ZOPE & PLONE.

Thierry Dumont [a]

Maply CNRS & Université Lyon 1

[a] GERONTE:

- Que diable allait-il faire dans cette galère ?

Zope.

But : *séparer contenu, logique et présentation.*

- base de données objet (vision hiérarchique, pas relationnelle).
- langage de script (python).
- patrons (templates) de pages web :
  - DTML
  - Page templates.
- outil de recherche intégré.

technologie

- Zope possède son serveur (communique avec Apache).
- base de donnée ZDB.
- écrit en Python et en C
- installation aisée (tar.gz)

gestion

- Administration par le web (la ZMI).
- utilisateurs avec droits.
- “undo” sophistiqué.

les objets Zope

- Objets typés ! Exemple : Page Templates, DTML, “Folders”, scripts... Plone !
- Folder : objet de base, pour construire une hiérarchie.
- Accessibilité : par URL.

  Exemple : Folder A à la racine ; B inclus dans A.

  `http ://<zope>/A/B`
- Héritage : dans B on cherche un objet... s’il n’existe pas, on le cherche dans A et ainsi de suite...
- possibilité d’ajouter ses propres objets.
- les propriétés des objets sont accessibles.

DTML : exemples

DTML

- exemple 1 (très simple !) :

```
<dtml-var standard_html_header>
  <h1> exemple tres simple </h1>
<dtml-var standard_html_footer>
```

- exemple 2 (boucle) :

```
<ul>
<dtml-in objectValues>
<li> <dtml-var getId></li>
</dtml-in>
</ul>
```

- DTML *se débrouille* :

```
<dtml-var doIt>
```

où `doIt` est un DTML, un script Python e.t.c...

- plus explicite (appel de script Python avec paramètres) :

```
<dtml-var expr=''doIt('withCare')''>
```

- conditionnels :

```
<dtml-if expr=''x < y''>
 x est plus petit que y.
</dtml-if>
```

- e.t.c..

Principal inconvénient : **le dtml n'est pas du html !**

$=>$ on ne peut pas déléguer facilement la *présentation*...

d'où les **Page Templates**.

Les Pages Template

Avantage : le patron est visible par un navigateur.

tal (template attribute language).

Exemples :

- `<title tal :content=”here/title”> Titre de la page</title>`
- ```
  <tr tal:repeat=
    ’’item container/ObjectValues’’>
  <td tal:content=’’item /getId’’> Id</td>
  <td tal:content=’’item/title’’> Titre</td>
  </tr>
  ```
- appel de scripts Python :

```

     bonjour 
```
```

Plone.

Un sytème de gestion de contenu.

- Zope : le système d'exploitation avec son file–system (la base de données objets)
- Plone : un applicatif construit au-dessus de Zope.

- utilisateurs : “login”, mot de passe, “répertoire” personnel.
- rôles : un utilisateur peut être membre, reviewer, administrateur ou anonyme.
- délégation (rôles locaux).
- acl’s.
- workflow.

gestion : uniquemement par interface web.

Le workflow

Pour chaque objet créé, plusieurs états :

1. privé (accessible uniquement au créateur).
2. visible (si on connait son url).
3. en attente (de modération),
4. public.

- un utilisateur “membre” peut gérer $1 -> 2 -> 3$ et $4 -> 1$,
- le passage $3 -> 4$: “reviewer” ou “administrateur”.

si l’auteur modifie l’objet : retour à l’état 1.

Méta–données. Le parcours d'un objet

Méta–données :

Pour chaque objet, on peut associer une liste de méta–données (= mots clés).

Parcours :

1. Un "membre" dépose un objet dans **son** répertoire et le modifie... puis demande la publication (état 3).
2. Un "modérateur" fait (ou non) passer l'objet à l'état "public".

Un exemple : le web Calcul

- Arborescence de répertoires, mais information pas hiérarchique !
- Solution : mots clés associés a chaque objet.
- Chaque feuille de l'arborescence est associée à un (des) mots clés.

L'entrée dans une feuille provoque l'éxécution d'un script Python qui extrait les objets correspondants aux mots clés du répertoire.

Avantages :

1. pas de duplication des données ;
2. localisation facile des données $=>$ ménage facile !

Remarques & Conclusion

Projets (Plone) :

- web de la fed. lyonnaise de calcul scientifique ;
- web des éditions des oeuvres de Dalembert. `http://dalembert.univ-lyon1.fr`

Avantages / Inconvénients :

- il faut connaître un peu Zope pour Plone.
- investissement non négligeable.
- Zope : gain ?

Appendice

Peut–on ignorer Python ?

- orienté objet (classes, héritage, opérateurs..) ; exceptions.
- très facile à apprendre.
- extensions simples à écrire

Applications :

- perleries,
- “glue” de codes de calcul, scripts (cf. code–aster)
- packages graphiques, numériques...
- beaux applicatifs (exemple : mayavi).

Bibliographie

* Site web de Zope : `http://www.zope.org`
* Site web de Plone : `http://www.plone.org`
* Site web de Python : `http://www.python.org`