Teil I.

Praxisteil

1. Versuchsaufbau

Der Optimierungsversuch kann leider nicht auf dem Livesystem durchgeführt werden, da dass die Stabilität beeinträchtigen würde. Deswegen wurde ein Entwicklungssystem genutzt, um die verschiedenen Methoden zu analysieren. Das Entwicklungssystem ist eine neuere Anschaffung der pludoni GmbH und dadurch, im Vergleich zum Livesystem, leistungsfähiger. Dadurch wird sich das Endergebnis, wenn es nach dem Abschluss der Arbeit auf das Livesystem übertragen wurde, in seinen Kenndaten unterscheiden. Trotz alledem werden die Verbesserungen relativ zum Testsystem proportional ausfallen.

- Testplattform: pludoni Server eq4
- Prozessor: Intel® Core™ i7-920
- Arbeitsspeicher: 8 GB DDR3 RAM
- Festplatten: 2 x 750 GB SATA-II HDD (Software-RAID 1)
- Netzwerkverbindung: 100Mbit
- Server Software: Apache/2.2.16 (Module)
- PHP Version: PHP 5.2.6-1+lenny13 with Suhosin-Patch 0.9.6.2
- Mysql Version: 14.12 Distrib 5.0.51a
- Betriebssystem: Debian 4.1.2-25 mit Kernel 2.6.26-26lenny2
- Serverstandort: Rechenzentrumspark Falkenstein

2. Testverfahren

Das Testen von Web-Performance hat sich in den letzten Jahren stetig weiterentwickelt. Eine Zeit lang galten sogenannte Backbonetests[1] als das Non-Plus-Ultra. Nachdem

[1] Tests, die direkt am Backbone, also im Rechenzentrum durchgeführt werden. Führt zu besseren Ergebnissen, da der Client und der Testserver direkt verbunden sind. Dieser Test spiegelt aber

aber immer deutlicher wurde, dass die tatsächlichen Ergebnisse sich stark von den Erwartungen unterscheiden, suchte man nach Alternativen. Der verlässlichste Test wird natürlich direkt beim Kunden beziehungsweise der Zielgruppe durchgeführt. Da es oft schwierig ist diese Tests durchzuführen, sollte zumindest mit ähnlichen Vorraussetzungen getestet werden.

2.1. Servertest

Die rein serverseitigen Optimierungen wurden mit „ab“ getestet. „ab“, das für „apache bench“ steht, wurde entwickelt, um Apache Server zu testen. Es analysiert, wie lange ein Server braucht, um Webseiten auszuliefern und wieviele Anfragen er pro Sekunde befriedigen kann, ohne dass es zu Ausfällen kommt. „ab“ führt die gewünschte Anzahl an Anfragen hintereinander, mit der gewählten Anzahl an nebenläufigen Zugriffen (Concurrency) auf die Ressource, aus und berechnet dann anhand der Ergebnisse die durchschnittliche Antwortzeit. Der Kommandozeilenbefehl, der zum Testen genutzt wurde, war denkbar einfach:

Listing 1: „ab“ mit Parametern

```
1 ab -n 50 -c 1 http://itsax.it-jobs-und-stellen.de/
```

Dies führt 50 Anfragen mit einer Concurrency von 1 auf itsax.it-jobs-und-stellen.de.

2.2. Browsertest

Browsertests können durch die komplexen Voraussetzungen keine exakten Ergebnisse für alle möglichen Konfigurationen liefern. Ein Netbook zum Beispiel wird eine Webseite immer langsamer darstellen als ein Desktop-PC der neuesten Generation. Man kann natürlich am eigenen Computer Tests durchführen. Dafür gibt es aber auch Programme, wie Firebug[2] und YSlow[3], die auch Verbesserungsvorschläge liefern können. Da aber möglichst objektiv getestet werden soll, ist es von Vorteil, eine standardisierte und automatisierte Umgebung zu nutzen. Für diesen Zweck bietet sich WebpageTest[4] an. Es wurde ursprünglich für den internen Gebrauch bei AOL entwickelt, steht aber seit 2008 unter der BSD-Lizenz und kann somit frei genutzt werden. Es ist eine umfassende Testsuite für Webseiten. Besonders die verschiedenen Analysen, die automatisch

nicht die Wirklichkeit wieder.

[2] Entwicklertool für Firefox `https://addons.mozilla.org/de/firefox/addon/firebug/`

[3] Erweiterung für Firebug, die Webseiten bewerten kann. `https://addons.mozilla.org/de/firefox/addon/yslow/`

[4] Link: `https://sites.google.com/a/webpagetest.org/docs/`

durchgeführt werden, sind positiv hervorzuheben. Das Programm ist in der Lage, die Webseite nach den „Google Page Speed“-Kriterien zu bewerten und ausführliche Inhaltsanalysen, aufgeschlüsselt nach Multipurpose Internet Mail Extensions (MIME)-Typen, zu liefern. Zusätzlich wird ein sehr hilfreiches Wasserfalldiagramm erzeugt und für jedes Element der Webseite ein Performance-Review erstellt. Darin werden unter anderem Kriterien wie Kompression und Caching betrachtet. Als besonderes Feature gibt es noch einen Videovergleich, in dem man Webseiten gegeneinander antreten lassen kann. Man kann WebpageTest sowohl selbst installieren als auch externe Anbieter nutzen. In diesem Fall wird www.webpagetest.org mit folgenden Parametern genutzt:

- 10 konsekutive Tests
- Standort des Clients: Frankfurt a. M.a
- Browser: IE9
- Connection: DSL 1,5 Mbps / 50ms RTT
- Only First View[5]

2.3. Andere Analyseverfahren

Profiling und Debugging auf Codeebene wurden nur sporadisch eingesetzt, da sie den Anwendungsfall Webseite nur eingeschränkt betreffen. Durch Caching kann die Codeausführungszeit größtenteils eliminiert werden. Bei Anwendungsfällen, die kein Caching erlauben, wie zum Beispiel eine Suche oder eine Applikation, ist es aber sehr hilfreich diese Tools zu verwenden. Sie ersparen viel Arbeit, da die Problemstellen schnell ausgemacht werden können. Von Hand diese Arbeiten durchzuführen, ist in kleinen Projekten sehr mühsam und in großen Projekten fast unmöglich. Jeder Entwickler sollte solche Programme in seinem Repertoire haben.

3. Ausgangszustand

Sreeram Ramachandran, ein Software Ingenieur der Firma Google, hat eine Analyse über 4.2 Milliarden Seiten veröffentlicht. Diese ist im Rahmen der Initiative „Let's make the web faster“ entstanden und zeigt häufige Fehlerquellen und ungenutztes Potential auf. Die durchschnittliche Webseite hat laut Ramachandran 320 kB Größe, 44 verschiedene Ressourcen und es werden nur 66% der komprimierbaren Inhalte tatsächlich

[5] Nur ungecachte Zugriffe werden getestet.

komprimiert.[Google, 2011a] ITsax.de hat 106 Ressourcen und 444 kB an Daten. Schon anhand dieser zwei Zahlen lässt sich eine vergleichsweise schwache Leistung vorhersehen. Besonders die Anzahl an verschiedenen Ressourcen deutet auf Missstände hin, da Parallelisierung von Zugriffen nur bis zu einem bestimmten Grad möglich ist. Die Time to First Byte (TtFB) von 674ms bezeichnet die Zeit, die vergeht bis der Webbrowser die ersten Daten empfängt. Der Nutzer hat aber zu diesem Zeitpunkt noch keinen Inhalt präsentiert bekommen. Die Inhalte werden erst angezeigt, nachdem die Time to Start Render (TtSR) vergangen ist. Der Nutzer muss demnach ungefähr zwei Sekunden warten, bis die Webseite im Browser anfängt sich aufzubauen. Die Load Time (LT) bezeichnet dann die Zeit die vergeht, bis die Seite komplett dargestellt wird und der Benutzer sie ohne Einschränkungen bedienen kann. Es können aber auch nach der LT noch Inhalte nachgeladen werden, wie zum Beispiel gestreamte Videos oder andere asynchrone Inhalte. Diese nachgeladenen Inhalte wirken sich aber nicht mehr negativ auf die User Experience aus, solange sie im Rahmen bleiben und nicht wichtige Teile der Webseite, wie zum Beispiel das Hauptmenü, noch per Flash geladen werden müssen. Die Anzahl der DOM Elemente bezeichnet alle vom Browser zu verarbeitenden Objekte und ist ein Indikator für die Komplexität der Webseite. Je mehr Elemente also vorhanden sind, desto länger muss der Browser die Positionierung und Darstellung berechnen. Mit 855 DOM Elementen gehört ITsax.de leider schon zu den komplexeren Seiten, was zu einem Großteil Drupal anzurechnen ist, welches auf Kosten der Performance andere Aspekte, wie Konfigurierbarkeit, in den Vordergrund stellt. Die Inhalte auf der Seite Itsax.de wurden mit dem Analysetool webpagetest.org ausgewertet. Die Ergebnisse sind in Abbildung 1 und 2 dargestellt, einmal in Bezug auf Größe und einmal aufgeschlüsselt nach der Anzahl der benötigten Requests, um die Inhalte vom Server anzufordern. Die numerischen Werte sind in der Tabelle 1 respektive Tabelle 2 erfasst.

Typ	**Byte**
image/jpeg	142251
image/png	101428
javascript	84758
image/gif	73378
text/css	30222
text/html	29917

Typ	**Anzahl**

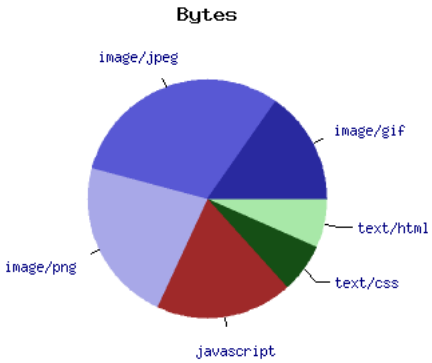

Abbildung 1: Größe der Inhalte

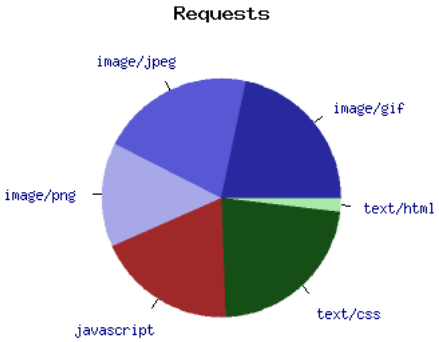

Abbildung 2: HTTP Anfragen

text/css	24
image/gif	23
image/jpeg	22
javascript	20
image/png	15
text/html	2

Leicht zu erkennen ist, dass der HTML Anteil, sowohl bei der Größe als auch bei Anzahl der HTTP Anfragen, eine untergeordnete Rolle spielt. Nicht vergessen werden darf aber die Zeit, die der Server benötigt, um den HTTP Code zu generieren. Dies kann man

sehr gut im Wasserfalldiagramm in Abbildung 3 erkennen, in welchem der Start des Ladevorgangs hervorgehoben wurde. Bevor der Initiale GET Request abgeschlossen ist, weiß der Browser noch nicht, welche Ressourcen er laden muss und es können noch keine anderen Aktionen ausgeführt werden.

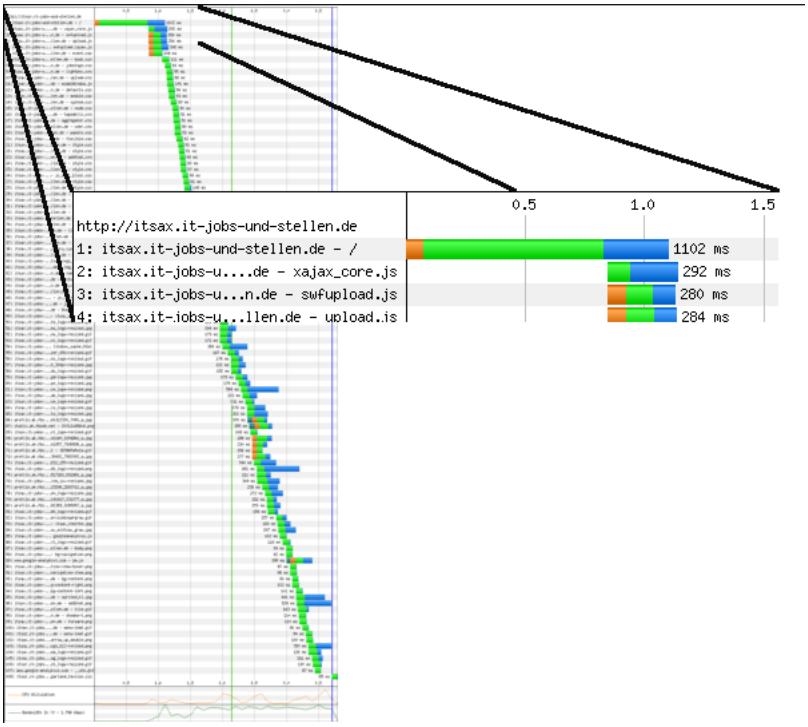

Abbildung 3: Wasserfalldiagramm: Ausgangszustand

Listing 2: Startwerte gemessen mit ab

```
1  Server Software:        Apache/2.2.16
2  Server Hostname:        itsax.it-jobs-und-stellen.deDocument Path:
                 /
3  Document Length:        65218 bytes
4
5  Concurrency Level:      1
6  Time taken for tests:   18.182 seconds
7  Complete requests:      50
8
9  Write errors:           0
10 Total transferred:      3266726 bytes
11 HTML transferred:       3239226 bytes
12 Requests per second:    2.75 [#/sec] (mean)
13 Time per request:       363.640 [ms] (mean)
14 Time per request:       363.640 [ms] (mean, across all concurrent
       requests)
15 Transfer rate:          175.46 [kBytes/sec] received
```

4. Implementierung und Test der einzelnen Methoden

Im folgenden Abschnitt werden die eingesetzten Methoden dargestellt und ihre Auswirkungen auf die Webperformance der Seite ITsax.de aufgezeigt. Dabei werden alle Methoden, die vom Autor, nach der Analyse des Ausgangszustands, für möglicherweise sinnvoll befunden wurden, getestet. Die Methode wird jeweils mit dem Startwert verglichen und der Aufwand eingeschätzt.

4.1. Alternative PHP Cache

Als allererste Maßnahme wurde ein PHP-Opcode-Cache untersucht. Opcode-Caches speichern das Kompilat vorangegangener Operationen und können so beim nächsten Aufruf direkt die Berechnung ausführen. Solche Systeme sind immer zu empfehlen, da sie normalerweise keinerlei Probleme bereiten und alle PHP Anfragen beschleunigen. Aktuell gibt es mehrere dieser, auch PHP-Beschleuniger genannte, Opcode-Caches. Unter ihnen nimmt aber der Alternative PHP Cache (APC)[6] eine führende Position ein. Hauptsächlich weil es der zuverlässigste unter den nicht-kommerziellen Caches ist. Ab PHP Version 5.4 wird er in PHP schon enthalten sein. Die Installation von APC

[6] `http://pecl.php.net/package/APC`

ist auf einigermaßen aktuellen Linux Servern auch denkbar einfach. Das PHP-Pear-Framework, was oft schon installiert ist, erlaubt mit einigen wenigen Befehlen einen funktionierenden Opcode-Cache zu installieren.

Listing 3: Installation von APC

```
1 apt-get install php-pear
2 pecl install apc
```

Nach der Installation wurde die Antwortzeit des Apacheservers gemessen und es kam zu folgendem Ergebnis. Statt 363 ms wurden nur noch 233 ms benötigt. Das bedeutet eine Verbesserung um 36% und es können 4,28 Anfragen je Sekunde bearbeitet werden gegenüber vorher nur 2,75. Alles unter der Prämisse, dass es keine nebenläufigen Anfragen gibt.

Listing 4: APC gemessen mit ab

```
1  Server Software:        Apache/2.2.16
2  Server Hostname:        itsax.it-jobs-und-stellen.de
3  Server Port:            80
4
5  Document Path:          /
6  Document Length:        64842 bytes
7
8  Concurrency Level:      1
9  Time taken for tests:   11.681 seconds
10 Complete requests:      50
11
12 Write errors:           0
13 Total transferred:      3261730 bytes
14 HTML transferred:       3234230 bytes
15 Requests per second:    4.28 [\#/sec] (mean)
16 Time per request:       233.620 [ms] (mean)
17 Time per request:       233.620 [ms] (mean, across all concurrent
       requests)
18 Transfer rate:          272.69 [kBytes/sec] received
```

4.2. Drupal Cache

Der Drupal Cache wurde mit Version 5 eingeführt und kann über das Administrationsinterface aktiviert werden. Durch ihn wird ein Caching aller Seiten ausgelöst. Dabei werden die fertig berechneten Webseiten beim ersten Aufruf komplett in der Datenbank

abgelegt. Wenn eine schon gecachte Seite aufgerufen wird, muss nur noch der fertige HTML-Code aus der Datenbank geholt werden. Diese Methode ist aber zu unflexibel, da nicht konfigurierbar ist, welche Seiten gecacht werden sollen. Für die Webseite ITsax.de müssen aber dynamische Inhalte ungecacht bleiben. Testweise wurde der Cache aktiviert und mit „ab“ getestet. Die Ergebnisse sind in folgender Ausgabe zu sehen.

Listing 5: Drupal Cache gemessen mit ab

```
1  Server Software:        Apache/2.2.16
2  Server Hostname:        itsax.it-jobs-und-stellen.de
3  Server Port:            80
4
5  Document Path:          /
6  Document Length:        65005 bytes
7
8  Concurrency Level:      1
9  Time taken for tests:   2.082 seconds
10 Complete requests:      50
11 Failed requests:        0
12 Write errors:           0
13 Total transferred:      3276900 bytes
14 HTML transferred:       3250250 bytes
15 Requests per second:    24.02 [\#/sec] (mean)
16 Time per request:       41.638 [ms] (mean)
17 Time per request:       41.638 [ms] (mean, across all concurrent requests
       )
18 Transfer rate:          1537.09 [kBytes/sec] received
```

Die Zeit pro Anfrage wird deutlich kleiner und es wird klar, dass Caching für den serverseitigen Teil die größten Verbesserungen bringt. Die PHP-Codeausführung kann auf ein notwendiges Minimum, dass heisst auf die Momente in denen der Cache neu geschrieben wird, reduziert werden. So kann zumindest der zu berechnende HTML-Code direkt ausgeliefert werden.

4.3. JS Aggregation und Minifizierung mit dem Javascript Aggregation Modul

Aggregation und Minifizierung sind Verfahren, die in der Web-Performance-Optimierung häufig eingesetzt werden. Für das Drupal 5 System gibt es fertige Module, die diese Aufgabe übernehmen. Das Modul interveniert innerhalb des Drupal Kerns. Dort werden die Javascripte ersetzt, die vorher direkt so ausgegeben wurden wie sie die Module

lieferten. Stattdessen wird eine einzelne, das heisst aggregierte Version, die optional minifiziert werden kann, ausgeliefert. Diese Minifizierung wird natürlich genutzt und spart einige kB. Ermöglicht wird dies durch die Nutzung von JSmin[7]. JSmin ist ein Filter, der unter anderem Kommentare entfernt und mehrere Leerzeichen zu einem zusammenfasst. Durch die Nutzung dieser Methode konnten 11 HTTP Abfragen und 11 kB an Datenvolumen eingespart werden.

Tabelle 1: Ergebnis der JS Aggregation und Komprimierung

Load Time:	3.658s
Time to First Byte:	0.595s
Time to Start Render:	1.915s
#DOM Elements:	844
#Requests:	95
Bytes In:	432 kB

4.4. CSS Aggregation und Komprimierung

Analog zur Javascript Optimierung kann man auch die CSS Aggregation betrachten. Dabei werden durch Zusammenfassung der einzelnen CSS Dateien HTTP Abfragen eingespart. Wie man an der gesunkenen Anzahl der Requests sehen kann, wurden 21 Abfragen nur durch Zusammenfügen der einzelnen CSS Dateien zu einer einzigen eingespart. Durch die Bereinigung beziehungsweise Komprimierung werden dabei außerdem 17 kB an überflüssigen Zeichen entfernt.

4.5. Drupal Boost Modul

Das Boost Modul ist ein Cache für statische Seiten, der allerdings nur für nichtangemeldete Gastnutzer funktioniert. Da ITsax.de fast ausschließlich von Gastnutzern besucht wird, bietet es sich an, dieses Modul zum Cachen kompletter Seiten zu nutzen. Dabei können HTML, XML, Ajax, CSS und Javascript Dokumente gecacht werden. Zusätzlich können diese durch gzipping komprimiert werden. Die genutzen Techniken sind sehr performant und bauen auf einem Dateisystemcache auf. Das bedeutet, jede

[7] Bibliothek zur Minifizierung von JavaScript. `https://github.com/rgrove/jsmin-php/`

Tabelle 3: Ergebnis der CSS Aggregation und Komprimierung

Load Time:	3.577s
Time to First Byte:	0.649s
Time to Start Render:	1.577s
#DOM Elements:	834
#Requests:	85
Bytes In:	427 kB

Tabelle 5: Ergebnis der Aktivierung des Drupal Boost Moduls

Load Time:	3.233s
Time to First Byte:	0.172s
Time to Start Render:	1.473s
#DOM Elements:	856
#Requests:	106
Bytes In:	444 kB

Seite wird komplett auf der Festplatte abgelegt. Alle Arten von serverseitigen Prozessen werden so nach dem initialem Seitenaufruf, der das Caching auslöst, nicht mehr durchlaufen. Dies sieht man sehr gut an der Time to First Byte. Von den 172 ms werden ca. 50 ms für den DNS Lookup benötigt und 70 ms für die erste HTTP Verbindung. Der Server braucht demnach nur ca. 50 ms um die Seite auszuliefern. Dies hat natürlich einen sehr positiven Einfluss auf die Gesamtperformance und macht sich besonders beim Endnutzer bemerkbar, da die Seite spürbar schneller anfängt, sich im Browser aufzubauen.

4.6. Bildkompression mit jpegoptim und OptiPNG

Bilder und Grafiken bieten oft großen Optimierungsspielraum. Zum einen erfolgt dies durch die richtige Auswahl der Dateiformate und zum anderen durch Komprimierung der Bilder. Da es auf www.itsax.de nicht nur statische Inhalte gibt, sondern auch durch Communitymitglieder und Communitymanager eingestellte Inhalte verwaltet werden müssen, sollte eine nachträgliche Qualitätsoptimierung der hochgeladenen Bilder durchgeführt werden. Um dies umzusetzen, sind die Programme OptiPNG und jpegOptim zu empfehlen. Da es sich bei beiden Programmen um Kommandozeilenprogramme handelt, kann man ihre Anwendung leicht automatisieren. Mit dem Linuxbefehl „find“, der praktischerweise eine Möglichkeit, Befehle auszuführen, besitzt, kann man direkt die entsprechenden Dateien an die Optimierer übergeben. Diese Aktionen können dann über einen Cronjob periodisch jede Nacht ausgeführt werden.

4.6.1. Verlustfreie Kompression

Technisch wird eine verlustfreie Kompression durch Verfahren wie die Huffman-Codierung oder die Lauflängenkodierung umgesetzt. Im Fall des PNG-Formats wird die Kompri-

Tabelle 7: Ergebnis der verlustfreien Kompression der Bilder

Load Time:	3.640s
Time to First Byte:	0.636s
Time to Start Render:	1.894s
#DOM Elements:	855
#Requests:	106
Bytes In:	429 kB

Tabelle 9: Ergebnis der verlustbehafteten Kompression der Bilder

Load Time:	3.255s
Time to First Byte:	0.669s
Time to Start Render:	1.908s
#DOM Elements:	855
#Requests:	106
Bytes In:	371 kB

mierungsmethode Deflate genutzt. Außerdem werden Vorfilter, in Form von prädikativer Kodierung, eingesetzt. Diese berechnen die wahrscheinlichen Farbwerte und es müssen nur die Abweichungen gespeichert werden.

Die Befehle sehen dann wie folgt aus:

Listing 6: verlustfreies Optimieren mit "find"

```
1 find . -name "*.png" -exec optipng -o7 {} \;
2 find . -name "*.jpg" -exec jpegoptim {} \;
```

Auf jeden Fall sollte eine verlustfreie Komprimierung durchgeführt werden, da die Bilder in diesem Fall nur an Dateigröße verlieren und die Bildqualität unberührt bleibt.

4.6.2. Verlustbehaftete Kompression

Da im Web kein besonders hoher Detailgrad gewährleistet werden muss, ist auch eine Kompression erwünscht, die auf Kosten der Bildqualität die Bildgröße verringert.

Listing 7: verlustbehaftetes Optimieren mit find

```
1 find . -name "*.png" -exec optipng -o7 {} \;
2 find . -name "*.jpg" -exec jpegoptim {} \;
```


4.7. Drupal 5 Fehler bei umgefärbten Themes

Das Framework Drupal 5 benutzt Themes zur Gestaltung der Oberfläche. Um diese farblich anpassen zu können, wurde das Color Modul installiert, welches Themeveränderungen ermöglicht. Aufgrund der Tatsache, dass das Theme nur kopiert wird und im Anschluss die Farben geändert werden, entstehen bei diesem Vorgang unnötige Duplikate, die beim Laden der Seite mitgeschleppt werden. Um diese zu entfernen, wird

Tabelle 11: Ergebnis der Fehlerbereinigung des Theme Moduls

Load Time:	3.626s
Time to First Byte:	0.629s
Time to Start Render:	1.890s
#DOM Elements:	854
#Requests:	104
Bytes In:	440 kB

Tabelle 13: Ergebnis des Spritings der Theme Bilder

Load Time:	3.707s
Time to First Byte:	0.669s
Time to Start Render:	1.968s
#DOM Elements:	855
#Requests:	103
Bytes In:	443 kB

einfach das Standardtheme durch das Modifizierte ersetzt. Dafür müssen nur noch einige Pfade in der style.css angepasst werden und man spart in dem Fall von itsax.de 4 kB, was immerhin ca 1% der übertragenen Datenmenge ausmacht. Umgesetzt wurde diese Maßnahme, indem die Duplikate manuell zu einer Datei zusammengeführt wurden. Oftmals müssen nur Farbangaben ersetzt werden, um die gewünschten Resultate zu erzielen. Danach wird das Theme durch die zusammengeführte Datei ersetzt. Das Color-Modul kann im Anschluss deaktiviert werden.

4.8. Theme Bilder spriten

Spriting wurde ursprünglich in der Videospielentwicklung verwendet, um Bilder in den Grafikspeicher zu laden. In der Webentwicklung ist es eine effektive Technik, um Bilder ohne mehrfachen Overhead zu laden. Beim Spriting wird aus vielen Einzelbildern eine Gesamtgrafik erstellt. Um auf die Ursprungsbilder dann einzeln zugreifen zu können, werden CSS Befehle genutzt, die es ermöglichen, die Größe und die Position eines Bildausschnittes anzuzeigen.

4.9. Module von Startseite entfernen

Um zu überprüfen, welchen Einfluss verschiedene, im Netzwerkgraphen auffällige, Module auf die Gesamtperformance haben, werden sie testweise komplett deaktiviert. So kann man entscheiden, bei welchen Modulen zusätzlicher Programmieraufwand lohnenswert ist.

4.9.1. Partnerslideshow

Das Deaktivieren der Partnerslideshow hat gravierenden Einfluss auf die Gesamtperformance. Das Modul lädt im aktiven Zustand alle Bilder, die es benötigt und bremst da-

Tabelle 15: Ergebnis der Deaktivierung der Partnerslideshow

Load Time:	2.695s
Time to First Byte:	0.636s
Time to Start Render:	1.838s
#DOM Elements:	790
#Requests:	80
Bytes In:	276 kB

Tabelle 17: Ergebnis der Deaktivierung des Facebook-Widgets

Load Time:	3.478s
Time to First Byte:	0.668s
Time to Start Render:	1.966s
#DOM Elements:	779
#Requests:	95
Bytes In:	406 kB

mit den gesamten Ladevorgang aus. Die Ladezeit verringert sich um 27,7%. Der größte Teil des Geschwindigkeitszuwachses ist der Verringerung der Übertragungsmenge zuzuschreiben.

4.9.2. Facebook Widget

In diesem Fall ist der Einfluss auf die Performance geringer, aber mit einer Verbesserung von 6,7% auf jeden Fall vorhanden. Das Widget besteht aus neun kleinen Fotos und dem Facebookrahmen. Den größten Effekt hat hier die Verringerung der Anzahl der HTTP Anfragen. So können wichtigere Elemente schneller geladen werden.

4.9.3. Jobleiste deaktiviert

Die Jobleiste hat einen geringfügig größeren Einfluss auf die Ladezeiten als das Facebook Widget. Die Ursache ist wahrscheinlich in den zusätzlichen Bibliotheken zu suchen, die zu diesem Modul gehören. Darunter sind Ajax Bibliotheken und anderer ThirdParty Code.

4.10. Umprogrammierung der Module

Um die langsamen Module weiterhin nutzen zu können, muss eine Lösung gefunden werden, die es ermöglicht, Inhalte nachzuladen, nachdem die Seite komplett geladen wurde. Um diese Asynchronität zu erreichen, wird mit Hilfe eines Timeout-Befehls, das Laden der nicht priorisierten Inhalte verzögert. Programmiert wird dieses Verhalten mit Javascript, genauer JQuery. Besonders betrachtet werden muss dabei, ob eine vorhandene Dynamik erhalten bleiben soll. Dazu gehören zum Beispiel zufällige Elemente oder Elemente mit besonders häufigen Aktualisierungen.

Tabelle 19: Ergebnis der Deaktivierung der Jobleiste

Load Time:	3.367s
Time to First Byte:	0.617s

4.10.1. Partnerslideshow

Abbildung 4: Partnerslideshow

Die Partnerslideshow ist eher ein kosmetisches Element der Startseite und für den Nutzer nicht notwendig. Daher kann es so programmiert werden, dass beim Laden nur ein leeres DIV übergeben wird. Dann wird ein Timeout gestartet und bei der Aktivierung des Timeoutevents wird das DIV mit den Slideshowelementen gefüllt sowie die Slideshow gestartet. Der Code wurde dahingehend angepasst, dass der HTML Code in eine Javascript Variable geschrieben wird, anstatt ihn direkt in der Seite einzufügen. Dadurch ist es möglich über eine DOM Manipulation den HTML Code später einzufügen. Das stellt sich dann wie im folgenden Codeaussschnitt dar.

Listing 8: Javascript - Slideshow

```
1  drupal_add_js('
2    if (Drupal.jsEnabled) {
3    $(document).ready(function() {
4       window.setTimeout(delayed_partnerbox,500);
5    });
6    function delayed_partnerbox(){
7      $("#projects").parent().html(partner_box_load);
8      $(".slideshow").cycle({
9          delay:  200 ,
10         height: "80px",
11         width: "180px",
12         containerResize: 0,
13         fit: 1,
14         fx: "fade"
15     });
16   }
17 }', 'inline')
```

An das window Objekt wird in Zeile 3 ein Timer gehängt. Der Timer löst nach 500 ms einen Callback aus, durch den die Funktion delayed_partnerbox ausgeführt wird. Durch diese Verlagerung des Inhalts braucht die Webseite nur noch 2.749s zum Laden. Bis die Webseite die nachgeladene Slideshow anzeigt, vergehen 3.794s. Die Webseite ist also eher benutzbar und später komplett geladen. In Testverfahren, wie zum Beispiel für das Google Ranking wird nur betrachtet, wann die Seite das Onload Event auslöst. Somit wurde die Seite für Google und den Nutzer schneller. Der Performancegewinn geht, wie in der vorherigen Analyse festgestellt, aus der verringerten Datenmenge hervor. Natürlich spielt auch die Reduzierung der HTTP Anfragen eine Rolle. Die Datenmenge wurde auf 284 kB verringert und es wurden statt 107 nur 82 HTTP Anfragen benötigt.

4.10.2. Facebook Widget

Das Modul für das Facebook Widget wurde durch Entwickler der pludoni GmbH so programmiert, dass es das von Facebook zur Verfügung gestellte Widget cacht und dann auf der Webseite darstellt. Das Hauptproblem bei dem Widget ist, dass die Profilbilder, die angezeigt werden, auf den Servern von Facebook liegen und sich somit der internen Kontrolle entziehen. Um das Modul abzukoppeln wird ein Timer analog zum vorhergehenden Modul programmiert.

Tabelle 21: Ergebnis der Umprogrammierung der Partner-Slideshow

Load Time:	2.749s
Time to First Byte:	0.626s
Time to Start Render:	1.832s
#DOM Elements:	855
#Requests:	82
Bytes In:	284 kB

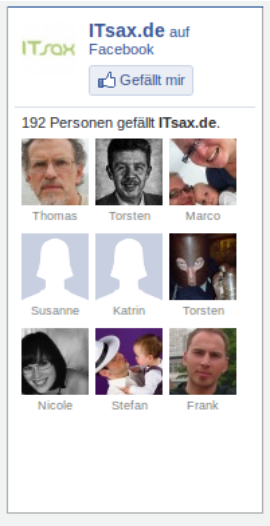

Abbildung 5: Das Facebook-Widget

Listing 9: Facebook Widget

```
1  function facebook_show_likebox() {
2    $local_path = drupal_get_path("module","facebook_pludoni");
3    drupal_add_js('
4    if (Drupal.jsEnabled) {
5        $(document).ready(function() {
6        window.setTimeout(delayed_facebookbox,400);
7      });
8      function delayed_facebookbox(){
9        $("#facebook-likebox").html("<iframe frameborder=\"0\" scrolling=\"
             no\" src=\"/'.$local_path.'/caching/likebox_cache.html\" id=\"
             fbooklikebox\"></iframe>");
10     }
11   }', 'inline');
12   return '<div id="facebook-likebox"></div>';
13 }
```


Wieder wird als initialer Rückgabewert nur ein leeres DIV-Element ausgegeben. Dies-

Tabelle 23: Ergebnis der Umprogrammierung des Facebook-Widgets

Load Time:	3.411s
Time to First Byte:	0.576s
Time to Start Render:	1.848s
#DOM Elements:	857
#Requests:	95
Bytes In:	406 kB

mal kann der HTML Code aber direkt eingefügt werden, ohne den Umweg über eine Javascriptvariable gehen zu müssen. Durch die Verringerung der HTTP Anfragen um 11 und der Webseitengröße um 38 kB kann eine Verbesserung der Ladezeit von 300 ms erzielt werden.

4.10.3. Partneranzeigen

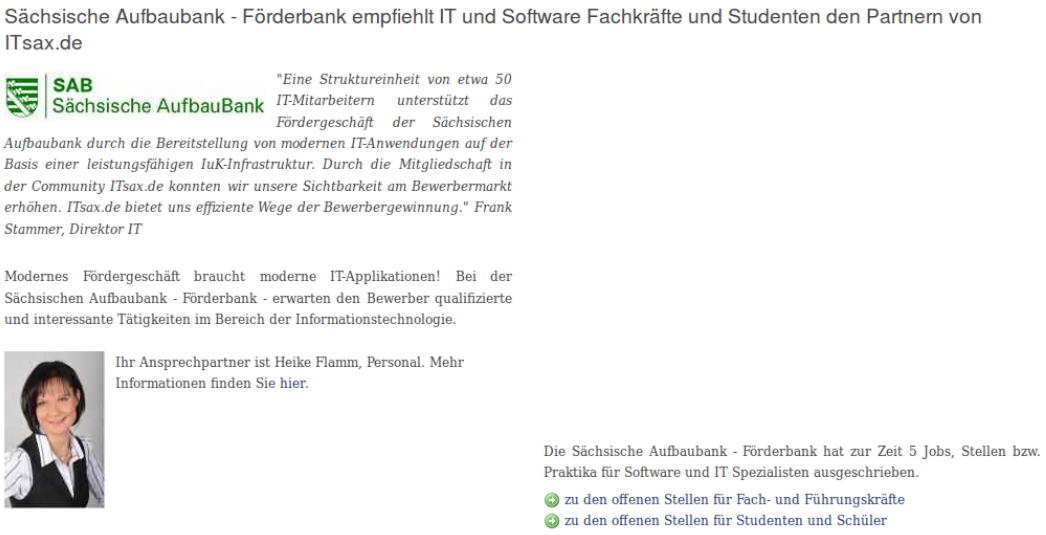

Abbildung 6: Partneranzeigen Block

Das Partneranzeigen-Modul musste umprogrammiert werden, da die Startseite durch das Boost Modul gecacht werden soll. Caching bedeutet normalerweise die Abwesenheit dynamischer oder zufälliger Elemente. Da aber bei jedem Besuch der Webseite

ein zufälliger Communitypartner angezeigt werden soll, musste eine Lösung gefunden werden. Gelöst wurde das Problem durch Auslagerung des Codes in einen Callback, der unter einer anderen URI angesprochen werden kann. Das Boost Modul kann solche Inhalte auch cachen. Es ist aber auch über einen URI Filter konfigurierbar. So kann eine performante Startseite mit dynamischen Inhalten realisiert werden. Die Ladezeit wird durch diesen Umweg leider negativ beeinflusst, aber 100 ms stehen in keinem Verhältnis zu dem Gewinn, der später erzielt wird. Da in einer Drupal Umgebung gearbeitet wird, kann leicht über einen Menü hook dem System die neue URI mitgeteilt werden. Wenn sie, nachdem sie bekannt gemacht wurde, aufgerufen wird, liefert sie den einzufügenden HTML Code aus.

Listing 10: Ajax - hook

```
1  function partneranzeigen_menu() {
2    $items = array();
3    $items[] = array(
4      'path'         => "partanz/ajax",
5      'callback'     => 'partneranzeigen_display_block_1',
6      'type'         => MENU_CALLBACK,
7      'access'       => true
8    );
9    return $items;
10 }
```

Eingefügt wird der HTML Code dann über eine Ajax GET Anfrage. Mit der schon bekannten Timeout Methode.

Listing 11: Ajax - GET

```
1 function delayed_partneranzeigen(){
2   $.get('/partanz/ajax',function(data){
3     $('#partneranzeigenbox').html(data);
4 });
```

Obwohl die Seite um 31 kB kleiner geworden ist und zwei HTTP Anfragen eingespart werden, wird sie, wie schon erwähnt, langsamer.

5. Endergebnis

Nach der Zusammenführung aller Optimierungen wurde die Webseite erneut getestet. Das Wasserfalldiagramm der optimierten Webseite ist in Abbildung 7 dargestellt. Nach nur einer Sekunde fängt der Browser an die Seite darzustellen. 2,913 Sekunden werden

Tabelle 25: Ergebnis der Umprogrammierung des Partneranzeigen-Moduls

Load Time:	3.871s
Time to First Byte:	0.669s
Time to Start Render:	2.255s
#DOM Elements:	857
#Requests:	105
Bytes In:	417 kB

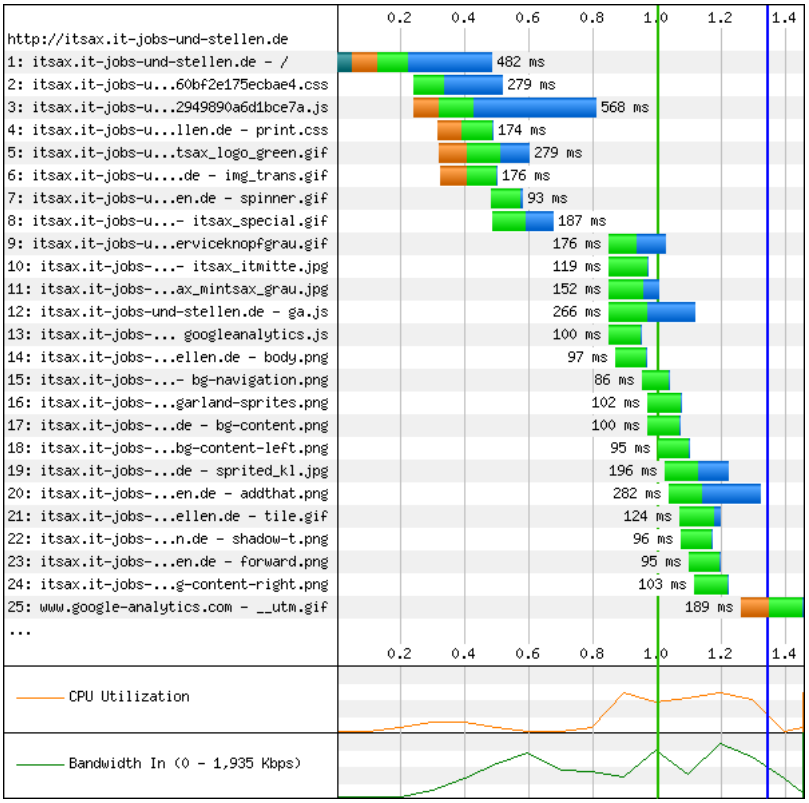

Abbildung 7: Wasserfalldiagramm: Optimierte Webseite

benötigt bis alle asynchronen Inhalte verfügbar sind. Die Webseitengröße wurde auf 138 kB komprimiert. Daher können auch langsame Internetverbindungen die Seite schnell laden. Von den 107 HTTP Anfragen sind nur noch 67 übrig geblieben, von denen

aber lediglich 25 vor dem Onload-Event benötigt werden. Dank dieser Reduzierung auf das Wesentliche erwartet den Besucher nun eine subjektiv wesentlich angenehmere Webseite.

6. Vergleich

Ein abschließender Vergleich aller Methoden wurde tabellarisch zusammengetragen. In der ersten Spalte steht der Methodenname, darauf folgen die Gesamtladezeit und die Zeit, die der Nutzer warten muss bis der Browser anfängt Inhalte darzustellen. Zu jedem Ergebnis wird noch eine Einschätzung abgegeben. Abschliessend wurde das Gesamtergebnis in der letzten Zeile dargestellt.

7. Zusammenfassung und Ausblick

Im Rahmen dieser Arbeit wurde beispielhaft an der Startseite der Community ITsax eine Web-Performanceanalyse durchgeführt und offensichtliche Fehler behoben. Zusätzlich wurden die Verbesserungen so dokumentiert, dass sie auf alle anderen Portale der pludoni GmbH leicht angewendet werden können. Auf dem Gebiet der Web-Performance gibt es ständig neue Entwicklungen, da sehr viele exzellente Programmierer in Firmen wie Google, Microsoft und Yahoo an dieser Entwicklung beteiligt sind. Oftmals ist es daher schwer die effektivsten Verbesserungen herauszufiltern und Prioritäten zu setzen. Die Arbeit hat gezeigt, dass ein Fokus auf die Bereiche der Verringerung der Anzahl der HTTP-Anfragen und der Webseitengröße zu sehr guten Ergebnissen führt. Weiterführende Maßnahmen könnten zum Beispiel die Nutzung von CDN Services sein. Auch eine aktuellere Drupal Version könnte in Betracht gezogen werden, dies ist aber mit einem erheblichen Zeitaufwand verbunden. Da der Fokus dieser Arbeit nur auf der Startseite lag, wurden andere kritische Bereiche, wie zum Beispiel die Suche, noch nicht betrachtet. Gerade bei dynamischen Elementen wäre eine weiterführende Beschäftigung mit den Möglichkeiten des Profiling und des Debugging anzuraten.

Tabelle 27: Auswertung der Ergebnisse

Methode	Load Time	TtSR	Kommentar
Ausgangszustand	3,728s (±0,00%)	2,002s (±0,00%)	Der Ausgangszustand ohne Optimierungen.
JS Aggregation	3,658s (−1,87%)	1,915s (−4,35%)	Bei geringen Aufwand können hier erste Verbesserungen erzielt werden.
CSS Aggregation	3,577s (−4,05%)	1,577s (−21,23%)	Diese Optimierung sollte auch auf jeden Fall durchgeführt werden.
Boost Modul	3,233s (−13,22%)	1,473s (−26,42%)	Caching ist Pflicht! Nur in seltenen Ausnahmen ist davon abzusehen.
Bildkompression verlustfrei	3,640s (−2,31%)	1,894s (−5,4%)	Durch Verringerung der Datenmenge können besonders Handynutzer profitieren.
Bildkompression verlustbehaftet	3,255s (−12,68%)	1,908s (−4,7%)	Zusätzliche verlustbehaftete Kompression sorgt oft für schnelleren Seitenaufbau, da große Mengen an Daten eingespart werden können.
Drupal 5 Theme Fehler	3,626s (−2,73%)	1,890s (−5,59%)	Durch diese sehr spezielle Optimierung können auch 100ms gewonnen werden.
Theme Bilder gesprited	3,707s (−0,56%)	1,968s (−1,69%)	Spriting ist eine gute Möglichkeit Requests zu sparen, macht sich aber leider nicht stark in der Ladezeit bemerkbar.
Partnerslideshow	2,749s (−26,26%)	1,842s (−7,99%)	Sehr deutlich wird die Geschwindigkeit durch das asynchrone Nachladen von Inhalten verbessern.
Facebook Widget	3,411s (−8,50%)	1,848s (−7,69%)	Besonders bei Inhalten von externen Quellen ist es nützlich die Ladezeiten abzukoppeln.
Partneranzeigen	3,871s (+3,83%)	2,225s (+11.13%)	An diesem Beispiel wird deutlich, dass nachladen nicht unbedingt Verbesserung sein muss. Trotzdem ist diese Umprogrammierung nötig, um das Caching der Startseite zu ermöglichen.
Gesamtergebnis	1.354s (−63,68%)	0.952s (−52,45%)	Wenn alle Verbesserungen auf einmal aktiviert werden, wird die Webseite dramatisch beschleunigt.

Abbildungsverzeichnis

Tabellenverzeichnis

Literatur

[AdWords 2008] AdWords, Inside: *Landing page load time will soon be incorporated into Quality Score - Inside AdWords.* http://adwords.blogspot.com/2008/03/landing-page-load-time-will-soon-be.html. 2008. – URL `http://adwords.blogspot.com/2008/03/landing-page-load-time-will-soon-be.html`

[Bixby 2011] Bixby, Joshua: *FAQs: The 12 most-asked questions about how Google factors page speed into its search rankings — Web Performance Today*. http://www.webperformancetoday.com/2011/08/05/faqs-google-seo-search-ranking-website-speed/. 2011. – URL `http://www.webperformancetoday.com/2011/08/05/faqs-google-seo-search-ranking-website-speed/`

[BMWi 2011] BMWi: *Breitbandportal des BMWi - Kartendownload*. http://www.zukunft-breitband.de/BBA/Navigation/Breitbandatlas/laenderkarten.html? 2011. – URL `http://www.zukunft-breitband.de/BBA/Navigation/Breitbandatlas/laenderkarten.html?`

[Fielding 1999] Fielding, R.: *Hypertext Transfer Protocol – HTTP/1.1*. http://www.ietf.org/rfc/rfc2616.txt. 1999. – URL `http://www.ietf.org/rfc/rfc2616.txt`

[Frucci 2009] Frucci, Adam: *Internet Speeds and Costs Around the World, Shown Visually*. http://gizmodo.com/5390014/internet-speeds-and-costs-around-the-world-shown-visually. 2009. – URL `http://gizmodo.com/5390014/internet-speeds-and-costs-around-the-world-shown-visually`

[Google 2011a] Google: *Let's make the web faster - Google Code*. http://code.google.com/intl/de/speed/articles/web-metrics.html. 2011. – URL `http://code.google.com/intl/de/speed/articles/web-metrics.html`

[Google 2011b] Google: *Let's make the web faster - Google Code*. http://code.google.com/intl/de/speed/. 2011. – URL `http://code.google.com/intl/de/speed/`

[Hitwise 2011] Hitwise: *Hitwise United States » - the power of competitive intelligence*. http://www.hitwise.com/us/datacenter/main/dashboard-23984.html. 2011. – URL `http://www.hitwise.com/us/datacenter/main/dashboard-23984.html`

[Kaspersky 2011] Kaspersky: *„Das haben wir nicht bestellt“: Miner-Botnetz attackiert deutsche Pizza-Lieferdienste*. http://www.kaspersky.com/de/news?id=207566473. 8 2011. – URL `http://www.kaspersky.com/de/news?id=207566473`

[Klukas 2011] Klukas, Jörg: *Unternehmen | pludoni GmbH - the community experts*. http://www.pludoni.de/unternehmen. Juni 2011. – URL `http://www.pludoni.de/unternehmen`

[Microsoft 2011] Microsoft: *IE 6 Warning | Educate Others to Stop Using IE6 | IE6 Countdown*. http://www.ie6countdown.com/educate-others.aspx. 2011. – URL `http://www.ie6countdown.com/educate-others.aspx`

[Netcraft 2011] Netcraft: *May 2011 Web Server Survey | Netcraft*. http://news.netcraft.com/archives/2011/05/02/may-2011-web-server-survey.html. 5 2011. – URL `http://news.netcraft.com/archives/2011/05/02/may-2011-web-server-survey.html`

[website optimization 2008] optimization website: *The Psychology of Web Performance - how slow response times affect user psychology*. http://www.websiteoptimization.com/speed/tweak/psychology-web-performance/. 5 2008. – URL `http://www.websiteoptimization.com/speed/tweak/psychology-web-performance/`

[PHP 2011a] PHP: *PHP: Was ist PHP? - Manual*. http://www.php.net/manual/de/intro-whatis.php. 2011. – URL `http://www.php.net/manual/de/intro-whatis.php`

[PHP 2011b] PHP: *PHP: Was kann PHP? - Manual*. http://www.php.net/manual/de/intro-whatcando.php. 2011. – URL `http://www.php.net/manual/de/intro-whatcando.php`

[StatCounter 2011] StatCounter: *Top 12 Browser Versions from Aug 10 to Aug 11 | StatCounter Global Stats*. http://gs.statcounter.com/#browser_version-ww-monthly-201008-201108. 8 2011. – URL `http://gs.statcounter.com/#browser_version-ww-monthly-201008-201108`

[Toll 2011] Toll, William: *Why Marketers Must Care About Site Speed*. http://searchenginewatch.com/article/2085970/Why-Marketers-Must-Care-About-Site-Speed. 7 2011. – URL `http://searchenginewatch.com/article/2085970/Why-Marketers-Must-Care-About-Site-Speed`

[w3c 2005] w3c: *W3C Document Object Model*. http://www.w3.org/DOM/#what. 1 2005. – URL `http://www.w3.org/DOM/#what`

[Witzmann 2011] Witzmann, Jan: *Existenzbedrohung DDoS-Attacke: Kleine Onlinehändler werden zur Zielscheibe für Erpressungsversuche | Onlinehändler-News*. http://www.onlinehaendler-news.de/2011/08/11/ddos-attacke-kleine-onlinehandler-werden-bedroht/.

8 2011. – URL http://www.onlinehaendler-news.de/2011/08/11/ddos-attacke-kleine-onlinehandler-werden-bedroht/

[YahooDevNetwork 2011] YahooDevNetwork: *Best Practices for Speeding Up Your Web Site*. http://developer.yahoo.com/performance/rules.html#min_dom. 8 2011. – URL http://developer.yahoo.com/performance/rules.html#min_dom