



Guide de Migration

LISE 2.x vers LISE 3.x



Version 1.5 du 30/01/2009

Etat : Rédaction



SUIVI DES MODIFICATIONS

Version	Rédaction	Description	Vérification	Date
1.0	G.PICAVET C.ROCHETEAU	Première version		26/06/08
1.4	M.Fraudeau	Mise à jour		08/07/08
1.5	P.BUGAN	Mise à jour LISE 3.0.0 rc2		11/11/08
Document validé dans sa version xxx				

LISTE DE DIFFUSION

Organisation	Nom	Info	Commentaire	Validation
		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>



SOMMAIRE

SUIVI DES MODIFICATIONS	2
LISTE DE DIFFUSION	2
SOMMAIRE	3
1 MOTIVATIONS	4
2 COMPATIBILITE	5
3 PROCEDURE DE MIGRATION.....	6
4 CONSERVATION DE FOP 0.2.....	11

DOCUMENTS DE REFERENCE

Version	Titre



1 MOTIVATIONS

- Fournir une procédure de migration d'un projet Acube LISE v2.x vers LISE v3.x, partiellement automatisée.
- Préciser les points de contrôle pour vérifier la compatibilité ascendante.



2 COMPATIBILITE

- ➔ compatibilité des classes Action dérivant de BaseAction
- ➔ compatibilité des fichiers xslt.
- ➔ réutilisation des fichiers variables_xx.xml
- ➔ Règles d'autorisation définies dans variables_xx.xml
- ➔ reprise du mapping url/action, et action/xslt
- ➔ Génération des pdf avec FOP (réutilisation de userconfig.xml)

Remarque : la procédure de migration suppose que l'application serveur à migrer respecte les normes Acube. En particulier l'arborescence et les fichiers présents dans le gabarit (présence de ErrorVO.xsl, SuccesVO.xsl, ...)



3 PROCEDURE DE MIGRATION

1. Vérifier la compilation et le bon fonctionnement de l'application à migrer
2. Supprimer les jars suivants de l'application à migrer :
 - struts-1.1.jar
 - struts-cx-0.9.5.jar
 - fop-0.20.5.jar
 - jdom.jar (0.9b)
 - fwacubej2ee-X.Y.jar (Framework Acube pour Struts 1)
3. Ajouter les jars suivants à l'application à migrer :
 - Nouvelles librairies du Framework Acube
4. Après recompilation, les classes référençant directement les API Struts 1, Struts CX, ou Servlet-API ne compilent plus.
 - ➔ Les classes d'action « Deconnecter » et « RedirectionIndex » sont intégrées dans le nouveau framework et peuvent donc être supprimées du projet. Par défaut ces actions sont associées aux URL « flux/protected/frameset/deconnecter » et « flux/protected/frameset/index ». mais pourront être redéfinies dans struts.xml si besoin (cf. plus loin).

```
<!-- action Redirection -->
<action name="index" class="acube.projet.action.frameset.RedirectionIndex">
...
</action>
```

```
<!-- action Deconnexion-->
<action name="deconnecter" class="acube.projet.action.Deconnecter">
...
</action>
```

- ➔ La classe `acube.projet.rules.Authentification` peut être supprimée. Elle est remplacée par le `SecurityInterceptor` du Framework (activé par défaut)
- ➔ Remplacer les imports de `StrutsCXConstants` par

```
import acube.framework.webcomp.technical.StrutsCXConstants;
```

- ➔ Pour les classes dérivant directement de la classe Action de Struts : dériver `ActionSupport`.
 - L'objet request peut être obtenu en implémentant `ServletRequestAware`. Struts 2 mappe automatiquement les paramètres de requêtes sur des propriétés de l'action. Pour cela, la propriété doit être du même nom que le paramètre, disposer de getter et setter, et avoir un type java compatible avec la valeur.
 - L'objet response peut être obtenu en implémentant l'interface `ServletResponseAware`
 - Le contexte de servlet peut être obtenu en implémentant l'interface `ApplicationAware` (ou `ServletContextAware`)



- ➔ Pour les classes dérivant de `baseAction` mais qui surcharge la méthode `execute()` :
 - Ce qui était fait dans le `execute()` doit être adapté et mis dans le `getBeansActions()` ;
- ➔ Pour l'upload de fichier, on adapte l'action façon `struts2` :
 - On se base sur `struts2`
- ➔ Pour les actions étendent `BaseActionFile`,
 - il faut surcharger la méthode `executeAtEnd()` pour pouvoir configurer la sortie correctement

5. copier le script « Migration-struts2.bat » à la racine du projet et l'exécuter.

Rajouter les librairies :

- contenues de l'archive `livrable-migrationtoollib.zip` dans votre répertoire `Serveur/WEB-INF/lib`.
- contenues de l'archive `livrable-Outillage_migrationTool_3.0.0-rc1.zip` dans votre répertoire `Serveur/WEB-INF/lib`.
-

Si tout se passe bien, un fichier `struts.xml` est créé dans `Serveur/src`.

Attention l'outil considère que les classes compilées se trouvent dans le répertoire `Serveur\WEB-INF\classes` du projet, si jamais votre configuration est différente, éditez le fichier `Migration-struts2.bat`.

Il faut également ajouter le `struts.xml` dans le `WEB-INF/classes` lors de la construction du livrable :

Dans le fichier `Server/build.xml` :

```
<!-- ===== CREATION de l'archive WAR de l'application ===== -->
<target name="createWAR" description="CREATION de l'archive WAR de l'application">
<echo message="*****" />
<echo message="CREATION DU FICHIER war" />
<echo message="*****" />
<war destfile="${rep.livraison}/${build.contexte.prod}.war" webxml="WEB-INF/web.xml">
<metainf dir="META-INF" />
<lib dir="${projet.lib}" />
<classes dir="${rep.livraison}/${projet.classes}" />
<zipfileset dir="WEB-INF" excludes="web.xml" includes="*.xml" prefix="WEB-INF" />
<zipfileset dir="WEB-INF" includes="*.dtd" prefix="WEB-INF" />
<zipfileset dir="WEB-INF/xml" excludes="CVS/*" prefix="WEB-INF/xml" />
<zipfileset dir="WEB-INF/xsl" excludes="CVS/*" prefix="WEB-INF/xsl" />
<zipfileset dir="authentification" excludes="CVS/*" prefix="authentification" />
<zipfileset dir="${rep.livraison}/${projet.src}" includes="*.properties"
prefix="WEB-INF/classes" />
<zipfileset dir="${rep.livraison}/${projet.src}" includes="*.xml" prefix="WEB-INF/classes" />
</war>
</target>
```



```
<zipfileset dir="flux" excludes="CVS/*" prefix="flux" />
<zipfileset dir="flux/protected" excludes="CVS/*" prefix="flux/protected" />
</war>
<delete dir="${rep.livraison}/WEB-INF" />
<delete dir="${rep.livraison}/${projet.src}" />
</target>
```

On pourra ensuite supprimer :

WEB-INF/struts-config_11.dtd

WEB-INF/struts-config_11.xml

WEB-INF/strutsconfig_095.dtd

WEB-INF/strutsconfig.xml

WEB-INF/strutsconfig-log4j-config.xml

Enfin supprimer les lib contenues dans l'archive livrable-migrationtoolib.zip dans votre répertoire Serveur/WEB-INF/lib précédemment ajouté pour lancer les Migration-struts2.bat.

6. Dans WEB-INF/web.xml, Supprimer les déclarations & mapping des ActionServlet et StrutsCXServlet :

```
<servlet>
    <servlet-name>action</servlet-name>
    <servlet-class>org.apache.struts.action.ActionServlet</servlet-class>
    <init-param>
        <param-name>config</param-name>
        <param-value>/WEB-INF/struts-config.xml</param-value>
    </init-param>
    <init-param>
        <param-name>debug</param-name>
        <param-value>2</param-value>
    </init-param>
    <load-on-startup>2</load-on-startup>
</servlet>
<servlet>
    <servlet-name>StrutsCXServlet</servlet-name>
    <servlet-
class>com.cappuccinonet.strutscx.xslt.StrutsCXServlet</servlet-class>
    <init-param>
        <param-name>debug</param-name>
        <param-value>>false</param-value>
    </init-param>
    <load-on-startup>3</load-on-startup>
</servlet>
<!-- servlet mappings -->
<servlet-mapping>
    <servlet-name>action</servlet-name>
    <url-pattern>*.xml</url-pattern>
</servlet-mapping>
<servlet-mapping>
    <servlet-name>action</servlet-name>
    <url-pattern>*.html</url-pattern>
</servlet-mapping>
<servlet-mapping>
    <servlet-name>action</servlet-name>
```




```
<url-pattern>*.csv</url-pattern>
</servlet-mapping>
<servlet-mapping>

    <servlet-name>action</servlet-name>
    <url-pattern>*.pdf</url-pattern>
</servlet-mapping>
<servlet-mapping>
    <servlet-name>action</servlet-name>
    <url-pattern>*.xls</url-pattern>
</servlet-mapping>
<servlet-mapping>
    <servlet-name>StrutsCXServlet</servlet-name>
    <url-pattern>/StrutsCXServlet</url-pattern>
</servlet-mapping>
```

Remplacer par les déclarations suivantes :

```
<filter>
    <filter-name>struts2</filter-name>
    <filter-class>
        org.apache.struts2.dispatcher.FilterDispatcher
    </filter-class>
</filter>

<filter-mapping>
    <filter-name>struts2</filter-name>
    <url-pattern>/flux/*</url-pattern>
</filter-mapping>

<context-param>
    <param-name>VariablesXmlLoader-config</param-name>
    <param-value>/WEB-INF/xml/variables_fr.xml</param-value>
</context-param>
<!-- Configuration de FOP si il existe -->
<context-param>
    <param-name>FopInitializer-config</param-name>
    <param-value>/WEB-INF/userconfig.xml</param-value>
</context-param>

<listener>
    <listener-
class>acube.framework.webcomp.listener.WebCompInitializer</listener-class>
</listener>
```

- La variable de contexte **VariablesXmlLoader-config** référence les fichiers variables_XX.xml à charger, séparés par des virgules
- La variable de contexte **FopInitializer-config** référence le fichier de configuration de FOP

7. Dans les feuilles de styles :

Remplacer tous les xsl:include d'autres feuilles de style :



```
<xsl:include href="/WEB-INF/xsl/utility/dateFormatter.xsl" />
```

Par ceci:

```
<xsl:include href="response:WEB-INF/xsl/utility/dateFormatter.xsl" />
```

Dans strutsCX, lorsqu'un élément avait une valeur null, `<xsl:value-of select="parametre" />` renvoyait un blanc, un champ vide. Par contre, Struts2 lui renvoie un champ contenant la chaîne 'null'. Il faut donc désormais tester la valeur du champ avant de l'ajouter dans la sortie.



4 CONSERVATION DE FOP 0.2

Les feuilles de styles XSL:FO compatibles FOP 0.2 ne sont pas compatibles avec FOP 0.9, si l'application à migrer contient des feuilles de styles XSL:FO, alors vous devez conserver l'ancienne version de FOP.

Procédure pour conserver FOP 0.2 :

1. Créer une classe « **XSLTResultCXFop02** » dans le package « **acube.framework.webcomp.views** » :

```
package acube.framework.webcomp.views;

import java.io.OutputStream;

import javax.servlet.http.HttpServletResponse;
import javax.xml.parsers.SAXParserFactory;
import javax.xml.transform.Source;
import javax.xml.transform.Templates;
import javax.xml.transform.Transformer;
import javax.xml.transform.TransformerFactory;
import javax.xml.transform.sax.SAXResult;
import javax.xml.transform.sax.SAXSource;
import javax.xml.transform.sax.SAXTransformerFactory;

import org.apache.fop.apps.Driver;
import org.apache.fop.apps.FOException;
import org.apache.struts2.ServletActionContext;
import org.xml.sax.XMLFilter;
import org.xml.sax.XMLReader;

import com.opensymphony.xwork2.ActionInvocation;
import com.opensymphony.xwork2.util.ValueStack;

public class XSLTResultCXFop02 extends XSLTResultCX {

    /**
     * <code>serialVersionUID</code> the serialVersionUID
     */
    private static final long serialVersionUID = 9106294445518946732L;

    /** {@inheritDoc} */
    public void execute(
        ActionInvocation invocation) throws Exception {

        long startTime = System.currentTimeMillis();
        String location = this.getStylesheetLocation();

        try {
            HttpServletResponse response = ServletActionContext.getResponse();

            // Create a transformer for the stylesheet.
```

```
Templates templates = null;
Transformer transformer;
if (location != null) {
    templates = this.getTemplates(location);
    transformer = templates.newTransformer();
} else {
    transformer = TransformerFactory.newInstance().newTransformer();
}
transformer.setURIResolver(this.getURIResolver());

Object result = invocation.getAction();
if (this.getExposedValue() != null) {
    ValueStack stack = invocation.getStack();
    result = stack.findValue(this.getExposedValue());
}
//ByteArrayOutputStream baos = new ByteArrayOutputStream();
Source source = this.getDOMSourceForStack(result);

TransformerFactory tFactory = TransformerFactory.newInstance();
// Determine whether the TransformerFactory supports the
// use of SAXSource and SAXResult
if (tFactory.getFeature(SAXSource.FEATURE)
    && tFactory.getFeature(SAXResult.FEATURE)) {
    // Cast the TransformerFactory to SAXTransformerFactory.
    SAXTransformerFactory saxTFactory = ((SAXTransformerFactory)
tFactory);

    // Create an XMLFilter for each stylesheet.
    XMLFilter xmlfilter = saxTFactory.newXMLFilter(templates);

    // Create an XMLReader.
    SAXParserFactory spf = javax.xml.parsers.SAXParserFactory
        .newInstance();
    spf.setNamespaceAware(true);
    XMLReader parser = spf.newSAXParser().getXMLReader();
    if (parser == null) {
        throw new FOPEException("Unable to create SAX parser");
    }

    // xmlfilter uses the XMLReader as its reader.
    xmlfilter.setParent(parser);

    // super important pour IE...
    response.reset();
    response.setContentType("application/pdf");

    OutputStream out = response.getOutputStream();

    Driver driver = new Driver();
    driver.setRenderer(Driver.RENDER_PDF);
    driver.setOutputStream(out);
    transformer.transform(
```



```
        source,
        new SAXResult(driver.getContentHandler()));

    out.flush();
    out.close(); // ...and flush...

    if (LOG.isDebugEnabled()) {
        LOG.debug("Time:" + (System.currentTimeMillis() - startTime)
            + "ms");
    }
} else {
    throw new FOPEException("Factory cannot support SAX
transformation");
}
} catch (Exception e) {
    LOG.error(
        "Unable to render XSLT Template, '" + location + "'", e);
    throw e;
}
}
```

2. Supprimer la déclaration dans le fichier web.xml du WebcompInitializer, car cette déclaration utilise des classes de Fop 0.9 :

```
<!-- listener>
  <listener-class>acube.framework.webcomp.listener.WebCompInitializer</listener-class>
</listener-->
```

3. Définir un « **result-type** » « **xsltCXFop02** » dans le fichier de configuration struts2 pour la webcomp « **struts-acube-webcomp-NOM_PROJET.xml** » (si le fichier n'existe pas dans le projet, copier celui du Framework contenu dans le jar webcomp, renommer le, puis dans le fichier struts.xml pointer dessus au lieu de pointer sur celui du Framework) :

A ajouter dans la définition des « result-types » :

```
[...]
<result-types>
  <result-type name="xsltCX" class="acube.framework.webcomp.views.XSLTResultCX" />
  <result-type name="xsltCXFop" class="acube.framework.webcomp.views.XSLTResultCXFop" />

  <result-type name="xsltCXFop02" class="acube.framework.webcomp.views.XSLTResultCXFop02" />
</result-types>
[...]
```

4. Utiliser le type « **xsltCXFop02** » pour les actions struts qui génèrent des pdf au moyen de feuilles de styles compatibles FOP 0.2

Exemple :

```
<action name="uneAction" class="acube.projet.web.action.UneAction">
```



```
<result name="success" type="xsltCXFop02">  
  <param name="location">/WEB-INF/xsl/XSL_FO.xsl</param>  
</result>  
</action>
```