Caching

Katrin Lányi | Computer- und Medienservice | lanyi@cms.hu-berlin.de
Bartosz Świątek | Computer- und Medienservice | bartosz.swiatek@cms.hu-berlin.de

Von Natur aus arbeiten Content-Management-Systeme langsamer als Systeme mit statischen HTML-Seiten, da sie die Seiten erst zur Laufzeit aus vielen einzelnen Elementen zusammensetzen müssen. Um dem entgegen zu wirken hat der richtige Einsatz des Cachings eine enorme Bedeutung.

Was ist Caching?

Unter Caching versteht man in der Informationsverarbeitung die Anlage eines schnellen Zwischen-Speichers, der Kopien von Original-Inhalten enthält. Caching findet zum einen in Geräten wie CPUs und Festplatten statt, zum anderen in vielen verschiedenen Software-Anwendungen. Der Zweck des Cachings ist meistens die Beschleunigung des Zugriffs auf nachgefragte Inhalte und die Entlastung des Originalservers. Es dient aber auch der Erhöhung der Zuverlässigkeit bei der Bereitstellung von Inhalten: Ist der Original-Inhalt gerade nicht verfügbar, wird ein älterer Inhalt ersatzweise zur Verfügung gestellt. Beispielsweise findet man im Google-Cache Inhalte oft noch lange, nachdem der Inhalt vom Originalserver bereits gelöscht wurde.

Warum hat Caching in Zope/Plone so große Bedeutung?

Im Gegensatz zu statischen HTML-Seiten werden Inhalte in Zope/Plone nicht als ganze Objekte vorgehalten. Inhalte in Content-Management-Systemen (CMS) – und das gilt für alle Arten von CMS – werden bei der Auslieferung grundsätzlich aus vielen einzelnen Teilen zusammenstellt. Die Seitennavigation beispielsweise wird aus den zur Auslieferungszeit vorhandenen Navigationselementen zusammengesucht und bei der Darstellung einer Seite werden möglicherweise alle für diesen Tag und dieses Thema interessanten Termine und Events automatisch eingefügt. Je nach Berechtigung des Nutzers können unterschiedliche Inhalte ausgeliefert werden – ein Riesenvorteil gegenüber statischen Seiten! Allerdings wird nun für die Berechnung der Seitenelemente teilweise sehr viel Zeit benötigt.

Effizientes Caching soll diesen erhöhten Zeitaufwand ausgleichen und die Seitenauslieferungszeiten wieder herabsetzen, indem es eine einmal abgefragte Inhaltszusammenstellung als statische Seite in einem Zwischenspeicher vorhält und unter der Bedingung, dass sich die Seitenelemente nicht verändert haben, bei der nächsten Anfrage wieder ausliefert. Das ist nicht trivial, denn wie kann überprüft werden, ob sich einzelne Teile, aus denen die Seite im Cache ursprünglich zusammengestellt wurde, geändert haben?

Wie arbeiten Web-Caches?

Alle Caches arbeiten nach Regeln, auf Grund derer sie Objekte ausliefern. Die Basis-Regeln für Web-Caches werden im HTTP-Protokoll festgeschrieben, andere legen die Administratoren der Caches fest.

Grundregeln:

- Alle Objekte werden im Cache gehalten, solange sie nicht als „verfallen“ gekennzeichnet sind.
- Ein im Cache vorhandenes Objekt wird als „verfallen“ betrachtet, wenn die im Header des Objektes angegebene Verfallszeit überschritten ist bzw. wenn es innerhalb einer bestimmten Zeit geändert wurde.
- Wenn im Header eines Objektes enthalten ist, dass dieses Objekt grundsätzlich nicht aus dem Cache ausgeliefert

werden soll, wird es dort auch nicht abgelegt. Gleiches gilt, wenn ein Caching zwar im Header nicht ausgeschlossen wird, aber auch kein Merkmal zur „Validierung“, also zur Überprüfung der Gültigkeit, enthalten ist.

- Wenn die Auslieferung eines Objektes eine Authentifizierung verlangt, wird es nicht im Zwischenspeicher gehalten.
- Wenn ein Objekt „verfallen“ ist, wird der Originalserver befragt, ob die im Cache vorgehaltene Kopie weiterhin ausgeliefert werden darf.

Die Beeinflussung des Caches

Gestalter von Webseiten können den Cache beeinflussen:

1. Über die Meta-Tags in HTML-Seiten
   Die Meta-Tags werden in den <HEAD>-Abschnitt eines HTML-Dokumentes eingefügt. Beispiele:
   a) Anweisung, dass dieses Objekt nach Ablauf der Verfallszeit vom Originalserver geladen werden soll:
   ```
   <meta http-equiv="expires"
   content="Sat,
   01 Dec 2001 00:00:00 GMT">
   ```
   b) Anweisung an den Browser, keinen Cache zu benutzen, sondern das Objekt in jedem Falle vom Originalserver zu laden:
   ```
   <meta http-equiv="cache-
   control" content="no-cache">
   ```
   c) Anweisung, dass diese Datei auf keinen Fall aus dem Cache ausgeliefert werden soll:
   ```
   <meta http-equiv="pragma"
   content="no-cache">
   ```

   Derartige Meta-Tags sind leicht zu setzen, allerdings werden sie nur von Browsern gelesen, nicht jedoch von Proxy-Caches.

2. Über den HTTP-Header
   Der HTTP-Header von Dokumenten wird über den Web-Server gesetzt und noch vor dem Beginn des HTML-Quellcodes ausgeliefert. Im Quellcode des Dokumentes ist er also nicht sichtbar, aber es gibt verschiedene Hilfsmittel, die den Header einer Seite sichtbar machen, z.B. die Web Developer Toolbar, die als Add On für verschiedene Browser existiert. Über den Menüpunkt „Information“ kann man sich dort den „Response Header“ einer dargestellten Webseite ansehen, siehe Abb. 1.

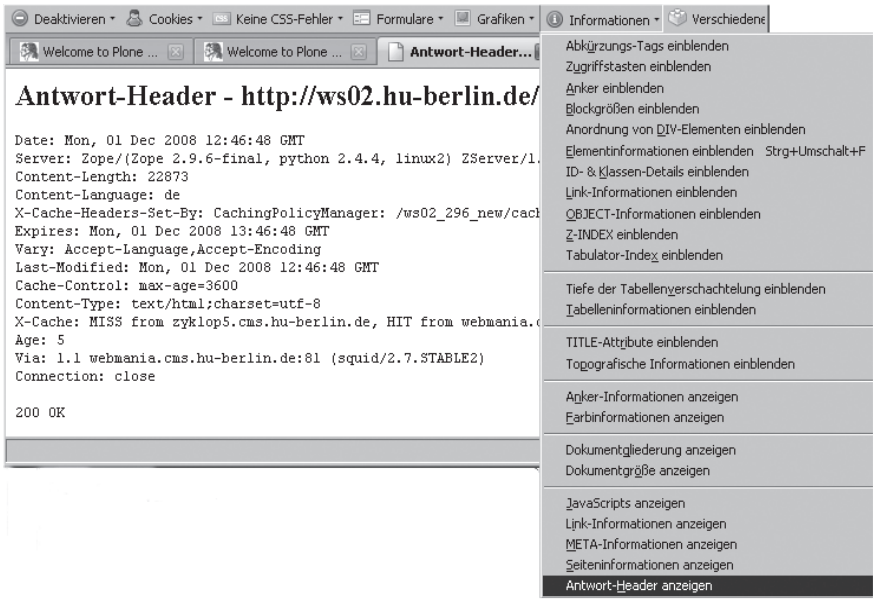

Abb. 1: Der Response Header einer Webseite

Eine andere Möglichkeit, die Header-Zeilen einzusehen, ist die Betrachtung mit wget:

```
wget -S --delete-after <URL>
```

Das Ergebnis sieht dann etwa wie folgt aus (Ausgabe gekürzt und anonymisiert):

```
einrechner:/usr/local/www/bin# wget -S --delete-after http://ws02.hu-
berlin.de/front-page
--16:37:56--  http://ws02.hu-berlin.de/front-page
          => `front-page'
Resolving ws02.hu-berlin.de... 141.20.9.999
Connecting to ws02.hu-berlin.de|141.20.9.999|:80... connected.
HTTP request sent, awaiting response...
  HTTP/1.1 200 OK
  Date: Mon, 01 Dec 2008 15:37:41 GMT
  Server: Zope/(Zope 2.9.6-final, python 2.4.4, linux2) ZServer/1.1
Plone/2.5.2
  Content-Length: 22873
  Content-Language: de
  X-Cache-Headers-Set-By: CachingPolicyManager: /ws02/caching_policy_manager
  Expires: Mon, 01 Dec 2008 16:37:41 GMT
  Vary: Accept-Language,Accept-Encoding
  Last-Modified: Mon, 01 Dec 2008 15:37:41 GMT
  Cache-Control: max-age=3600
  Content-Type: text/html;charset=utf-8
  Age: 15
  X-Cache: HIT from einrechner.cms.hu-berlin.de
  Via: 1.1 einrechner.cms.hu-berlin.de:81 (squid/2.7.STABLE2)
  Connection: close
Length: 22,873 (22K) [text/html]
```

Der Eintrag „Expires“

Webserver können für alle ausgelieferten Objekte in deren Response-Header eine grundsätzliche Verfallszeit festlegen. Alternativ können sie einen Wert für ein „maximales Alter“ des Objektes (max-age) angeben. Dazu muss ein max-age-Tag in den Kopf des Dokuments eingefügt werden. Wenn der max-age-Wert des Response-Headers nicht überschritten wurde, wird das Objekt weiterhin im Cache gehalten. Zu beachten ist: Wenn der Header eines Objektes sowohl einen max-age- als auch einen expires-Wert enthält, hat der max-age-Wert Vorrang und überschreibt ggf. den expires-Eintrag. Das Vorhandensein eines max-age-Wertes impliziert, dass ein Objekt im Cache abgelegt werden darf, auch wenn es daneben im Header noch andere, restriktivere Regeln gibt. Wenn ein Objekt z. B. aus Sicherheitsgründen nicht gecachet werden soll, muss auf diesen Eintrag also verzichtet werden![1]

Beispiel für einen Expires-Eintrag: Die einzige gültige Form für einen Expires-Eintrag ist ein Datum inkl. einer Zeit, die dem GMT-Standard, nicht der lokalen Zeit, entspricht, beispielsweise:

```
Expires: Fri, 05 Sep 2008 11:16:44
GMT
```

Der Eintrag „Cache Control“

Beispiel für einen max-age-Eintrag:

```
Cache-Control: max-age=3600
```

Die verbreitete Auffassung, die Anweisung „Pragma: no-cache“ führe dazu, dass eine Seite gar nicht im Cache gehalten werde, ist so nicht wahr. Die HTTP-Spezifikation schreibt diese Verfahrensweise nicht zwingend vor.[2] Und so werten viele Caches diesen Eintrag nicht aus. An Stelle dessen sollte man „Cache-Control: no-cache“ verwenden.

Cache-Formen und Cache-Manager in Zope/Plone

Die Beeinflussung der verschiedenen Caches erfolgt über die Cache-Manager, die als Objekte in Zope/Plone-Instanzen eingebunden werden können.

Browserseitiges Caching

Zum einen ist die Standardeinstellung der meisten Webbrowser so festgelegt, dass ein Caching auf Nutzerseite erfolgen soll. Hier also geht es um einen Zwischenspeicher auf dem Rechner des Benutzers, der vom ausliefernden Webserver nicht immer beeinflussbar ist. Er ist besonders dann ärgerlich, wenn Nutzer in ihrer Eigenschaft als Redakteure die Inhalte mancher Webseiten verändern dürfen, aber regelmäßig veraltete Inhalte vorfinden (siehe auch Tipps am Ende des Artikels).

Caching im RAM des Zope/Plone-Servers

Der RAM-Cache übernimmt das Vorhalten von einzelnen Webseiten-Objekten. Er speichert grundsätzlich keine Bilder, Files u. ä., sondern das Ergebnis aufwändiger Berechnungen. Die zu cachenden Objekte werden im Arbeitsspeicher des Servers gehalten. Das macht einerseits die Auslieferung der darin enthaltenen Objekte sehr schnell, bedeutet aber auch, dass Plone mehr Arbeitsspeicher belegt.[3] Plone generiert standardmäßig ein RAMCache-Objekt in der Wurzel der Plone-Site.

Proxy-Caching

Effektives serverseitiges Caching erreicht man hauptsächlich über einen Proxy-Server bzw. wie im Falle der HU ein entsprechendes Proxy-Server-Cluster. Dort werden alle Cache-fähigen Objekte aller Zope/Plone-Instanzen abgelegt und bei gleicher Anfrage wieder ausgeliefert ohne den Originalserver nochmals mit der Seitenberechnung belasten zu müssen. Zur Steuerung dieses Caches steht unter Zope/Plone der Accelerated HTTP Cache Manager zur Verfügung, siehe Abb. 2.

Seine Parameter überschreiben die globalen Cache-Einstellungen (ZMI: “portal_skins->plone_templates-> global_cache_settings”).

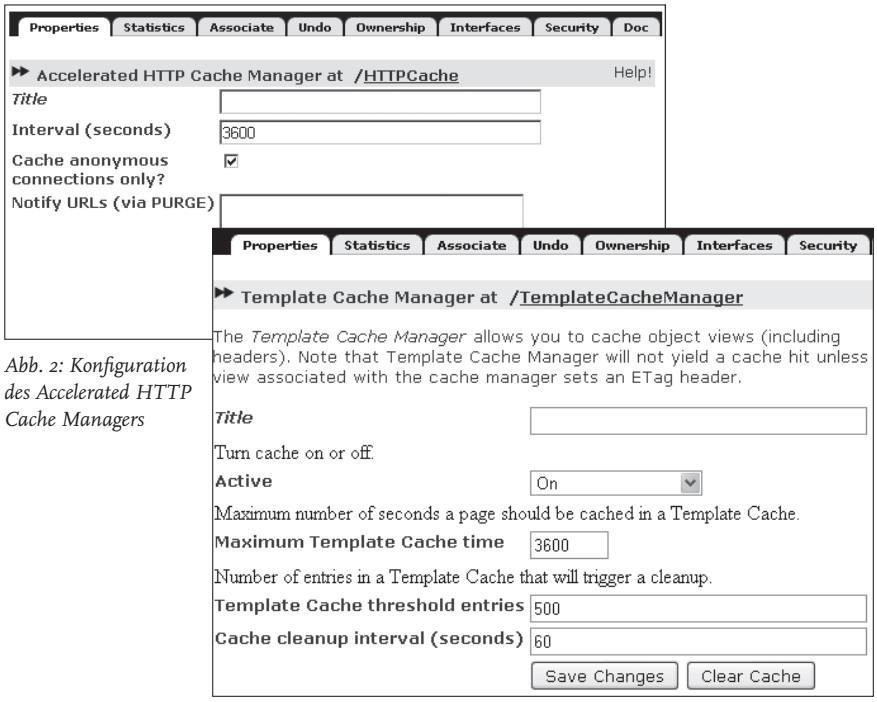

Abb. 2: Konfiguration des Accelerated HTTP Cache Managers

Abb. 3: Konfiguration des Template Cache Managers

1 http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html
2 http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.32
3 http://www.galileocomputing.de/openbook/plone25/plone_kap_12_002.htm

Beispiel 1 für die Datei global_cache_settings (die gecachten Seiten werden nach einer Stunde (3600 s) wieder aus dem externen Cache gelöscht):

```
<metal:block tal:define="dummy python:request.RESPONSE.setHeader(‚Vary‘,
‚Accept-Language,Accept-Encoding,Cookie‘)" />
<metal:block tal:define="dummy python:request.RESPONSE.setHeader(‚Cache-
Control‘, ‚max-age=0,s-maxage=3600‘)" />
<metal:block tal:define="dummy python:request.RESPONSE.setHeader(‚Last-
Modified‘, here.modified().toZone(‚GMT‘).rfc822())" />
```

Beispiel 2 (Nutzer, die sich eingeloggt haben, erhalten keine Cache-Kopien mehr):

```
<metal:block tal:define="dummy python:request.RESPONSE.setHeader(‚Vary‘,
‚Accept-Language,Accept-Encoding,Cookie‘)" />
<tal:block condition="isAnon">
<metal:block tal:define="dummy python:request.RESPONSE.setHeader(‚Cache-
Control‘, ‚max-age=0,s-maxage=3600‘)" />
<metal:block tal:define="dummy python:request.RESPONSE.setHeader(‚Last-
Modified‘, here.modified().toZone(‚GMT‘).rfc822())" />
</tal:block>
<tal:block condition="not:isAnon">
<metal:block tal:define="dummy python:request.RESPONSE.setHeader(‚Cache-
Control‘, ‚must-revalidate,max-age=0,no-cache‘)" />
</tal:block>
```

Seit einiger Zeit steht auch der in Marburg entwickelte Template Cache Manager zur Verfügung, der die Möglichkeit bietet, zusätzlich ein ETag auszuliefern. Ein ETag (auch „entity tag“ genannt) ist ein erweiterter Header-Eintrag, der mit HTTP/1.1 eingeführt wurde. Er dient zur Bestimmung von Änderungen an der angeforderten Datei. Wenn eine neue HTTP-Antwort denselben ETag wie eine zuvor erhaltene HTTP-Antwort enthält, ist die angeforderte Seite mit der zuvor angeforderten identisch und muss nicht noch einmal herunter geladen werden. Der ETag kann eine beliebige Zeichenkette sein. Es bietet sich an, die Dateigröße oder das Änderungsdatum zu verwenden. Die Header-Zeilen sehen dann wie folgt aus:

a) bei anonymen Nutzern

```
Date: Wed, 04 Jun 2008 12:57:04 GMT
Server: Zope/(Zope 2.9.6-final, python 2.4.4, linux2) ZServer/1.1 Plone/2.5.2
Vary: Host
Content-Length: 19523
Content-Language: de
Expires: Wed, 04 Jun 2008 13:37:06 GMT
Last-Modified: Wed, 04 Jun 2008 12:57:05 GMT
Etag: 1212584225.3|QW5IwNjUzL2I3YmViOTNhOGViNjEwZTIwYTRkYzE5YTk4MTQyMmU=
Cache-Control: max-age=3600, s-maxage=3600
Content-Type: text/html;charset=utf-8
X-Cache-Lookup: HIT from localhost.localdomain:81
Via: 1.0 localhost.localdomain:81 (squid/2.6.STABLE5)
```

b) bei authentifizierten Nutzern

```
Date: Wed, 04 Jun 2008 13:01:44 GMT
Server: Zope/(Zope 2.9.6-final, python 2.4.4, linux2) ZServer/1.1 Plone/2.5.2
Vary: Host
Content-Length: 36306
Content-Language: de
Expires: Wed, 04 Jun 2008 13:01:45 GMT
Last-Modified: Wed, 04 Jun 2008 13:01:45 GMT
Etag: 1212584505.8NjM0NTMuMDgvYjdiZWI5M2E4ZWI2TBlMjBhNGRjMTlhOTgxNDIyZQ==
Cache-Control: max-age=0, s-maxage=0, must-revalidate, no-cache
Content-Type: text/html;charset=utf-8
X-Cache-Lookup: HIT from localhost.localdomain:81
Via: 1.0 localhost.localdomain:81 (squid/2.6.STABLE5)
```

Welcher Cache-Manager für welche Objekte verwendet werden soll, kann zum einen über die Angaben cache = <Cache-Manager> in den .metadata-Dateien der Objekte (Bilder, JavaScript-Skripte, CSS) festgelegt werden. Zum anderen lassen sich Objekte auch über den „Associate“-Button in der Zope-Oberfläche einem Cache-Manager zuweisen. Objekten, die in den Ordner custom kopiert wurden, um sie zu modifizieren, muss dort nachträglich der zu verwendende Cache nochmals zugewiesen werden! Für die allermeisten Objekte sollte der HTTPCache verwendet werden.

Caching Policies

Um Plone-Content-Typen richtig cachen zu können, müssen zusätzlich zur Konfiguration der Cache-Manager noch Caching Policies, also Regeln des Cachens, bestimmt werden. Dazu steht im ZMI der Caching Policy Manager zur Verfügung. Die hier festgelegten Regeln erweitern oder überschreiben jene, die unter portal_skins->plone_templates->global_cache_settings definiert wurden. Beispiel siehe Abb. 3.

Hinweis: Unter Plone 2.5 werden später die Werte von pre-check (Das ist der Wert in Sekunden, nachdem geprüft werden muss, ob ein Objekt noch „frisch“ ist bevor die Seite an den Nutzer ausgeliefert wird.) und post-check (Das ist der Wert in Sekunden, der angibt, wann ein Objekt neu in den Cache geladen werden muss. Dies geschieht möglicherweise erst, nachdem die Seite an den Nutzer ausgeliefert wurde!) nicht angezeigt, aber richtig ausgewertet.[4]

Proxies können Cache-Angaben ignorieren

Um Systeme auch bei großem Andrang von Redakteuren und Besuchern noch stabil betreiben zu können, ist es von Zeit zu Zeit erforderlich, das Caching aggressiver zu betreiben. An der HU nehmen wir diese Einstellung beispielsweise zum Semesterwechsel vor. Dann

4 http://msdn.microsoft.com/en-us/library/ms533020(VS.85).aspx

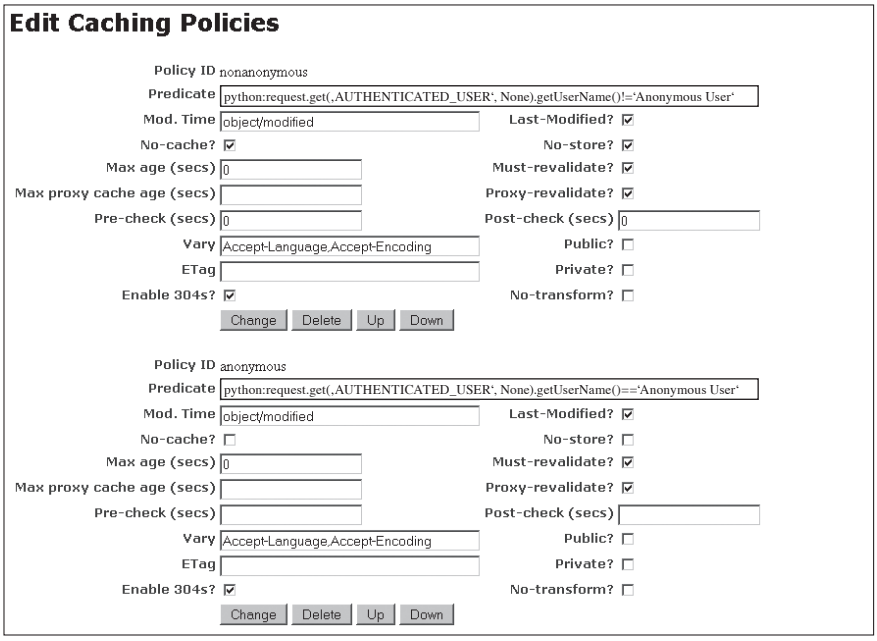

Abb. 3: Einrichten einer Caching-Policy, die Seiten nicht aus dem Cache ausliefert, wenn ein Nutzer sich bei Plone authentifiziert hat.

ist es egal, ob ein Objekt, z. B. eine CSS-Datei, dem Cache mitteilt, sie habe sich verändert oder nicht, es wird immer eine Mindestzeit lang eine Cache-Kopie ausgeliefert. Lediglich Administratoren haben dann noch die Möglichkeit, über Skripte ein so genanntes „Purging" zu veranlassen, also das Löschen eines Objektes aus dem Cache.

Tipps für Plone-Redakteure

- Löschen Sie manuell den Inhalt Ihres Browser-Caches, bevor Sie mit der Bearbeitung von Plone-Seiten beginnen. Die Browser bieten inzwischen einige Erweiterungen, um diesen Vorgang auf einen Klick zu reduzieren.
- Schalten Sie den Cache Ihres Browsers während der Bearbeitung von Plone-Seiten ab.
- Prägen Sie sich die Tastenkombination für das Neuladen einer Seite mit Überschreiben des Caches ein. Diese hängt vom jeweiligen Browser ab, aber viele Browser unterstützen hier „Strg+F5".[5]
- Vermeiden Sie die Integration häufig wechselnder Elemente in eine Plone-Seite. Beispielsweise ist das Kalender-Portlet ein optisch durchaus attraktives Seitenelement, das jedoch beim Laden der Seite viel Zeit braucht, weil es zunächst alle Termine einer Site ausfindig machen muss, um die Darstellung eines Tages zu berechnen.
- Nachrichten, die auf der Startseite eingebunden werden, sollten im Laufe des Tages nicht ausgewechselt werden.
- Auch die Navigationselemente sollten so selten wie möglich verändert werden.

5 http://support.mozilla.com/de/kb/Keyboard+Shortcuts

Literatur

[1] McKay, Andy: *The Definitive Guide to Plone*. APRESS, http://www.zope.de/dokumentation/plonebuch/ch14.rst, 2004