|  |
| --- |
|  |
| **Documentation technique – Docflow Process** |
| **Objet Document :**  Ce document a pour but d’expliquer les différentes techniques d’intégration des processus métiers au niveau de **Docflow Process**. |

**Auteurs du document :**

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Auteur | Version | Commentaire |
| 19/03/2019 | Yassine EL GHARBI | 1.0 |  |
| 25/03/2019 | Marouane EL MABROUK | 1.0.1 |  |
| 16/05/2019 | Zakaria MOULOU | 1.1 |  |

**Sommaire**

1. **Schéma d’un processus sous Docflow Process:**

Un processus est décrit par un fichier XML contenant les 3 sections suivantes à l’intérieur de la balise <Process> :

1. **DataSources :** la section contenant les sources de données.
2. **MetaDatas :** l’ensemble des métadonnées du processus.
3. **Steps :** les différentes étapes du processus.

**I.1 Sources de données (DataSources):**

Les sources de données (DataSources) sont utilisées pour remplir un champ avec un requête :

Afin de créer une source de données, il faut ajouter la balise <DataSource> sous la section <DataSources> :

<DataSource name="GetCollaborateurs" type="MetaDataItems">

<Param name="Query" value="IF (EXISTS(select \* from GROUPES g,

GROUPE\_COLLABO gc where g.Libelle=''Groupe SP'' and g.code\_groupe=gc.Code\_groupe and gc.collaborateur=@$ConnectedUserName)) Begin SELECT \* FROM (values ('''','''')) List(id, Label) UNION SELECT CAST(CODE\_COLLABORATEUR as nvarchar(max)) as id, CONCATENER\_COLLABORATEUR as Label FROM COLLABORATEUR end else Begin SELECT \* FROM (values ('''','''')) List(id, Label) UNION SELECT \* FROM (values

(@$ConnectedUserName,@$ConnectedUserName)) List(id, Label) end;" />

</DataSource>

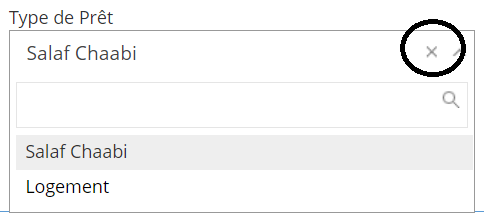
* Afin de remplir une liste déroulante avec la possibilité d’effacer la sélection il faut sélectionner une ligne vide dans la requête à exécuter :

<DataSource name="GetTypesPret" type="MetaDataItems">

<Param name="Query" value="SELECT \* FROM (values ('''','''')) List(id, Label) union SELECT \* FROM (values (''1'', ''Salaf Chaabi''), (''2'', ''Logement'')) List(id, Label)" />

</DataSource>

Résultat :



* Afin de remplir un champ de type date avec la date courante sous le format **dd/MM/yyyy**, il faut spécifier le format avec la requête suivante :

<DataSource name="GetCurrentDate" type="MetaDataItems">

<Param name="Query" value="SELECT \* FROM (values (convert(varchar, getdate(), 103),convert(varchar, getdate(), 103))) List(id, Label);" />

</DataSource>

N.B : Une liste peut être remplit soit avec une sélection de la base de données :

<DataSource name="GetEntite" type="MetaDataItems">

<Param name="Query" value="select DESIGNATION\_ENTITE as id, DESIGNATION\_ENTITE as Label from dbo.ENTITE where CODE\_ENTITE = (select ENTITE\_COLLABORATEUR from dbo.COLLABORATEUR where CONCATENER\_COLLABORATEUR = @NomPrenom)" />

</DataSource>

Soit en remplissant les données manuellement :

<DataSource name="GetTypesPret" type="MetaDataItems">

<Param name="Query" value="SELECT \* FROM (values ('''','''')) List(id, Label) union SELECT \* FROM (values (''1'', ''Salaf Chaabi''), (''2'', ''Logement'')) List(id, Label)" />

</DataSource>

* Au niveau des sources des données, on peut accéder aux variables de session ainsi que les variables du processus courant. Ces variables sont les suivantes :

@CurrentUser / $ConnectedUserName / $ConnectedUserGroupes / $ConnectedUserSerial / $ConnectedUserGrade / $ConnectedUserRole / $ConnectedUserEntity / $ConnectedUserEntitySup / \_Creator / \_CreatorEntity / \_CreatorGroupes / \_CreatorRole / @idProcess

**I.2 Métadonnées (MetaDatas):**

Cette section contient les métadonnées du processus, ces métadonnées sont les champs à remplir et à afficher dans les différentes étapes du processus. La section <MetaDatas> contient des balises de type <Data> qui décrivent les métadonnées du processus :

<Data name="Partenaire" label="Partenaire" type="List">

<DataSource ref="SelectPartenaire" />

         <Constraints>

<Constraint type="DependOn" ref="Type" />

</Constraints>

</Data>

Dans cette partie on décrit :

* Le nom de la métadonnée, sont type, ainsi que le libellé qui sera affiché pour une donnée dans les différents formulaires :

<Data name="Partenaire" label="Partenaire" type="List">

* Les types disponibles sont les suivants : **Text / Date / List / Textarea / File / Numeric / Money**.
* Sa source de données. On fait appel aux sources de données décrites dans la première partie.

<Data name="Partenaire" label="Partenaire" type="List">

<DataSource ref="SelectPartenaire" />

        <Constraints>

<Constraint type="DependOn" ref="Type" />

</Constraints>

</Data>

* Les contraintes sur cette métadonnée :

<Data name="Partenaire" label="Partenaire" type="List">

<DataSource ref="SelectPartenaire" />

        <Constraints>

<Constraint type="DependOn" ref="Type" />

</Constraints>

</Data>

Dans cet exemple, la métadonnée **Partenaire** dépend d’une autre métadonnée appelée **Type**, quand on fait cette déclaration, on peut récupérer la valeur de la métadonnée référencée quand elle est changée afin d’effectuer un traitement sur la métadonnée courante.

**NB :** Une métadonnée de type File contient la paramètre **Multi**, qui est de type booléen et qui est utilisé pour activer ou désactiver la possibilité d’ajouter ce fichier plusieurs fois :

<Data name="ConventionSignee" label="Convention signée" type="File" col="1" Multi="false" />

* Si on veut créer une liste qui se remplit automatiquement en choisissant une valeur de la première liste comme un appel AJAX on fait comme suit :

<Data name="Famille" label="Famille" type="List">

<DataSource ref="GetFamille" />

</Data>

<Data name="Produits" label="Désignation produits" type="List">

<DataSource ref="GetDesignation" />

<Constraints>

<Constraint type="DependOn" ref="Famille" />

</Constraints>

</Data>

avec le DataSource GetFamille reçoit une requête de sélection de la table Familles comme suit :

<DataSource name="GetFamille" type="MetaDataItems">

<Param name="Query" value="SELECT \* FROM (values ('','')) List(id, Label) union SELECT cast(id as nvarchar(max)), cast(Label as nvarchar(max)) FROM Familles" />

</DataSource>

Et le DataSource GetDesignation reçoit une requête de la table Produits et Familles comme suit :

<DataSource name="GetDesignation" type="MetaDataItems">

<Param name="Query" value="SELECT \* FROM (values ('','')) List(id, Label) union select cast(id as nvarchar(max)), cast(Label as nvarchar(max)) from produits where FamilleID = (select id from familles where label = @Famille)" />

</DataSource>

**1.2.1 Générer des documents PDF :**

* Pour générer automatiquement un fichier PDF à partir des métadonnées, on ajoute une section Documents avec un Data de type File comme suit :

<Section name="Documents">

<Data name="BonPrelevementTechnique" label="Bon de prélévement techniques" type="File" />

</Section>

[Uncomplet]

**I.3 Les étapes du processus (Steps):**

Dans cette étape, on décrit les différentes étapes du processus. Dans un processus on a trois types des étapes :

* Une étape initiale : Un processus contient une seule étape initiale avec le paramètre **Initial="true" :**

<Steps>

<Step name="Initiation" Initial="true" Final="false">

<Description>c est la première étape du process</Description>

<MetaDataConstraints/>

<Collaborators>

<AnyUser />

</Collaborators>

<Duration value="1" Variable="true" />

<Notifications />

<Alerts />

<NextSteps>

………………

Des étapes finales : Un proces

* sus peut avoir plusieurs étapes finales. Ce type des étapes met fin à l’exécution du processus. Sur le fichier XML, une étape finale a le paramètre

<Step name="Convention clôturée" Initial="false" Final="true">

     <MetaDataConstraints>

         <Data ref="Partenaire" constraint="HideInThisStepOnly">

<Condition>

<Expression>

                 ${Type} != "Partenariat"

            </Expression>

</Condition>

</Data>

………………

* Des étapes intermédiaires : Les différentes étapes de traitement d’un processus. Une étape intermédiaire a la valeur False pour les deux paramètres Initial et Final :

<Step name="Signature note de désignation" Initial="false" Final="false">

<MetaDataConstraints>

<Data ref="Type" constraint="DisabledInThisStepOnly" />

<Data ref="Partenaire" constraint="HideInThisStepOnly">

<Condition>

<Expression>

                ${Type} != "Partenariat"

            </Expression>

</Condition>

</Data>

………………

Pour chaque étape, on décrit les éléments suivants :

* Les contraintes sur les métadonnées à l’intérieur de la balise **<MetaDataConstraints>**:

Ces contraintes sont liées à la visibilité de chaque métadonnée au niveau de l’étapes courante. Afin d’ajouter une contrainte pour une métadonnée, il faut ajouter cette métadonnée dans la balise **Data** avec les deux paramètres : **ref** qui prend comme valeur le nom de la métadonnée en question, et **constraint** pour lequel on donne la contrainte de visibilité de cette métadonnée. Il existe trois types de contraintes de visibilité :

à **RequiredInThisStepOnly** :

<MetaDataConstraints>

<Data ref="Convention" constraint="RequiredInThisStepOnly" />

La métadonnée sera obligatoire afin de valider l’étape et passer à l’étape suivante.

à **HideInThisStepOnly** :

<MetaDataConstraints>

<Data ref="DateSignature" constraint="HideInThisStepOnly" />

La métadonnée sera invisible pour cette étape.

à **DisabledInThisStepOnly** :

<MetaDataConstraints>

<Data ref="NomPrenom" constraint="DisabledInThisStepOnly" />

La métadonnée sera visible mais désactivée pour cette étape.

* Une contrainte sur une métadonnée peut être conditionnelle. Si la condition est satisfaite, la contrainte sera appliquée à la métadonnée.

On peut déclarer une condition sous la balise **<Condition>** à l’intérieur de la balise **<Data>**. Une condition sur la visibilité d’une métadonnée est de type expression :

* + - On déclare cette expression à l’intérieur de la balise **<Expression> :**

<Data ref="NomBanque" constraint="HideInThisStepOnly">

<Condition>

<Expression>

${PieceDemandee} != "Attestation de virement irrévocable"

</Expression>

</Condition>

</Data>

<Data ref="NomBanque" constraint="RequiredInThisStepOnly">

<Condition>

<Expression>

${PieceDemandee} == "Attestation de virement irrévocable"

</Expression>

</Condition>

</Data>

Dans cet exemple, on évalue la valeur d’un autre champ afin d’afficher ou de cacher la champ nom de la banque.

Au niveau d’une expression, on peut utiliser les valeurs des autres métadonnées, comme on peut évaluer les variables de session ainsi que les variables du processus (le nom d’utilisateur connecté, l’id du processus …).

Au niveau des contraintes de visibilité des métadonnées, on peut aussi récupérer le nom de l’action effectuée :

<Data ref="BulletinPaie" constraint="RequiredInThisStepOnly">

      <Condition>

        <Expression>

          ${$ActionName}.ToString() == "Valider"

        </Expression>

      </Condition>

    </Data>

**NB :** Ce cas est important si par exemple une pièce jointe est obligatoire seulement quand on valide l’étape afin de passer vers l’étape suivante, et elle n’est pas obligatoire si on fait un retour vers l’étape précédente.

**Exemple :**

Dans ce processus, quand on valide l’étape d’initiation, la métadonnée « Pôle Concerné »  est obligatoire si on choisit l’action « signature avec convention signée », sinon si on choisit  l’action « signature sans convention signée » cette métadonnée sera invisible :

<Data ref="PoleConcerne" constraint="RequiredInThisStepOnly">

            <Condition>

                <Expression>

                    ${$ActionName}.ToString() == "ConvSignnee"

                </Expression>

            </Condition>

</Data>

        <Data ref="PoleConcerne" constraint="HideInThisStepOnly">

            <Condition>

                <Expression>

                    ${$ActionName}.ToString() != "ConvSignnee"

                </Expression>

            </Condition>

</Data>

La variable **ActionName** sera présentée par la suite dans la partie **NextSteps.**

* Le deuxième élément à déclarer au niveau d’une étape c’est la section **<Collaborators> :**

Dans cette étape, on déclare les utilisateurs concernés par l’étape courante.

Les utilisateurs concernés par une étapes, peuvent être :

* + Un groupe : Tous les collaborateurs appartenant à ce groupe peuvent traiter cette étape

<Collaborators>

<Group name="Groupe Service Comptabilité" mode="Many" count="1" />

</Collaborators>

* + Un collaborateur : Un collaborateur bien précis va traiter cette étape. On récupère ce collaborateur en appelant une procédure stockée :

<Collaborators>

<Procedure name="Action\_GetChefDeProjet" mode="Many" count="1" reexecute="true" />

</Collaborators>

* L’élément **<Duration>** au niveau d’une étape permet de définir le délai du traitement d’une étape en jours avec le paramètre **value**. Ce délai peut être ouvrable avec **Ouvrable="true"**, comme il peut être calendaire avec **Ouvrable="false"**.

<Duration value="15" Ouvrable="true" />

* Une section **<Notifications>** peut aussi être présente dans une étape. Cette section permet de définir les notifications qui seront envoyées par mail aux collaborateurs concernés :

Dans une étape on peut envoyer plusieurs notifications en mettant chaque notification dans un **mail group.** Le paramètre **MailGroup** permet de mettre une notification dans un mail group.

<Notifications>

<CurrentUser target="To" template="DemandeValidationCooperation" MailGroup="Mg1" />

<CurrentUser target="To" template="SignatureConvention" file="Note;ConventionSignee" MailGroup="Mg2" />

</Notifications>

La balise **CurrentUser** permet d’envoyer la notification aux collaborateurs concernés par cette étape.

On peut aussi envoyer la notification au collaborateur qui a initié le processus avec la balise **<Creator> :**

<Notifications>

<Creator target="To" template="DemandeNonAccepteePret" />

</Notifications>

Le paramètre Target peut avoir deux valeurs :

* **To :** Le collaborateur concerné par l’étape sera le destinataire de la notification.
* **Cc :** Le collaborateur concerné par l’étape sera en copie dans l’email envoyé comme notification pour cette étape.

Le paramètre **template** permet de donner le nom de la template de l’email à envoyer pour la notification. Cette template est stockée dans la table **MailTemplate** dans la base de données **PrestigeMailServiceDb.**

Au niveau d’une template, on peut récupérer les métadonnées du processus. Voici un exemple d’une template de notification :

<p>Bonjour,</p>

<p>Votre demande d&rsquo;avance sur salaire a &eacute;t&eacute; accept&eacute;e, dont ci-dessous les &eacute;l&eacute;ments cl&eacute;s :</p>

<p><strong>Type de demande :</strong> @ProcessName</p>

<p><strong>Etat :</strong> @StepName</p>

<p><strong>Exp&eacute;diteur :</strong> @Sender</p>

<p><strong>Nom et Prénom du demandeur :</strong> @NomPrenom</p>

<p>----</p>

<p><cite>Ce message est g&eacute;n&eacute;r&eacute; automatiquement. Veuillez ne pas y r&eacute;pondre.</cite></p>

**NB :**

Il est possible d’envoyer des pièces jointes avec l’email de notification. En séparant les noms des métadonnées qui représentent ces pièces jointes avec un point virgule :

<Notifications>

<CurrentUser target="To" template="DemandeValidationCooperation" MailGroup="Mg1" />

<CurrentUser target="To" template="SignatureConvention" file="Note;ConventionSignee" MailGroup="Mg2" />

</Notifications>

On peut envoyer des notifications à un groupe des collaborateurs comme suit :

<Notifications>

<Group value="Groupe Validateurs" target="To" template="NotificationPreLevementTechniqueMail" file="PieceJustif" MailGroup="Mg1" />

</Notifications>

Balises Notifications :

      <Notifications>

        <Creator target="To" template="template1" MailGroup="Mg1" file="File1;File4" />

        <Group value="groupX" target="To" template="template1" MailGroup="Mg1" />

        <Procedure value="ProcedureZ" target="To" template="template1" MailGroup="Mg1" />

        <CurrentUser target="To" template="template1" MailGroup="Mg1"/>

        <Procedure value="ProcedureY" target="To" template="template1" MailGroup="Mg1" />

        <MetaData value="UserAddress" target="To" template="template1" MailGroup="Mg1" />

        <ManagerCurrentUser target="Cc" template="template1" MailGroup="Mg1" />

        <Manager target="Cc" template="template1" MailGroup="Mg1" />

<Role value="Responsable" dept="Direction" target="Cc" template="template1" MailGroup="Mg1" />

        <CurrentUser target="To" template="template2" MailGroup="Mg2"/>

        <Mail value="[ClientAddress@gmail.com](mailto:ClientAddress@gmail.com)" target="Cc" template="template2" MailGroup="Mg2"/>

        <Role value="RoleG" target="Cc" template="template2" MailGroup="Mg2" />

      </Notifications>

* *Au niveau d’une étape, on peut définir les étapes qu’on peut atteindre à partir de l’étape courante au niveau de la*  *balise* ***<NextSteps>.*** *Ces étapes sont représentées par des balies* ***<Step>.***

<NextSteps>

<Step name="Etude de la demande" ActionName="Valider" ActionLabel="Valider" />

<Step name="Demande non acceptée" ActionName="Rejeter" ActionLabel="Rejeter" />

</NextSteps>

La balise **<Step>** contient les paramètres suivants :

* **name :** le nom de l’étape qu’on peut atteindre en choisissant cette action **(ActionName).**
* **ActionName :** le nom de l’action à effectuer. Cette variable peut être récupérée dans des conditions telles que les conditions de visibilité des métadonnées. Par exemple :

<Data ref="Decision" constraint="RequiredInThisStepOnly">

       <Condition>

         <Expression>

           ${$ActionName}.ToString() == "Valider"

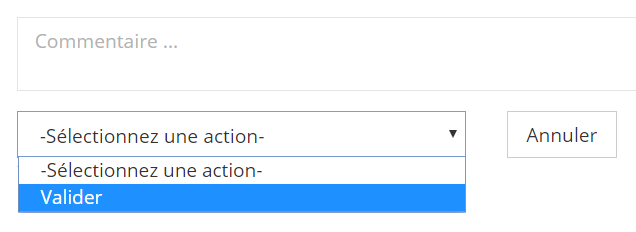
         </Expression>

       </Condition>

     </Data>

Dans ce cas, si on choisit de valider cette étape, le champ **Decision** sera obligatoire.

* **ActionLabel :** l’action qui sera visible pour l’utilisateur sur l’interface de traitement de cette étape :



Dans cet exemple, **ActionLabel**est **Valider.**

**NB :** L’affichage des actions peut être conditionnel. Dans ce cas l’action ne sera visible que si la condition est satisfaite.

Une condition est évaluée en appelant une procédure stockée de la manière suivante :

<NextSteps>

    <Step name="Préparation dossier" ActionName="Valider" ActionLabel="Valider" >

      <Condition>

        <Procedure name="Action\_Est\_Chef" />

      </Condition>

    </Step>

    <Step name="Validation Pôle" ActionName="Valider" ActionLabel="Valider" >

      <Condition>

        <Procedure name="Action\_Not\_Appartient\_Cooperation" />

      </Condition>

    </Step>

    <Step name="Désignation des équipes" ActionName="ConvSignnee" ActionLabel="Présence Convention signée">

      <Condition>

        <Procedure name="Action\_Appartient\_Cooperation" />

      </Condition>

    </Step>

    <Step name="Préparation à la signature" ActionName="ConvNotSignnee" ActionLabel="Absence Convention signée">

      <Condition>

        <Procedure name="Action\_Appartient\_Cooperation" />

      </Condition>

    </Step>

</NextSteps>

Si la procédure appelée dans une condition retourne **True**, l’action sera visible.

Pour afficher liste des documents PDF générés automatiquement on ajoute ce qui suit après <NextSteps /> :

<CustomActions>

<Generate fileFormat="pdf" template="BonPrelevementTechnique.cshtml" filename="BonPrelevementTechnique" />

</CustomActions>

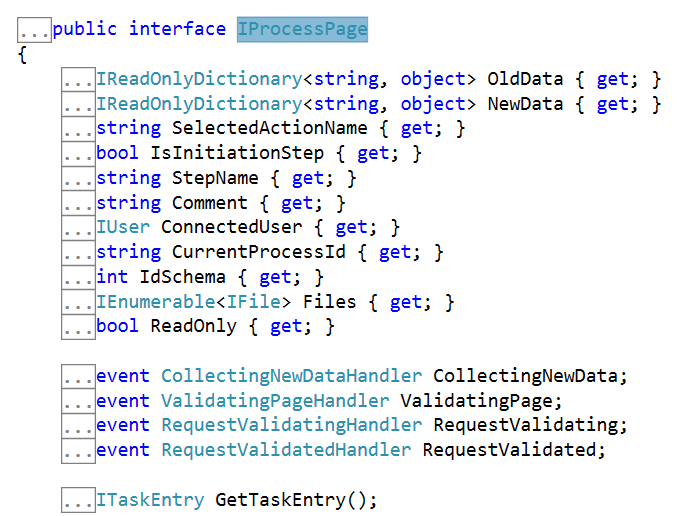
La propriété template est le modèle du fichier PDF qui prend le fichier cshtml se trouvant dans le Workflow.Webservice sur le chemin suivant : C:\inetpub\wwwroot\<Nom\_WorkFlow.Webservice>\Templates

"BonPrelevementTechnique.cshtml" est le nom du template razor sur le chemin ci-dessous.

1. **Parties spécifiques des formulaires :**

Si vous avez une partie d’un formulaire qui demande un traitement spécifique, DFP permet d’afficher un **user control** pour un type de processus dans des étapes bien précises. Il suffit juste d’implémenter l’interface IProcessPage

Cette méthode contient les informations suivantes :



Pour faire la vérification des processus et des étapes ainsi que l’affichage des contrôles on créer une méthode par exemple **DoWork()** avec un type de retour booléen, après on met une condition dans l’événment page\_load pour exécuter le RequestValidating ou RequestValidated comme suit :

try

{

if (DoWork())

{

ProcessPage.RequestValidated += RequestValidated;

}

}

catch (Exception ex)

{

Tracing.Trace(Enumerations.TraceType.Error, string.Empty, "UCProcessExportCSV.Page\_Load", ex);

((Page)Page.Master).ShowInformationPopup(HelperClass.GeneralErrorMessageTitle, HelperClass.GeneralErrorMessage, TypeImage.Error);

}

L’ajout et la modification des valeurs des métadonnées se fait au niveau de la méthode **RequestValidating(Dictionary<string, object> newData)**, cette méthode est pour les événements qui s’exécute au moment de la validation. **newData** c’est le dictionnaire qui contient les métadonnées du processus avec leurs nouvelle valeurs pour l’étape courante. Sur ce dictionnaire, on peut ajouter les nouvelles métadonnées avec leurs valeurs à partir des contrôles du **user control** spécifique.

L’événement **RequestValidated(IProcessDto processDto)**, s’exécute après la validation.