Zope

Gaël LE MIGNOT – Pilot Systems

Mars 2007

Plan

Introduction

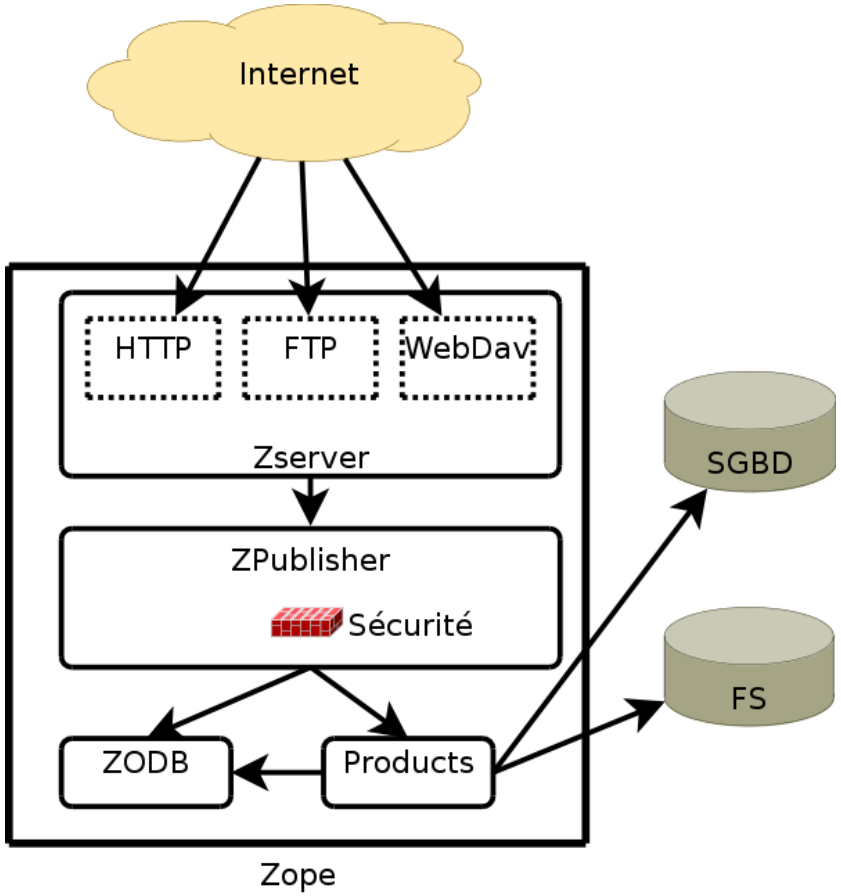
Généralités

- Serveur d'application
- Écrit en Python à 95%, 5% en C
- Multiplateforme
- Administrable via le Web (ZMI)

Zope 2 et Zope 3

- Zope 2 : actuellement utilisé
- Zope 3 : très différent, non compatible
- Plan de migration : Five

Architecture générale



Publication d'objets

Principes

- Faire correspondre une arboresence et un schéma d'addressage (type URL)
- Appel de méthodes ou d'attributs spéciaux
- Permet de raisonner purement en objet
- Donne des URLs lisibles

Attention

- Risque d'exposer involontairement des informations
- Pour accéder au protocole (redirections, . . .) on a `REQUEST`

Points clés (1)

Fonctionnalités standards

- Multi-protocoles
- Base de données objet
- Gestion de la sécurité
- Gestion des transactions
- Gestion du clustering (ZEO)
- Fonctionne par instances

Points clés (2)

Extensions

- De très nombreux produits d'extensions
- Gestion des bases de données relationnelles
- Connexion à des annuaires LDAP

Utilisable avec Apache et Squid

- ZServer parler HTTP, pas HTTPS
- Apache pour la gestion de l'HTTPS
- Squid ou équivalent pour la gestion du cache

Base de données objet

Généralités

- Stoque une hierarchie d'objets
- Transparent pour l'utilisation en Python
- Limitations : pas d'objets systèmes

Stockage

- Par défaut, dans un fichier Data.fs
- Méthode `pickle` de Python
- Possibilité d'autres stockages
- Possibilité de monter plusieurs ZODB

Gestion des transactions

Principes

- Maintenir la cohérence
- Revenir à un état stable en cas d'erreur
- Possibilité d'annuler les transactions

Précautions

- Attention aux `ConflictError`
- Ne **jamais** intercepter `ConflictError`
- Nécessité de *commit* partiels pour des grosses opérations
- Nécessité de gérer les ressources externes (mails, . . .)

Contrôle de la persistance

Attributs non persistant

- Utilité : cache, objets systèmes, ...
- Utilisation : `_v_` (`_v_socket`, ...)

Persistance des sous-objets

- Sous-objets persistant conversés
- Attention aux types standards (listes, dictionnaires)
- Solutions : `PersistentMapping`, `PersistentList`, `OOBTree`
- Autre solution : réaffecter l'attribut

Exemple de persistance d'une liste

```
class Brouette(Persistent):
    def __init__(self):
        self.inside = []

    def add(self, obj):
        self.inside.append(obj)

    def add(self, obj):
        inside = self.inside
        inside.append(obj)
        self.inside = inside
```

Principes de l'acquisition

Généralités

- Permet de retrouver un objet ou un attribut
- Exemple théorique : l'attribut "position"
- Utilisation : scripts, templates, outils

Fonctionnement

- L'attribut est cherché sur l'objet lui-même
- Sinon, sur son parent dans le *contexte d'acquisition*
- Sinon, sur le parent du parent
- Et ainsi de suite

Acquisition avancée

Attributs et méthodes utiles

- `aq_base`, `aq_inner`, `aq_parent`
- `__of__`
- `getParentNode`

L'acquisition hors de Zope

- `Acquisition` un module Python
- Acquisition explicite et implicite

Concepts de la sécurité dans Zope

Principaux concepts

- Utilisateurs
- Rôles
- Droits

Concepts avancés

- Rôles locaux
- Acquisition des droits

Environement d'exécution restreint

Qu'est-ce qui est restreint ?

- Tout ce qui vient du Web (ZPT, Scripts)
- Les mêmes types d'objets en File System

Qu'est-ce qui n'est pas restreint ?

- Les produits
- Les `External Method`
- L'exception `Unauthorized`
- Les méthodes : `has_permission`, `getRolesInContext`

Effets d'un environnement restreint

Contrôles globaux

- Utilisation de `import` limitée
- Aucun accès direct au système
- `print` redéfini

Contrôles locaux

- Le contrôle se fait sur l'opérateur « . »
- Les droits sont définis dans le code du produit
- Pour les objets Web, les droits sont standards

Principes du catalogue Zope

Utilité

- Permet une recherche rapide sur des critères
- Évite de réveiller les objets
- Supporte des plugins, comme `TextIndexNG`

Concepts

- Principes des Index et Metadata
- Objets *catalog-aware*
- Les `brains`
- La méthode `getObject`

Utilisation du catalogue Zope

Recherche simple

- Appel du catalogue
- Passage de paramètres Python standard
- Exemple : `catalog(title="kangourou", creator="kilobug")`
- Listes ou tuples pour effectuer des `ou`

Recherche avancée

- Possibilité de définir des `range`
- Critères dépendant des index, par exemple `depth`

Principes des Zope Page Template

Atouts

- Namespaces XMLs standards
- Attributs de balises normales
- Support des macros
- Traduction via `i18n`

Concepts

- Namespaces `tal` et `metal`
- Expressions `path`, `string` et `python`

Utilisation des Zope Page Template

Attributs simples

- `tal:content`, `tal:replace`
- `tal:condition`
- `tal:define`
- `tal:attributes`

Attributes avancés

- `tal:repeat`, variable `repeat`
- `structure`
- `omit-tag`

Ordre d'interprétation

1. `define`
2. `condition`
3. `repeat`
4. `content / replace`
5. `attributes`
6. `omit-tag`

Conclusion

Conclusion

- Zope serveur d'application très puissant
- Attention aux performances du code restreint
- Manque de documentation, mais code clair

Ressources

- http://www.zope.org
- http://www.pilotsystems.net
- Mailing lists, IRC, ...