Vortrag zum

Proseminar „*Website-Management-Systeme*“

im Wintersemester 2003/04

Thema „*Session-Management*“

von Christian Bindseil

Oktober 2003

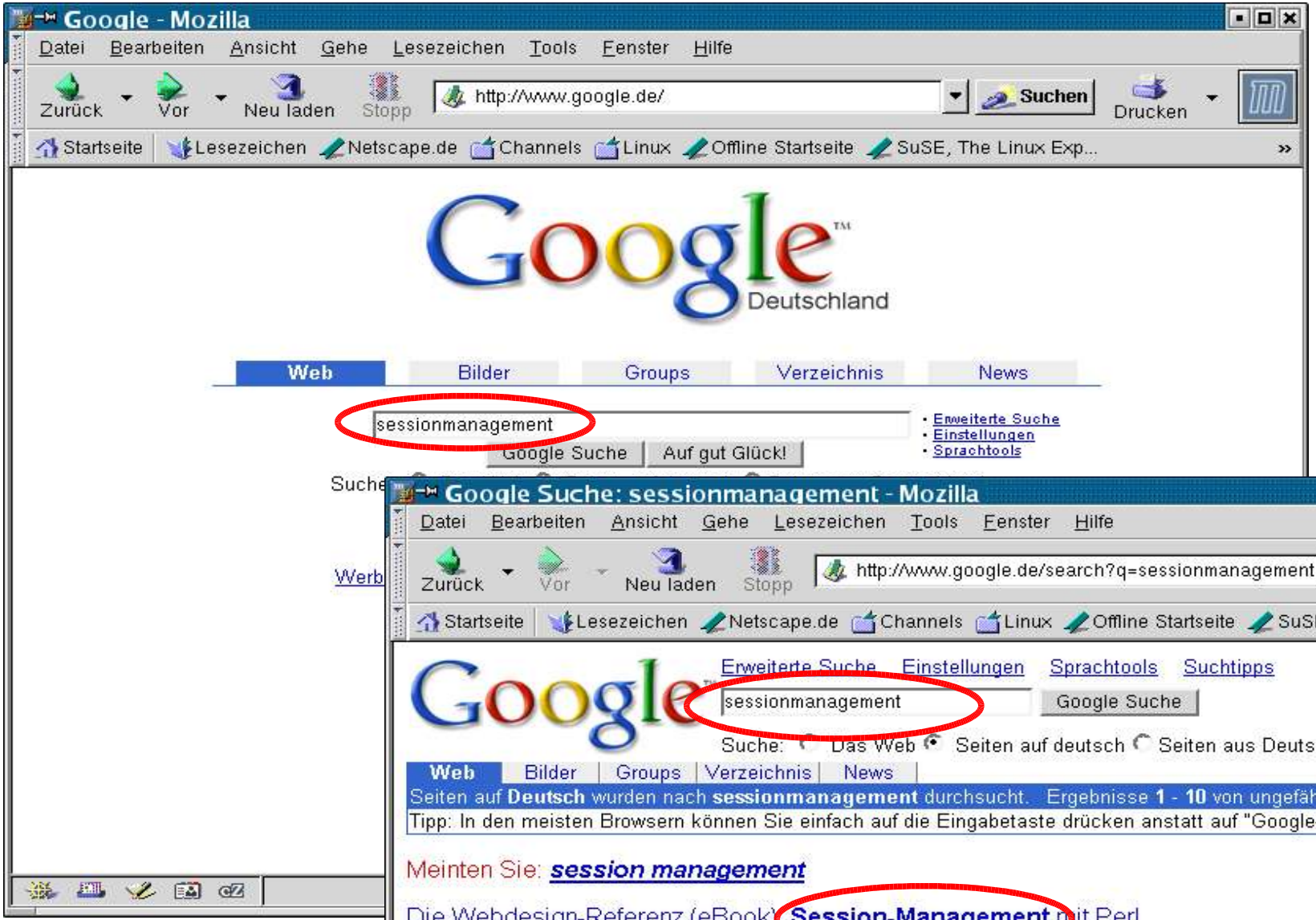
Theoretische Grundlagen

Begriffsklärung: *Session*

- Ablauf einer Benutzer-Server-Interaktion
- (Mehr oder weniger genau) definierter Beginn
- Ebensolches Ende
- Zustand (z. B. Warenkorb)

Beispiel 1 – Page Session

Schon bei einfachen Webanwendungen ist der Inhalt einer Seite abhängig von den Eingaben auf der vorhergehenden Seite, also einem oder mehreren Parametern.


Google - Mozilla
Datei Bearbeiten Ansicht Gehe Lesezeichen Tools Fenster Hilfe
Zurück Vor Neu laden Stopp http://www.google.de/ Suchen Drucken
Startseite Lesezeichen Netscape.de Channels Linux Offline Startseite SuSE, The Linux Exp...
Google Deutschland
Web Bilder Groups Verzeichnis News
sessionmanagement
Google Suche Auf gut Glück!
Erweiterte Suche
Einstellungen
Sprachtools
Google Suche: sessionmanagement - Mozilla
Datei Bearbeiten Ansicht Gehe Lesezeichen Tools Fenster Hilfe
Zurück Vor Neu laden Stopp http://www.google.de/search?q=sessionmanagement Suchen Drucken
Startseite Lesezeichen Netscape.de Channels Linux Offline Startseite SuSE, The Linux Exp...
Google
Erweiterte Suche Einstellungen Sprachtools Suchtipps
sessionmanagement
Google Suche
Suche: Das Web Seiten auf deutsch Seiten aus Deutschland
Web Bilder Groups Verzeichnis News
Seiten auf Deutsch wurden nach sessionmanagement durchsucht. Ergebnisse 1 - 10 von ungefähr 6,810. Suchdauer: 0.47 Sekunden.
Tipp: In den meisten Browsern können Sie einfach auf die Eingabetaste drücken anstatt auf "Google-Suche" zu klicken.
Meinten Sie: session management
Die Webdesign-Referenz (eBook) Session-Management mit Perl
eBook-Project by AboutWebDesign.de [LOGO] 7.6 Session-Management
mit Perl. ... Eine davon kümmert sich um Session-Management. Die Perl ...
www.webdesign-referenz.de/technikII_session.html - 45k - Im Cache - Ähnliche Seiten
Anzeigen
iNKA -win a world of time
Ein health care management system
das Ihre Arbeitsprozesse integriert

Ist dies eine Session? Ja, denn...

- Beginn: Betreten der Seite durch den Benutzer
- Zustand: Suchwort
- Ende: Verlassen der Seite (ungenau: Schließen des Fensters)

Wichtig:

Kennzeichnend für die Page Session ist, dass ein Benutzer in mehreren Browser-Fenstern unabhängige Sessions gleichzeitig benutzen kann.

Beispiel 2 – Browser Session

Kompliziertere Fälle (z. B. Benutzerzählung auf einer Webpräsenz) haben andere Anforderungen:

- Beginn: Betreten der Seite durch den Benutzer
- Zustand: Historie+Zählung aller Seitenaufrufe des Benutzers
- Ende: Verlassen der Seite (ungenau: Timeout)

Wichtig:

Die Browser-Session gilt global für den gesamten Browser, unabhängig vom benutzten Browserfenster.

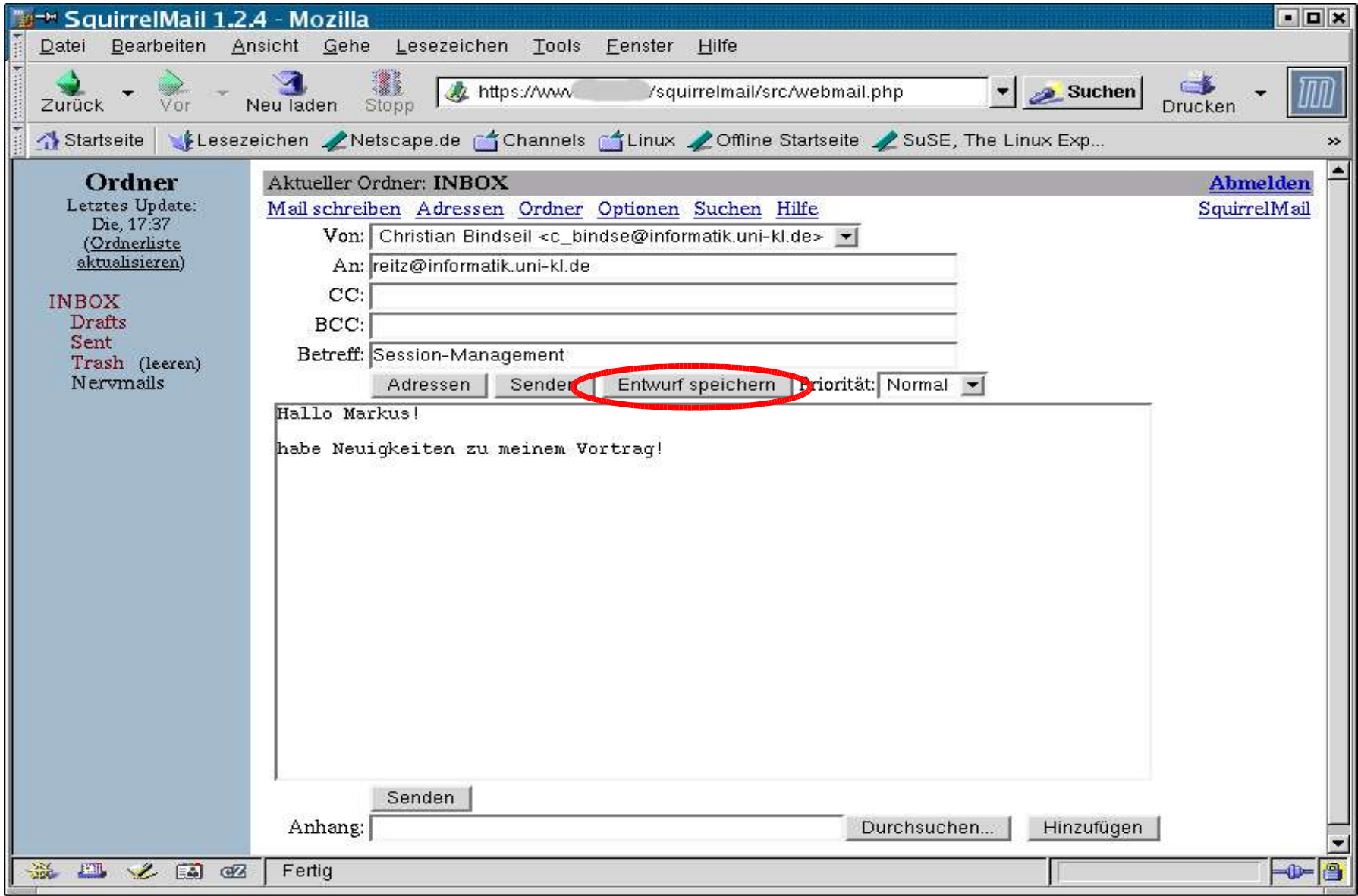
Beispiel 3 – User Session

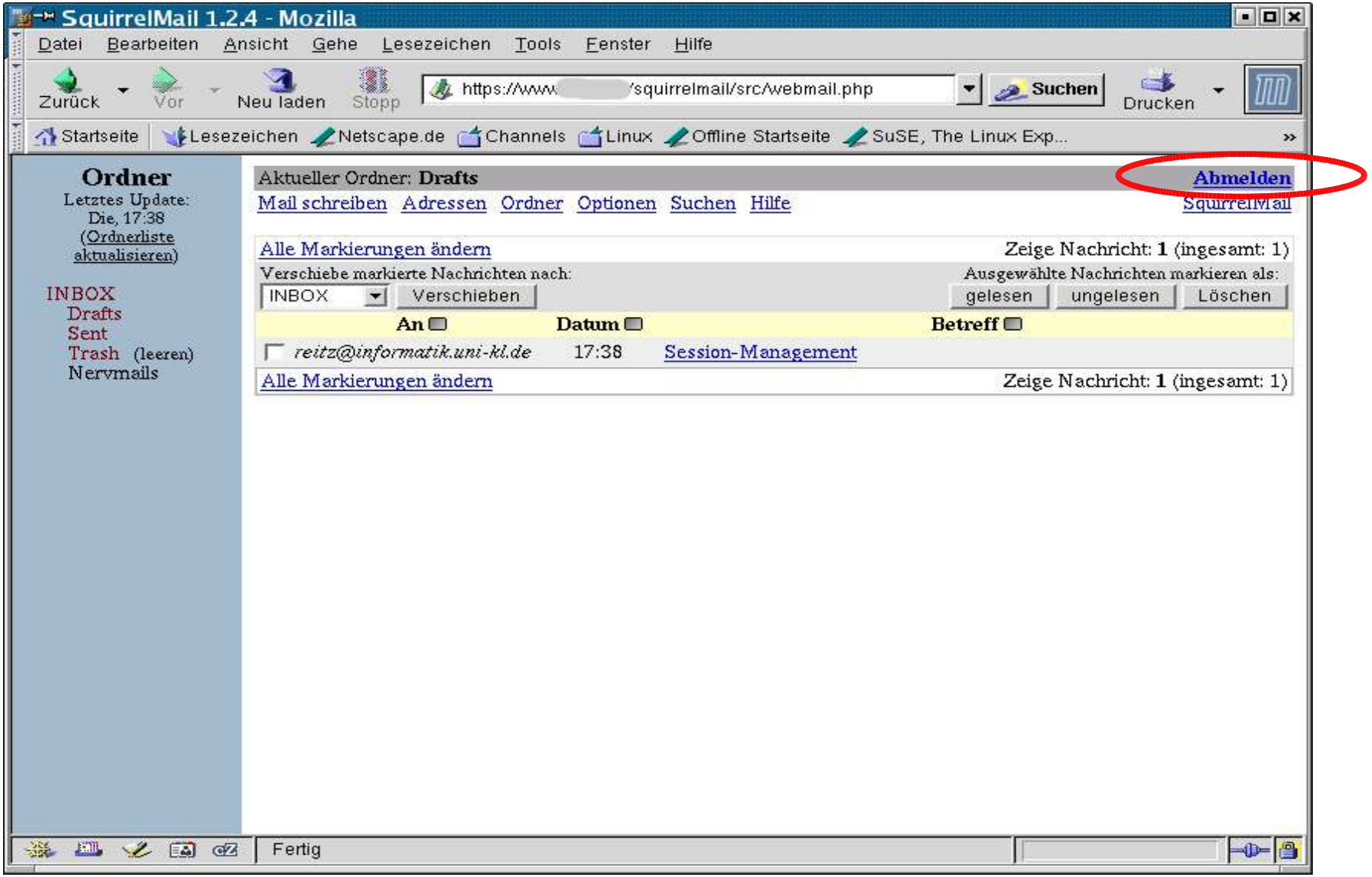
Wieder anders sieht es aus, wenn es darum geht, einen User immer eindeutig identifizieren zu können – also nicht nur ein Browserfenster oder einen Browser.

Bekannte Beispiele:

- Online-Shop-Systeme
- Webmail-Interfaces

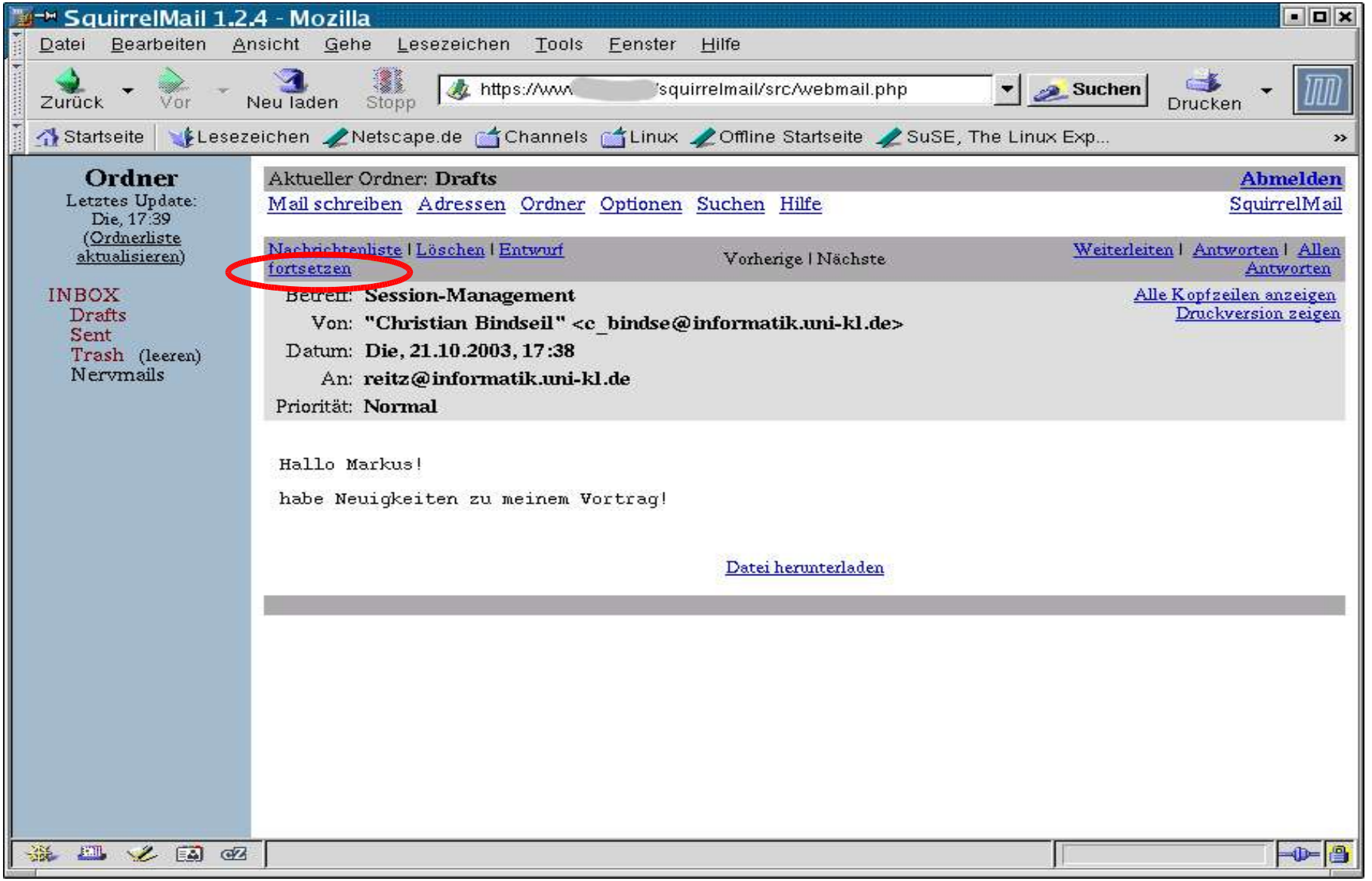
Nach dem ersten Login...

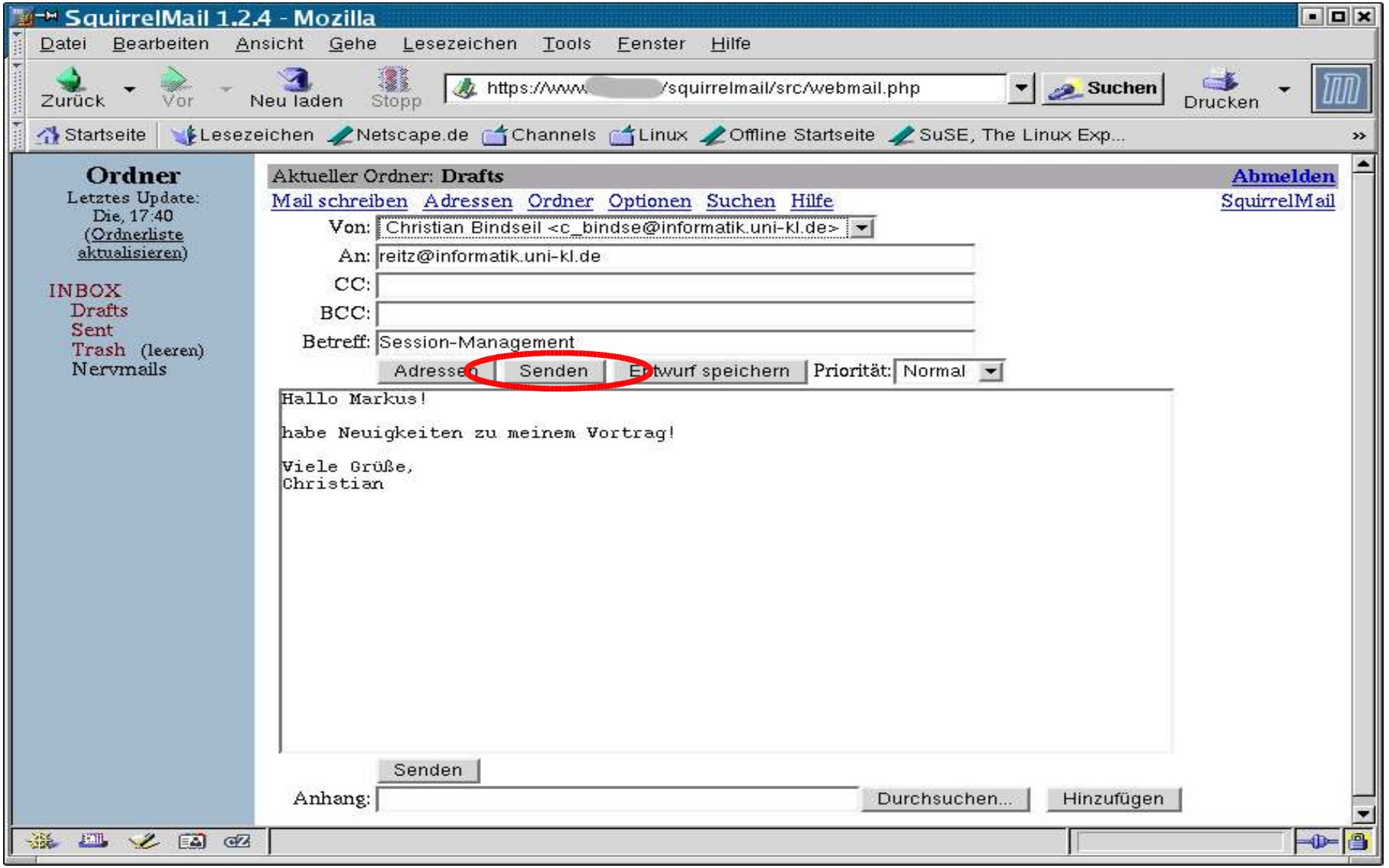




…anschließend Logout

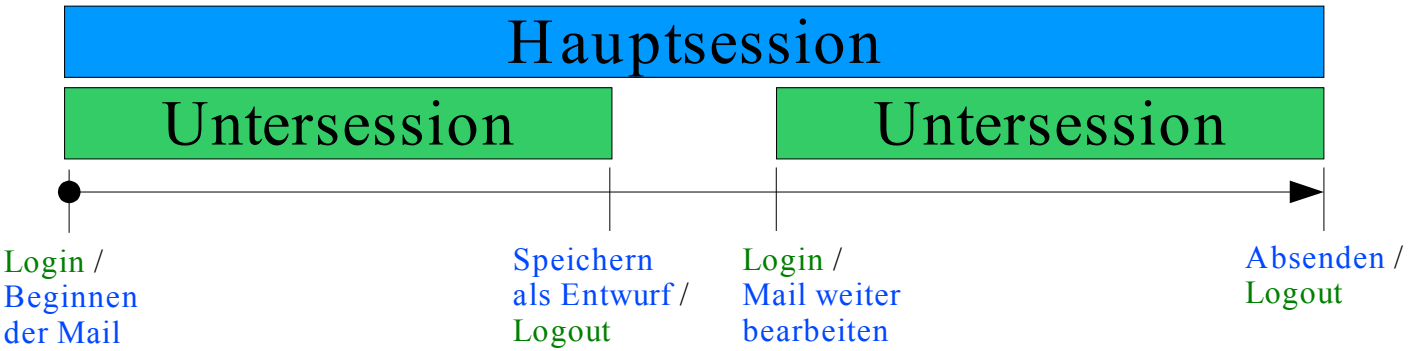
Nach dem zweiten Login...





...anschließend Logout

Bemerkungen:



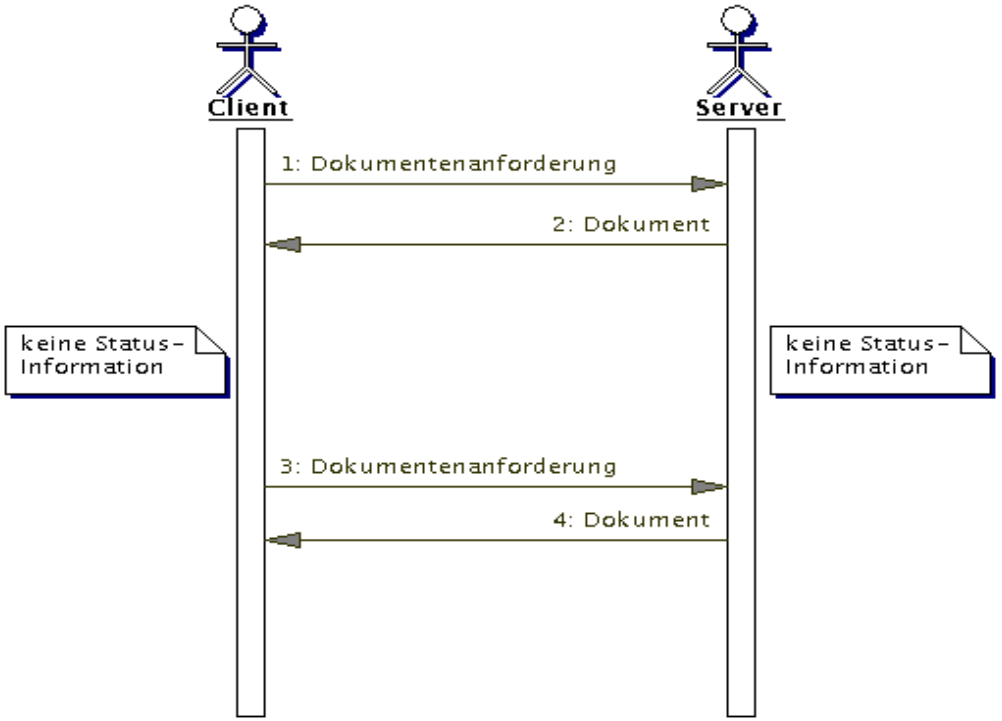
In der Hauptsession ist der Benutzer selbst das einzige gleichbleibende Element!

Für die einzelnen Sitzungen als Untersessions reichen die bislang erwähnten Session-Formen aus.

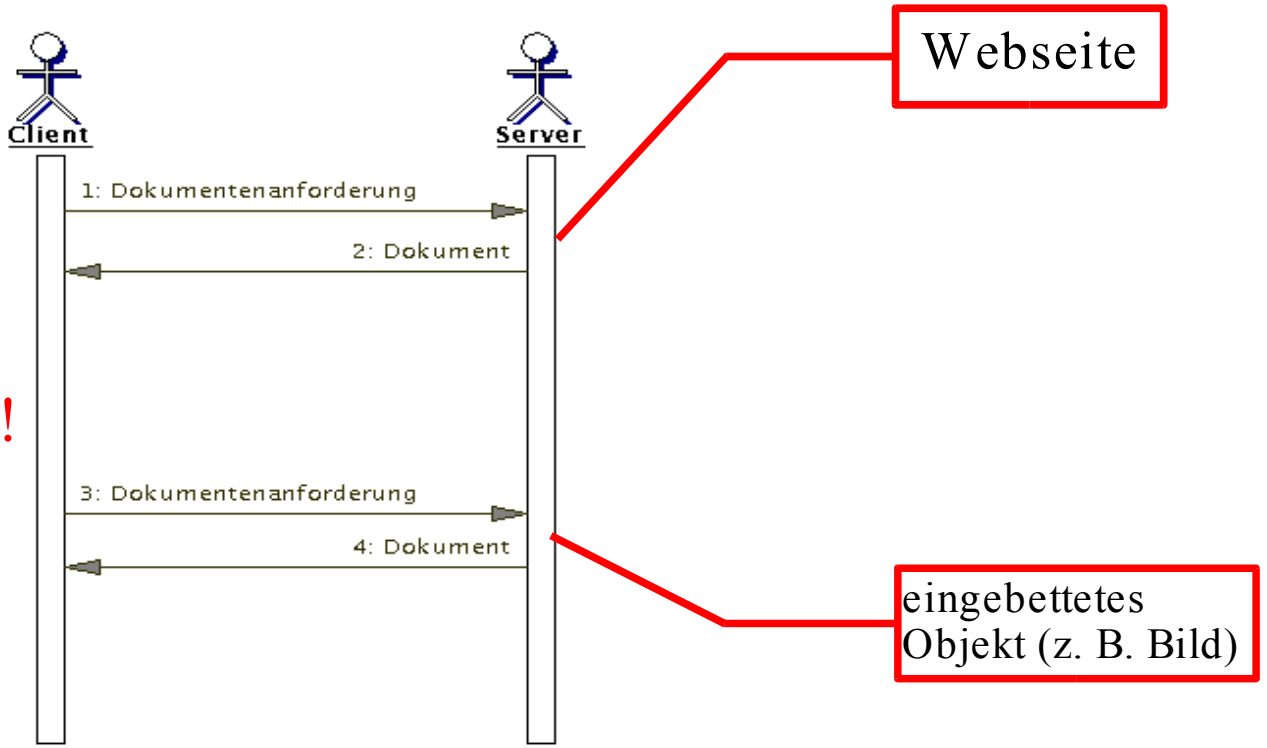
Praktische Umsetzung

Historisch bedingte Probleme

Das HTTP (HyperTextTransferProtocol) als Grundlage des WWW hat eine historisch bedingte Schwäche:
es ist zustandslos.



Typischer Verbindungsablauf bei HTTP:



Zwei komplett getrennte Verbindungen!

Anfangszeit des WWW:

- Server-Ressourcen begrenzt
- kleine Webseiten
- wenige eingebettete Objekte

=> ressourcensparendes Protokoll

Heute:

- Verbindungs-Ressourcen begrenzt
- große Webseiten
- viele eingebettete Objekte

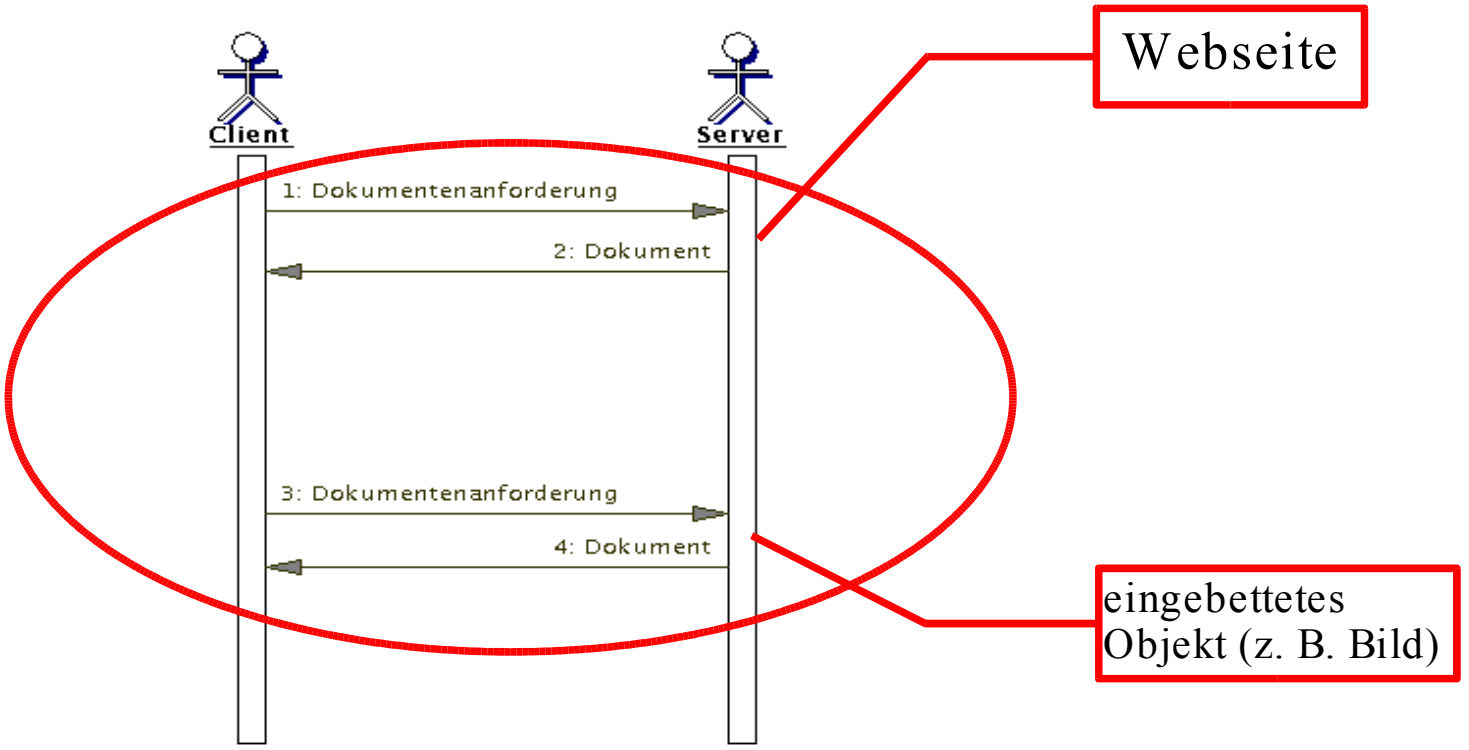
=> ineffektiv, verschwenderisch

Weiterentwicklung (HTTP 1.1):

- Geöffnete Verbindung kann mehrfach genutzt werden
- Verbindungsende durch Browser oder (sehr kurzen) Timeout

=> ressourcensparend

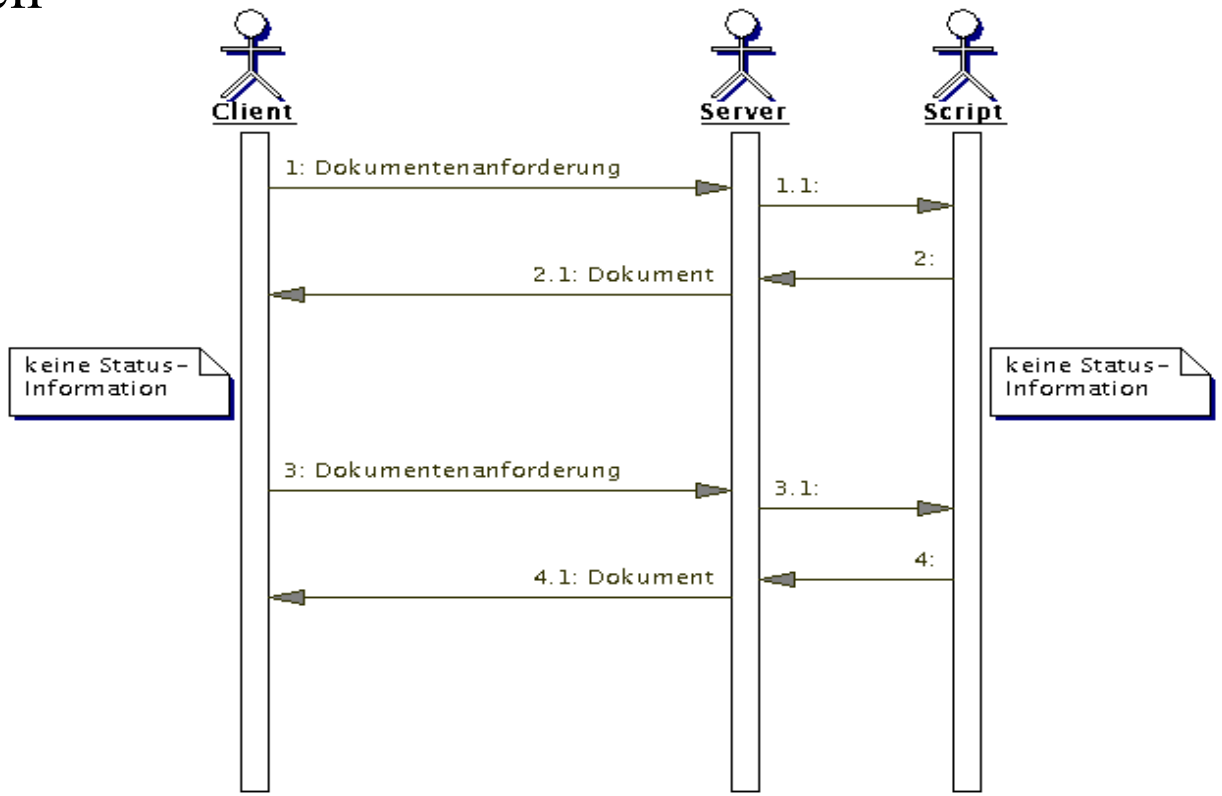
Eine einzige Verbindung für mehrere Downloads!



Wichtig:

HTTP 1.1 noch immer zustandslos
=> keine Lösung für Session-Management-Probleme

Für dynamische Webseiten
nicht ausreichende Basis!



Lösungswege

Da HTTP keine Fähigkeit zur Zustandsspeicherung besitzt, müssen andere Möglichkeiten untersucht werden.

Es ergeben sich folgende zwei Ansätze:

- Stateful Server
- Stateless Server = Stateful Client

Stateful Server

Der Server speichert alle Session-Daten

Vorteile:

- ✓ Informationen werden dauerhaft und sicher zentral gespeichert
- ✓ Sensitive Informationen leichter gegen Ausspähen schützbar

Nachteile:

- ✗ Erhöhter Speicherbedarf auf dem Server
- ✗ Client-Identifizierung u. U. schwierig

Stateless Server = Stateful Client

Der Client speichert alle Session-Daten

Vorteile:

- ✓ Benutzer hat mehr Transparenz über gespeicherte Daten
- ✓ Einfache Client-Identifizierung

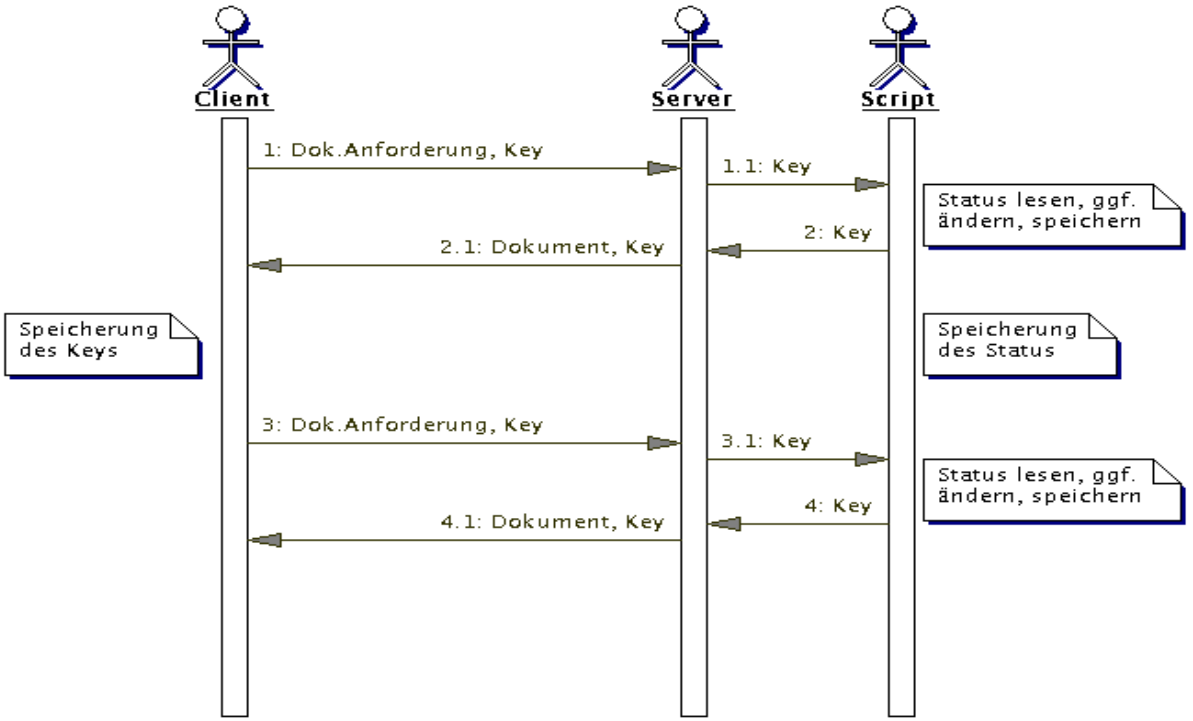
Nachteile:

- ✗ Browserabhängig (Status nicht transportabel auf andere Systeme)
- ✗ Sensitive Daten müssen übertragen werden

Folgerung aus den Ansätzen

Wie man leicht erkennt, ist der Stateful Server die bessere Wahl:

- Die Session-Daten liegen auf dem Server
- Der Client speichert eindeutigen Key (Session-ID, SID)
- Den Daten wird der Key fest zugeordnet



Technische Realisierung

Wie bereits festgestellt, liegt das eigentliche Problem in der Identifizierung des Clients. Zum Besseren Verständnis der Problematik lohnt ein Blick auf nicht anwendbare Ansätze:

Identifizierung der IP-Adresse

Scheitert daran, dass nicht jedem Internet-Teilnehmer eine eindeutige und unveränderliche IP-Adresse zugeordnet ist. Stichworte: DHCP, Firewall mit NAT, Multiusersysteme, Proxy

Identifizierung der Standard-Browser-ID

Scheitert daran, dass die ID bei identischen Browser-Versionen gleich und somit keinem einzelnen Browser zugeordnet ist.

Funktionierende Techniken

- Cookies
- Versteckte Formularfelder
- URL-Rewriting
- Benutzerauthentifizierung

Cookies

- Kleine Datenobjekte
- Auf HTTP-Ebene zum Browser übertragbar
- Ebenso wieder auslesbar
- Felder: Name, Wert, Lebensdauer, Domäne, Pfad, SSL-Info

Problematisch sind begrenzte Lebensdauer (mit Cookie übergeben, durch den Browser geregelt oder durch manuelles Löschen) und die Tatsache, dass viele Benutzer Cookies aus Gründen der Privatsphäre abschalten.

Moderne Web-Sprachen (hier z. B. PHP) ermöglichen einfache Umsetzung des Cookie-Konzepts:

Setzen des Cookies:

Syntax:

```
<?php
 setcookie(Name, Wert*, Lebensdauer*,
           Pfad*, Domain*, Sicher*)
?>
```

* = optionale Parameter

Beispiel:

```
<?php
 setcookie(„SID“, „ED3cx$dky“)
?>
```

Auslesen des Cookies:

Syntax:

```
<?php
 $wert = $HTTP_COOKIE_VARS[Name];
?>
```

Beispiel:

```
<?php
 $sid = $HTTP_COOKIE_VARS[„SID“];
?>
```

Versteckte Formularfelder

- Zusätzliches unsichtbares Feld mit SID
- Wird mit allen anderen Feldern übermittelt

```
<form>
...
<input type="hidden" name ="SID" value="ED3cx$dky">
</form>
```

Nachteil: Nur auf Webseiten mit Formularen einsetzbar

URL Rewriting

Sei folgende Adresse die eines Scripts:
`http://www.uni-kl.de/test/mein-script.cgi`

Übergabe der SID mittels gewöhnlicher Script-Parameter:
`http://www.uni-kl.de/test/mein-script.cgi?ED3cx$dky`

bzw. in folgender Form am Beispiel von Java:
`http://www.uni-kl.de/test/mein-script.jsp;jsessionid=ED3cx$dky`

oder mittels URL-Ergänzung:
`http://www.uni-kl.de/test/mein-script.cgi/ED3cx$dky`

Gewöhnliche Script-Parameter und URL-Ergänzung lassen sich auch kombinieren und werden folgendermaßen durch das Script ausgewertet:

- Die Scriptparameter stehen in der Variablen QUERY_STRING
- Die Parameter der URL-Ergänzung stehen in PATH_INFO

Benutzerauthentifizierung

Benutzer müssen sich authentifizieren, um eine HTTP-Verbindung aufbauen zu können.

- Bei Erstkontakt (seit Browserstart) erscheint Login-Box
- Eingabe von Username und Password
- Bei Korrektheit vom Browser gespeichert für weitere Aufrufe
- Anmeldedaten stehen serverseitig in Variable REMOTE_USER

Dies ist die einzige Möglichkeit, Benutzer wirklich genau zu authentifizieren!

Nachteile:

- Viele Systeme => viele getrennte Anmeldungen
- Oftmals unterschiedliche Anmeldedaten
- Meist unterschiedliche Regeln für Passwörter
- Jedes System in der Regel in anderer Sicherheitszone

=> Flut von Benutzernamen und Passwörtern!

Abschlussbemerkungen

Obwohl beide Techniken von modernen Web-Sprachen gut unterstützt werden, sind Cookies dem URL-Rewriting vorzuziehen, da sie einfacher anwendbar, effizienter (beim Rewriting müssen immer alle Links einer Seite geändert werden) und sicherer sind.

Gefahr besteht vor allem durch Session-Entführung, da

- Suchmaschinen die SID in Links mit abspeichern,
- die SID bei externen Links fremden Seitenbetreiber zukommen,
- Adressen mit SID in Bookmarks gespeichert werden können.

Da aber oftmals Cookies durch den Benutzer unterbunden werden, wird häufig eine Fallback-Lösung mit URL Rewriting eingesetzt.

Versteckte Formularfelder spielen aufgrund ihres beschränkten Einsatzbereichs eine untergeordnete Rolle.

Benutzerauthentifizierung wird oft eingesetzt, meist in Kombination mit anderen Session-Management-Techniken in Form verschachtelter Sessions. Häufig wird dabei eine von der Websprache abhängige Authentifizierung über ein HTML-Formular anstatt der Loginbox verwendet.

Anhang

Quellenverzeichnis

Jan Newmarch: „HTTP-Sessionmanagement“, August 2000, http://jan.netcomp.monash.edu.au/ecommerce/session.html

Martin Nilsson: „Session Variables Revisited“, September 2001, http://community.roxen.com/articles/015_sessions_2/

Dirk Louis, Christian Wenz: „Dynamic Web Publishing“, Markt+Technik Verlag, München, 2001

http://www11.informatik.tu-muenchen.de/lehre/praktika/vap/docSS2003/doc_cas_5_ger_html/html/url_rewriting.html

Impressum

Vortrag zum Proseminar „Website-Management-Systeme“ im Wintersemester 2003/04 an der Technischen Universität Kaiserslautern, Fachbereich Informatik, Arbeitsgruppe Softwaretechnik

Thema: „Session-Management“

Verfasser: Christian Bindseil
Ort: Kaiserslautern
Datum: Oktober 2003