TECHNISCHE UNIVERSITÄT
KAISERSLAUTERN

Fachbereich Informatik
AG Softwaretechnik

Ausarbeitung zum

Proseminar „*Website-Management-Systeme*“ im Wintersemester 2003/04

Thema „*Session-Management*“

von Christian Bindseil
Oktober 2003

Inhaltsverzeichnis

1 Theoretische Grundlagen

1.1 Was ist eine Session?

Für die Zwecke dieser Ausarbeitung erscheint es sinnvoll, dieser Frage zunächst mit einer Begriffsklärung zu begegnen, bevor man sich einigen konkreten Anwendungsfällen zuwendet:

Begriffsklärung: *Session*
Eine Session ist der Ablauf einer Benutzer-Server-Interaktion mit einem (mehr oder weniger genau) definierten Beginn und einem ebensolchen Ende, sowie einem Zustand (z. B. vom Benutzer wählbare Hintergrundfarbe), der über diesen Zeitraum hinweg fest bleiben oder auch veränderlich sein kann.

Um diese Erklärung greifbarer zu machen, hilft ein Blick auf reale Anwendungsfälle, wie sie wohl jedem Internet-Nutzer schon begegnet sind.

1.2 Beispiele

1.2.1 Page Session

In der Natur von Webanwendungen liegt es, dass der Inhalt der betrachteten Webseite abhängig ist von den Aktionen des Benutzers auf der vorhergehenden Seite. Ein einfaches Beispiel ist eine normale Suchmaschine, bei der natürlich das auf einer oder mehreren Seiten präsentierte Suchergebnis abhängig von den Suchwörtern ist, die der Benutzer zuvor auf der Startseite eingegeben hat. In diesem Fall ist also das Suchwort der den Zustand bestimmende Parameter.
Eine etwas komliziertere Anwendung ist eine Seite mit mehreren Parametern, beispielsweise wenn verschiedene Tabellen auf einer Seite dargestellt sind, die der Benutzer, jede für sich, nach der gewünschten Spalte sortieren lassen kann. Hierbei hätte jede Tabelle ihren eigenen Parameter für den gewünschten Sortierstatus (Zustand; in Sessiondaten gespeichert).
Auch wenn das Ende, in diesem Fall als das Verlassen der Seite durch den Benutzer, nicht wirklich genau definiert werden kann, so hat man hier ein Beispiel für eine Session gefunden.
Bleibt noch festzuhalten, dass es im gezeigten Fall nützlich ist, dass ein User in mehreren Browser-Fenstern unabhängige Sessions gleichzeitig benutzen können sollte.

1.2.2 Browser Session

Der Betreiber einer Website möchte oftmals wissen, wieviele Benutzer die Website nutzen und wie lange sie dort verweilen. Hierbei reicht natürlich eine Page Session wie im ersten Fall nicht aus. Die Session muss restriktiver festgelegt werden:
Beginn ist der erste Aufruf der Website durch den Benutzer. Von nun an muss jede Seite der Website, die der Benutzer auswählt, registriert und der entsprechende Aufruf (Zustandsänderung: Sessiondaten z. B. Aufrufhistorie) dem Benutzer zugeordnet werden können, auch unabhängig davon, ob er dafür ein weiteres Browser-Fenster verwendet oder nicht.
Das Ende der Session ist wieder nicht genau festlegbar, so dass hier eine Timeout-Lösung angebracht scheint: wird innerhalb einer bestimmten Zeitspanne keine weitere Seite durch den Benutzer aufgerufen, endet seine Session.
Man benötigt also eine Technik, die es erlaubt, einen Benutzer (und nicht nur ein Browser-Fenster) während seines gesamten Aufenthalts auf der Website eindeutig identifizieren zu können.

1.2.3 User Session

In diesem Fall diene eine Online-Shopping-Website als Beispiel. Hierbei reichen weder die Page Session noch die Browser Session aus, da man die Anforderung hat, einen Benutzer wirklich immer eindeutig identifizieren zu können, unabhängig vom benutzten PC, Browser, Browserfenster, etc..

Ein typischer Vorgang könnte so aussehen:
Der User beginnt sein Online-Shopping morgens daheim, indem er Produkte zu seinem Warenkorb hinzufügt. Nun geht er zur Arbeit. In der Mittagspause möchte er weitere Sachen in seinen Warenkorb legen. Hierzu benötigt er mehrere Browserfenster, um Produkte vergleichen zu können. Anschließend erst vollendet er seine Bestellung, und die Waren können ihm zugesandt werden..

Betrachtet man den Ablauf des gesamten Bestellvorgangs (Zustand und Sessiondaten: u. a. der Warenkorb) als Haupt-Session (mit den einzelnen Sitzungen als Unter-Sessions), so ist bei diesem Vorgang der Benutzer selbst das einzige gleichbleibende Element, weshalb eine Technik benötigt wird, die ihn - unabhängig von allem anderen - identifizieren und ihm seinen Warenkorb zuordnen kann.

Ein weiteres Beispiel für eine User Session sind auch die Web-Frontends der bekannten Freemail-Anbieter – hier kann man z. B. das (ggf. unterbrochene) Verfassen und Senden einer Mail als Session betrachten.

In beiden Fällen kann man aber auch die Dauer der gesamten Mitgliedschaft mit allen einzelnen Sitzungen als derartige Haupt-Session mit Unter-Sessions ansehen.

Nach Betrachtung der gängigsten Session-Arten am Beispiel, wobei die verwendeten Namen (z. B. User Session) nicht offiziell, aber eine treffende Bezeichnung sind, soll nun die technische Seite betrachtet werden.

2 Praktische Umsetzung

2.1 Historisch bedingte Probleme

Von Anfang an bis heute ist das HTTP (HyperTextTransferProtocol) Grundlage des World Wide Web (nicht des Internets!). Dieses Protokoll setzt auf TCP/IP auf (vergleiche z. B. Vorlesung Systemsoftware) und ist der Standard für die Kommunikation zwischen Web-Server und Web-Browser.

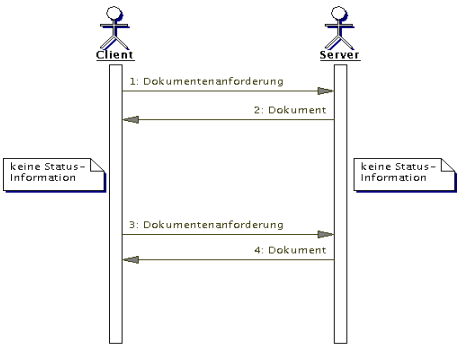
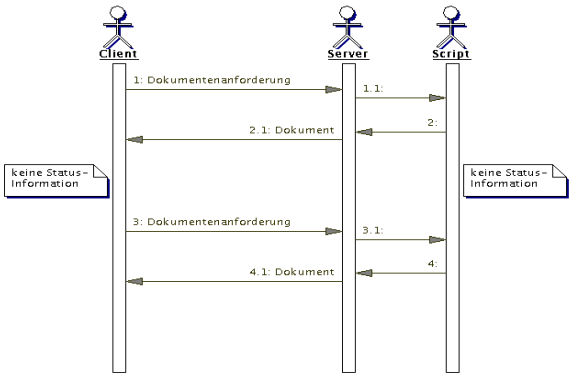
Aufgrund seines verhältnismäßig hohen Alters hat das HTTP eine für die Anwendung von Session-Management gravierende Schwäche: es ist zustandslos. Zur Verdeutlichung diene folgende kurze Erläuterung eines Verbindungs-Ablaufs mit HTTP:
Zunächst öffnet der Browser eine Verbindung zum Webserver und lädt das gewünschte Dokument, anschließend wird die Verbindung wieder beendet.
Diese gesamte Prozedur wird für jede benötigte URL einzeln ausgeführt, also für eine HTML-Seite mit fünf Bildern insgesamt sechsmal.

In der Entstehungszeit des WWW war das noch sinnvoll, zum einen weil auch die Server nur sehr begrenzte Ressourcen besaßen und somit jegliche Informationsspeicherung vermieden werden musste – also auch das Aufrechterhalten einer Verbindung. Zum anderen waren auch die HTML-Seiten weitaus simpler, und Bedarf an Session-Management war noch nicht absehbar.
Heutzutage hingegen ist dieses Vorgehen aufgrund sehr komplexer Webseiten mit unzähligen

eingebetteten Objekten (Bilder, Flash-Animationen, etc.) sehr ineffizient und verschwenderisch, da das damit verbundene sehr häufige Verbindungsauf- und -abbauen weit mehr Ressourcen verschlingt, als eine Aufrechterhaltung. Deshalb wurde HTTP 1.1 entwickelt, bei dem eine Verbindung auch für unmittelbar folgende Anfragen geöffnet bleiben kann. Die Verbindung wird entweder vom Browser nach Abschluss aller Downloads oder aber nach einer sehr kurzen Inaktivitäts-Zeit automatisch beendet.
Auch dieses Vorgehen bietet keine Lösung für die Problemstellung des Session-Managements, da eine derart kurzzeitige Aufrechterhaltung der Verbindung keine Wiedererkennung des Benutzers nach einer längeren Pause erlaubt.

2.2 Lösungswege

Zusammenfassend kann man festhalten:
Das WWW bietet „dank“ HTTP keinerlei Unterstützung für die zu Beginn diskutierten Session-Management-Szenarien, die es erfordern, dass an irgendeiner Stelle der aktuelle Zustand der Anwendung gespeichert wird. Dies verdeutlichen auch die beiden folgenden Diagramme:

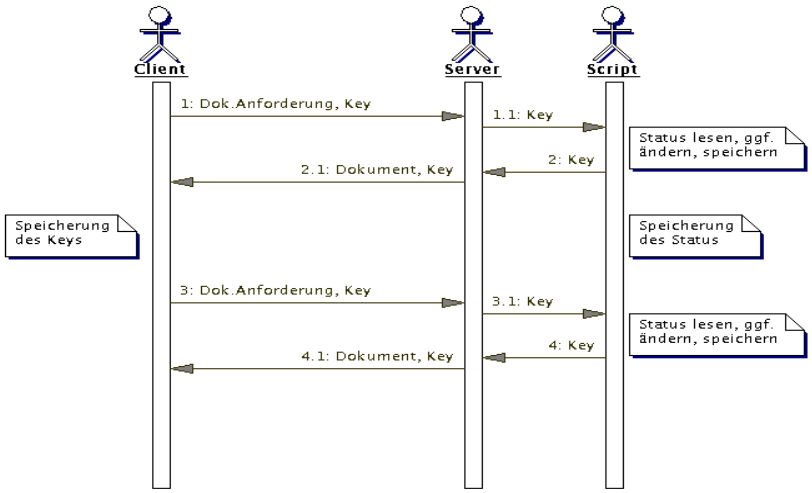
Stateless Connection: *Verbindung ohne Statusspeicherung: statische Webseiten, HTTP*	**Serverseitige Scripte:** *Im Prinzip zustandslos (stateless), lediglich während des Scriptablaufs existiert ein Status*
	

Aus den Überlegungen bzgl. des Speicherorts folgen nun zwei Ideen zur Speicherung des Anwendungszustands:

Stateful Server	***Stateless Server = Stateful Client***
Der Server speichert alle Session-Daten	Der Client speichert alle Session-Daten
Vorteile: ✔ Informationen werden dauerhaft und sicherer zentral gehalten ✔ Sensitive Informationen können leichter gegen Ausspähen geschützt werden	Vorteile: ✔ Benutzer hat mehr Transparenz über gespeicherte Daten ✔ Einfache Client-Identifizierung

Stateful Server	*Stateless Server = Stateful Client*
Nachteile: ✗ Erhöhter Speicherbedarf auf dem Server ✗ Client-Identifizierung u. U. schwierig	Nachteile: ✗ Browserabhängig (Status nicht transportabel auf andere Browser, PCs) ✗ Sensitive Daten müssen übertragen werden

Wie man leicht erkennen kann, ist der Stateful Server die bessere Wahl: Die eigentlichen Daten einer Session bleiben auf dem Server, der Client speichert nur einen eindeutigen Key, der diesen Session-Daten auf dem Server zugeordnet wird:



Der Session-Key ist in der Regel eine vom Server bei Beginn der Session erzeugt Zufallssequenz (Session-ID, SID), kann aber (vgl. 1.2.3) auch ein persönlicher Benutzername sein.

2.3 Technische Realisierung

2.3.1 Einleitung

Wie bereits festgestellt, liegt das eigentliche Problem in der Identifizierung des Clients. Die Speicherung der Daten auf dem Server hingegen ist eher eine Frage der jeweils eingesetzten Sprachen und Server.
Um den Hintergrund der heute eingesetzten Techniken (auf die nachfolgend eingegangen wird) besser zu verstehen, sollte man zuerst einige Ansätze, die nicht angewendet werden, betrachten:

Ansatz	*Warum es nicht klappt*
Identifizierung der IP-Adresse	Nicht jeder Teilnehmer des Internets hat eine eindeutige und unveränderliche IP-Adresse, bedingt durch dynamische IP per DHCP (z. B. bei Interneteinwahl), Firewalls mit NetworkAdress-Translation, Proxy-Server und Multi-User-Umgebungen.
Identifizierung der Standard-Browser-ID	Diese ID ist in sehr vielen Fällen identisch, da sie einer Browser-Version und nicht einem einzelnen Browser zugeordnet ist.

Im Laufe der Jahre wurden verschiedene Techniken entwickelt, den Session-Key auf User-Seite zu speichern und zwischen Browser und Server zu übertragen. Die wichtigsten sind im Folgenden aufgeführt:

2.3.2 Cookies

Cookies sind kleine Daten-Objekte, die auf HTTP-Ebene übertragen, im Browser gespeichert und auf gleichem Wege auch wieder ausgelesen werden. Die Felder eines solchen Cookie-Objekts sind Name, Wert, Lebensdauer, Domäne, Pfad und Verschlüsselungs-Info (ob der Cookie über HTTPS übertragen werden darf oder nicht).

Probleme mit Cookies sind die begrenzte Lebensdauer (entweder im Cookie übergeben oder durch den Browser begrenzt oder durch manuelles Löschen durch den Benutzer) und die Tatsache, dass immer mehr Benutzer Cookies im Browser abschalten, meist aus Gründen der Privatsphäre, da Cookies oft auch missbraucht werden, um das Surfverhalten von Benutzern auszuspähen.

Moderne Web-Sprachen ermöglichen eine einfache Umsetzung dieses Konzepts, wie wir hier am Beispiel von PHP (leicht vereinfacht) sehen können:

Setzen des Cookies	***Auslesen des Cookies***

Synthax:
`<?php`
`setcookie(Name, Wert*, Lebensdauer*, Pfad*, Domain*, Sicher*)`
`?>`
* = optionale Parameter
	Synthax:
	`<?php`
	`$wert = $HTTP_COOKIE_VARS[Name];`
	`?>`

Beispiel:
`<?php`
`setcookie(„SID“, „ED3cx$dky“)`
`?>`
	Beispiel:
	`<?php`
	`$sid = $HTTP_COOKIE_VARS[„SID“];`
	`?>`

Dieser PHP-Code steht an erster Stelle in einer HTML-Datei, gefolgt vom eigentlichen Seiteninhalt und einem oder mehreren PHP-Blöcken zur Verarbeitung der Daten. Die HTML-Datei trägt dann die Endung „.php“.

2.3.3 Versteckte Formularfelder

Wie der Name schon sagt, ist deren Benutzung nur auf Webseiten mit Formularen möglich. Hier wird dem üblichen HTML-Formular ein verstecktes Feld hinzugefügt, das den Session-Key enthält und wie alle anderen Formularfelder auch übermittelt wird:

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

2.3.4 URL Rewriting

Angenommen, dass folgende Adresse die eines Scripts ist:
`http://www.uni-kl.de/test/mein-script.cgi`

Soll nun unter Beibehaltung der aktuellen Session darauf verwiesen werden, so muss der entsprechende Link vor der Auslieferung der aktuellen Seite entsprechend manipuliert werden.
Dazu gibt es zwei Möglichkeiten, dem Script Optionen mitzugeben, entweder mittels gewöhnlicher Script-Parameter (nach [3], [4]):
`http://www.uni-kl.de/test/mein-script.cgi?ED3cx$dky`

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

oder mittels URL-Ergänzung (nach [1]):
```
http://www.uni-kl.de/test/mein-script.cgi/ED3cx$dky
```

Nach [1] lässt sich beides auch kombinieren, in dem der Teil beginnend mit dem „?“ am Ende der ergänzten URL angefügt wird. Die Parameter nach dem „?“ können anschließend durch das Script in der Variable QUERY-STRING ausgelesen werden, die Parameter der URL-Ergänzung stehen dann in der Variablen PATH_INFO.

2.3.5 Benutzerauthentifizierung

Beim ersten Kontakt seit Browserstart wird dem Benutzer eine Login-Box gezeigt, in die er seinen Benutzernamen und sein Passwort eingeben muss. Ist dies erfolgreich, wird die angeforderte Seite ausgeliefert, und der Browser speichert die eingegebenen Daten zur automatischen Wieder-Authentifizierung bei erneuten Aufrufen (z. B. verlinkter Seiten) intern – andernfalls wird statt der angeforderten Seite eine Fehlermeldung ausgeliefert.
Die Anmeldedaten werden serverseitig nach Übertragung durch den Browser in der Umgebungsvariablen REMOTE_USER gespeichert.

Dies ist die einzige Methode, einen Benutzer wirklich genau zu authentifizieren. Nachteilig hierbei ist natürlich, dass man sich somit auf vielen Systemen getrennt anmelden muss und oftmal nicht den gleichen Benutzernamen und in vielen Fällen auch nicht das gleiche Kennwort nehmen kann, sowohl aus persönlichen Sicherheitsgründen, als auch aufgrund unterschiedlicher Passwort-Richtlinien auf Serverseite – somit wird die Menge von Benutzernamen und Passwörtern schnell unüberschaubar.

3 Abschlussbemerkungen

In der Praxis werden für Session-Management häufig Cookies benutzt, da diese gegenüber dem URL Rewriting einfacher anzuwenden sind. Zwar werden beide Verfahren von modernen Web-Sprachen gut unterstützt, aber URL Rewriting ist mit einem größeren Aufwand (es müssen sämtliche Links jeder Seite geändert werden) verbunden und somit nicht nur langsamer sondern auch fehleranfälliger. Zudem ist URL Rewriting sicherheitskritischer, da andere Benutzer durch „Entführung“ von Sessions an fremde Session-Daten gelangen können. Problematisch ist insbesondere, dass

1) Suchmaschinen die SID in den Links mit abspeichern und somit anderen Usern zur Verfügung stellen können,
2) Adressen mit SID in Bookmarks gespeichert werden und
3) bei Verweisen aus der Session auf fremde Seiten die SID als Teil der Adresse der verweisenden Seite dem Betreiber der verlinkten Seite zugänglich wird.

Da aber oftmals Cookies durch den Benutzer unterbunden werden (vgl. 2.3.2), wird häufig eine Fallback-Lösung mit URL Rewriting eingesetzt.

Versteckte Formularfelder spielen aufgrund ihres beschränkten Einsatzbereichs (vgl. 2.3.3) eine untergeordnete Rolle.

Benutzerauthentifizierung wird oft eingesetzt, meist zusammen mit anderen Session-Management-Techniken in Form verschachtelter Sessions (vgl. 1.2.3). Häufig wird dabei eine von der Web-sprache abhängige Authentifizierung über ein HTML-Formular anstatt der Loginbox verwendet.

4 Anhang

4.1 Quellenverzeichnis

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

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

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

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

4.2 Impressum

Ausarbeitung 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