

Titre: Mise en œuvre de mod_WebObjects

Version: 1.1

Dernière modification: 2009/07/02 19:00

Auteur: Aurélien Minet <aurelien dot minet at cocktail dot org>

Statut: version finale

Licence: Creative Commons - by-nc-sa 2.0

(<http://creativecommons.org/licenses/by-nc-sa/2.0/fr/>)

Remarques:

- les pré-requis sont de savoir installer et configurer Apache, avoir des notions de compilation en C, de l'outil make.
- le document est "orienté" Apache sous Linux mais la procédure d'installation reste applicable à d'autres Unix
- rien dans ce document n'est supporté par Apple
- ce document dans sa globalité n'est pas spécifique Cocktail
- au niveau de la typographie: le code, les lignes de commandes, le contenu des fichiers sont en italique

Mise en œuvre de mod_WebObjects



Objectif:

- Comprendre le fonctionnement de mod_WebObjects
- Installer mod_WebObjects
- Configurer mod_WebObjects

Introduction:

Apple livrait avec WebObjects 5.2 les sources de mod_WebObjects sous "APPLE PUBLIC SOURCE LICENSE" en différentes versions: IIS, Apache 1.3 et CGI. La Licence permet la modification, c'est pourquoi le support d'Apache 2 et 2.2 a été rajouté par le Project Wonder en plus de divers corrections.

Avec WebObjects 5.4 Apple a livré une version reprenant les corrections et supportant officiellement apache 2 et 2.2 mais ne supportant plus IIS (NSAPI).

L'installation peut se baser sur la version XCode 3.0, qui est compatible (niveau des URL) avec les anciennes versions de WebObjects, ou sur la version Wonder qui est une version patchée de la version XCode 3.0 comprenant des corrections de bugs notamment pour x86_64.

1. Rôle et fonctionnement

mod_WebObjects peut aussi être appelé Adaptor (ou encore HTTP Adaptor) étant donné qu'il reçoit les requêtes http qu'il transmet à l'application afin d'obtenir une réponse qu'il renvoie au client. Et que cette dernière à un WOAdaptor qui par défaut est le WODefaultAdaptor, il fournit le support http à l'application c'est grâce à lui que l'on peut se connecter directement à l'application via son port via un navigateur web (il existe d'autres WOAdaptors).

Ainsi mod_WebObjects fait interface entre les clients et les applications, il transforme donc le serveur http en proxy. Mais pas seulement car il peut découvrir applications avec leurs instances, c'est pourquoi le proxy est dynamique et qu'il fait load-balancer lorsqu'il y a différentes instances pour une même application.

Pour cela il fonctionne avec 2 listes:

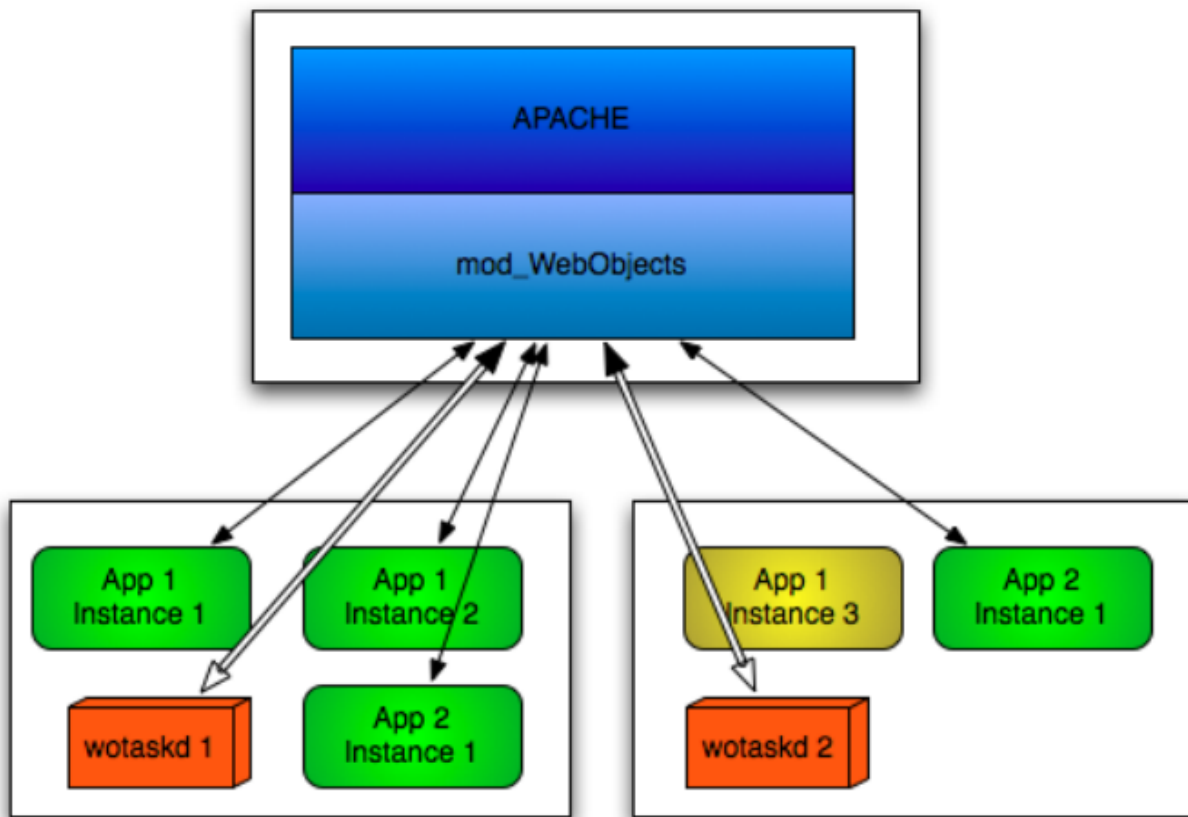
- une liste de serveurs d'applications
- une liste d'instances disponibles pour chacun des serveurs de la liste précédente.

Ces 2 listes peuvent d'être dynamiques ou statiques, on obtient 3 configurations possibles:

- multicast: découverte des serveurs dynamique (liste des instances forcément dynamique)
- multihosts: la liste de serveurs fixe (n wotaskd de déclarés) mais la liste des instances est dynamique
- statique: les 2 listes sont fixes (instances disponibles ou non)

La configuration multihosts est recommandée. Dans cette configuration mod_WebObjects va donc régulièrement contacter chaque wotaskd pour obtenir la liste des instances disponibles.

Exemple:



Sur le site, il y a 2 deux serveurs, mod_WebObjects contacte les wotaskds de chaque serveur et obtient un liste de 4 instances correspondant à 2 applications:

- App1 a 4 instances: 3 sont démarrées sur le premier serveur, la 4ième est déclarée mais non démarrée sur le deuxième serveur.
- App2 a 1 seule instance et elle est sur le deuxième serveur.

Exemple de ce que connaît au final mod_WebObjects si toutes les instances était démarrées dans l'exemple précédent.

```
<?xml version="1.0" encoding="ASCII"?>
<adaptor>
  <application name="app1" retries="5" scheduler="ROUNDROBIN" urlVersion="4">
    <instance id="1" port="2001" host="appserver1.mydomain.com"
sendTimeout="3" recvTimeout="10"/>
    <instance id="2" port="2002" host="appserver1.mydomain.com"/>
    <instance id="3" port="2003" host="appserver1.mydomain.com"/>
    <instance id="4" port="2001" host="appserver2.mydomain.com"/>
  </application>
  <application name="app2" retries="1" scheduler="RANDOM" urlVersion="4">
    <instance id="1" port="2002" host="appserver2.mydomain.com"/>
  </application>
</adaptor>
```

Par instance il peut y avoir des paramètres spécifiques. Ce fichier XML correspond à la configuration statique.

Apache avec mod_Webobjects peut être sur l'un des serveurs WebObjects ou sur un autre serveur. Il peut même y avoir plusieurs serveur pour Apache avec mod_WebObjects, chacun contactant les wotaskds des serveurs d'applications indépendamment.

2. Installation

pré-requis:

- Apache (et devel pour avoir les headers)
- APR (Apache Portable Runtime) et APR-utils (peut être inclu dans Apache)
- gcc >= 3.4
- make >= 3.8
- glibc >= 2.2

La version à installer est donc celle issues d'Xcode 3.0 (/Developer/Examples/WebObjects/Sources) ou l'archive woadaptor-Xcode_3.0.tgz qui est légèrement modifiée au niveau des Makefile pour compiler sur UNIX:

à la place de:

```
STRIP_FLAGS = -S  
$(CC) $(CFLAGS) -c -macosx_version_min 10.5 $<
```

il y a:

```
#STRIP_FLAGS = -S  
$(CC) $(CFLAGS) -c $<
```

Aussi le fonctionnement de la "configuration" est toujours basé sur un make.config et non un ./configure, le make.config de l'archive est simplifié: il ne reste qu'à adapter les chemins :

```
ADAPTOR_OS = Linux  
OS=${ADAPTOR_OS}  
ADAPTORS = Apache2.2  
APXS = /usr/sbin/apxs2  
APXS2.2 = /usr/sbin/apxs2  
APACHEINCLUDE_DIR = /usr/include/apache2  
APACHEMODULE_DIR = /usr/lib/apache2/modules  
CC=gcc  
RC_CFLAGS = -arch i386  
CFLAGS += -I/usr/local/include  
LDFLAGS += -L/usr/local/lib
```

(les CFLAGS peuvent être adaptés à votre environnement, ex: "-fpic -Fpic" dans le cas de l'utilisation d'un GCC Hardened et d'un Kernel avec PaX)

La modification du fichier make.config faite il faut lancer la compilation avec *make*, il y a quelques warning dans la fonction xmlTokenizerNextToken ne sont pas problématique. Aussi la compilation est "gérée" par apxs donc ne pas tenir compte des warning de make. S'il n'y a pas eu de problème mod_WebObjects.so est dans le sous-répertoire correspondant à la version souhaitée. Il faut le copier dans dans /usr/lib/apache2/modules (ou plus exactement dans celui contenant les modules d'apache, en cas de doute faire 'apxs -q LIBEXECDIR' pour le déterminer).

3. Configuration

Dans le sous-répertoire correspondant à la version d'Apache choisie copier le fichier `apache.conf` dans le sous répertoire d'Apache contenant la configuration des modules (`/etc/apache2/modules.d/` ou `/etc/httpd/conf.d ?`), le nommer `webobjects.conf`.

Ensuite il faut éditer `webobjects.conf` :

- commenter la ligne `"LoadModule WebObjects_module SYSTEM_LIBRARY_DIR/WebObjects/Adaptors/Apache/mod_WebObjects.so"`
- corriger la valeur pour `WebObjectsDocumentRoot` même si l'adaptor n'utilise pas cette valeur.
- choisir le mode de fonctionnement en changeant la valeur de `WebObjectsConfig`, il est recommandé d'utiliser le `multihost` qui a la forme:
`WebObjectsConfig http://<name-of-a-host>:<port-on-a-host>,http://<name-of-another-host>:<port-on-a-host> <interval>`
- activer la page d'information <http://webserver/cgi-bin/WebObjects/WOAdaptorInfo?user+password> en décommentant/changeant `WebObjectsAdminUsername` et `WebObjectsAdminPassword`
- décommenter `WebObjectsLog` afin de pouvoir activer les logs lorsque `/tmp/logWebObjects` existe
- on peut ajouter: `WebObjectsOptions redirect=http://.../erreur.html` pour personnaliser la page d'erreur lorsque l'application demandée n'existe pas ou qu'elle n'a pas d'instance démarrée.

Puis il faut modifier la configuration d'apache (`httpd.conf ?`):

- il faut charger le module `WebObjects` donc dans la section où il y a les `LoadModule` ajouter:

```
LoadModule WebObjects_module modules/mod_WebObjects.so
```

- si vous utilisez `mod_Rewrite`, les `LoadModule` doivent être dans cette ordre:

```
LoadModule WebObjects_module modules/mod_WebObjects.so
LoadModule rewrite_module modules/mod_rewrite.so
```

- il faut commenter la ligne `ScriptAlias /cgi-bin/ "/path/to/cgi/dir..."` si elle existe.
- il faut changer les droits d'accès aux répertoires:
par défaut il y a

```
<Directory />
    Order deny,allow
    Deny from all
</Directory>
```

(ce qui est très sécuritaire)

- donc il faut inverser les droits

```
<Directory />  
    Order allow,deny  
    allow from all  
</Directory>
```

(ce n'est pas vraiment l'idéal)

ou ajouter:

```
<LocationMatch /cgi-bin/WebObjects/.*>  
    Order allow,deny  
    Allow from all  
</LocationMatch>  
  
<Location /WebObjects>  
    Order allow,deny  
    Allow from All  
</Location>
```

(ce qui mieux niveau sécurité)

- enfin il faut s'assurer que webobjects.conf est bien chargé par une directive Include (Include /.../webobjects.conf ou Include /etc/apache2/modules.d/*.conf)

Il ne reste plus qu'à vérifier que l'Adaptor est bien chargé après avoir redémarré Apache

- vérifier que <http://webserver/cgi-bin/WebObjects/WOAdaptorInfo?user+password> fonctionne et indique la bonne configuration.
- une URL du type <http://webserver/cgi-bin/WebObjects/AppQuiNExistePas> doit afficher "The requested application was not found on this server." (ou la page d'erreur maison) si le mod_WebObjects tourne et s'il est accessible.
- après `touch /tmp/logWebObjects` le fichier `/tmp/WebObjects.log` doit être créé par mod_WebObjects ce qui indique qu'il est chargé. Le contenu peut aider à l'analyse de certains problèmes.

Aussi il ne faut pas oublier de copier les WebServerResources dans `/.../htdocs/WebObjects`, signer les jar dans `/.../htdocs/WebObjects/Java ...`

Conclusion

La dernière version de mod_WebObjects est plus simple à compiler, la configuration est bête et méchante: il faut respecter certains points précis. Il y a encore pas mal de limitations (pas de gestion virtual host ...) mais ce module est critique pour le déploiement et la disponibilité des applications, prendre un peu de temps pour l'installer est nécessaire. Aussi il est fortement recommandé de lire la documentation officielle d'Apple sur l'Adaptor dans le [WebObjects Deployment Guide](#) .