

**MiTTWALD**

Webhosting. Einfach intelligent.

# TYPO3 & Varnish

Konfiguration optimieren & Fehler vermeiden

[www.mittwald.de](http://www.mittwald.de)

# Oliver Thiele

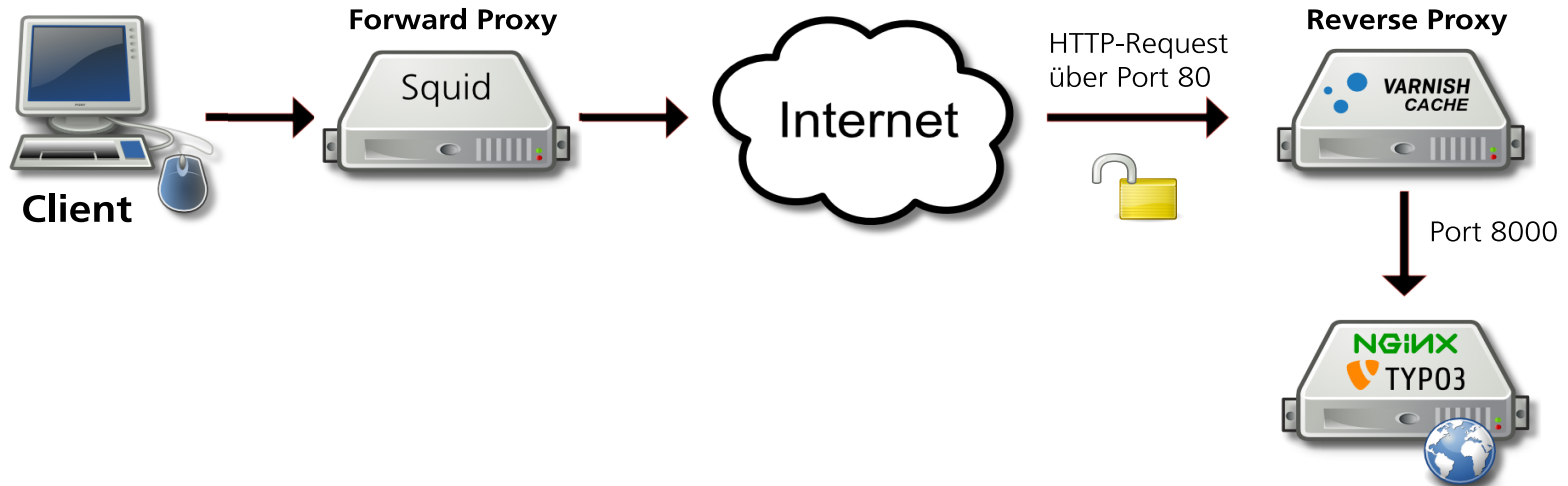
Dozent bei Mittwald CM Service



- Alter: 42 Jahre
- Wohnort: Lübbecke
- Certified TYPO3 Integrator
- Re-Zertifiziert 2015
- Seit 2003 als Dozent für TYPO3-Schulungen festangestellt

# Forward & Reverse Proxy

Wo wird etwas gecached?



# Reverse Proxy

Warum sollten wir einen Reverse Proxy nutzen?

- Seiten werden sehr viel schneller ausgeliefert
- Weniger Last auf dem Webserver
- Höhere Ausfall-Sicherheit
- Besseres Ranking bei Google durch das schnellere Ausliefern der Webseiten

# Probleme durch Caching?

Häufige Probleme sind:

- Es werden für alle Besucher zu alte Inhalte ausgeliefert.
- HTTPS-Verbindungen, bei denen die SSL-Verschlüsselung bis zum Webserver reicht, können nicht zwischengespeichert werden.
- Inhalte, die je nach eingeloggter Person unterschiedlich sind, dürfen nicht gecacht werden (außer, es gibt für jeden Login einen separaten Cache).
- Bei falscher Konfiguration könnten Personen Inhalte sehen, die nicht für sie gedacht sind.
- Redakteure in TYPO3 könnten ihre Änderungen nicht sehen.

# Was verhindert das Caching im Varnish?

- Sessions / Cookies
- no\_cache=1 (GET, POST, TypoScript, ...)
- Die ungecachten cObjects USER\_INT und COA\_INT
- Auslieferung unterschiedlicher Inhalte je nach Environment-Variablen wie IP, User-Agent
- SSL-Verschlüsselung

# Sessions

## **Cookies, die bei TYPO3 gesetzt werden :**

- *fe\_typo\_user* Erst in der aktuellen Version nur nach Anmeldung gesetzt
- *be\_typo\_user* Ein einziger Aufruf vom Backend-Anmeldeformular reicht, um das Cookie zusetzen
- Google Analytics
- Piwik
- ...

# no\_cache

Einstellung in TYPO3

Wird gesetzt über:

- GET/POST-Parameter
- TypoScript Setup Einstellung  
`config.no_cache=1`  
bzw.  
`page.config.no_cache=1`
- Seiten-Eigenschaften



# USER\_INT & COA\_INT

## Ungecachte Content Objects (cObjects)

- Diese Content-Objekte deaktivieren die Cache-Control-Header für die Seite, die der Varnish benötigt, um Seiten zu cachen.
- Vorsicht bei Verwendung von \*\_INT-Objekten bei der FLUIDTEMPLATE Eigenschaft *variables*! Es reicht die Definition eines \*\_INT-Objektes und es wird für sämtliche Backend-Layouts der Cache deaktiviert.
- Besser **AJAX** verwenden oder in einem BE-Layout mit `<f:cObject typoscriptObjectPath="lib.myUncachedElement">` arbeiten! Hier werden nur die Seiten ungecacht ausgeliefert, in denen über das Argument `typoscriptObjectPath` ein \*\_INT-Objekt aufgerufen wird.

# Einstellungen im TYPO3 Install-Tool

- **['SYS']['reverseProxyIP'] = '127.0.0.1'**  
Hier muss die Adresse vom Varnish-Server eingetragen werden.
- **['SYS']['reverseProxyHeaderMultiValue'] = 'first | last'**  
Mit dieser Änderung wird die erste/letzte X-Forwarded-For IP im TypoScript abrufbar über  
`stdWrap.data = getIndpEnv:REMOTE_ADDR`
- **['FE']['dontSetCookie'] = 1**  
Muss nicht mehr zwingend eingetragen werden. TYPO3 setzt in aktuellen TYPO3-Versionen das FE-Cookie nur noch bei Bedarf.
- **['FE']['disableNoCacheParameter'] = 0/1**  
Optimal wäre „0“, aber dann darf keine Extension  
`<input type="hidden" name="no_cache" value="1">`  
im Formular verwenden!

**MiTTWALD**

Webhosting. Einfach intelligent.

# TYPO3 Backend & TypoScript

Einstellungen optimieren

[www.mittwald.de](http://www.mittwald.de)

# TypoScript

## Setup



```
config {  
    sendCacheHeaders = 1  
    cache_period = 86400  
}
```

Mit diesen Zeilen werden die Cache-Control-Header gesendet, die ein Reverse-Proxy benötigen, um die HTML-Seiten cachen zu können.

# TYPO3 Seiteneigenschaften

Benutzeranmeldung deaktivieren?!? Was ist das?!?

### Veröffentlichungsdaten und Zugriffsrechte

**Veröffentlichungsdatum**   **Ablaufdatum**   **Auf Unterseiten ausdehnen**  Aktiviert

**Zugriffsrechte für Benutzergruppen**

**Ausgewählte Objekte** **Verfügbare Objekte**

	<p>⤴</p> <p>⤵</p> <p>⤶</p> <p>⤷</p> <p>✕</p> <p>Nach Anmeldung verbergen Anzeigen, wenn angemeldet __Benutzergruppen: __ Admin Mittwald</p>
--	---

**Anmeldeverhalten**

▾

- Benutzeranmeldung freigeben
- Benutzeranmeldung deaktivieren
- Anmeldung für Benutzer freigeben, aber für Benutzergruppen deaktivieren
- Anmeldung wieder freigeben

# Anmeldeverhalten

Steuert die Ausgabe der Cache-Control-Header

- **Benutzeranmeldung freigeben** (Standard)  
Die Seite wird nicht gecacht, wenn es einen Login gibt.
- **Benutzeranmeldung deaktivieren**  
Die Seite kann von einem Proxy gespeichert werden. Das Cookie `fe_typo_user` wird nicht mehr ausgegeben. Sowohl eingeloggte als auch ausgeloggte Personen erhalten die selbe Seite.
- **Anmeldung für Benutzer freigeben, aber für Benutzergruppen deaktivieren**  
Hier könnte ein spezieller Header ausgegeben werden, mit dem man dem Varnish mitteilt, dass eine zweite Version der Seite zwischengespeichert werden soll.
- **Anmeldung wieder freigeben**  
Die Seite kann von einem Proxy nicht mehr gespeichert werden. Das Cookie `fe_typo_user` wird wieder ausgegeben, auch wenn auf einer übergeordneten Seite „Benutzeranmeldung deaktivieren“ ausgewählt war.

**MITTWALD**

Webhosting. Einfach intelligent.

# Nginx

Anpassungen

[www.mittwald.de](http://www.mittwald.de)

# nginx Logfiles optimieren

IP des Clients anstatt des Servers loggen

## **Problem:**

Normalerweise werden die IP-Adressen von einem Besucher im access.log des Webservers gespeichert. Mit einem vorgeschalteten Varnish-Server ist die mitgeloggte IP-Adresse immer die vom Varnish.

## **Lösung:**

Der Varnish-Server gibt die IP des Besuchers an den Webserver per **X-Forwarded-For** Header. Dieser muss dann geloggt werden.



# nginx Logfiles optimieren

## Webserver

- ***/etc/nginx/nginx.conf*** (Sektion: http) oder Virtual-Host Datei (Sektion: server)
- `set_real_ip_from 127.0.0.1; # IP of your varnish/proxy`
- `real_ip_header X-Forwarded-For; # Header from varnish`
- ***/etc/varnish/default.vcl***  
Wenn das Loggen noch nicht klappt, im `vcl_recv` Abschnitt hinzufügen:  
`set req.http.X-Forwarded-For = client.ip;`

**MITTWALD**

Webhosting. Einfach intelligent.

# HTTPS mit Varnish

SSL-gesicherte Verbindungen cachen

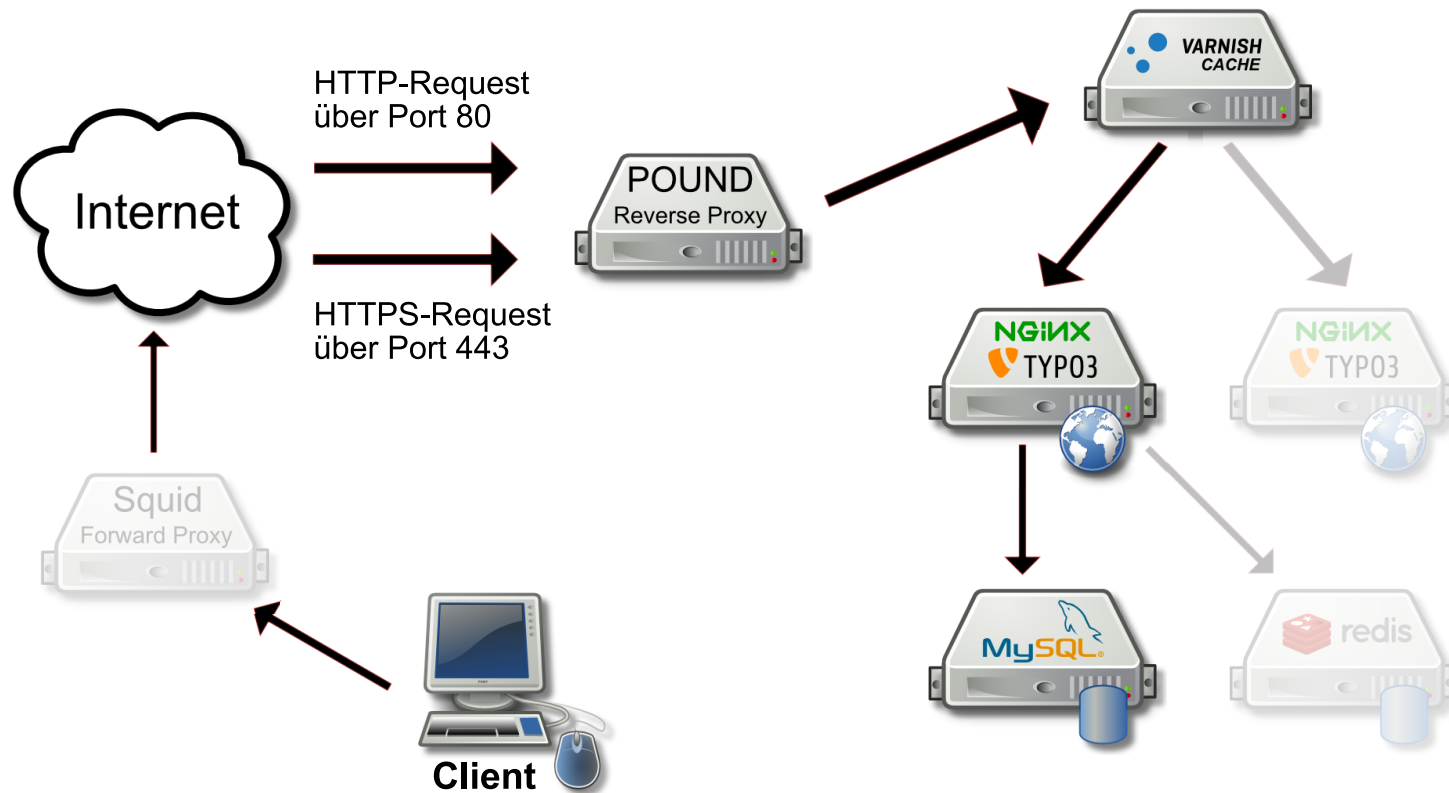
[www.mittwald.de](http://www.mittwald.de)

# Reverse Proxy Pound

als https-Wrapper nutzen

- Pound ist ein Reverse-Proxy, ein Lastverteiler und ein HTTPS-Wrapper.
- Pound kann Anfragen auf Regeln basierend an einen oder mehrere Webserver weiterleiten.
- Pound kann Sitzungen erkennen, zum Beispiel durch Cookies oder Anmelde-Zeichenfolgen.
- Eine einmal erkannte Sitzung wird in der Folge immer an den gleichen Webserver geleitet.
- Man kann Pound als Lastverteiler einsetzen, um eingehende Anfragen auf mehrere Webserver zu verteilen und so die Gesamtbelastung zu minimieren.
- Pound reicht die Anfragen zum größten Teil komplett unverändert durch - ist dadurch sehr kompatibel.
- Pound verarbeitet keinerlei Anfragen selbst.
- Pound ist kein Webcache.

# Reverse Proxy Pound



# Reverse Proxy Pound

## Ausschnitt aus der Konfiguration

```
ListenHTTP
# Address Default: 127.0.0.1, for all use 0.0.0.0
# put your server's public IP address here
Address 0.0.0.0
Port 80
RewriteLocation 0 # Default 1 => infinite loop!
# HeadRemove "X-Forwarded-For"
## allow PUT and DELETE also (by default only GET, POST and HEAD)?:
xHTTP 2
Service
    HeadRequire "(Host: vm-typo3.loc|Host: opcache.vm-typo3.loc)"
    BackEnd
        Address 127.0.0.1
        Port 8000
    End
End
Service
    BackEnd
        Address 127.0.0.1
        Port 6081
    End
End
End
```

**MiTTWALD**

Webhosting. Einfach intelligent.

# Varnish-Konfiguration

Anpassen der default.vcl Datei

[www.mittwald.de](http://www.mittwald.de)

# Varnish-Konfiguration in der default.vcl

## Caching prüfen

Zum Zählen der aus dem Cache ausgelieferten Seiten kann dieses Snippet verwendet werden:

```
sub vcl_deliver {  
    if (obj.hits > 0) {  
        set resp.http.X-Cache = "Hit (" +  
obj.hits + ")";  
    } else {  
        set resp.http.X-Cache = "MISS";  
    }  
}
```

# Varnish-Konfiguration in der default.vcl

## Dateien schützen (Abschnitt: vcl\_recv)

```
/**
 * Added security, the "w00tw00t" and other attacks are pretty annoying
 * so lets block it before it reaches our webserver
 * config, backup and other private files are also blocked
 */
if (req.url ~ "^/w00tw00t" ||
    req.url ~ "^/phppath/" ||
#   req.url ~ "^/pma/" ||
    req.url ~ "^/phpMyAdmin" ||
    req.url ~ "^/phpmyadmin" ||
    req.url ~ "wp-(admin|login|content)" ||
    req.url ~ "\.inc\.php" ||
    req.url ~ "^/%70%68%70%70%61%74%68/" ||
    req.url ~ "\.(bak|backup|conf|log|properties|sql|tar)$" ||
    req.url ~ "/Private/" ||
    req.url ~
"typo3conf/ext/(.*)/Configuration/(TypoScript|FlexForms|TCA|ExtensionBuilder)/"
) {
    return (synth(403, "Not permitted"));
}
```



**MITTWALD**

Webhosting. Einfach intelligent.

ESI

Edge Side Includes

[www.mittwald.de](http://www.mittwald.de)

# Edge Side Includes (ESI)

<https://www.varnish-cache.org/docs/4.0/users-guide/esi.html>

```
#!/bin/sh
echo 'Content-type: text/html'
echo ''
date "+%Y-%m-%d %H:%M"
```

---

```
<HTML>
<BODY>
The time is: <esi:include src="/cgi-
bin/date.cgi"/>
at this very moment.
</BODY>
</HTML>
```

# Edge Side Includes

<https://www.varnish-cache.org/docs/4.0/users-guide/esi.html>

## ESI Aktivierung:

```
sub vcl_backend_response {
    if (bereq.url == "/test.html") {
        set beresp.do_esi = true; // Do ESI processing
        set beresp.ttl = 24 h;    // Sets the TTL on the HTML above
    } elseif (bereq.url == "/cgi-bin/date.cgi") {
        set beresp.ttl = 1m;     // Sets a one minute TTL on
                                // the included object
    }
}
```

## Tipp:

**Aktivierung über Header-Daten. Diese könnten in Abhängigkeit zu einer Checkbox bei den Seiteneigenschaften übermittelt werden.**

# Edge Side Includes

<https://www.varnish-cache.org/docs/4.0/users-guide/esi.html>

## **esi:remove and <!--esi ... →**

```
<esi:remove>
  <a href="http://www.example.com/LICENSE">The
license</a>
</esi:remove>
<!--esi
<p>The full text of the license:</p>
<esi:include src="http://example.com/LICENSE" />
-->
```

**MiTTWALD**

Webhosting. Einfach intelligent.

# Hilfreiche Extension vcc

Varnish Cache Control

[www.mittwald.de](http://www.mittwald.de)

# Hilfreiche Extension „vcc“

Varnish Cache Control

Das Löschen des Varnish-Caches kann über die Extension  
**„Varnish Cache Control“**

mit dem Extension-Key „vcc“ auch über das TYPO3 Backend erfolgen.

Mit der installierten und konfigurierten Extension kann ein Redakteur eine Seite über einen Klick im Backend aus dem Varnish-Cache löschen.

**MITTWALD**

Webhosting. Einfach intelligent.

# Lasttest

mit dem Linux-Tool „siege“

[www.mittwald.de](http://www.mittwald.de)

# Last-Test

mit und ohne Varnish

Dazu die Config-Dateien so einstellen, dass der Zugriff beim Varnish und Nginx nicht nur über 127.0.0.1 erfolgen kann! Nach dem Test gegebenenfalls wieder zurückstellen!

Test mit Varnish: `siege -b -c 10 -t10s http://domain.tld:6081`

Test ohne Varnish: `siege -b -c 10 -t10s http://domain.tld:8000`



# Ergebnis des Siege Tests

## Ohne Varnish

```
root@myserver ~ # siege -b -c 10 -t10s "http://dev.oliver-thiele.de:8000"
** SIEGE 3.0.8
** Preparing 10 concurrent users for battle.
The server is now under siege...
Lifting the server siege...      done.
```

```
Transactions:                1086 hits
Availability:                 100.00 %
Elapsed time:                 9.36 secs
Data transferred:            7.50 MB
Response time:                0.09 secs
Transaction rate:             116.03 trans/sec
Throughput:                   0.80 MB/sec
Concurrency:                  9.96
Successful transactions:      1086
Failed transactions:          0
Longest transaction:          0.12
Shortest transaction:         0.03
```

# Ergebnis des Siege Tests

## Mit Varnish

```
root@myserver ~ # siege -b -c 10 -t10s "http://dev.oliver-thiele.de:6081"  
** SIEGE 3.0.8  
** Preparing 10 concurrent users for battle.  
The server is now under siege...  
Lifting the server siege...      done.
```

Transactions:	<b>95064 hits</b>
Availability:	100.00 %
Elapsed time:	9.37 secs
Data transferred:	<b>656.74 MB</b>
Response time:	0.00 secs
Transaction rate:	<b>10145.57 trans/sec</b>
Throughput:	70.09 MB/sec
Concurrency:	9.66
Successful transactions:	95064
Failed transactions:	0
Longest transaction:	0.57
Shortest transaction:	0.00