Documentation du mode Portail de FormPublisher

Bienvenue dans la documentation du mode Portail de FormPublisher : intégration de l'application générée dans un portail, dialogue et échanges entre l'application (formulaires) et le portail.

1. Le cycle de vie d'un formulaire en trois étapes

Un formulaire FormPublisher connait un cycle de vie en trois étapes principales :

- **l'activation** du formulaire ;
- les **sauvegardes** en cours de remplissage ;
- la **validation** du formulaire.

Le choix d'une étape est formalisé à travers une URL envoyée au serveur.

Les étapes sauvegarde et validation n'étant pas accessibles si l'étape d'activation n'a pas été conduite.

L'étape de sauvegarde n'étant plus possible après validation.

Trois étapes optionnelles peuvent être envisagées (après l'activation et avant la validation) :

- **l'abandon** du remplissage d'un formulaire ;
- les **échanges** programmés ;
- le **chargement** des données.

2. Le mode portail de FormPublisher

Le mode portail englobe les capacités d'une web application produite par FormPublisher à échanger des informations avec l'extérieur lors des étapes activation, sauvegarde et validation.

Dans sa configuration de base, FormPublisher connait trois implémentations du mode portail, à savoir :

- BasicPortal
- JwayPortal
- MixedPortal

Il est possible de choisir le mode portail pour l'une des étapes principales en positionnant le paramètre portal dans l'URL utilisée. Durant le cycle de vie d'un formulaire, il est possible d'utiliser plusieurs dialogues portails même si cela n'est pas courant.

2.1. BasicPortal

Son comportement est le suivant :

- activation : ne fait rien à ce stade ;
- sauvegarde : permet de sauvegarder le contexte de remplissage du formulaire sur le poste de l'utilisateur comme un fichier ;
- validation : produit différents éléments à l'issue du remplissage du formulaire. Ces éléments peuvent être les données saisies au format XML, le pdf rempli. Ces éléments sont communiqués hors de la web application par un envoi SMTP.

Pour les étapes optionnelles :

- abandon : arrêt de la cession en cours ;
- échanges programmés : en fonction de l'auteur ;
- chargement : à partir d'une sauvegarde locale sous forme de fichier.

2.2. JWayPortal

Ce mode portail implique un composant serveur qui offrira des services sollicités durant les trois étapes principales de la vie du formulaire :

- activation : le formulaire va faire une demande de données au service adéquat.
- sauvegarde : le formulaire va envoyé des données au service adéquat.
- validation : le formulaire va générer un ensemble d'éléments qu'il va communiquer au service adéquat.

Pour les étapes optionnelles :

- abandon : arrêt de la cession en cours avec éventuellement une redirection ;
- échanges programmés : en fonction de l'auteur ;
- chargement : désactivée.

Le composant serveur doit donc implémenter les services détaillés par le dialogue portail **JWayPortal** pour que FormPublisher puisse fonctionner ainsi.

2.3. MixedPortal

C'est le mode portail de fonctionnement par défaut de FormPublisher.

Sur validation, il permet le dépôt du datastore dans un dossier du serveur.

Exemple de paramétrage :

- jway.MixedPortal.valid.mode=**store**
- jway.MixedPortal.pathstore=/myfolder

3. Le dialogue portail JWayPortal

Les URLs employées pour solliciter un formulaire FormPublisher permettent de choisir un mode portail plutôt qu'un autre lors d'une des étapes principales du cycle de vie de celui-ci.

Ainsi les étapes principales se traduisent par une action particulière au niveau de l'URL d'appel :

- **activation** : action=login
- **sauvegarde** : action=save
- **validation** : action=valid

A l'action retenue est associée différents paramètres qui vont indiquer les comportements à prendre par FormPublisher :

- formulaire sur lequel porte la demande : documentId=<identifiant jxml>
- type de contenu à produire : mediaType=< {ji_html,jr_pdf, jr_html, jr_jxml}>
- ji_html : html en mode dynamique interview
- jr_html : html en mode dynamique mais report
- jr_jxml : jxml en mode dynamique
- jr_pdf : pdf en mode dynamique de type report

Tous ces types de contenu ne sont pas par défaut disponibles pour un document dans une publication.

En effet, lors de la phase de publication FormPublisher va analyser les liens existants de sorte à générer uniquement les contenu demandés explicitement.

- espace où mettre les données : sds=<nom de l'espace>
- données issues de ces espaces pour produire le contenu dans le type spécifié ids=<nom de l'espace1,...>

L'usage du mode portail JWayPortal implique donc une syntaxe particulière (la version présentée sera disponible à partir de FormPublisher 1.2a pour les versions précédentes nous contacter) :

- **activation**

  (BasicPortal) *http://localhost/qualicite/Controler?action=login& documentId=formulaire& mediaType=ji_html& ids=userData%2CdraftData& sds=draftData*

  (JWayPortal) *http://localhost/qualicite/Controler?action=login& documentId=formulaire& mediaType=ji_html& ids=userData%2CdraftData& sds=draftData&**portal**=JWayPortal&**dialogUSERID**=admin& **dialogDRAFTID**=workspace%3A%2F%2FSpacesStore%2F27007713-23d0-497d-904e-d3459f4c54b3&**dialogUrl**=http%3A%2F%2Flocalhost%3A8080%2Falfresco%2Fservice%2Flu%2Fjway%2Fjway%3Falf_ticket%3DTICKET_46eb09052a32949e02bd402e5ee3153e4776df94*

  Les paramètres importants sont ici **dialogUSERID** (identifiant de l'utilisateur du côté du serveur de services), **dialogDRAFTID** (identifiant de la session en cours) et **dialogURL** (l'URL permettant d'accéder au service) et enfin le paramètre **portal** permettant de sélectionner le mode portail **JWayPortal**.

  Ces paramètres vont être utilisés par la partie serveur du formulaire pour dialoguer avec le serveur de services de dialogue.

- **sauvegarder**

  (BasicPortal) *http://localhost/qualicite/Controler?action=save& documentId=formulaire& mediaType=ji_html& ids=userData%2CdraftData& sds=draftData*

  (JWayPortal) *http://localhost/qualicite/Controler?action=save& documentId=formulaire& mediaType=ji_html& ids=userData%2CdraftData& sds=draftData& portal=JWayPortal*

- **validation**

  (BasicPortal) *http://localhost/qualicite/Controler?action=valid& documentId=formulaire& mediaType=jr_pdf& ids=userData%2CdraftData&sds=draftData*

  (JWayPortal) *http://localhost/qualicite/Controler?action=valid& documentId=formulaire& mediaType=jr_pdf& ids=userData%2CdraftData& sds=draftData& portal=JWayPortal*

  Il est possible d'ajouter le paramètre **dialogMAKE** à l'url de validation.

  Si ce paramètre n'a pas de valeur, le pdf ne sera pas créé.

  S'il a la valeur pdf, le pdf sera généré et enfin s'il contient une autre valeur le comportement suivant sera adopté :

  Si **dialogMAKE** contient la valeur word, on va chercher la classe suivante,

  lu.jway.webapp.builder.WORDBuilder pour créer un fichier de sortie. Il est ainsi théoriquement possible de produire différents formats documentaires en sortie de FormPublisher.

Le dialogue portail JWayPortal se traduit sous forme d'échanges par le biais du protocole HTTP et des méthodes GET et POST de demandes auxquelles il sera répondu un messages XML.

Les demandes émises par la partie serveur du formulaire prendront la forme suivante :

- **activation** :

  (GET*) http://localhost:8080/alfresco/service/lu/jway/jway?action=getData& userID=admin& draftID=workspace://SpacesStore/27007713-23d0-497d-904e-d3459f4c54b3& formID=qualicite& alf_ticket=TICKET_1eafecdfe62666237fc30307f8968a017265ce9e*

- **sauvegarde**

  (POST) *http://localhost:8080/alfresco/service/lu/jway/jway?action=saveData& userID=admin& draftID=workspace://SpacesStore/27007713-23d0-497d-904e-d3459f4c54b3&formID=qualicite& alf_ticket=TICKET_1eafecdfe62666237fc30307f8968a017265ce9e*

  Ainsi que dans le corps de l'envoi (multipart/form-data) un paramètre *draftData* contenant un dataStore.

- **validation**

  (POST) *http://localhost:8080/alfresco/service/lu/jway/jway?action=validForm&userID=admin& draftID=workspace://SpacesStore/27007713-23d0-497d-904e-d3459f4c54b3& formID=qualicite&alf_ticket=TICKET_1eafecdfe62666237fc30307f8968a017265ce9e*

  Ainsi que dans le corps de l'envoi (multipart/form-data) un paramètre *draftData* contenant un dataStore et un second *pdfData* contenant le pdf produit.

  A noter que suivant la valeur de **dialogMAKE**, ce dernier paramètre peut ne pas exister (si **dialogMAKE** est vide) ou contenir autre chose que un pdf.

Les réponses produites par le composant serveur vont donc se traduire par une enveloppe XML (encodée en UTF-8 et indiqué comme tel dans l'en-tête HTTP au niveau du CHARSET) particulière de la forme :

- Enveloppe XML

```
<?xml version="1.0" encoding="UTF-8"?>
<PortalControlerAnswer>
   <userData><!-- un contenu --></userData>
   <draftData><!-- un contenu --></draftData>
   <alertMessage><!-- un contenu --></alertMessage>
   <redirectURL><!-- un contenu --></redirectURL>
</PortalControlerAnswer>
```

Suivant les étapes principales du cycle de vie, certaines parties de cette enveloppe vont changées :

- **activation**

  les éléments *userData* et *draftData* vont contenir des données au format dataStore pour pré-remplir le formulaire.

  alertMessage peut contenir une information qui suivant la configuration de l'interface du formulaire sera affichée ou non.

- **sauvegarde**

  les éléments *userData* et *draftData* seront ignorés.

  alertMessage peut contenir une information qui suivant la configuration de l'interface du formulaire sera affichée ou non.

  redirectURL peut contenir une URL vers laquelle l'utilisateur sera redirigé à l'issu de la sauvegarde.

- **validation**

  les éléments *userData* et *draftData* seront ignorés.

  alertMessage peut contenir une information qui suivant la configuration de l'interface du formulaire sera affichée ou non.

  redirectURL peut contenir une URL vers laquelle l'utilisateur sera redirigé à l'issu de la sauvegarde.

Il existe différentes implémentations du composant serveur de dialogue portail JWayPortal et ce dans différents langages de Java à PHP en passant par Grails.

JWay a pour sa part développer un tel composant s'intégrant dans Alfresco (nous contacter pour de plus ample information).

A noter qu'il est possible de traiter d'autres formats XML que celui de dataStores conetnus dans l'enveloppe XML en entrée du dialogue portail JWayPortal.

Cela peut se faire en implémentant un filtre de données sous forme d'une classe Java dans le package *lu.jway.webapp.ds* et ayant comme nom celui de la racine du flux XML à traiter *[E]lement root name + "Filter"* et qui étend la classe *lu.jway.webapp.ds.Filter*.

Le processus d'importation ainsi créé va traiter le flux de sorte à produire une structure de données JIL en correspondance avec la classe Filter étendue.

4. Un cas particulier, le SigningPortal

Le SigningPortal n'implémente que les étapes activation et validation d'un portail et ce d'une manière très particulière en se basant de surcroit sur un dialogue portail lui aussi spécifique.

En effet, l'objet du SigningPortal est la délégation de l'étape de validation a un composant serveur extérieur qui va peut être effectuer des traitements complémentaires avant les traitements de validation proprement dits.

Ainsi lors de l'étape de délégation de la validation, le dialogue portail SigningPortal va communiquer toutes les informations de son contexte de fonctionnement :

- comment le formulaire a été activé (BasicPortal, JWayPortal);
- quelles sont les données spécifiques à cette activation ;
- quelles sont les données issues de remplissage du formulaire.

Le SigningPortal permet aussi d'abandonner la validation et de revenir au formulaire.

La reprise du remplissage du formulaire par la biais de l'étape d'activation est dans ce contexte versatile : le formulaire est ré-activé en mode SigningPortal puis passe dans le mode portail de remplissage initial du formulaire.

- activation :

  http://<serveur>/<webappbyfp>/Controler?action=login&documentId=formulaire& mediaType=ji_html& ids=userData%2CdraftData& sds=draftData& portal=SigningPortal& dialogUrl=http://<serveur>/<service>& dialogUSERID=anonymous &dialogDRAFTID=<identifiant>

- validation :

  http://<serveur>/<webappbyfp>/Controler?action=valid& documentId=formulaire &mediaType=jr_html &ids=userDraft%2CdraftData& dialogMAKE=&portal=SigningPortal& dialogUrl=http://<serveur>/<service>& dialogUSERID=anonymous& dialogDRAFTID=anonymous

Un composant serveur de dialogue SigningPortal a été développé sur la base de SpringSurf (nous contacter pour de plus ample information).

5. Remarque sur les actions

Le contrôleur de FormPublisher côté serveur comprend un ensemble fini d'actions qui correspondent pour certaines à des étapes du cycle de vie du formulaire.

FormPublisher peut voir cette liste d'actions enrichie par le développement de nouvelles.

Pour ce faire, il faut :

- positionner la propriété lu.jway.webapp.action.extension.permited à true
- développer l'action dont le profile réponde à :

  package *lu.jway.webapp.action*

  la nom de la classe étant le *<nom de l'action en majuscule>Action* et qui implémente l'interface *WebAction* .

Ce mécanisme permettant de modifier le comportement standard de FormPublisher, il est à manipuler avec beaucoup de précautions, n'hésitez pas à faire garantir le bon fonctionnement de nouvelles actions par JWay.

En général, il est préféré à ce mécanisme, l'usage des extensions ou échanges programmés.