Mehrsprachige Web-Sites mit Apache

Content Negotiation statt Länderflaggen

Stefan Kuhlins

Lehrstuhl für Wirtschaftsinformatik III
Universität Mannheim
68131 Mannheim
http://www.kuhlins.de/

Zusammenfassung Auf vielen Web-Sites, die ihre Inhalte in verschiedenen Sprachen anbieten, sind Länderflaggen und Ähnliches zu sehen, die den Besuchern die Auswahl einer Sprache per Mausklick erlauben. Im Apache schlummert ein viel zu selten eingesetztes Feature namens „Content Negotiation“, das die Auswahl der für den jeweiligen Besucher am besten passenden Sprachversion automatisiert. Gegenüber handgestrickten Lösungen bietet Content Negotiation einfachere Wartung und professionelleren Auftritt. Darüber hinaus eignet sich der Mechanismus zur Auswahl von Ressourcen wie beispielsweise Bildformaten und komprimierten Dateien.

1 Einleitung

Apache ist einer der wenigen Web-Server, die Content Negotiation unterstützen (siehe Anhang A). Übersetzen lässt sich „Negotiation“ mit „Verhandlung“. Web-Client und -Server verhandeln demnach darüber, welche Variante einer Ressource ausgewählt werden soll, damit eine möglichst gute Darstellung resultiert. Speziell auf Sprachen angewandt, heißt dies, dass die Benutzer die von ihnen bevorzugten Sprachen im Browser einstellen – beim *Netscape Navigator* unter *Einstellungen*, *Navigator*, *Sprachen* zu finden (siehe Abb. 1). Der Browser teilt dem Web-Server bei Anfragen dann mit, welche Sprachen bevorzugt werden. Darüber hinaus funktioniert der Mechanismus z. B. auch für Bildformate wie GIF und JPEG. Außerdem kann der Browser mitteilen, welche Plug-Ins installiert sind, wovon jedoch kaum einer Gebrauch macht.

Um ein Dokument in verschiedenen Sprachen präsentieren zu können, werden Übersetzungen des Dokuments für ausgewählte Sprachen erstellt. Anschließend teilt man dem Web-Server mit, für welche Sprachen Dokumente vorliegen. Apache stellt dafür zwei Mechanismen zur Verfügung: *Varianten* und *Multiviews*. Das Zweite basiert dabei auf dem Ersten.

2 Varianten

In einer Variantendatei werden alle bereitgestellten Sprachversionen eines Dokuments aufgelistet und jeweils einer oder auch mehreren Sprachen zugeordnet.

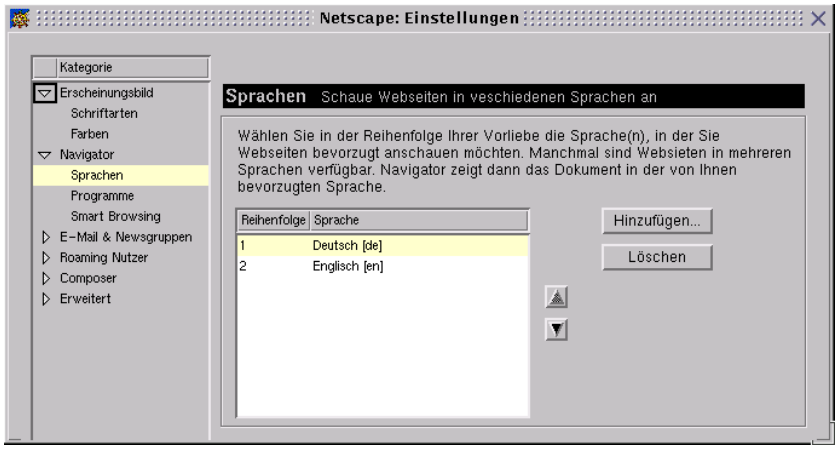

Abbildung 1. Bevorzugte Sprachen im *Netscape Navigator* einstellen

Man spricht deshalb auch von „Type-Maps“. Empfängt der Web-Server eine Anfrage für die Variantendatei, liefert er die am besten passende Version, indem die vorhandenen Varianten gemäß den vom Browser übermittelten Präferenzen geordnet werden.

Damit Apache Variantendateien (konventionsgemäß `*.var`) auswertet, ist ein entsprechender „Handler“ zu aktivieren, was normalerweise in der Server-Konfiguration durch die Zeile `AddHandler type-map var` bereits voreingestellt ist.

Für zwei HTML-Dokumente – eines in Deutsch (`test.de.html`) und eines in Englisch (`test.en.html`) – kann die Variantendatei so wie `test.var` aufgebaut werden (siehe Abb. 2).

```
URI: test

URI: test.de.html
Content-Type: text/html
Content-Language: de
Description: "Das deutsche Original des Testdokuments"

URI: test.en.html
Content-Type: text/html
Content-Language: en
Description: "English translation of the test document"
```

Abbildung 2. Variantendatei `test.var` für sprachspezifische HTML-Dateien

Die Datei `test.var` enthält drei Sektionen, die jeweils durch eine Leerzeile voneinander getrennt sind. In jeder Sektion wird ein Dokument mit seinen speziellen Eigenschaften beschrieben. Die erste Sektion enthält konventionsgemäß einen Eintrag für die Datei als Ganzes. Dies ist nicht zwingend notwendig und wird von Apache ignoriert. Auch Kommentarzeilen sind möglich; sie werden mit `#` eingeleitet.

Im Beispiel werden vier von sechs möglichen Headern eingesetzt. Nur `URI` und `Content-Type` müssen für jede Variante angegeben werden. `URI` verweist auf eine Dateiversion und wird als URL relativ zur Variantendatei interpretiert. Die Datei muss auf demselben Web-Server liegen und zugreifbar sein.

Der Medientyp oder auch MIME-Typ wird mittels `Content-Type` angegeben. Dieser Eintrag kann zusätzlich einen Zeichensatz, Level sowie Qualitätswert beinhalten (siehe Kap. 7).

`Content-Language` spezifiziert die Sprache mit den üblichen Länderkürzeln gemäß [3]. Für einige Sprachen sind überdies Untervarianten möglich wie z. B. `de-AT` (Österreich), `de-CH` (Schweiz) und `de-DE` (Deutschland).

Falls Apache keine passende Variante ermitteln kann, generiert er eine Übersicht über die verfügbaren Varianten, aus der Benutzer die für sie beste auswählen können (siehe Kap. 4). Hilfreich sind dabei Kurzbeschreibungen für die einzelnen Varianten, die mittels `Description` formuliert werden.

Darüber hinaus kann mit `Content-Length` die Dateigröße in Bytes angegeben werden, so dass Apache Dateigrößen vergleichen kann, ohne auf die Dateien zuzugreifen. Bei sonst gleichwertigen Alternativen wählt Apache die mit der kleinsten Größe.

Außerdem kann für komprimierte oder anderweitig codierte Dateien mittels `Content-Encoding` spezifiziert werden, in welcher Codierung (z. B. `x-compress` oder `x-gzip`) sie vorliegen (siehe Kap. 6).

Bei einer Anfrage nach `test.var` wertet Apache die Variantendatei aus und schickt das passende Dokument, wobei der vom Browser übermittelte HTTP-Header `Accept-Language` berücksichtigt wird (siehe Anhang B). Wenn dort nur eine Sprache angegeben ist und zu dieser auch ein Eintrag in der Variantendatei vorliegt, wird das Dokument in dieser Sprache geschickt. Sind mehrere Sprachen angegeben, wird die zuerst verfügbare gesendet. Falls kein passender Eintrag in der Variantendatei vorhanden ist, wird eine Übersicht über die verfügbaren Dokumente erzeugt (siehe Kap. 4).

Mit Variantendateien hat man die volle Kontrolle über den Auswahlmechanismus, aber man muss sie für jede Ressource zeitaufwändig erstellen und pflegen. Diese Aufgabe kann Apache übertragen werden, wenn es möglich ist, allein anhand der Dateiendung die zugehörige Variante zu bestimmen.

3 Multiviews

Um Multiviews für unser Beispiel zu aktivieren, sind folgende Einstellungen innerhalb einer `.htaccess`-Datei vorzunehmen:

```
Options +MultiViews
AddLanguage de .de
AddLanguage en .en
LanguagePriority de en
```

`Options All` umfasst `MultiViews` nicht, deshalb ist es explizit einzustellen. Mit `AddLanguage` werden für einzelne Sprachen (üblicherweise `de`, `en` usw.) die zu verwendenden Dateiendungen (`.de`, `.en` usw.) festgelegt. `LanguagePriority` steuert, welches Dokument zu liefern ist, wenn der Client keine Priorität äußert. Das ist beispielsweise der Fall, wenn `Accept-Language` in der Anfrage fehlt. Die zuerst aufgeführte Sprache hat die höchste Priorität. Für korrekte Anfragen gemäß HTTP 1.1 hat `LanguagePriority` keinen Effekt.

Die deutsche Version erhält den Dateinamen `test.html.de` und entsprechend beinhaltet die Datei `test.html.en` das Dokument in der englischen Version. Empfängt Apache nun eine Anfrage nach der nicht existierenden Datei `test.html`, sucht er nach Dateien, die zur Maske `test.html.*` passen. Dabei werden im Beispiel die Dateien `test.html.de` und `test.html.en` gefunden. Für diese beiden Dateien erzeugt Apache automatisch eine temporäre Variantendatei, wobei er die Sprachversionen anhand der Dateiendungen zuordnet.

Eine Anfrage nach `test` ergibt die Suchmaske `test.*`, was im Beispiel zum gleichen Ergebnis führt. Werden die sprachspezifischen Dateien dagegen mit `test.de.html` und `test.en.html` benannt, liefern nur noch Anfragen nach `test` – nicht aber nach `test.html` – das gewünschte Ergebnis. Für die Erkennung des MIME-Typs spielt die Reihenfolge der Dateiendungen (beispielsweise `.de` und `.html`) dagegen keine Rolle. Interessant ist, dass Links, die keine Dateiendung beinhalten wie `test`, nicht nur in beiden Fällen funktionieren, sondern es auch noch ermöglichen, die Dateiendung nachträglich zu ändern, z. B. von `.html` auf `.shtml`, ohne die Links ändern zu müssen. Falls die Suchmaske keine Treffer liefert, wird der Fehler “404 Not Found” gemeldet.

4 Kein Anschluss unter dieser Nummer...

Wenn beispielsweise im Browser eine Sprache eingestellt ist, zu der keine Version des Dokuments existiert, generiert Apache eine Fehlerseite (“406 Not Acceptable”), die eine Übersicht der zur Verfügung stehenden Dokumente enthält, aus der die Besucher auswählen können (siehe Abb. 3).

Statt der von Apache generierten Übersicht kann auch ein selbst definiertes Standarddokument angezeigt werden, indem in die Variantendatei ein Eintrag aufgenommen wird, für den keine Sprache spezifiziert wird (siehe Abb. 4).

Im Beispiel wird das Dokument `test.html` geschickt, wenn `Accept-Language` weder `de` noch `en` beinhaltet.

5 Bilder

Auch für Bilder, die in verschiedenen Sprachen vorliegen, ist Content Negotiation analog zu HTML-Dokumenten einsetzbar. Zwei Bilddateien `bild.de.gif`

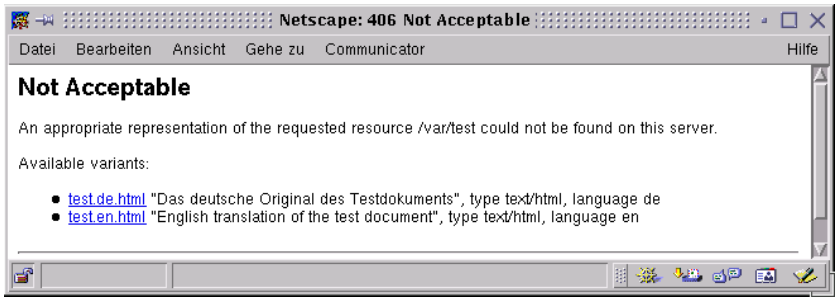

Abbildung 3. Kein passendes Dokument vorhanden

```
URI: test

URI: test.de.html
Content-Type: text/html
Content-Language: de

URI: test.en.html
Content-Type: text/html
Content-Language: en

URI: test.html
Content-Type: text/html
```

Abbildung 4. Variantendatei `test.var` mit Standarddokument

und `bild.en.gif` können mit zugehöriger Variantendatei `bild.var` oder per Multiviews einfach mit `bild` angesprochen werden (siehe Abb. 5).

6 Was geht sonst noch?

Neben `Accept-Language` gibt es noch die HTTP-Header `Accept`, `Accept-Charset` und `Accept-Encoding`. Damit lassen sich außer der Sprache auch der MIME-Typ, der Zeichensatz und die Kodierung variieren. Letzteres kann man beispielsweise nutzen, um zu umfangreichen HTML-Dokumenten komprimierte Versionen anzubieten, wodurch die Übertragungszeit reduziert werden kann, wenn der Client in der Lage ist, die Datei zu decodieren. Die Eintragungen in der Datei `doc.var` bewirken, dass Clients, die mit `gzip` gepackte Dateien nicht decodieren können, den Inhalt als (ungepacktes) HTML-Dokument erhalten (siehe Abb. 6).

7 Qualitätsaspekte

Die Bildqualität von Grafiken hängt vom verwendeten Dateiformat ab. Liegt beispielsweise eine technische Zeichnung in den Dateien `grafik.gif` und `gra-`

```
URI: bild

URI: bild.de.gif
Content-Type: image/gif
Content-Language: de
Description: "Bild mit deutschem Text"

URI: bild.en.gif
Content-Type: image/gif
Content-Language: en
Description: "Image with English text"
```

Abbildung 5. Sprachvarianten für Bilder: `bild.var`

```
URI: doc

URI: doc.html.gz
Content-Type: text/html
Content-Length: 12038
Content-Encoding: x-gzip

URI: doc.html
Content-Type: text/html
Content-Length: 33952
```

Abbildung 6. Variantendatei `doc.var` für eine komprimierte Datei

`fik.jpg` vor, so liefert die GIF-Datei in der Regel das bessere Ergebnis – bei einem Urlaubsbild wäre es umgekehrt. Kann ein Browser beide Bildformate darstellen, möchten wir ihm daher die GIF-Datei schicken. Dies erreicht man, indem innerhalb der Variantendatei für die GIF-Datei ein höherer Qualitätswert ("*quality score*") angegeben wird als für die JPG-Datei.

Qualitätswerte definieren die relative Qualität einer Variante im Vergleich zu den anderen. Sie liegen im Wertebereich 0.000 bis 1.000, wobei eine Variante mit dem Qualitätswert 0 nie ausgewählt wird. Fehlt die Angabe, wird sie von Apache auf 1 gesetzt. Für unser Beispiel erzeugen wir die Variantendatei `grafik.var` (siehe Abb. 7).

Eine Anfrage nach `grafik.var`, die den HTTP-Header `Accept: image/gif; q=0.5, image/jpeg;q=0.8` enthält, bevorzugt von sich aus die JPG-Datei gegenüber der GIF-Datei. Apache multipliziert die Qualitätswerte der Anfrage mit denen der Variantendatei, woraus hier die Werte 0.45 für GIF und 0.40 für JPG resultieren. Da die GIF-Variante den höchsten Wert erreicht, wird sie ausgewählt. Fehlen die Qualitätswerte in der Anfrage, werden sie auf 1 gesetzt.

```
URI: grafik

URI: grafik.gif
Content-Type: image/gif; qs=0.9
Content-Length: 1245
Description: "Grafik als GIF (gute Qualität)"

URI: grafik.jpg
Content-Type: image/jpeg; qs=0.5
Content-Length: 915
Description: "Grafik als JPEG (nicht so gute Qualität)"
```

Abbildung 7. Qualitätswerte für Grafikdateien: `grafik.var`

8 Cache

Da sich beim Einsatz von Content Negotiation hinter einer URL verschiedene Dokumente verbergen, können die Antworten des Web-Servers nicht ohne weiteres im Cache gespeichert werden. Es muss nämlich sichergestellt werden, dass eine Anfrage inklusive der Parameter (Accept-Header) zum Dokument im Cache passt. HTTP 1.1 verfügt hier über entsprechende Möglichkeiten, die Apache auch unterstützt. Für HTTP 1.0 kennzeichnet Apache die Antworten dagegen so, dass sie nicht in den Cache aufgenommen werden, was sich mittels `CacheNegotiatedDocs` abschalten lässt. Für Anfragen gemäß HTTP 1.1 hat `CacheNegotiatedDocs` keinen Effekt.

9 Fazit

Multiviews sind einerseits komfortabler als Varianten, andererseits aber nicht so mächtig, weil keine Detaileinstellungen wie z. B. für Qualitätswerte vorgenommen werden können. Für verschiedene Sprachversionen eignen sie sich aber ausgezeichnet. Insgesamt ist Content Negotiation ein vielseitiges Feature, das nicht nach dem Auswechseln der Standardindexseite des Apache in der Versenkung verschwinden sollte.

Literatur

1. Apache Homepage, http://www.apache.org/
2. Glahn, Kay: Winnetous Erben, Linux Enterprise, Heft 1/2000, S. 54–59
3. ISO 639: Code for the representation of names of languages, International Information Centre for Terminology (INFOTERM), 1988

A Apache einrichten

Zum Nachvollziehen der Beispiele wird eine funktionsfähige Installation des Apache Web-Servers vorausgesetzt. Die meisten Linux-Distributionen installieren den Apache per Vorgabe zusammen mit dem Betriebssystem. Auf der Apache Homepage (siehe [1]) wird die aktuelle Version zum Download angeboten. Zum Zeitpunkt der Entstehung dieses Artikels war dies die Version 1.3.12. Einen Überblick über die Apache Software Foundation gibt [2].

Per Voreinstellungen wird Apache im Verzeichnis `/usr/local/apache/` installiert. (Einige Linux-Distributionen wie z. B. *SuSE* halten sich nicht an diese Verzeichnisvorgaben.) Die Konfigurationsdateien – insbesondere `httpd.conf` – befinden sich im Unterverzeichnis `conf`. HTML-Dokumente und alles, was dazugehört, liegen im Unterverzeichnis `htdocs`. Gestartet werden kann der Web-Server mit `/usr/local/apache/bin/apachectl start`, sofern dies nicht schon automatisch beim Systemstart geschieht. Ob der Web-Server läuft, sieht man durch einen Zugriff auf `http://localhost/`.

Content Negotiation wird durch das Modul `mod_negotiation` bereitgestellt, das per Vorgabe aktiviert ist.

Einstellungen für den Web-Server können global in der Datei `httpd.conf` vorgenommen werden und lokal innerhalb von Dokumentverzeichnissen mittels `.htaccess`-Dateien. Damit letzteres für die Beispiele hier funktioniert, müssen in `httpd.conf` die beiden folgenden Zeilen vorhanden sein:

```
AccessFileName .htaccess
AllowOverride All
```

Damit Änderungen in `httpd.conf` vom Apache angenommen werden, ist ein Neustart notwendig.

B Content Negotiation auf Protokollebene

Auf HTTP-Ebene sieht eine vom *Netscape Navigator* formulierte Anfrage so aus:

```
GET /test.var HTTP/1.0
Connection: Keep-Alive
User-Agent: Mozilla/4.74 [de] (X11; U; Linux 2.2.16 i586)
Host: localhost:80
Accept: image/gif, image/x-xbitmap, image/jpeg, image/png, */*
Accept-Encoding: gzip
Accept-Language: de, en
Accept-Charset: iso-8859-1,*,utf-8
```

Apache antwortet darauf mit:

```
HTTP/1.1 200 OK
Date: Sun, 01 Oct 2000 21:06:07 GMT
Server: Apache/1.3.12 (Unix)
Content-Location: test.de.html
Vary: negotiate,accept-language
TCN: choice
Last-Modified: Sun, 01 Oct 2000 19:25:24 GMT
ETag: "388b2-89-39d78fa4;388b4-f9-39d78fa4"
Accept-Ranges: bytes
Content-Length: 137
Connection: close
Content-Type: text/html
Content-Language: de
Expires: Sun, 01 Oct 2000 21:06:07 GMT

<!DOCTYPE html PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html>
<head><title>Test Deutsch</title></head>
<body><h1>Deutsch</h1></body>
</html>
```

Bei der Antwort ist kein Unterschied zwischen Multiviews und Varianten festzustellen.