Die Krux mit dem Browser

Probleme bei der Realisierung web-basierter Geschäftsanwendungen

JavaForum Stuttgart
27. Juni 2002

Roman Seibold und Jörg Hettel

Inhalt

- Charakteristik von Web-Anwendungen
  - Das HTTP-Protokoll
- Probleme mit dem Browser
- Problemlösungen
  - "s*endRedirect*" als Lösung für das Back- und Reload-Button Problem
  - Web-Anwendung als Zustandsautomat zur Lösung der Navigationsproblematik
- Zusammenfassung

Charakteristik von Web-Anwendungen

- Bei der Entwicklung von Web-Anwendungen müssen verschiedene Problembereiche adressiert werden
- Beispiele:
  - Benutzerfreundlichkeit/Benutzerführung
  - Security
  - Ausfallsicherheit
  - Mehrsprachigkeit
  - Integration von Bestandssystemen
  - Schnelle Entwicklung
  - etc.

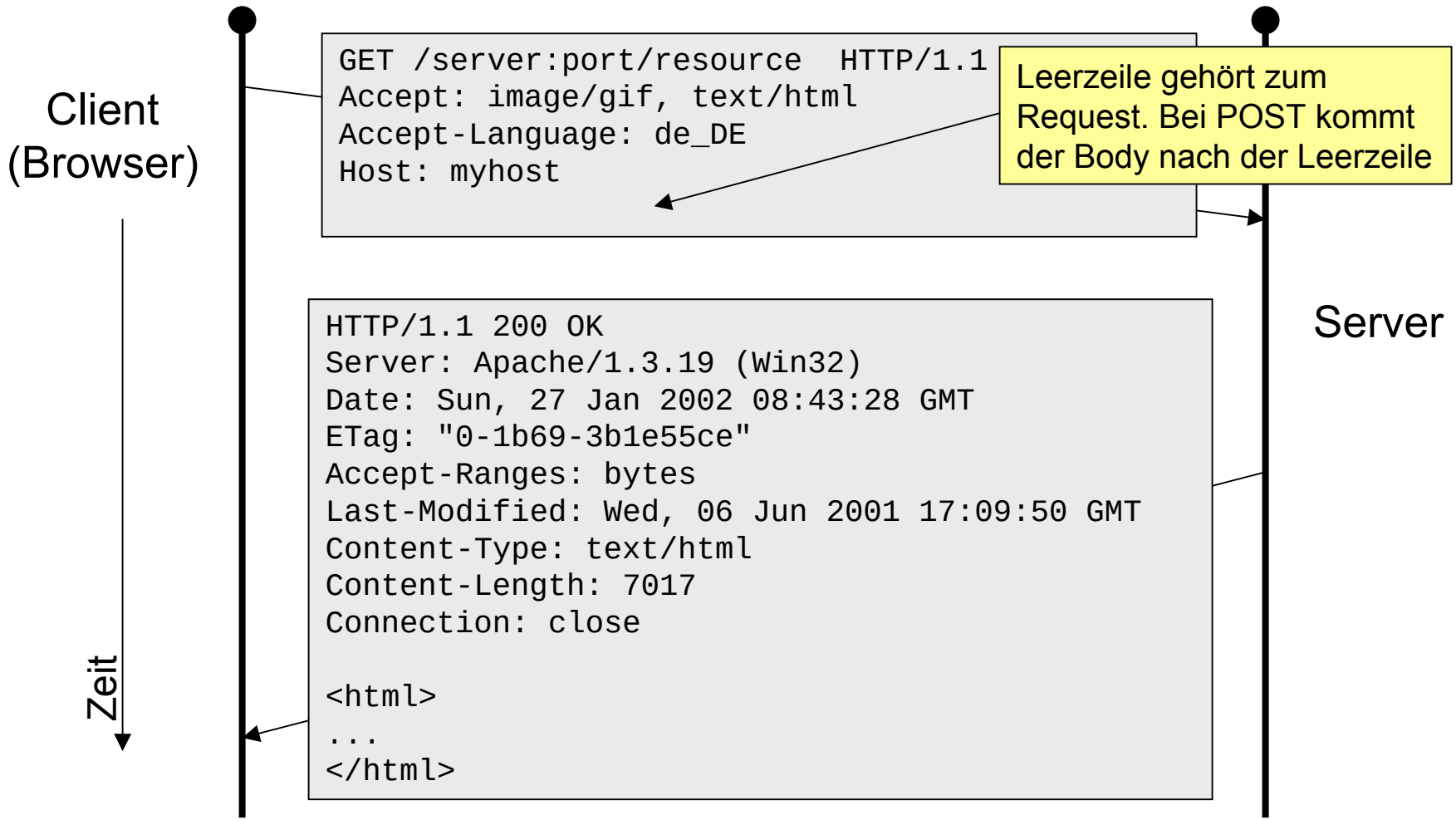
Charakteristik von Web-Anwendungen

- Neue technologische Herausforderungen
  - HTTP als Standard-Protokoll
    - http ist zustandslos
  - Browser als GUI-Frontend
  - HTML als hauptsächliche Client-Sprache
    - Keine direkte Interaktionen möglich (z.B. Eingabeüberprüfungen, kontextbezogenes Freischalten von Eingabefelder, etc.)
  - Web-Server-Plugins als Schnittstelle zu Programmen
    - CGI, Fast-CGI, ISAPI, NSAPI, Servlet-Engine, etc.

Das HTTP-Protokoll

- HTTP = Hyper Text Transfer Protocol
- HTTP-Protokoll besitzt eine Request-Response Semantik
- Das Protokoll ist zustandslos
- Spezifikation unterliegt dem W3C
  - URL: http://www.w3.org/
- Kennt verschiedene Request-Arten
  - Bsp: GET, POST, etc.
- Antworten haben einen Nummerncode
  - Bsp: 404 Not Found, 200 OK, etc.

Request- und Response-Ablauf



Wiedererkennung

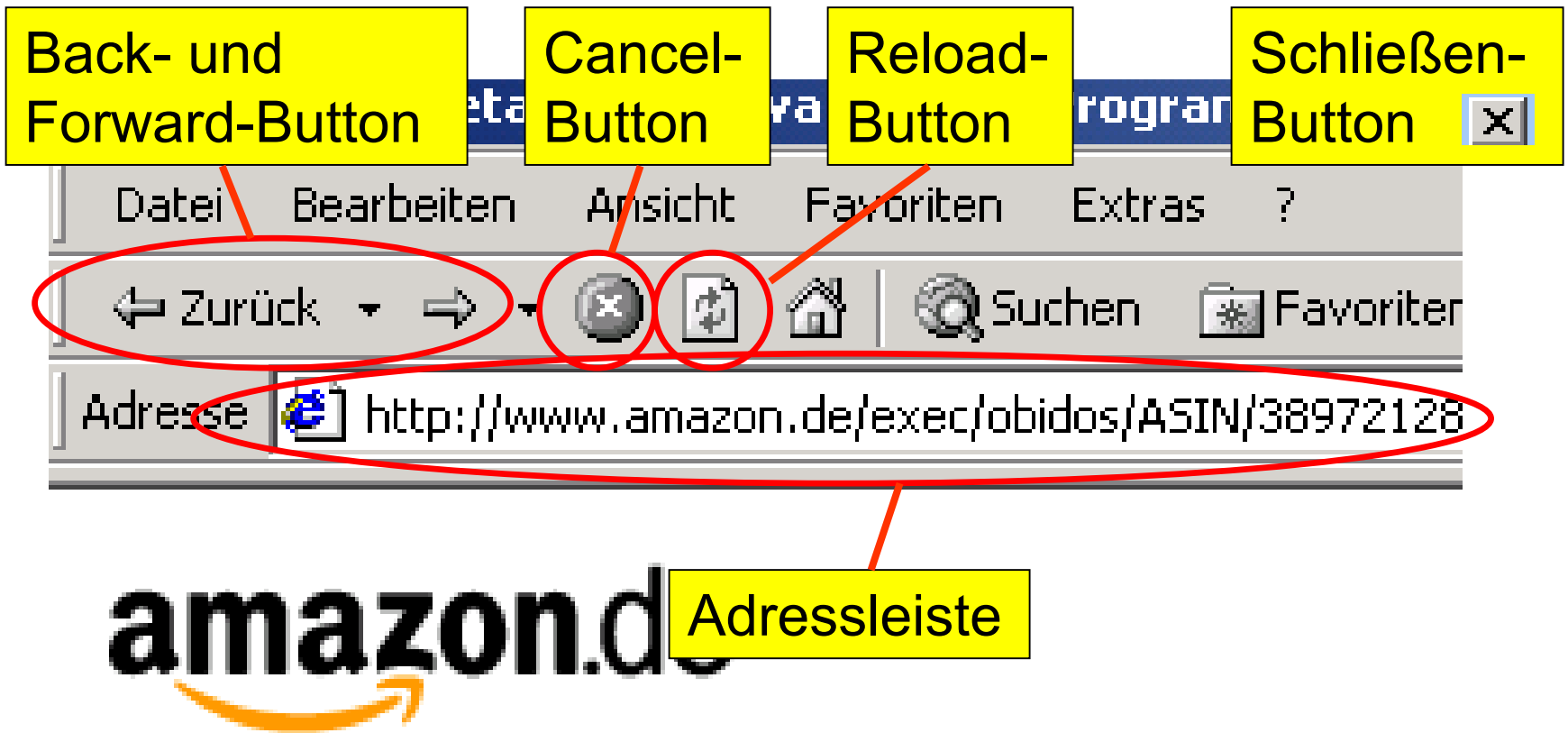
- HTTP kennt keine "Sessions"
- Cookies und URL-Rewriting sind Techniken zur Simulation von "Sessions"
  - Cookie-Unterstützung kann am Browser ausgeschaltet werden
  - Cookies können nichts anstellen
    - Cookies können keine Informationen stehlen (ActiveX oder Scripts sind in dieser Hinsicht viel gefährlicher)
  - Cookies können aber zum Datensammeln missbraucht werden
    - Siehe Firma *DoubleClick*, die Werbung auf vielen kommerziellen Sites platziert und über Cookies das Browse-Verhalten der Interessenten aufzeichnet und auswertet

Inhalt

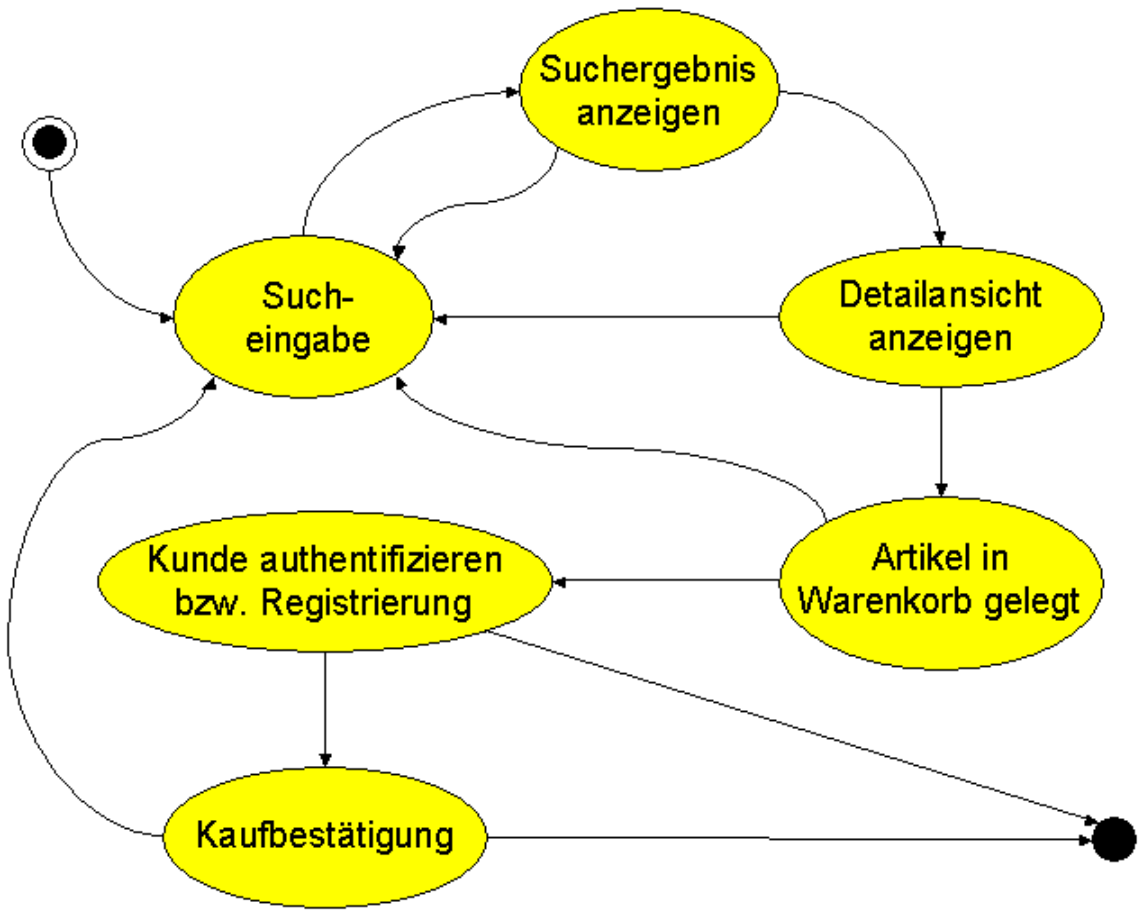
- ✓ Charakteristik von Web-Anwendungen
  - ✓ Das HTTP-Protokoll
- □ Probleme mit dem Browser als Client
- □ Problemlösungen
  - □ "sendRedirect" als Lösung für das Back- und Reload-Button Problem
  - □ Web-Anwendung als Zustandsautomat zur Lösung der Navigationsproblematik
- □ Zusammenfassung

Probleme mit dem Browser als Client

- "Gefährliche Stellen" am Browser

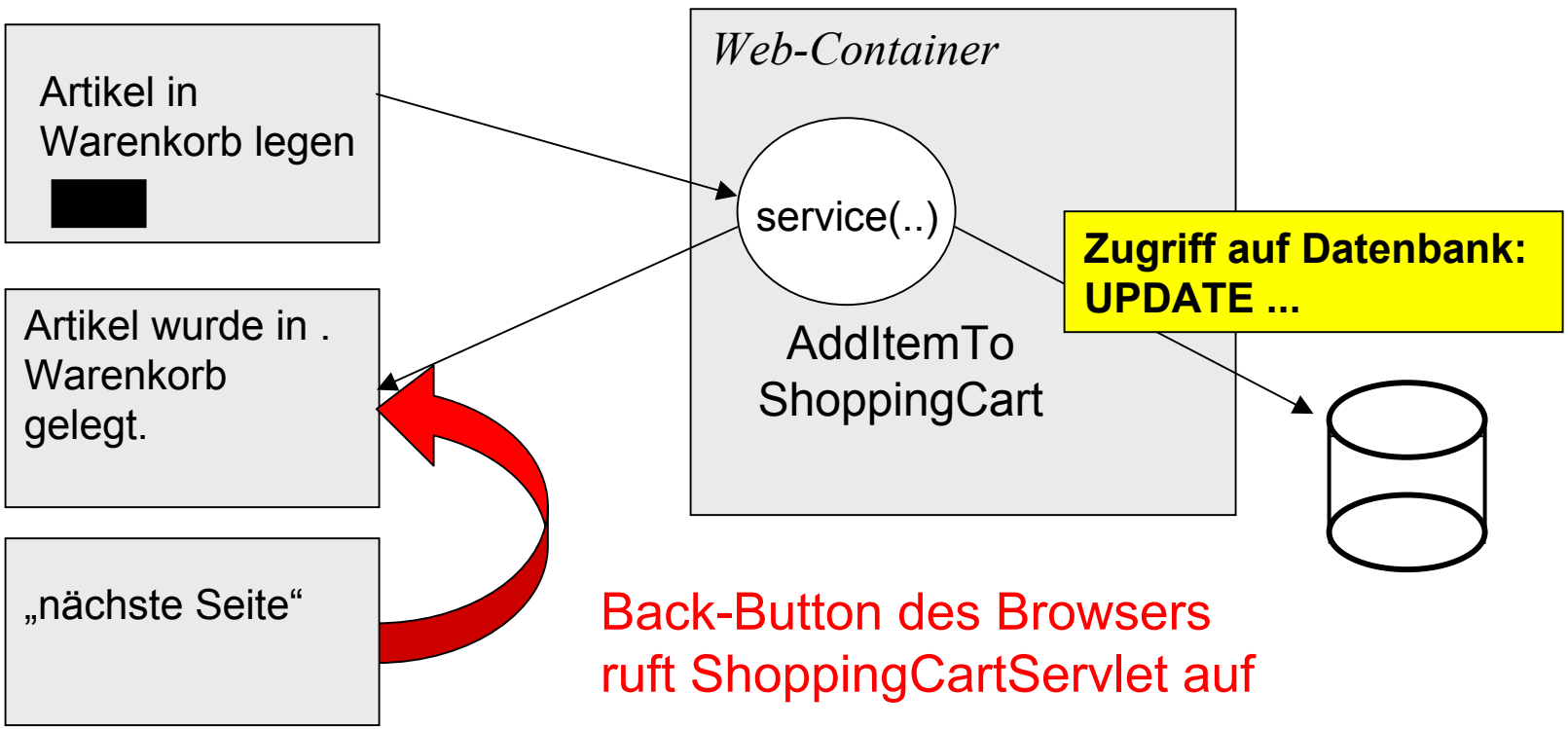


Motivationsbeispiel: Web-Shop



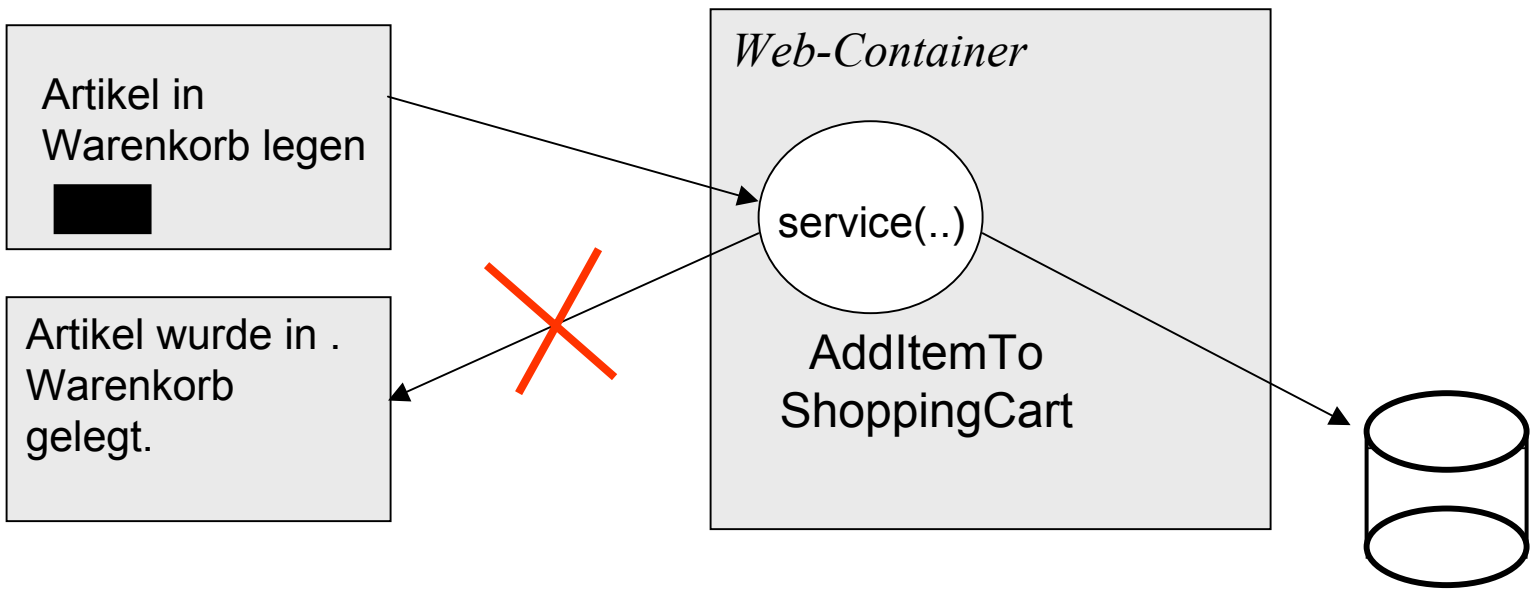
Back- und Reload-Button

- Durch den Back- und Reload-Button können unbeabsichtigt Requests ausgelöst werden

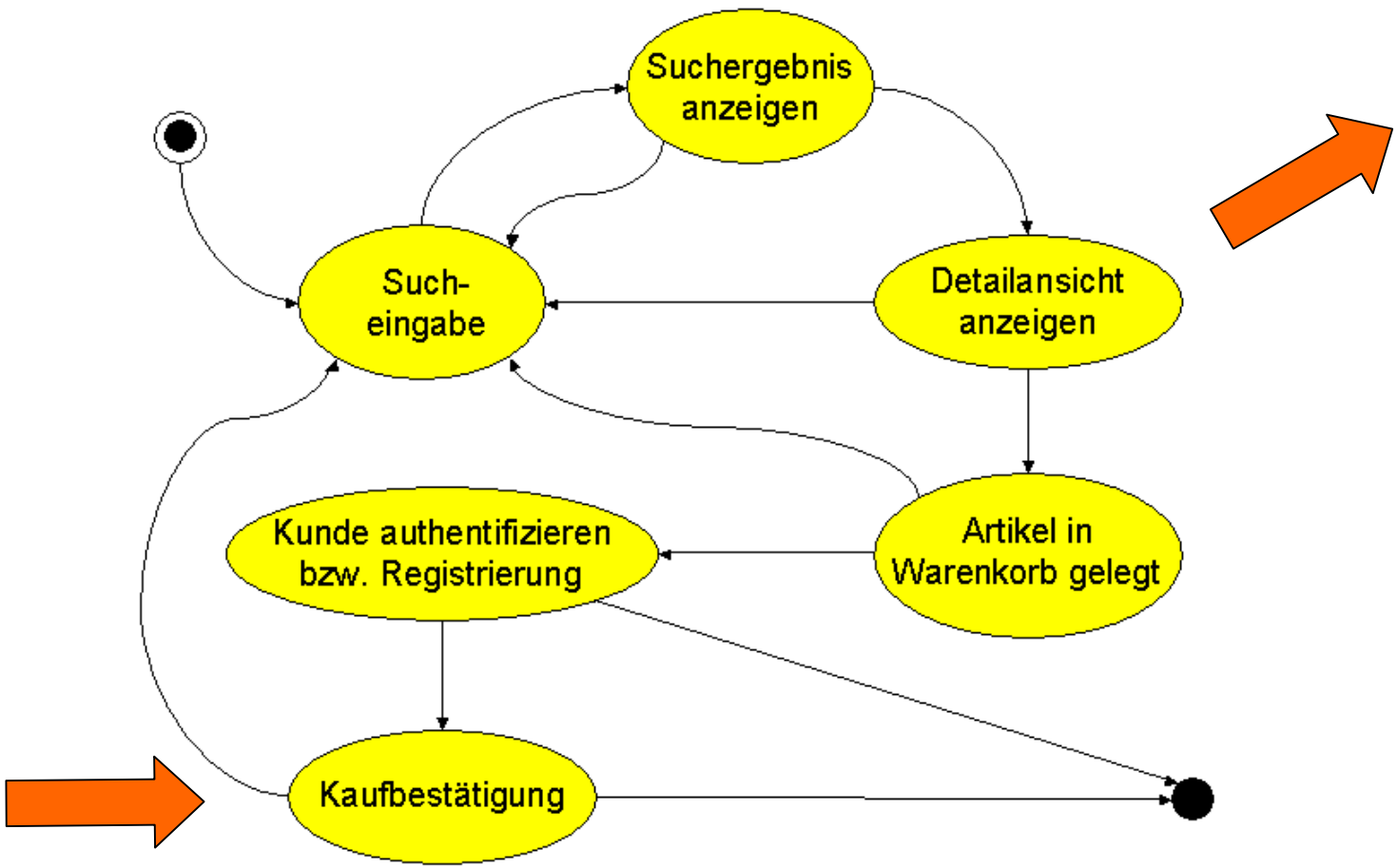


Cancel-Button

- Das Drücken des Cancel-Buttons verhindert die Auslieferung der Antwort



Direkte Navigationseingabe



Inhalt

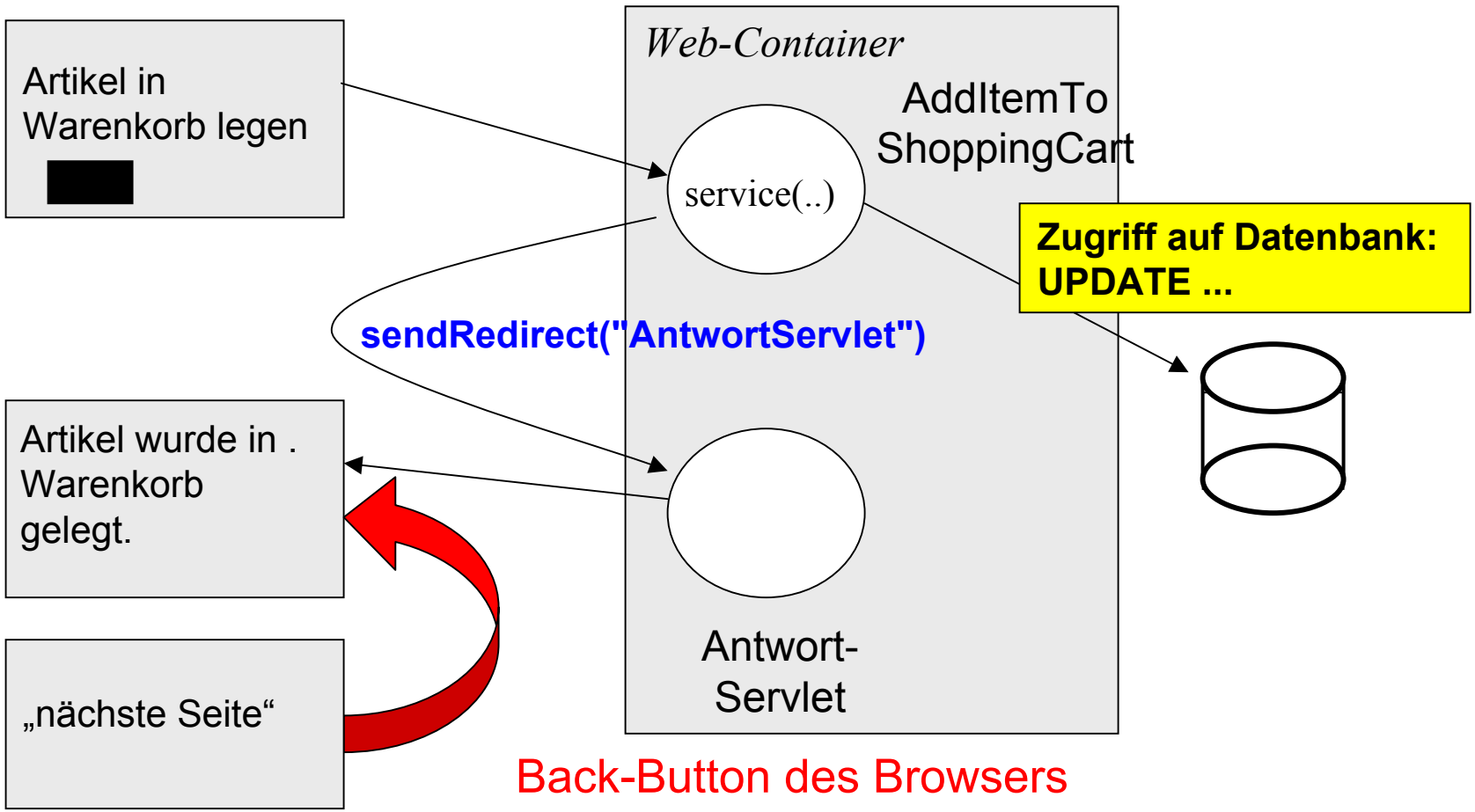
- ✓ Charakteristik von Web-Anwendungen
  - ✓ Das HTTP-Protokoll
- ✓ Probleme mit dem Browser als Client
- □ Problemlösungen
  - □ "sendRedirect" als Lösung für das Back- und Reload-Button Problem
  - □ Web-Anwendung als Zustandsautomat zur Lösung der Navigationsproblematik
- □ Zusammenfassung

sendRedirect-Lösung

- Verwenden von *sendRedirect* löst auf einfache Art und Weise das Back- und Reload-Button Problem

- **Idee:** Antwort kommt von einer "anderen" URL

Wirkung von *sendRedirect*



Code-Beispiel: *sendRedirect*

```
// Benutzung von sendRedirect
response.sendRedirect( webResource );
```

```
HTTP/1.1 302 Found
Date: Sun, 27 Jan 2002 10:35:33 GMT
Server: Apache/1.3.19 (Win32)
Location: http://server/webApp/webResource
Content-Length: 0
Content-Type: text/html
Content-Language: de
```

HTTP-Antwort an den Client.

Der Client fordert eine neue Ressource an.

Probleme mit dem *sendRedirect*

- Ein sendRedirect setzt in vielen Fällen einen Return-Code *302 Found*
  - Bei *302* ist nicht klar spezifiziert, ob die neue Ressource über GET oder POST angefordert wird
  - Oft wird GET genommen, unabhängig von der Art des aufrufenden Requests

Sichereres *sendRedirect*

- Benutzung der Return-Codes *303 See Other* oder *307 Temporary Redirect*
- Bei *303* wird neue Ressource immer über GET geholt
- Bei *307* wird die ursprüngliche Requestart beibehalten
  - Bei einem POST-Request fragt der Browser üblicherweise nach, ob der Benutzer mit einem Redirect einverstanden ist, da hier ja auch potentiell Daten an eine andere Ressource geschickt werden

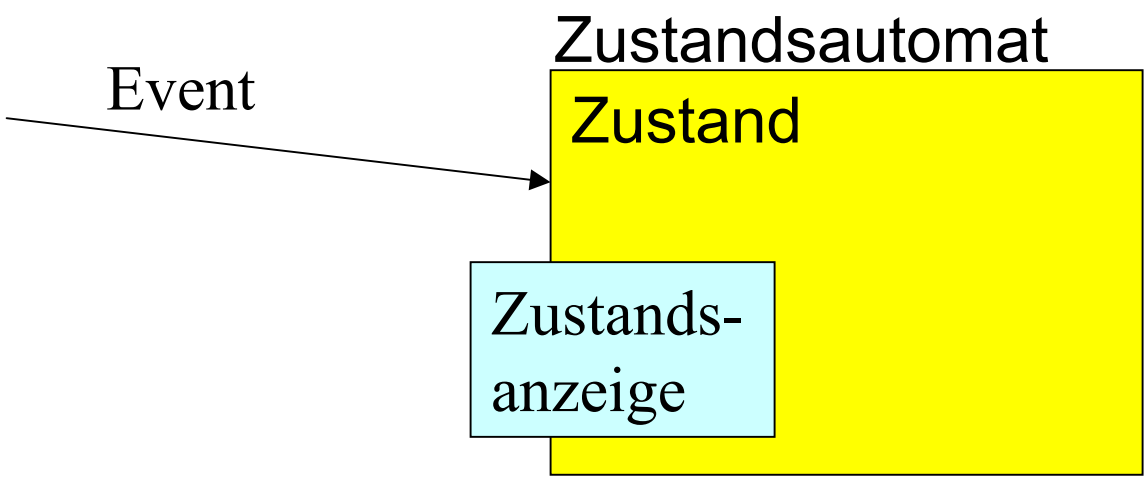
Direktes Setzen des Headers

```
// Alternative:
// Explizites setzten von Header-Einträgen
response.setStatus(
         HttpServletResponse.SC_MOVED_TEMPORARILY);
response.setHeader("Location", newAbsolutURL );
```

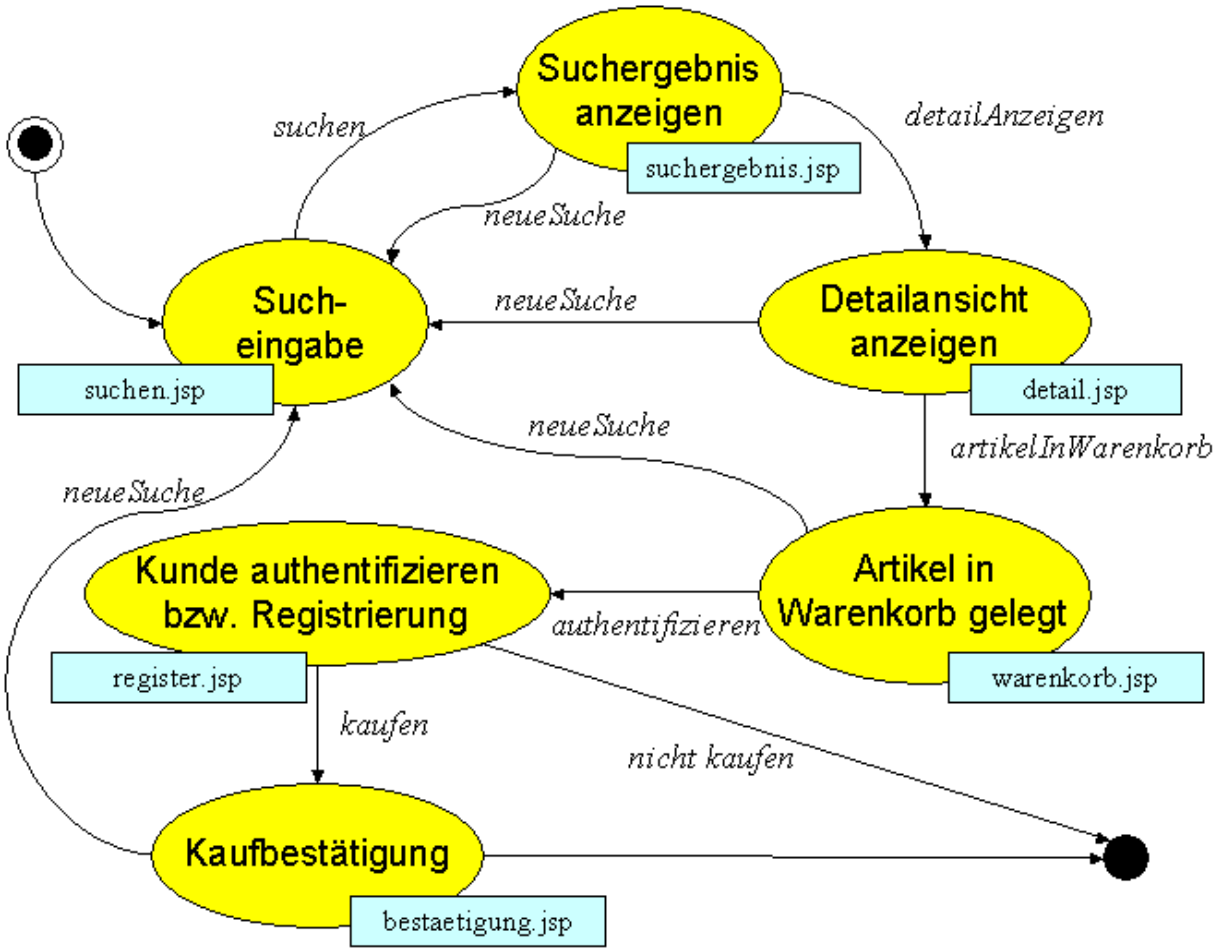
- **Problem:** Nicht alle Browser unterstützen die "neuen" Return-Codes

Web-Anwendung als Zustandsautomat

- Requests entsprechen Events an einen Zustandsautomaten
- Events können Zustandsänderungen hervorrufen



Zustandsdiagramm



Code-Beispiel (1)

```
// Liste alle Zustände
State suchEingabe   =
     new State("Sucheingabe", "suchen.jsp");
State suchErgebnis  =
     new State("Suchergebnis", "suchergebnis.jsp");
State detailAnzeige =
     new State("Detail Anzeige", "detail.jsp");
....


// Liste der Übergänge
Transition fromSucheingabeToSuchergebnis =
       new Transition("Sucheingabe",
                      suchEingabe, suchErgebnis );
....
```

Code-Beispiel (2)

```
....
// Zuordnung der Übergänge und der auslösenden
// Events zu den Zuständen

suchEingabe.addTransition("suchen",
                    fromSucheingabeToSuchergebnis);
....
```

- Besserer Weg:
  Konfiguration des Automaten in einer XML-Datei hinterlegen

Problem mit Caching

- Caching ist aus Performance-Gründen reizvoll
- **Gefahr:** "Zustand der Anwendung im Browser" und Zustand der eigentlichen Anwendung divergieren.
- Caching kann ausgeschaltet werden:

```
// disable caching
response.setHeader("Pragma","no-cache");
response.setHeader("Cache-Control",
                   "no-cache");
response.setDateHeader("Expires",0);
```

Problem mit Browsern

- **Achtung:** Unterschiedliches Browserverhalten
  - IE 5 ignoriert das Ausschalten des Browsercaches
    - http://support.microsoft.com/support/kb/articles/Q222/0/64.ASP
    - http://support.microsoft.com/support/kb/articles/Q234/2/47.ASP

Inhalt

- ✓ Charakteristik von Web-Anwendungen
  - ✓ Das HTTP-Protokoll
- ✓ Probleme mit dem Browser als Client
- ✓ Problemlösungen
  - ✓ "sendRedirect" als Lösung für das Back- und Reload-Button Problem
  - ✓ Web-Anwendung als Zustandsautomat zur Lösung der Navigationsproblematik
- □ Zusammenfassung

Zusammenfassung

- Bei Web-Anwendungen hat man es mit entkoppelten Clients zu tun.
  - Folge des zustandslosen HTTP-Protokolls
- Über entsprechende Techniken und Anwendungsdesigns können die daraus resultierenden Schwierigkeiten gelöst werden
- Web-Anwendungen müssen von vornherein als Web-Anwendungen konzipiert werden
  - Analogon: Bei Verteilten Anwendungen ist der *Unified Object View* auch nicht realisierbar

Inhalt

- ✓ Charakteristik von Web-Anwendungen
  - ✓ Das HTTP-Protokoll
- ✓ Probleme mit dem Browser als Client
- ✓ Problemlösungen
  - ✓ "sendRedirect" als Lösung für das Back- und Reload-Button Problem
  - ✓ Web-Anwendung als Zustandsautomat zur Lösung der Navigationsproblematik
- ✓ Zusammenfassung

Literatur

1. Avedal, Karl et al.: *Professional JSP*, Wrox Press Ltd, 2000
2. Hunter, J. und Crawford W.: *Java Servlet Programming*, O'Reilly, 2001
3. Jakarta Struts Homepage: http://jakarta.apache.org/struts
4. Http-Spezification: http://www.w3c.org
5. Treese, G.W. und Stewart, L.C.: *Designing Systems for Internet Commerce*, Addison Wesley, 1998
6. SUN‘s J2EE Spezifikation: http://java.sun.com/products/j2ee
7. Yacoub, S.M. und Ammar, H.H.: *Finite State Machine Patterns*, White Paper