Contact : luis@itaapy.com
Revision : 1.15

1 Documentation d'administration du produit Scrib

Table des matières

1.1 Installation de Python et de Zope

L'installation s'effectue automatiquement à l'aide du script Itaapy *installZope.py*, celui-ci effectue les opérations suivantes :

```
- Décomprime les sources de Python et Zope.
- Crée un répertoire qui va contenir les Python, Zope, la ou les ins-
tances de Zope
- Compile Python, puis Zope avec le Python précédemment compilé
- Crée une instance de de Zope
- Instal dans l'instance de Zope le Produit Epoz
- Crée dans l'instance de Zope un répertoire ``Packages`` ou pourrons être dé-
posé les  paquets livré par Itaapy.
```

1.1.1 Remarques

Il faut a ce stade vérifier que le python installé dispose des modules pythons suivant :
– python-imaging -- http ://www.pythonware.com/downloads/index.htm
– python-mysqldb -- http ://sourceforge.net/projects/mysql-python
On peut voir dans l'annexe les interfaces (entrée/sortie) du script.

1.2 Le répertoire `Packages`

Le script d'installation a crée un répertoire `Packages` dans l'instance de Zope.
On y déposeras les paquets livrés par Itaapy que l'on peut décomprimer sur place :

```
$ tree -L 1 ./Packages/
./Packages/
|-- culture--main--0.2--patch-110
|-- culture--main--0.2--patch-110.tgz
|-- culture--main--0.2--patch-96
|-- culture--main--0.2--patch-96.tgz
|-- iHotfix--main--0.5--patch-1
|-- iHotfix--main--0.5--patch-1.tgz
|-- ikaaro--main--0.9--patch-104
|-- ikaaro--main--0.9--patch-104.tgz
|-- ikaaro--main--0.9--patch-128
|-- ikaaro--main--0.9--patch-128.tgz
|-- itools--main--0.5--patch-42
`-- itools--main--0.5--patch-42.tgz
```

Il ne faut pas renommer ces paquets, en effet leur nom permet de retrouver un état unique de l'arbre de développement arch.
Arch aka *tla* est le système de gestion de version distribué utilisé par Itaapy.

1.3 Installation des produits Zopes

L'installation des produits Zope se fait par l'ajout de liens symboliques dans le répertoire `Products` de l'instance de Zope :

```
$ tree -L 1  ./Products/
./Products/
|-- Epoz
|-- culture -> ../Packages/culture--main--0.2--patch-110
|-- iHotfix -> ../Packages/iHotfix--main--0.5--patch-1
`-- ikaaro -> ../Packages/ikaaro--main--0.9--patch-128
```

1.4 Installation du module Python : `itools`

Il faut appeler le script d'installation **setup.py** avec le python dans lequel on veux installer le le module :

```
$ pwd
/home/luis/Zopes/scrib/scrib.culture.fr
$ ls
bin  Extensions  inituser  Packages etc  import      log       Pro-
ducts  var
$ cd Packages/itools--main--0.5--patch-42/
$ ~/Zopes/scrib/Python-2.3.4/bin/python setup.py clean build install
```

1.5 Test et initialisation de Zope

Afin de tester que tout est en place il faut à ce stade lancer le serveur Zope :

```
$ ls
bin  Extensions  inituser  Packages  etc  import  log  Products  var
luis@petitBar :scrib2.culture.fr$ ./bin/runzope
```

Cette étape sert également à initialiser le Produit `culture`, c'est à dire a informer se produit de là ou se trouve l'interpréteur python qu'il devras utiliser. Cf. `.culture/input_data/where_is_python.txt`

1.6 Initialisation du produit `culture`

Un seul fichier de configuration : `Setup.conf` définis la configuration de l'aspect SQL et Mail de l'application :

```
## Sql configuration
SqlHost = localhost
SqlDatabase = Sscrib
SqlUser = Sscrib
SqlPasswd = SScrib-2005*

## Mail configuration
# SMTPServer=cyr2.culture.fr
# SMTPServer=smtp.wanadoo.fr
# SMTPServer=localhost
SMTPServer = smtp.wanadoo.fr
MailResponsableBM = luis@itaapy.com
MailResponsableBDP = luis@itaapy.com
```

Pour la génération et l'exécutions des scripts tout est centralisé dans le *Makefile*, il suffit donc d'exécuter :

```
$ make clean
find ./ -name "*.pyc" -exec rm -f {} \;
find ./ -name "*_autogen*" -exec rm -f {} \;
rm -f *.bak
rm -f pot po bin
```

Puis `make gen_all` qui va
- créer les script pour remplir les tables SQL
- créer les contrôles pour les BM et les BDP
- créer les formulaire XML pour les BM et les BDP

```
$ make gen_all
```

```
autogen SQL tables   ./input_data/fill_bm04_autogen.sql
autogen SQL tables   ./input_data/fill_bm05_autogen.sql
autogen SQL tables   ./input_data/fill_bm06_autogen.sql
autogen SQL tables   ./input_data/fill_bm07_autogen.sql
autogen SQL tables   ./input_data/fill_bdp04_autogen.sql
autogen SQL tables   ./input_data/fill_bdp05_autogen.sql
autogen SQL tables   ./input_data/fill_bdp06_autogen.sql
autogen SQL tables   ./input_data/fill_bdp07_autogen.sql

autogen controles   ./controlesBDP.py
autogen controles   ./controlesBM.py

autogen STL template   ui/FormBDP_report1_autogen.xml
autogen STL template   ui/FormBDP_report2_autogen.xml
autogen STL template   ui/FormBDP_report4_autogen.xml
autogen STL template   ui/FormBDP_report5_autogen.xml
autogen STL template   ui/FormBDP_report6_autogen.xml
autogen STL template   ui/FormBDP_report7_autogen.xml
autogen STL template   ui/FormBDP_report9_autogen.xml

autogen STL template   ui/FormBM_report1_autogen.xml
autogen STL template   ui/FormBM_report2_autogen.xml
autogen STL template   ui/FormBM_report4_autogen.xml
autogen STL template   ui/FormBM_report5_autogen.xml
autogen STL template   ui/FormBM_report6_autogen.xml
autogen STL template   ui/FormBM_report8_autogen.xml
autogen STL template   ui/FormBM_report9_autogen.xml
```

Puis `make create_database` qui va créer la base **scrib** :

```
$ make create_database
Do you want to re/create SQL scrib database? : (y/N) y
Enter password :
./input_data/create_scrib.SQL OK
```

Enfin `make fill_tables` qui va remplir les tables des BM et des BDP :

```
$ make fill_tables
Do you want fill SQL tables? : (y/N) y
./input_data/fill_adresse.sql OK
./input_data/fill_bdp04.sql OK
./input_data/fill_bm04_autogen.sql OK
./input_data/fill_bm05_autogen.sql OK
./input_data/fill_bm06_autogen.sql OK
./input_data/fill_bm07_autogen.sql OK
./input_data/fill_bdp04_autogen.sql OK
./input_data/fill_bdp05_autogen.sql OK
./input_data/fill_bdp06_autogen.sql OK
./input_data/fill_bdp07_autogen.sql OK
luis@potlatch :culture$
```

1.7 Annexe A : script d'installation installZope.py

Execution du script installZope.py :

```
$ python installZope.py
```

```
download_path : [/home/luis/Zopes/downloads]
root_path : [/home/luis/Zopes]
project_name : [scrib]
zope_instance_name : [scrib.culture.fr] scrib.culture.fr
zope_port_mask : [6000]
admin_passwd : [rinLopjen5]


zope_name Zope-2.7.3-0
for python use /home/luis/Zopes/scrib/Python-2.3.4/bin/python
make zope_instance ok
Folder


Packages created
untar and install Epoz-0.8.6.tar.gz Zope Product
tar -xzf /home/luis/Zopes/downloads/Epoz-0.8.6.tar.gz
  - Total time : 0 min 2 sec
**********************************************************************
*    download_path = /home/luis/Zopes/downloads                      *
*    python_bin = /home/luis/Zopes/scrib/Python-2.3.4/bin/python     *
*    zope_instance_path = /home/luis/Zopes/scrib/scrib.culture.fr    *
*    ftpPort = 6021                                                  *
*    httpPort = 6080                                                 *
**********************************************************************

  - Run zope on port 6080 with
    /home/luis/Zopes/scrib/scrib.culture.fr/bin/runzope
```

Le répertoire du projet `scrib` a été crée, il contient un Python, un Zope et une instance de Zope nommée `scrib.culture.fr` :

```
$ ls -la scrib/
Python-2.3.4/  scrib.culture.fr/  Zope-2.7.3-0/
```

L'instance de Zope contient en plus des répertoires habituel le répertoire `Packages` :

```
$ tree -d -L 1 scrib.culture.fr/
scrib.culture.fr/
|-- Extensions
|-- Packages
|-- Products
|-- bin
|-- etc
|-- import
|-- log
‘-- var
```

1.8 Annexe B : Organisation du Code

Voici comment s'organise le code au sain du produit `culture` :
A la racine ce trouve le Makefile et le code python, la **logique** de l'application.
L'arborescence comprend à la racine de produit culture le code python :

```
$ tree -L 1
  .
```

```
|-- __init__.py
|-- Culture.py
|-- CultureTypes.py
|-- Folder.py
|-- Forms.py
|-- User.py
|-- Form.py
|-- FormBDP.py
|-- FormBM.py
|-- controlesBDP.py
|-- controlesBM.py
|-- schemaBDP.py
|-- schemaBM.py
|-- utils.py
|-- setup.py
|-- gen_all.py

|-- Changelog
|-- Makefile
|-- README
|-- refresh.txt
|-- Setup.conf
|-- TODO

|-- gen
|   |-- ...

|-- input_data
|   |-- ...
|
`-- ui
    |-- ...
```

Le répertoire **gen** comprend le code servant à générer automatiquement une partie des formulaire et des contrôles. Le répertoire **input_data** contient l'information servant à initialiser l'application, créations des bibliothèques municipales et départementales de prêt, ainsi que les scripts de création et de mise à jour des bases et des tables MySQL.

```
$ tree  -L 1 gen input_data ui
gen
|-- FillSql.py
|-- GenFormsTemplates.py
|-- __init__.py
|-- gen_BDPreports.py
|-- gen_BMreports.py
|-- gen_controles.py
`-- gen_sql.py

input_data
|-- create_scrib.SQL
|-- fill_adresse.sql
|-- fill_bdp04.sql
|-- fill_bm04.sql_orig
|-- init_BDP.txt
```

```
  |-- init_BM.txt
  `-- where_is_python.txt
```

Enfin le répertoire **ui** comprend les interfaces, le code XML et XHTML servant à la présentation des interfaces, les fichier se terminant par autogen ont été généré automatiquement à l'aide de la commande `make gen_all`.

```
ui
|-- Folder_browse_list.xml.en
|-- FormBDP_help.xml
|-- FormBDP_help1.xml
|-- ...
|-- FormBDP_report0.xml
|-- ...
|-- FormBM_help.xml
|-- FormBM_help1.xml
|-- ..
|-- FormBM_report9.xml
|-- ..
|-- Form_controles.xml
|-- Form_controlesInfo.xml
|-- Form_fill_report.xml
|-- Form_report_csv.xml
|-- Forms_browse_list.xml
|-- Forms_search.xml
|-- User_home.xml
|-- bibUserFolder_search.xml
|-- debug.xml
|-- form.css
|-- form.js
|-- images
|-- main.css
|-- onglet.css
|-- printable_template.xhtml
`-- template.xhtml
```