Installation canal Annuaire

Canal Annuaire

Installation, paramétrage du canal du canal
Auteur : Sébastien GAUDIN (Université nancy 2)

- Canal Annuaire
- Configuration du canal
  - Le fichier de configuration
    - Les textes de la configuration
    - Les tables
      - Tables LOCALE
      - Tables XML
      - Tables LDAP
      - Tables SQL
    - Les annuaires
      - Les formulaires
      - Les fiches
  - La classe de customMail
    - getUrlChoose
    - getUrlMailToLink
    - getUrlMailToChannel
- Internationalisation
  - Au niveau de la configuration
  - Les textes "normaux"
- Déploiement

Configuration du canal

Le fichier de configuration

Vous pouvez télécharger ici l'exemple que nous allons détailler.

Les textes de la configuration

```
<languages><varsOfText>
                    <varOfText name="LOGIN" />
                    <varOfText name="NAME" />
                    <varOfText name="FIRST_NAME" />
                    <varOfText name="FORMATION" />
                    <varOfText name="MAIL" />
                    <varOfText name="COMPOSANTE" />
                    <varOfText name="DISCIPLINE" />
                    <varOfText name="STATUS" />
                    <varOfText name="RESPONSABILITY" />
                    <varOfText name="CAMPUS" />
                    <varOfText name="PHONE" />
                    <varOfText name="FAX" />
                    <varOfText name="ADDRESS" />
                    <varOfText name="PERSO_PAGE" />
                    <varOfText name="INTROPUBLIC" />
                    <varOfText name="NAME_ANNU_1" />
                    <varOfText name="NAME_ANNU_2" />
                    <varOfText name="NAME_ANNU_3" />
                    <varOfText name="NAME_ANNU_4" />
                    <varOfText name="NAME_ANNU_5" />
          </varsOfText>
          <language langId="en_US" />
          <language langId="fr_FR" />
</languages>
```

Tous les textes utilisés dans ce fichier de configuration ne doivent pas être écrits ici, mais dans les fichiers de langues (répertoire /properties/languages). Vous devez déclarer dans ici les noms des variables de texte que vous allez utiliser ainsi que les langages disponibles (le premier précisé est le langage par défaut si un texte n'existe pas dans la langue spécifiée par l'utilisateur).

Dans notre exemple, le fichier /properties/languages/config_fr_FR.properties contient par exemple :

ADDRESS = Adresse

INTROPUBLIC = Cet annuaire mentionne les personnes ayant accepté d'y figurer.

Les tables

Les tables permettent d'avoir une correspondance entre un code et un libellé. Ces tables peuvent être de 3 types, selon la provenance des valeurs :

Tables LOCALE

```
<table name="listeCampus" type="static">
              <item code="plg" value="PÔLE LORRAIN DE GESTION" fieldOrder="10" />
              <item code="droit" value="CAMPUS CARNOT-RAVINELLE, DROIT, SCIENCES ECONOMIQUES ET AES"
fieldOrder="1000" />
</table>
```

Les items de cette table sont décrits directement dans le fichier de configuration. L'ordre est spécifié selon la valeur de l'attribut "fieldOrder", par défaut c'est dans l'ordre d'aparition dans le fichier.

Tables XML

```
<table name="listeStatuts" type="xml"
              file="statut.xml" />
```

Les items de cette table se trouvent dans le ficher xml /prperties/xmlTables/statut.xml. L'ordre est spécifié selon la valeur de l'attribut "fieldOrder", par défaut c'est dans l'ordre d'aparition dans le fichier :

```
<list name="listeStatuts">
                        <item  code="PERS_ENS" value="Enseignant / Chercheur" />
                        <item code="PERS_ENSV" value="Enseignant vacataire" />
                        <item code="PERS_ATER" value="ATER / Moniteur / Lecteur" />
                        <item code="PERS_THES" value="Etudiant en thèse" />
                        <item code="PERS_EMP" value="Personnel ''IATOS/ITARF''" />
                        <item code="PERS_STAG" value="Stagiaire" />
                        <item code="PERS_RET" value="Personnel retraité" />
                        <item code="AUTRE" value="Autre" />
</list>
```

Tables LDAP

```
<table name="listeComposantes"
            type="ldap"
            attrCode="cn"
            attrValue="description"
            fieldOrder="description"
            base="ou=Structures"
            scope="one"
            filter="objectclass=ClassComposante"
            url="ldap://ldap1.univ.fr:392/dc=univ,dc=fr ldap://ldap2.univ.fr:392/dc=univ,dc=fr" />
```

Les items de cette table sont récupéré par la requete LDAP. L'ordre est spécifié selon un l'attribut "fieldOrder", par défaut c'est dans l'ordre de la récupération.

Tables SQL

```
<table name="listeComposantes2"
            type="sql"
            bddType="JDBC"
            bindpass="loginDeConnexion"
            binddn="PassDeConnexion"
            attrCode="idComp"
            attrValue="descComp"
            fieldOrder="descComp"
            filter="(cond1='A') AND (cond2='true')"
            url="jdbc:oracle:thin:@//oracle.univ.fr:1521/BASE"
            driverClassName="oracle.jdbc.OracleDriver"
            tableName="nomDeLaTable"
            />
```

Dans l'attribut "tableName", il est possible de spécifier plusieurs tables pour faire une jointure. Par exemple "Table1 A, Table2 B". Dans ce cas, les attributs de code, valeur et tri peuvent utiliser les noms logiques des tables ("A.idComp").

Si vous utilisez un pool JNDI, certains attribut ne sont plus nécessaire :

```
<table name="portalUser"
                        type="sql"
                        bddType="JNDI"
                        url="NomDuPoolTomcat"
                        tableName="nomDeLaTable"
                        attrValue="descComp"
                        attrCode="idComp"
                        fieldOrder="descComp"
                        filter="(cond1='A') AND (cond2='true')"/>
```

Quel que soit le type de table utililsé, il est possible de lier les valeurs à celles d'une autre table. Pour cela, précisez l'attribut "tableLink" pour dire à quelle

Les annuaires

```
<directory name="rechercheIndividu"
          title="NAME_ANNU_1" intro="INTROPUBLIC" allowExport="true" >
          <server url="ldap://ldap1.univ.fr:392/dc=univ,dc=fr
          ldap://ldap2.univ.fr:392/dc=univ,dc=fr" /> ...
          </directory>
```

Un annuaire est définit par le tag "directory"

Le "name" est le nom de l'annuaire. Il sert à savoir sur quel annuaire on effectue la recherche mais c'est aussi le nom que l'on utilise lors d'un appel en mode servant.

Le "title" est le texte qui apparait pour que l'utilisateur choisisse un annuaire.

Si l'attribut "intro" est valué, cette phrase apparaitra sous le titre de l'annuaire dans le formulaire. Cela permet de donner une description sur l'annuaire.

Si "allowExport" est à true, il sera possible pour l'utilisateur d'exporter en csv le résultat de ses recherche dans cet annuaire.

Les formulaires

```
<request base="ou=people" scope="sub"  maxEntries="100" identifier="uid">
                        <elemOfRequest  type="direct" filter="!(eduPersonPrimaryAffiliation=student)"/>
            <elemOfRequest varI18n="LOGIN" type="text" filter="uid=%s*" minChar="3" delStar="false"/>
            <elemOfRequest varI18n="NAME" type="text" filter="sn=%s*" minChar="4" delStar="false"/>
            <elemOfRequest varI18n="COMPOSANTE" type="listTable" table="listeComposantes" filter="composante=%
c" />
            <!-- <
            elemOfRequest varI18n="COMPOSANTE" type="textTable" table="listeComposantes" filter="composante=%c"
searchValue="*%s*"/>
            <elemOfRequest varI18n="COMPOSANTE" type="radioTable" table="listeComposantes" filter="composante=%
c" />
            <elemOfRequest varI18n="COMPOSANTE" type="checkboxTable" table="listeComposantes" filter="
composante=%c" />
            <elemOfRequest varI18n="COMPOSANTE" type="multiListTable" table="listeComposantes" filter="
n2composante=%c" />
            <elemOfRequest varI18n="COMPOSANTE" type="subTable" table="listeFormations" filter="(n2composante=%
cm)(n2formation=%cc)" />
            -->
</request>
```

Dans le tag "request" il faut positionner tous les éléments qui constitueront le formulaire. Ces éléments sont des "elemOfRequest".

Le libellé du champ est indiqué dans l'attribut "varI18n", il s'agit du nom de la variable de texte à utiliser.

Il est possible de limiter la saisie de l'utilisateur en donnant un nombre minimum de caractères dans la saisie d'un champs texte (minChar) et de supprimer automatiquement le caractère * (delstar).

Il en existe de plusieurs types :

- direct : le filtre est automatiquement appliqué et est directement spécifié dans le l'attribut "filter".
- text : Ceci va afficher une zone de saisie textuelle. Le filtre engendré est précisé dans l'attribut "filter" et la saisie utilisateur se positionnera à la place du "%s".
- listTable : Ceci va afficher une liste déroulante dont les valeurs sont prises dans une des tables déclarée. Le filtre engendré est précisé dans l'attribut "filter" et la saisie utilisateur se positionnera à la place du "%c".
- texteTable : Ceci va afficher une zone de saisie textuelle. L'utilisateur va pouvoir saisir une chaine de caractères pour retrouver des valeurs (pas les codes) des items d'une table déclarée. Le filtre engendré est précisé dans l'attribut "filter" et est répété pour chaque réponse correspondant à la saisie utilisateur avec un OU. Le code de chaque réponse sera positionné à la placedu %c. Les réponses convenant à la saisie sont recherchée dansles valeurs de la table selon le masque "searchValue". Le %sreprésente la partie que l'utilisateur a saisi.
- radioTable : Ceci va afficher une liste à puce dont les valeurs sont prises dans une des tables déclarée. Le filtre engendré est précisé dans l'attribut "filter" et la saisie utilisateur se positionnera à la place du "%c".

- checkboxTable : Ceci va afficher une liste de cases à cocher dont les valeurs sont prises dans une des tables déclarée. Le filtre engendré est précisé dans l'attribut "filter" et la saisie utilisateur se positionnera à la place du "%c".
- multiListTable : Ceci va afficher une liste à sélection multiple dont les valeurs sont prises dans une des tables déclarée. Le filtre engendré est précisé dans l'attribut "filter" et la saisie utilisateur se positionnera à la place du "%c".
- subTable : Ce type de champ crée deux listes déroulantes. L'une déterminant les valeurs de l'autre. Le filtre engendré est précisé dans l'attribut "filter" et le code la table maitre se positionnera à la place du "%cm" et le code de la table fille se positionnera à la place de l'attribut %cc. Si vous utilisez ce type de table, vous pouvrez utiliser l'attribut allowAllSubValues qui, à "true", permet de sélectionner "toutes les valeurs de la seconde liste".

  ⚠ pour plus d'informations sur l'utilisation de subtable cf. http://listes.esup-portail.org/sympa/arc/esup-utilisateurs/2005-11/msg00074.html

Les fiches

```
<card maxCard="1"> ...
           </card>
```

Sur le tag "card" il faut préciser l'attribut "maxcard" pour limitter le nombre de fiches à l'écran avant de passer en mode tableau.

```
<sort order="ascending">
                       <sortOn attribute="sn" />
                       <sortOn attribute="cn" />
</sort>
```

Vous pouvez préciser l'ordre d'affichage des résultats selon un ou plusieurs atributs de la personne.

```
<line varI18n="LOGIN" visibility="list" linkType="toCard"> ... </line>
<line varI18n="MAIL" visibility="card" linkType="mail"  mailClass="org.esupportail.portal.channels.CAnnuaire.
config.custom.CustomMail"> ... </line>
<line varI18n="COMPOSANTE" visibility="card"> ... </line>
<line varI18n="CAMPUS" linkType="external" externalLink="http://www.univ.fr/presentation/campus/%c.html?
depuis_id=99"> ... </line>
<line varI18n="ADDRESS" visibility="list"> ... </line>
<line varI18n="PERSO_PAGE" linkType="attribute"> ... </line>
```

Une ligne de fiche doit obligatoirement posséder un libellé ("varI18n").

Une ligne peut avoir 3 types de visibilité (quelque soit le type, l'attribut est récupérable en mode servant et en export) :

- list : cet attribut sera visibile en mode tableau et en mode fiche.
- card : cet attribut sera visible uniquement en mode fiche (mode par défaut).
- none : cet attribut ne sera pas visible dans le canal.
  Une ligne peut être cliquable. Pour cela, il faut préciser le type de lien ("linkType") :

- toCard : permet de passer du mode tableau au mode fiche.
- external : permet d'acceder à un site externe. La valeur de l'attribut est placée à la place du %c.
- attribute : le lien est la valeur de l'attribut
- mail : il s'agit d'un mail cliquable. La classe de personnalisation des lien doit être renseignée dans l'attribut "mailClass". La classe par défaut teste si le canal est déjà en mode servant ou en mode anonyme. Dans l'un de ces 2 cas, c'est un lien "mailto" qui est positionné, sinon c'est le canal "mailTo" qui est appellé en mode servant.

```
<line varI18n="LOGIN" visibility="list"linkType="toCard">
                                <attributeLine attribute="uid" type="direct" />
</line>
<line varI18n="NAME" visibility="list" fieldConnection=", ">
                                <attributeLine attribute="sn" type="direct" />
                                <attributeLine attribute="givenname" type="direct" /> </line>
<line varI18n="STATUS" visibility="list">
                                <attributeLine attribute="status" type="table" linkTable="code" table="
listeStatuts" />
</line>
```

Une ligne peut avoir un ou plusieurs attributs concaténés. Vous devez donc décrire les attributs visibles par ligne. Dans le cas où une ligne contient plusieurs attributs, fous pouvez spécifier une expression de séparation de ces champs. Dans l'éxemple ci-dessus, en positionnant fieldConnection à", ", chaque valeur des attributs à récupérés seront séparées par une ",". Si le champs n'est pas valué, le séparateur est le retour à la ligne.

Un "attributeLine" doit obligatoirement avoir un "attribute" précisé. Il peut être de 2 types :

- direct : la valeur à afficher est tirée directement du LDAP
- table : la valeur à afficher provient d'une table déclarée.
  Si paramètre wrapToAscii permet de transformer la saisie utilisateur en ascii (élimination des caractères accentués).

La classe de customMail

Dans le cas d'un champ de type "mail", vous pouvez paramétrer le type de lien. La classe "ACustomMail" a pour rôle de tester le déploiement du canal mailTo. Si le canal est déployé correctement, la classe fait appel à la méthode getUrlChoose, sinon, c'est automatiquement la méthode getUrlMailToLink qui est invoquée.

Vous devez donc créer un classe qui implémente cette classe abstraite (dans le répertoire "custom") et contenant les 3 méthodes :

getUrlChoose

C'est dans cette méthode que vous écrivez l'algo de choix entre un lien mailto ou l'appel au canal. Vous disposez alors d'un certains nombre de paramètres :

- boolean servantActive : true si le canal annuaire est déjà en mode servant
- boolean guestMode : true si le canal annuaire fonctionne en mode anonyme
- String attributeName : nom de l'attribut LDAP contenant le mail
- String idPeople : identifiant de la personne
- String mailAddress : adresse mail récupérée de LDAP
  Cette méthode doit retourner un entier : LINK_MAIL_TO pour lancer la méthode getUrlMailToLink() ou CHANNEL_MAIL_TO pour lancer getUrlMailToChannel();

getUrlMailToLink

Cette méthode retourne un lien de type mailto. Les paramètres disponibles sont :

- boolean servantActive
- boolean guestMode
- String mailAddress

getUrlMailToChannel

Cette méthode retourne un lien vers le canal mailto. Les paramètres sont :

- boolean servantActive
- boolean guestMode
- String attributeName
- String idPeople

Internationalisation

Cette partie conserne les administrateurs car elle décrit comment modifier les textes d'affichage. Celà se situe soit au niveau de la configuration (nom des champs, nom des annuaires ... etc), soit au niveau des textes normaux.

Au niveau de la configuration

Comme vous l'avez vu dans le détail du fichier de configuration, es entêtes des champs sont multilangues. Dans le fichier de configuration vous avez un lot de déclaration de variables de texte.

Ce sont ces variables que vous devez écrire dans des fichiers de propriétés (dans le répertoire properties/languages) nommés config<<langue>>.properties

Les textes "normaux"

Les textes des menus et tous les textes affiché doivent se trouvés dans le fichier de propriété correspondant à la langue en cours et doit être dans le répertoire languages à la racine du package. Les fichiers de langues s'appellent (dans le répertoire languages) CAnnuaire<<langue>>.properties

Déploiement

Préparation du fichier de configuration CAnnuaire.xlm.: mettre les renseignements relatifs aux serveurs et aux attributs importants.

Préparation du déploiement : modifier les chemins d'installation du fichier build.properties.

Si vous aviez déjà installé une ancienne version du canal, lancez la commande **ant undeploy**, pour effacer toute trace du canal dans l'environnement de production.

Lancement de la commande **ant deploy**, pour déployer le canal dans les bons répertoires.

L'administrateur peut maintenant publier les canaux dont il a besoin en fonction du paramètre d'instanciation : "serverView". Pour cela, il y a 2 méthodes :

La plus simple étant de publier le canal de manière normale, par l'interface WEB prévue à cet effet.

La seconde étant d'utiliser la commande ant pubchan.

Cette méthode remplace toute la démarche de publication.

Utilisez la commande ant deploypubchan qui va déposer le fichier de déploiment (pubchan_CAnnuaire.xml) dans le répertoire {uPortal}/properties /chanpub. Les informations nécessaires à la publication sont décrites dans des tags XML, vous permettant ainsi même de spécifier l'attribut "serverView".

Ensuite, lancez la commande "ant pubchan -Dchannel=CAnnuaire.xml". depuis le répertoire "racine esup".

Vous pouvez bien sur créer autant de fichier xml que vous désirez. Par exemple, rien n'interdit d'en avoir un pour la vue en anonyme, un pour les étudiants et un pour le personnel.

Si vous décidez de mettre en place l'utilisation du canal MailTo, vous devez déployer ce canal. Il faut au moins la version 1.0.2 du MailTo (documentation ici).