



## Guide de Migration

LISE 2.x vers LISE 3.2.x



Version 1.8 du 20/04/2010

Etat : Validé



## 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
1.6	A. Lesuffleur	Mise à jour LISE 3.1.0		06/11/09
1.7	C. Richomme A. Lesuffleur	Mise à jour Lise 3.2.0 Composant Captcha		16/04/10
1.8	A. Lesuffleur	Correction suite aux remarques MAEE du 22/04/10		22/04/10
		<b>Document validé dans sa version xxx</b>		

## 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

<b>SUIVI DES MODIFICATIONS .....</b>	<b>2</b>
<b>LISTE DE DIFFUSION .....</b>	<b>2</b>
<b>SOMMAIRE .....</b>	<b>3</b>
<b>1 MOTIVATIONS.....</b>	<b>4</b>
<b>2 COMPATIBILITE .....</b>	<b>5</b>
<b>3 PROCEDURE DE MIGRATION.....</b>	<b>6</b>

## DOCUMENTS DE REFERENCE

Version	Titre



# 1 MOTIVATIONS

- Fournir une procédure de migration d'un projet Acube LISE v2.x vers LISE v3.2.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 des feuilles FOP 0.20)

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)
  - jcaptcha-all-1.0-RC6.jar
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 `strut2` :
  - 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" />
<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.

Pour FOP, par défaut avec la webcomp la version 0.20 est utilisée avec le result-type « **xsltCXFop** ». Pour utiliser FOP 0.9, il faut utiliser le result-type « **xsltFop** », remplacer le jar de FOP et convertir les feuilles XSL non compatible.

Attention, il n'est pas possible d'utiliser FOP 0.20 et FOP 0.90 au sein d'un même projet.

8. En cas d'utilisation du **Captcha** :

Il faut passer de l'utilisation d'une Servlet sous la pile 2 au captcha appelé par Spring.

- Supprimer l'ancien fichier de configuration : /WebContent/WEB-INF/config/captcha.xml

Ce fichier n'a plus lieu d'être car l'ensemble de la configuration du captcha est maintenant contenu dans le fichier de configuration captcha du Framework (spring-captcha.xml).

- Dans le fichier « **web.xml** » :

Supprimer l'appel à la Servlet du Captcha en supprimant les lignes suivantes du fichier « web.xml » :

```
<servlet-mapping>
  <servlet-name>jcaptcha</servlet-name>
  <url-pattern>/flux/framework/captcha</url-pattern>
</servlet-mapping>
```

Et ajouter le fichier de configuration Spring du captcha

```
<context-param>
  <param-name>contextConfigLocation</param-name>
  <param-value>
    ...
    classpath*:spring-captcha.xml
  </param-value>
</context-param>
```

Le fichier « spring-captcha.xml » fait appel à la configuration par défaut du captcha du Framework. Il est possible de redéfinir sa propre configuration par projet avec la création de son propre fichier Spring que l'on appellera à la place de celui-ci (se reporter au guide du développeur).



Pour vérifier que l'action Captcha du Framework est correctement appelée, tapez dans l'url de votre navigateur et une image doit apparaître :

`http://{Serveur}/{Projet}/flux/protected/framework/captcha.xml`



*Exemple d'image de Captcha*

- **Appel de l'action de contrôle**

Modifier la classe action de vérification du Captcha de la manière suivante :

```
public class Captcha extends ActionSupport implements ServletRequestAware {

    /**HttpServletRequest*/
    private HttpServletRequest servletRequest;

    /**Booléen indiquant si le captcha est correct*/
    private Boolean validCaptcha;

    /** Valeur à tester, entrée par l'utilisateur*/
    private String captchaTest;

    /**Captcha Service
    private transient DefaultManageableImageCaptchaService captchaService;

    public String execute() {

        // Récupération de l'id du captcha
        String captchaId = this.servletRequest.getSession().getId();

        // On teste si l'utilisateur a entré le bon captcha
        this.validCaptcha = this.captchaService.validateResponseForID(
            captchaId, this.captchaTest);

        return Action.SUCCESS;
    }

    ...
}
```

L'action récupère maintenant directement la valeur entrée par l'utilisateur (variable « captchaTest » ici) afin de tester celle-ci.

Pour utiliser cette action telle quelle, il sera peut être nécessaire de modifier la partie cliente existante afin de n'envoyer que cette information sous forme de chaîne de caractères.

C'est la valeur de la variable « validCaptcha » qui sera donc renvoyée au client qui pourra dire si oui ou non le test a été fructueux.

- **Appel de l'action**

Le struts.xml doit comporter l'appel à l'action de test.



```
<package name="captchaCheck"
  extends="struts-acube-fwacubej2ee" namespace="/flux/protected/captcha">
  <action name="captchaCheck"
    class="acube.projet.web.action.captcha.Captcha">
    <result name="success" type="dispatcher">
      <param name="location">
        /templates/captcha/captchaCheck.jsp
      </param>
    </result>
  </action>
</package>
```

Le nom de l'action doit être différent de « captcha » car ce nom est utilisé par l'action de génération d'image du Framework.

- Modifier le **flux de retour** : « captchaCheck.jsp », mentionné dans le struts ci-dessus.

```
<?xml version="1.0" encoding="UTF-8"?>
<%@ page contentType="text/xml; charset=utf-8" %>
<%@ taglib uri="/struts-tags" prefix="s"%>
<CONTROLE>
  <RESULTAT>
    <VALUE><s:property value="validCaptcha" /></VALUE>
  </RESULTAT>
</CONTROLE>
```

Le flux retourne au client la valeur du test réalisé par l'action de contrôle. La partie client doit ensuite gérer ce retour pour afficher le résultat à l'utilisateur.