Vitefait: exemple de script de création automatique de formulaire Vitefait à partir d'un fichier existant

Il est possible à partir d'un simple script d'automatiser la création d'un formulaire Vitefait.

L'exemple qui suit permet de créer un formulaire de recherche multicritaire sur les codes FINESS (Fichier National des Etablissements Sanitaires et Sociaux) après avoir téléchargé ceux ci à partir du site officiel (http://finess.sante.gouv.fr/index.jsp)

1) télécharger les codes finess en format CSV

   allez sur le site: http://finess.sante.gouv.fr/index.jsp

   demandez « recherche libre »

   choisissez vos critères et demandez « chercher »

   choisissez « export format CSV », vous obtiendrez un fichier « export.csv »

2) allez dans l'interface de management de Vitefait sur le dossier « vitefait_prog »

   http://monserveur:monport/vitefait/vitefait_prog/manage

3) créer un objet de type « file », appelez le « finessCSV » et avec la touche « parcourir » allez y insérer votre fichier « export.csv » téléchargé

4) créez un objet de type script que vous nomme comme vous voulez (par exmple « crer_finess »). Cliquez sur « add and edit ».

5) supprimez tout le code du script Python et faites un copier-coller du code qui est sur les pages suivantes. Ciquez sur « Save Changes »

Le code source est totalement commenté, vous pourrez l'adapter à vos propres besoins.

Une fois le questionnaire généré, vous pourrez l'adapter à vos propres besoins par Vitefait.

Exemple: dans mon cas (import de tous les Finess de type établissement du nord). J'ai une question « Lib categorie » qui est la question 0200 et qui s'appelle champ20 (cela peut être différent pour vous en fonction de votre import) et je veux faire une recherche par liste déroulant sur les libellés de catégorie.

Je modifie ma question comme suit:

je coche « type liste ».

je supprime « criterContient » et j'ajoute « critereEgal » dans « tests ou traitements »

Dans « nom du formulaire de liste » je met « FINESS »

dans « critères » je met « FINESS_groupe='champ20' »(champ 20 est entre apostrophes et non guillemets)

dans « nom des questions » je met « champ20 »

je demande « valider » puis « tester » et ma modification est prise en compte, je peux désormais choisir ma catégorie par liste déroulante. Mais mon formulaire étant multicritère, je peux choisir une catégorie dans une vile...

Vous pouvez tester le formulaire décrit sur

http://www.medecinelibre.org/vitefait/vitefait_prog/FINESS_prog

Si vous avez des questions, n'hésitez pas à m'écrire sur la liste de diffusion de Vitefait.

```
## CODE SOURCE DE creer_FINESS
prg = context
val = container.REQUEST
## supprimer formulaire FINESS
## appel des fonctions existantes de ViteFait (utilisées par la fonction utilitaire de Vitefait
## les champs k_nomform et confsup sont initialisé à "FINESS"
## on appelle ensuite la fonction "détruire table" en premier puique qu'elle utilise les programmes créés par Vitefait pour le formulaire
## ensuite on appelle la fonction "supprimer" pour supprimer totalement le formulaire et tout le code associé
## dans les 2 cas les erreurs sont ignorées (cas où la table SQL et le formulaire n'existeraient pas
##
## le nom du formulaire peut être changé
##
val.set("k_nomform","FINESS")
##
val.set("utilitdem","detruire table")
val.set("confsup",val.k_nomform)
try:
   prg.executer_prog_dtml(prg,val)
except:
   pass
val.set("utilitdem","supprimer")
try:
   prg.executer_prog_dtml(prg,val)
except:
   pass
##
##
## ajouter formulaire FINESS
## l'ajout du formulaire se fait en 2 temps en utilisant les fonction de Vitefait
## Ajout du formulaire (page formulaire de Vitefait
## les fonctions SQL de Vitefait sont utilisées en direct pour remplir le dictionnaire de Vitefait
## seul les champs "nom du formulaire", "titre" et "touche lister" sont remplis
##
prg.dict.viteDict_sql_ajouter(k_nomform=val.k_nomform, \
```

```
                titre="Liste des codes FINESS", \
                voir="", \
                ajouter="", \
                modifier="", \
                supprimer="", \
                lister="X", \
                valider="", \
                rab="X", \
                formretour="", \
                fa1="", \
                fa2="", \
                fa3="", \
                fa4="", \
                fa5="", \
                fa6="", \
                fa7="", \
                fa8="", \
                fa9="", \
                faa="")
##
##
## 2ème temps: ajouter champs FINESS
## le dictionnaire des champs de Vitefait pour le formulaire Vitefait sera créé avec pour nom les noms des champs se trouvant dans la première
## ligne du fichier CSV
## la longueur des champs sera calculée avec la plus grande longueur trouvée pour chaque colonne
## les champs non remplis ne seront pas créés pour simplifier le questionnaire
##
##
## récupération du csv dans la variable fichier
## note sur le fichier FINESS
## c'est du CSV pas trop satndard:
## les champs sont séparés par des virgules
## chaque champ est précédé par égal et mis entre guillemets, sauf les champs vides ou parfois les virgules sont simplement doublées
## parfois des virgules se trouvent à l'intériueur des champs
```

```
## exemple:
## ="truc",="machin avec , vigurle à l'intérieur",,,,="autre champ"
##
##
fichier = str(prg.finessCSV.data)
## transformation de toutes les virgules doublées en y insérant ="" afin de pouvoir traiter tous les champs de manière identique
## en se basant sur la chaine =" comme délimiteur
##
while fichier.count(",,"):
    fichier = fichier.replace(',,',',="",')
##
## création des lignes à partir du fichier (chaque ligne est séparée par les caractères \r\n
##
lignes = fichier.split("\r\n")
##
## découpage de la première ligne pour avoir le nom des champs
##
chmp = lignes[0].split('="')
nomchamps = []
##
## suppression de la virgule et du caractère " dans le nom du champ
##
for v in chmp:
    nomchamps += [v.strip(",").strip('"')]
##
## calcul des longueurs maximales pour chaque champ
## la table des longueur est initialisée à 0
## ensuite tout le fichier est parcouru (à partir de la ligne 1)
## et sur chaque ligne, chaque champ est parcouru.
## on stocke ensuite dans la table, en fonction de la position du champ la plus grande longueuer entre celle du champ en cours et celle déjà stockée
##
longueurs = [0] * len(nomchamps)
##
## parcours de chaque ligne
```

```
##
for lgn in lignes[1:]:
    ##
    ## le compteur de numéro de champs est initialisé à 1 pour chaque ligne (saut du premier champ) et sera incrémenté pendant le parcours des champs
    ##
    numero = 1
    ##
    ## découpage de chaque ligne en champs et parcours des champs
    ## note: le premier champ étant systématiquement vide (à cause de l'éclatement par =" on commence à partir du 2ème champ (noté 1)
    ##
    for v in lgn.split('="')[1:]:
        ##
        ## suppression de la virgule et du caractère " dans le nom du champ
        ##
        chmp = v.strip(",").strip('"')
        ##
        ## choix pour le champ de la longueur maxi entre celle qui se trouve déjà en table et celle trouvée sur le champ en cours
        ##
        longueurs[numero] = max(len(chmp),longueurs[numero])
        ##
        ## incrémentation du numéro de champs pour pouvoir traiter le champ suivant
        ##
        numero += 1
##
## création des champs dans le dictionnaire de Vitefait
## note: le premier champ étant systématiquement vide (à cause de l'éclatement par =" on commence à partir du 2ème champ (noté 1)
## c'est pourquoi numero qui est le numéro de champs est initialisé à 1
##
numero = 1
for v in nomchamps[1:]:
    ##
    ## seuls les champs ayant remplits au moins une fois dans la table seront créés (longueur différente de 0)
```

```
   ##
   if longueurs[numero]:
      ##
      ## envoi du récapitulatif nom du champ, titre et longueur pour chaque champ créé dans
Vitefait
      ##
      print "champ"+str(numero),nomchamps[numero],longueurs[numero],"ajouté"
      ##
      ## le premier champ (numéro FINESS) sera créé comme "CleSansPrefixe" les autres en
"nonCle"
      ##
      if numero == 1:
         typcle = "CleSansPrefixe"
      else:
         typcle = "nonCle"
      ## notes: la recherche de Vitefait est limitée aux 130 premier champs,
      ## pour cela si numéro est supérieur à 130 on met "pasEnListe" dans typtest
      ## pour les autres champs on met "critereContient"
      if numero <= 130:
         typtest = "critereContient"
      else:
         typtest = "pasEnListe"
      ##
      ## ajout via la méthode SQL de Vitefait de chaque champ
      ## avec nom du formulaire = FINESS (initialisé en début de script)
      ## incrémentation 10 par 10 du numéro de champ
      ## nom du champ créé automatiquemant à champsX avec X = numéro du champ
      ## longueur calculée
      ## type: texte
      ## "CleSansPrefixe" ou "nonCle"
      ## et d'office traitement "critereContient" pour les recherches (130 premiers champs)
      ##
      prg.dict.viteDict_champ_sql_ajouter(k_nomform=val.k_nomform, \
                                          k_rang=("0000"+str(numero))[-3:]+"0", \
                                          titre=nomchamps[numero], \
                                          nomchamp="champ"+str(numero), \
```

```
                                typex="texte", \
                                longx=str(longueurs[numero]), \
                                nbl="", \
                                occursx="", \
                                clenmodntab=typcle, \
                                typtest=typtest, \
                                listtab="", \
                                listcrit="", \
                                listques="", \
                                tests="", \
                                calcecr="", \
                                calcvisu="", \
                                calclist="")
    ##
    ## incrémentation du numéro de champ
    ##
    numero += 1
##
## ajout d'une ligne spécifique de code
## par défaut, le nombre maximum de lignes sélectionnées est de 1000 dans chaque méthode SQL
## ce code après génération va permettre à la méthode sql_lister de modifier ce défaut pour ne pas avoir de limite
## ceci est fait en créant un programme "apres_generation" ayant pour code
## prg.vitefait_prog.FINESS_sql_lister.manage_advanced(0,100,0,",",)
##
prg.dict.viteDict_champ_sql_ajouter(k_nomform=val.k_nomform, \
                                k_rang="9999", \
                                titre="code", \
                                nomchamp="code", \
                                typex="texte", \
                                longx="1", \
                                nbl="", \
                                occursx="", \
                                clenmodntab="nonEnFormulaire", \
                                typtest="", \
```

```
                        listtab="", \
                        listcrit="", \
                        listques="", \
                        tests="", \
                        calcecr="", \
                        calcvisu="/**/\n#prog=apres_generation\nval.set('max_row',0)\nprg.vitefait_
prog."+val.k_nomform+"_sql_lister.manage_advanced(0,100,0,",",)", \
                        calclist="")
##
## appel du module de génération de programme de Vitefait
##
prg.viteDict_generer()
##
## chargement de la table SQL en utilisant l'objet d'ajout de FINESS généré
## parcours des lignes du fichier en sautant l'entête
##
for lgn in lignes[1:]:
   ##
   ## initialisation du numéro de champ à 1 pour sauter le premier champ systématiquement vide
   ##
   numero = 1
   ##
   ## découpage de la ligne en champ en sautant le premier champ
   ##
   for v in lgn.split('="')[1:]:
      ##
      ## suppression de la virgule et du " dans le champ
      ##
      chmp = v.strip(",").strip('"')
      ##
      ## affectation de la variable en cours
      ##
      val.set("champ"+str(numero),chmp)
      ##
      ## incrémentation du numéro de champ
```

```
      ##
      numero += 1
   ##
   ## appel de l'objet d'ajout de ligne SQL de FINESS dans Vitefait
   ##
   prg.vitefait_prog[val.k_nomform+"_sql_ajouter"]()
   ##
   ## exécution du formulaire créé
   ##
   val.RESPONSE.redirect(val.URL2+'/vitefait_prog/'+val.k_nomform.strip()+'_prog')
return printed
```