per integrar les 2 definicions anteriors)

Definició dels gestors d'Excepcions

```
<bean id="XXXHandler"
      ...>
```

Path desenvolupament: main/resources/spring/openFrame-services-exceptions.xml
Path deployment web: WEB-INF/classes/spring/openFrame-services-exceptions.xml

Per cada gestor d'excepcions que volem crear (es pot crear un únic per una classe pare d'excepció, una per cada subtipus d'excepció, ..) crearem un bean de configuració amb la següent informació:

Atributs:

Atributs	Requerit	Descripció
id	Sí	Identificador del gestor
class	Sí	Classe gestora que implementa la interfície 'ExceptionHandler'

Podem afegir propietats a cadascun dels gestors (per exemple el seu Servei de Traces) dins la definició del gestor.

Exemple:

```
<beanid="systemExceptionHandler"
class="net.opentrends.openframe.services.exceptions.handlers.SystemExceptionHandle
r">
   <property name="loggingService">
          <ref local="loggingService"/>
   </property>
</bean>
```

Definició del mapeig de tipus d'excepcions i gestors

```
<bean
id="net.opentrends.openframe.services.e
xceptions.aop.aspectwerkz.ExceptionHand
lerAspect"...>
```

Path desenvolupament: main/resources/spring/openFrame-services-exceptions.xml
Path deployment web: WEB-INF/classes/spring/openFrame-services-exceptions.xml

Atributs:

Atributs	Requerit	Descripció
id	Sí	Usar 'net.opentrends.openframe.services.exceptions.aop.aspectwerkz.ExceptionHandlerAspect'
class	Sí	Usar' 'net.opentrends.openframe.services.exceptions.aop.aspectwerkz.ExceptionHandlerAspect'
singleton	Sí	Especificar 'true'. D'aquesta forma es crearà una instància per a cada petició

Propietats:

Propietat	Requerit	Descripció
exceptionHandlers	Sí	Mapa de clau-valor on la clau ha de ser la classe excepció full qualified i el valor la referència al gestor.

Exemple:

```
<bean
id="net.opentrends.openframe.services.exceptions.aop.aspectwerkz.ExceptionHandlerA
spect"
```

```
class="net.opentrends.openframe.services.exceptions.aop.aspectwerkz.ExceptionHandl
erAspect"
        singleton="false">
   <property name="exceptionHandlers">
     <map>
       <entry>

<key><value>net.opentrends.openframe.services.exceptions.SystemException</value></
key>
              <ref bean="systemExceptionHandler"/>

<key><value>net.opentrends.openframe.services.mail.exception.MailServiceException<
/value></key>
              <ref bean="unAltreExceptionHandler"/>
           ...
           <!-- tantes entrades com gestors -->
       </entry>
     </map>
   </property>
</bean>
```

A l'exemple es mostra cóm s'ha configurat el gestor 'systemExceptionHandler' per l'excepció de tipus 'net.opentrends.openframe.services.exceptions.SystemException'.

Definició del mecanisme d'intercepció automàtica

```
<aspectwerkz>
    <system id="ExceptionHandlingAspect">
       <package
name="net.opentrends.openframe.services.exceptions.aop.aspectwerkz">
              <aspect class="ExceptionHandlerAspect"...
```

Path desenvolupament: main/resources/aop.xml
Path deployment web: WEB-INF/classes/aop.xml

Aquest fitxer defineix: les característiques de l'AOP AspectWerkz per definir l'aspecte. Per a més informació es recomana consultar 'http://aspectwerkz.codehaus.org/'.

Definirem els següents elements:

a) Aspecte
b) Advice. En quins punts s'afegeix l'aspecte

- Aspecte

```
<aspect class="ExceptionHandlerAspect"...
```

Definir els següents atributs:

Atributs	Requerit	Descripció
class	Sí	Classe que defineix l'aspecte. Utilitzar: 'ExceptionHandlerAspect'
container	Sí	Contenidor d'integració Usar 'net.opentrends.openframe.services.exceptions.aop.aspectwerkz.container.SpringAspectContainer'

```
<!-- ExceptionHandlerAspect: classe que defineix la funcionalitat comuna
implementada per aquest aspecte;
     El atribut container indica la classe que proporciona les instàncies del
gestor -->
<aspect                                                class="ExceptionHandlerAspect"
container="net.opentrends.openframe.services.exceptions.aop.aspectwerkz.container.
SpringAspectContainer">

</aspect>
```

- Advice. Definició de la intercepció

```
<advice type="" ...
```

Definir els següents atributs:

Atributs	Requerit	Descripció
type	Sí	Usar 'after throwing(exception)' per indicar que volem realitzar la intercepció en qualsevol punt de l'aplicació en el que es llenci una excepció
bind-to	Sí	Usar 'execution(* *.*(..))' per indicar que interceptarem el llençament d'excepcions que es faci des de qualsevol mètode
name	Sí	Usar 'handleException(java.lang.Throwable exception)' per indicar que es cridi a aquest mètode de l'aspecte

```
...
<!-- Els asteriscs indiquen qualsevol paquet, qualsevol classe i qualsevol mètode;
     El tipus es “després de produïr-se” qualsevol tipus d'exceptió  -->
<advice    type="after    throwing(exception)"    bind-to="execution(*    *.*(..))"
name="handleException(java.lang.Throwable exception)"/>
...
```

A la següent figura es presenta un exemple d'aquest fitxer:

```
<aspectwerkz>
    <system id="ExceptionHandlingAspect">
        <package
name="net.opentrends.openframe.services.exceptions.aop.aspectwerkz">
            <aspect           class="ExceptionHandlerAspect"           container="n
   et.opentrends.openframe.services.exceptions.aop.aspectwerkz.container.SpringAsp
ectContainer">
                    <advice type="after throwing(exception)"
                                                bind-to="execution(*   *.*(..))"

name="handleException(java.lang.Throwable exception)"/>
            </aspect>
        </package>
    </system>
</aspectwerkz>
```

2.3. Utilització del Servei

2.3.1. Creació d'Excepcions

La creació d'excepcions és un aspecte de disseny molt important. Quants tipus d'excepcions generar?

La resposta no és única i si bé existeixen diferents possibilitats la recomanació és:

- Definir una excepció base per les excepcions d'un subsistema en concret. Per exemple, podem definir un tipus d'excepció per un módul concret de l'aplicació
- Definir una excepció específica per cada classe de negoci. D'aquesta forma, podrem determinar de forma més senzilla des de quin punt s'ha llençat una excepció i realitzar tractaments diferenciats

A més, seguirem els següents patrons d'ús:

- Si l'excepció base o específica és de negoci, aquesta haurà d'heretar de 'BusinessException'
- Si l'excepció base o específica és d'aspectes de serveis o infrastructura (DAOs, serveis d'integració, etc.), aquesta haurà d'heretar de 'SystemException'

2.3.2. Declaració d'Excepcions llençades als mètodes

Segons si l'excepció és de subtipus 'BusinessException' o de subtipus 'SystemException' es seguiran diferents estratègies:

1) Excepció de subtipus 'BusinessException'

   En cas de que el mètode llenci una excepció de subtipus 'BusinessException' declarar '**throws Exception**'.

Com hem comentat, BusinessException és una excepció de tipus 'checked', el que implica que segons el mecanisme tradicional de Java qualsevol classe que rebi l'excepció hauria de control·lar-la mitjançant 'try-catch' o declarar-la de nou als seus mètodes (pràctica totalment prohibida).
En els mecanismes tradicionals de Java, capturar una excepció de tipus checked implica afegir un control poc útil d'encapsulació de l'excepció rebuda en una nova excepció. Aquest control és degut a la pràctica poc correcta d'haver de gestionar obligatóriament les excepcions rebudes.

Segons el comentat, evitem el tractament innecessari declarant l'excepció amb 'throws Exception'. Aquesta incorporació és necessària per a que el compilador no es queixi de que no s'ha declarat l'excepció.

Exemple:

```
public interface AccountBO {
   public void save(Account account) throws Exception;
   public void saveOrUpdate(Account account) throws Exception;
    public void update(Account account) throws Exception;
   public void delete(Account account) throws Exception ;
   public Account load(Account account);
}
```

2) Excepció de subtipus 'SystemException'

Si el mètode llença una excepció de tipus pare 'SystemException' hem de mantenir el mecanisme de Java, ja que aquestes excepcions, per ser de tipus 'unchecked' no requereixen del control al nivell superior.

```
public void send(String from, String subject, String aMessage, boolean isHtml,
String to) throws MailServiceException;
```

2.3.3. Generació d'Excepcions

La generació d'excepcions es realitza mitjançant la clàusula 'throws' de Java.

En la generació de les excepcions afegirem el màxim de detall possible mitjançant l'ús de la classe 'ExceptionDetails' (consultar l'apartat 'Arquitectura i Components' per a més informació). Aquesta classe permet que es pugui informar entre d'altres del nivell (`Level`), de la capa (`Layer`) i del subsistema (`Subsystem`) en que s'ha produit l'excepció que llencen l'excepció. A més, es pot informat un codi d'error (que tindrà la seva descripció en un fitxer de propietats) i unes propietats adicionals que serveixen per depurar l'error.

El patró recomanat per llençar una excepció és el mostrat a continuació:

```
ExceptionDetails exDetails = new ExceptionDetails(...)
exDetails.setProperties(...);
...
throw new XXXException(exDetails);
```

2.3.4. Encapsulació d'Excepcions rebudes de terceres parts

Si volem integrar tercers components externs a openFrame és convenient que s'encapsulin les excepcions rebudes en el mecanisme d'excepcions de openFrame. El patró de tractament pot ser com el mostrat a continuació:

```
try {
   ...
    cridaServeiExtern();
} catch(ServeiExternException ex){
  ExceptionDetails exDetails = new
ExceptionDetails("codiExcepcio",...,Layer.XXXX,Subsystem.XXXX);
   exDetails.setProperties(mailProperties);
   throw new XXXOpenFrameException(ex,exDetails);
```

Cóm veiem, capturem l'excepció del component de tercers i li afegim la informació necessària. Per últim llencem l'excepció incorporant com a segon paràmetre els detalls que hem indicat i en el primer paràmetre l'excepció arrel del tercer component.

Exemple:

```
package net.opentrends.openframe.services.mail.impl;
...
public class SpringMailServiceImpl implements MailService {
...
    public void send(String from, String subject, String aMessage,
boolean isHtml, Map recipients, List attachments) throws MailException {
     ...
       try {
         ...
           message = this.mailSender.createMimeMessage();
         ...
           helper.setFrom(from);
           helper.setSubject(subject);
           helper.setText(aMessage,isHtml);
         ...
           this.mailSender.send(message);
       }
       catch(MessagingException ex){
```

```
            ExceptionDetails          exDetails          =          new
ExceptionDetails("openFrame.services.mail.error_preparing_addresses",nul
l,Layer.SERVICES,Subsystem.MAIL_SERVICES);
            exDetails.setProperties(mailProperties);
            throw new MailServiceException(ex,exDetails);
        }
        catch(Exception ex){
            ExceptionDetails          exDetails          =          new
ExceptionDetails("openFrame.services.mail.error_sending_mail",null,Layer
.SERVICES,Subsystem.MAIL_SERVICES);
            exDetails.setProperties(mailProperties);
            throw new MailServiceException(ex,exDetails);
        }
    }
...
}
```

2.3.5. Intercepció i tractament de les Excepcions

Mitjançant l'ús de la programació orientada a aspectes (AOP) i les classes proporcionades pel servei d'excepcions s'obté una solució integral a la gestió intel.ligent d'excepcions.

La intercepció de les excepcions es realitza mitjançant el mecanisme definit als fitxers de configuració (veure apartat 'Configuració i Instal·lació). Per definir un gestor s'han de seguir els següents pasos:

1) Extendre de la classe 'ExceptionHandlerAdapter'
2) Definir el mètode 'public void handleException(Throwable e) throws Throwable'
3) Dins aquest mètode realitzar la implementació del tractament
4) Si l'excepció rebuda és de negoci encapsular-la en una excepció de tipus 'WrappedCheckedException'. D'aquesta forma capes superiors no hauran de definir la declaració de l'excepció ni capturar-la.

```
public class XXXExceptionHandler extends ExceptionHandlerAdapter {
...
    public void handleException(Throwable e) throws Throwable {
         ...
         WrappedCheckedException          wExc          =          new
WrappedCheckedException(e,...);
         throws wExc;
        }
    }
...
}
```

Aquest gestor pot definir objectes atribut per usar altres serveis de openFrame com per exemple per:

- Generar mitjançant el Servei de Traces un missatge de tipus informatiu de l'excepció.
- Generar mitjançant el Servei de Traces un missatge de diferent nivell segons el nivell de l'excepció rebuda (es pot obtenir mitjançant l'atribut 'level' de l'objecte 'ExceptionDetails').
- Integració amb serveis externs que necessiten del coneixement de determinades excepcions
- Altres necessitats

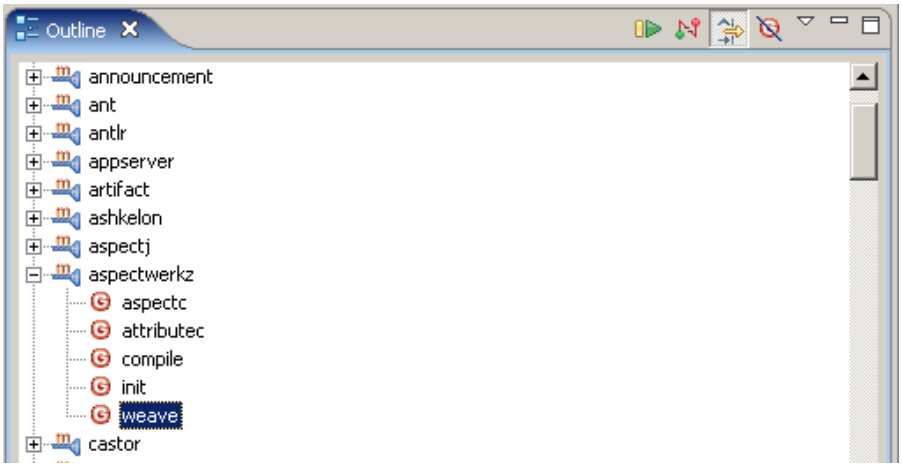
Exemple:

```
package net.opentrends.openframe.services.exceptions.handlers;

...

public class SystemExceptionHandler extends ExceptionHandlerAdapter {
...
  public void handleException(Throwable e) throws Throwable {
     Log log = this.loggingService.getLog(this.getClass());
  log.debug("Exception   received:   SystemExceptionHandler,   Exception
catched:" + e);
    throw e;
    }

...
}
```

2.4. Eines de Suport

2.4.1. Generació del bytecode d'intercepció amb AspectWerkz

Per tal de que funcioni en execució el mecanisme d'intercepció cal un procés adicional de compilació a les classes. Aquest procés adicional es pot executar en Maven mitjançant el goal 'aspectwerkz:weave' tal i com es mostra en la figura següent:



Aquest procés fa servir les següents variables de configuració que hauran d'estar en el nostre fitxer 'project.properties':

```
<!-- mode de màxima informació en el procés de modificació de bytecode -->
maven.aspectwerkz.verbose=true
<!-- compilació basada en anotacions -->
maven.aspectwerkz.mode=attribdef
maven.aspectwerkz.definition.validate=true
<!-- fitxer de propietats del servei (veure Configuració) o de definició del
nostre aspecte -->
maven.aspectwerkz.definition.file.src=${basedir}/src/main/resources/aop.xml
<!-- directori a on es copien totes les classes que s'han de passar pel nostre
procés addicional de modificació de bytecode -->
maven.aspectwerkz.build.dest=${basedir}/target/aspectwerkz/classes
<!-- directori a on s'ubicaran les classes modificades. És important que aquest
directori sigui el directori WEB-INF/classes de l'aplicació desplegada al servidor
-->
maven.aspectwerkz.weave.build.dir=${basedir}/.deployables/${opentrends.war.wtp.mod
ule}/WEB-INF/classes
```

Per evitar la necessitat manual de realitzar aquest goal, podem incorporar el següent codi 'Maven' en la creació de l'aplicatiu Web o de la llibreria:

```
<u:available file="${maven.aspectwerkz.weave.build.dir}/aop.xml">
       <attainGoal name="aspectwerkz:weave"/>
          <ant:echo>Moving  file  ${maven.aspectwerkz.weave.build.dir}/aop.xml  to
${maven.aspectwerkz.weave.build.dir}/../aop.xml</ant:echo>
          <ant:move                                          failonerror="false"
file="${maven.aspectwerkz.weave.build.dir}/aop.xml"
tofile="${maven.aspectwerkz.weave.build.dir}/../aop.xml"/>
</u:available>
```

2.5. Integració amb Altres Serveis

2.5.1. Integració amb el Servei de Presentació (Arquitectura Base)

Aplicar les estratègies definides prèviament implica que l'excepció és llençada a les capes superiors. Aquestes poden decidir si tractar-la o no tractar-la. Si la petició ha estat realitzada per un usuari en mode on-line és convenient que li informem de l'excepció que s'ha produit.

Tota excepció produïda per l'aplicació arriba finalment a la classe principal de l'arquitectura de openFrame 'ExtendedDelegatingTilesRequestProcessor' (veure el document 'Serveis de Presentació' per a més informació).

Aquesta classe captura l'excepció rebuda i realitza els següents pasos:

1) Extreu la següent informació de detall de l'excepció ('ExceptionDetails'):

   - Codi d'error
   - Arguments per construir un missatge a partir del codi d'error
   - Propietats adicionals de l'excepció.
   - Nivel de l'error
   - 'Layer' o capa d'on prové
   - Subsistema d'on prové

2) Construeix varis objectes de tipus 'ActionMessage' de Struts a partir d'aquesta informació que afegeix a l'object 'ActionMessages'.

   - El primer missatge conté el missatge d'error (construit mitjançant el Servei de Multiidioma segons el codi d'error de l'excepció i els arguments).

```
message = new ActionMessage(exDetails.getErrorCode(),
                  exDetails.getArguments());
```

     Afegeix aquest missatge a l'objecte 'ActionMessages' amb clau 'org.apache.struts.action.ERROR'.

   - El segon missatge es construeix amb els detalls de l'excepció. Si l'excepció rebuda és de tipus 'SystemException' la clau del missatge és ' 'net.opentrends.openframe.services.exceptions.Layer.error', mentre que si és de tipus 'WrappedCheckedException' el deixa a ' net.opentrends.openframe.services.exceptions.Layer.warning'.

3) L'objecte 'ActionMessages' és deixat a l'atribut del request 'Globals.ERROR_KEY' per integració amb Struts.

```
request.setAttribute(Globals.ERROR_KEY, messages);
```

Els errors que s'insereixen fan ús del Servei Multidioma. És necessari afegir al fitxer de multiidioma les següents entrades:

```
net.opentrends.openframe.services.exceptions.WrappedCheckedException=<br>Error
type={0}<br>Localized                              message={1}<br>Properties={2}<br>
Layer={3}<br>Subsystem={4}<br>Error code={5}<br>Arguments={6}
net.opentrends.openframe.services.exceptions.SystemException=<br>Error
type={0}<br>Localized                              message={1}<br>Properties={2}<br>
Layer={3}<br>Subsystem={4}<br>Error code={5}<br>Arguments={6}
```

4) Finalment, consulta a la configuració de l'acció d'Struts si existeix un 'ActionForward' anomenat **'error'**. Si es troba, es redirigeix la petició cap aquest 'forward' (típicament per mostrar la mateixa pantalla d'on venim amb els missatges d'error). Si no es troba, l'excepció puja i es delega la possibilitat del tractament a algun gestor d'excepcions d'Struts (típicament definint dintre del tag <global-exceptions/> l'excepció que volem manegar i el gestor corresponent).

Pàgina per presentar els errors

Podem representar els errors en una pàgina JSP fent ús del tag 'messages' de Struts.

A continuació es mostra un exemple de la seva utilització, on es mostren els missatges en colors diferents segons la severitat de l'error:

```
<html:messages id="msg" property="org.apache.struts.action.ERROR">
   <div class="error" style="margin-right: 10px; margin-bottom: 3px; margin-top:
3px">
          <img src="images/iconWarning.gif" class="icon" alt="Warning" />
          <bean:write     name="msg"     filter="false"/>       [     <a
href="javascript:showEL('errorDetails')"><b><bean:message
key="jsp.includes.see_error_details"/></b></a>       ]   [       <a
href="javascript:showEL('stackTrace')"><b><bean:message
key="jsp.includes.see_stack_trace"/></b></a>]<br>
          <html:messages                                                  id="msg"
property="net.opentrends.openframe.services.exceptions.Level.WARNING">
          <div       id="errorDetails"       style="display:       none"><br><font
color="#ff0000"><bean:write name="msg" filter="false"/></font></div>
          </html:messages>
          <html:messages                                                  id="msg"
property="net.opentrends.openframe.services.exceptions.Level.ERROR">
          <div        id="errorDetails"        style="display:        none"><font
color="#00cc00"><bean:write name="msg" filter="false"/></font></div>
          </html:messages>
          <html:messages                                                  id="msg"
property="net.opentrends.openframe.services.exceptions.STACKTRACE">
          <div    id="stackTrace"    style="display:    none"><br><pre><bean:write
name="msg" filter="false"/></pre></div>
          </html:messages>
   </div> 
```

```
</html:messages>
```

2.6. Preguntes Freqüènts

3. Exemples

4. Annexos