OBA onafhankelijk van Gravatar

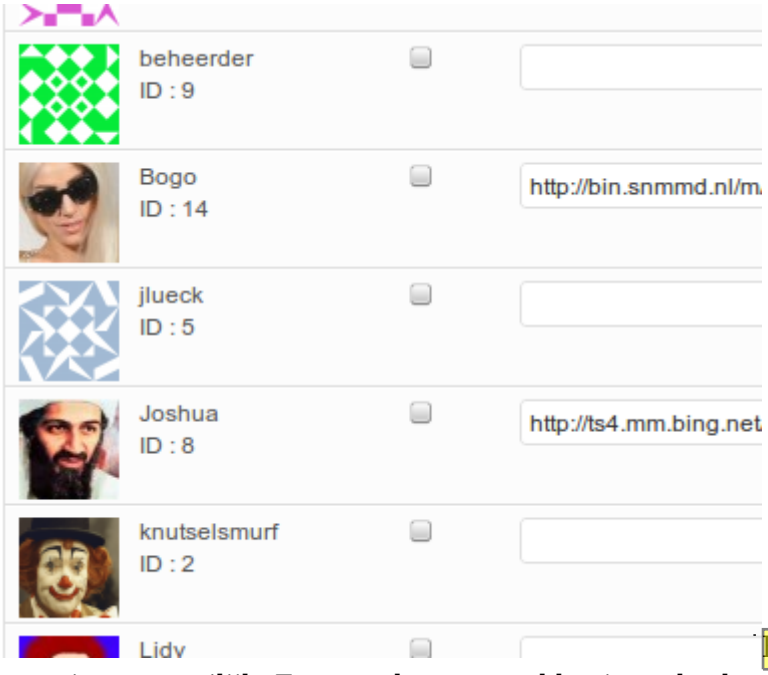
We heten ten slotte niet voor niets “onafhankelijke bloggers”
Gravatar is heel geschikt om iedereen die ergens reageert met zijn e-mail adres als identificatie een gezicht te geven. Het is ook helemaal geïntegreerd in wordpress software, dus ook al heb je je blog zelf gehost op je eigen domein, dan zit je nog aan de eigenaars van wordpress.com vast.

Blogspot wil daar natuurlijk niets mee te maken hebben, dus veel blogspotters hebben buiten blogspot geen gezicht. Mensen die lid worden van OBA, moeten ons een e-mail adres vertellen.

Of het nu is, omdat mensen ons niet vertrouwen en dus een fake-email adres opgeven, of ze onhandig zijn met computers of gewoon te lui om even een plaatje te uploaden, veel leden hebben geen avatar.
En dan krijgen ze default een monstertje.
Dat ziet er niet aantrekkelijk uit op de voorpagina.

In het verleden heb ik OBA al een keer voorzien van een cache, zodat avatars voortaan op onze eigen server gehost worden. Dat heb ik gedaan omdat een na-aper alle gravatars geoogst had zodat het leek alsof hij heel veel leden had. Hot-linken naar onze eigen avatars staan we niet toe.
Ik heb het systeem nu uitgebreid : De bron van de plaatjes hoeft niet meer “gravatar.com” te zijn. Als we maar een URL van een vierkant plaatje hebben, dan komt het helemaal goed. Het plaatje wordt opgeslagen op de server, en telkens als de wordpress software een gravatar probeert te laden, grijpt onze plugin in en maakt er een OBA-avatar van.
gravatar.com levert plaatjes in elk formaat.
Dat kan onze plugin ook : de grafische engine van PHP maakt bliksemsnel verkleinde versies, die vervolgens opgeborgen worden in de cache.

Het systeem werkt nog niet voor comments. Van een commenter weet je alleen het e-mail adres, hij hoeft geen lid te zijn van OBA. Zonder zijn user-id heeft het systeem geen idee wat zijn avatar zou zijn, dat moeten we nog aan gravatar.com vragen.
Het is wel zo dat ook die avatars afgeschermd zijn voor na-apers. Uit de url van onze avatars kan geen gravatar-url worden afgeleid, omdat we het e-mail adres veranderen voordat de md5 hash berekend wordt (gravatar en wordpress gebruiken md5($author-email), dus als je die hash-waarde weet kun je gravatars in alle afmetingen opvragen.



Het aanpassen van de “obagrcache” plugin was niet zo moeilijk. Er zat echter een addertje onder het gras. OBA zit achter een reverse caching proxy, en ook als je alle 10 de versies van een avatar vervangt door nieuwe, blijven de oude avatars nog heel lang in de proxy cache (en in de browser-cache).
Vanuit de achterliggende server de proxy-cache kunnen bewerken, dat zou toch wel heel mooi zijn. Normaal moet ik dat met de hand doen, en het is nogal een gepruts. Nginx gebruikt de url om een dir/subdir/filenaam te berekenen. Dat moet je nabootsen en dan moet je dat file-tje weggooien. Nginx ziet dan dat het filetje er niet meer is, en hij haalt het opnieuw op bij de achterliggende server.
Tot nu toe moest ik via ssh een terminalsessie starten op de host, en dan met de hand een voor een die filenamen berekenen en weggooien.
Een webinterface is er niet, want op onze host draait alleen een proxyserver zonder PHP. Een webserver met PHP pakt meteen 100 Mb, en die hebben we niet.

Ik wilde een script maken dat ik als daemon kon laten starten op het moment dat ik het nodig had. Na wat lezen en experimenteren, kwam ik tot de conclusie dat Python me ging helpen. Python is een programmeertaal die ongeveer hetzelfde kan als PHP. Maar Python is bedacht als scripttaal voor beheertoepassingen, terwijl PHP bedacht is als manier om HTML op te leuken. Het toeval wil, dat de 2 steeds meer naar elkaar toe groeien.
De filosofie van Python is dat programmeurs er snel mee moeten kunnen werken, de computer sneller laten werken kan later.
De waarschijnlijke reden dat Python en PHP in dezelfde vijver vissen is, dat Python door Google intern ontwikkeld is om iets te hebben dat de concurrentie niet had. Het bestond al lang voor PHP.
De reden dat ik het gebruik, is dat Ubuntu aan elkaar hangt van Python. Omdat onze host draait onder Ubuntu Linux, kan ik daar gratis beschikken over de krachtigste scripttaal die er is. Het bleek heel simpel om een script te maken dat via cgi door een webserver te starten is. CGI is het oudste protocol om actieve content voor het web te maken. Bijna elke webserver ondersteunt het. Nginx niet, maar we hebben nog een tweede webserver, eentje die het altijd doet en bijna geen geheugen gebruikt : Mathopd. (je kan meerdere webservers op 1 host draaien, als je ze maar allemaal een ander poortnummer geeft)
Het kost een paar regels code om Python te vertellen dat de webserver de plaats van het scherm en toetsenbord inneemt, en het kost ook maar een paar regels om Mathopd te vertellen dat Python alle files afhandelt die eindigen op .py en dat alleen ik bij die files mag.
En zo kun je dus complete menu’s maken om veelvoorkomende beheertaken makkelijker te maken.
De eerste taak die ik aldus geautomatiseerd heb was dus : nginx cache opschonen.

Een formulier invullen op internet is natuurlijk niets anders dan een url maken : http:/domein.nl/cgi-bin/script.py?veld1=waarde1&veld2=waarde2…., dus dat kan een wordpress-plugin ook.
CGI scripts zijn traag, want de interpreter moet eerst starten, maar op deze manier kost het beheer nauwelijks geheugen.

Dat ik het hele Pythonscript, waarvoor ik informatie uit het laatste hoofdstuk van het manual nodig had, geschreven heb op dezelfde dag waarop ik Python geleerd heb, wil misschien iets zeggen over mij, maar ook over het gemak waarmee deze taal aan te leren is voor een programmeur. Een programmeur heeft meestal een bepaalde logische manier van denken, en als er een taal is die daar automatisch bij aansluit, zonder dat je allerlei uitzonderingen uit je hoofd hoeft te leren, dan kan een programmeur die taal extreem snel leren. Zo'n taal is Python. (CSS ga ik bijvoorbeeld nooit leren, je blijft gokken en proberen)