Praktikum

DBAE / WI

Servlets

Pascal Reuss. M.Sc.

Raum A08b Spl.
Email: reusspa@uni-hildesheim.de

Intelligente Informationssysteme

Stiftung Universität Hildesheim 2003

Client-Server-Architektur

Typische Netzwerkanwendungen bestehen aus *Clients* und *Server*

Client : *(Gast)*

Initiiert den Kontakt mit dem Server („speaks first“)

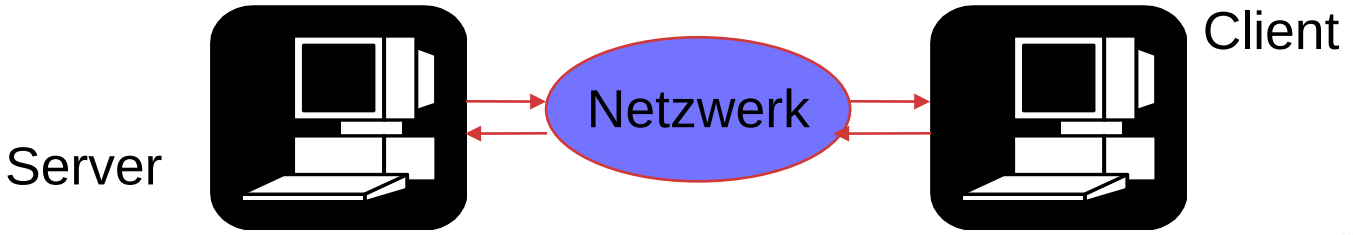
Benutzt Services des Servers

Für Web ist der Client der Web Browser, für E-Mail – Mail Client

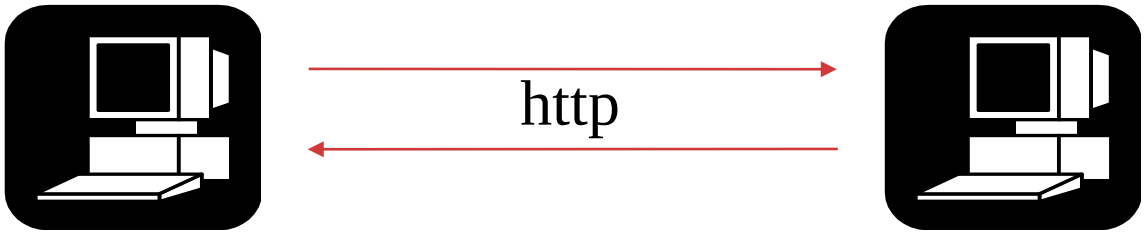
Server : (Kellner)

liefert die Services für den Client

z.B. Web-Server liefert die angeforderte Seite, IMAP, POP3-Server liefert die E-Mail.



Web-basierter Client-Server Verbund



Web Browser

Web Server

Benutzer - Interface:

- HTML pages
- JavaScript
- Java Applets

liefert:

- statische/dynamische Seiten
- Applet code
- „server side business logic“

HTTP Basics

HTTP ist ein einfaches zustandloses Protokoll.

Der Client (der Web Browser), macht eine Anfrage, der Web Server liefert eine Antwort, (Ressource) → die Transaktion ist erledigt.

Detailiert:

Der Client spezifiziert den HTTP-Befehl, genannt Methode, die dem Server mitteilt, welche Aktion ausgeführt werden soll

Der Client spezifiziert die Adresse des Dokuments (URL)

Der Client spezifiziert die Version des von ihm benutzten HTTP-Protokolls

Der Client spezifiziert optionale Information

GET /intro.html HTTP/1.0

(Diese Anfrage verwendet die GET-Methode, um nach dem Dokument *intro.html* zu fragen. Benutzt dabei wird HTTP Version 1.0)

HTTP Basics

- Der Server spezifiziert die Version des von ihm verwendeten HTTP
- Der Server spezifiziert den Status
- Der Server spezifiziert die Beschreibung des Status

HTTP/1.0 200 OK oder *HTTP/1.0 404 Not Found*

Nach der Statuszeile sendet der Server die Kopfdaten, die dem Client Informationen mitteilen → welche Software benutzt der Server, Typ der Antwortdatei, ...

Date: Saturday, 23-May-00 03:25:12 GMT
Server: Tomcat Web Server/3.2
MIME-version: 1.0
Content-type: text/html
Content-length: 1029
Last-modified: Thursday, 7-May-00 12:15:35 GMT

Wurde die Anfrage erfolgreich bearbeitet, sendet der Server die Daten. Andernfalls liefert der Server eine (für den Menschen verständliche) Erklärung, wieso die Anfrage nicht erledigt ist.

HTTP (GET and POST)

Die meist benutzten Methoden sind GET und POST

Die GET- Methode fragt Information ab (ein Dokument / Ergebnisse der Anfrage an eine Datenbank) „Fragen“

Die POST- Methode versendet Informationen (Kreditkartennummer, die in eine Datenbank abzulegende Information) „Sagen“

Andere Methoden sind: PUT, DELETE, HEAD, TRACE, OPTIONS

Dynamische Herstellung von Antworten (HTML-Seiten) : Servlets

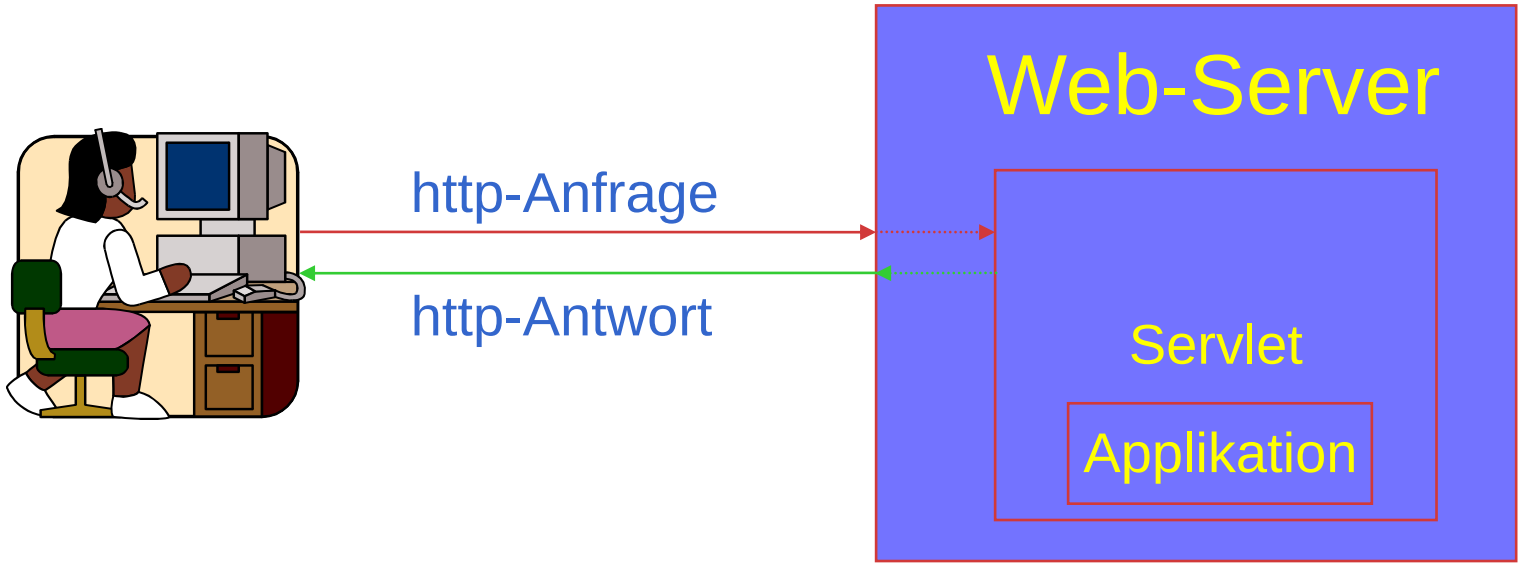
Servlet

Ein Servlet ist eine kleine Applikation

Es wird auf dem Server ausgeführt

Es wird wie eine HTML Seite adressiert

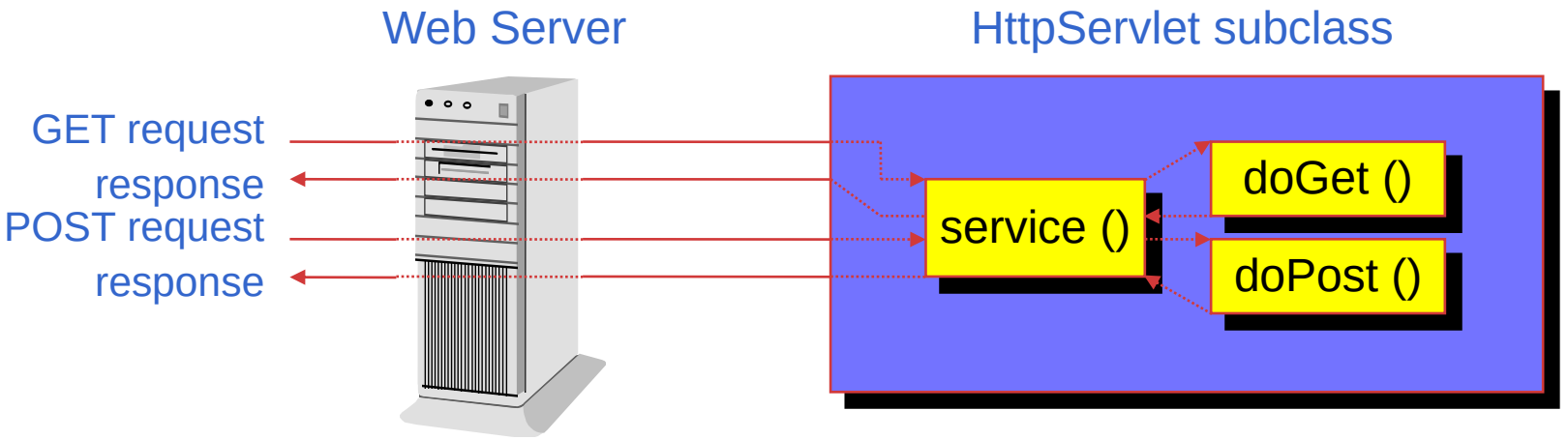
Es erzeugt die vom Client erwartete Antwort (HTML) on-the-fly.



Servlets-Technologie

- Web-Server muss für Servlets Java-fähig sein
- Servlet ↔ Applet : Applet Clientseitige Javaanwendung, Servlet serverseitig
- Java-Klassen des Servlets können auf dem Server ausgeführt werden
- Servlets erweitern die Funktionalitäten des Web-Servers genau so wie Applets die Funktionalitäten von Web-Seiten erweitern
- Servlets sind plattform-unabhängig (Java)
- Servlets laufen als Thread innerhalb des Webservers
- Servlets können mehrere Eigenschaften des Servers benutzen
  - z.B. in die log-Datei des Servers schreiben

Servlet: Umgang mit GET und POST requests



Servlets benutzen Klassen und Interfaces aus zwei Packages: *javax.servlet* und *javax.servlet.http*

Das *javax.servlet* Package enthält Klassen und Interfaces für die Unterstützung der generischen, protokoll-unabhängigen Servlets.

Für die HTTP-Funktionalität werden diese Klassen durch die Klassen im *javax.servlet.http* Package erweitert.

Für die Verarbeitung der GET und POST Anfragen überschreibt ein HTTP-Servlet die Methoden `doGet()` und `doPost()`.

Hello World - Servlet

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class HelloWorld extends HttpServlet {

   public void doGet(HttpServletRequest req, HttpServletResponse res) throws
   ServletException, IOException {          // reagiere auf req mit res

      res.setContentType("text/html");        // res = welcher Art
      PrintWriter out = res.getWriter();      // res = schreibe in res

      out.println("<HTML>");                                  // res schreiben
      out.println("<HEAD><TITLE>Hello World</TITLE></HEAD>");
      out.println("<BODY>");
      out.println("<BIG>Hello World</BIG>");
      out.println("</BODY></HTML>");
      out.close();
   }
}
```

Hello World - Servlet

Dieses Servlet erweitert HttpServlet Klasse und überschreibt die Methode `doGet()`

Bekommt der Web-Server eine GET-Anfrage, dann ruft er die Methode `doGet()` auf und übergibt das *HttpServletRequest* Objekt (req) und das *HttpServletResponse* Objekt (res) .

Das *HttpServletRequest* Objekt repräsentiert die Anfrage des Clients.

Das *HttpServletResponse* Objekt repräsentiert die Antwort des Servlets.

Formulardaten verarbeiten

If you don't mind me asking, what is your name?

Submit Query

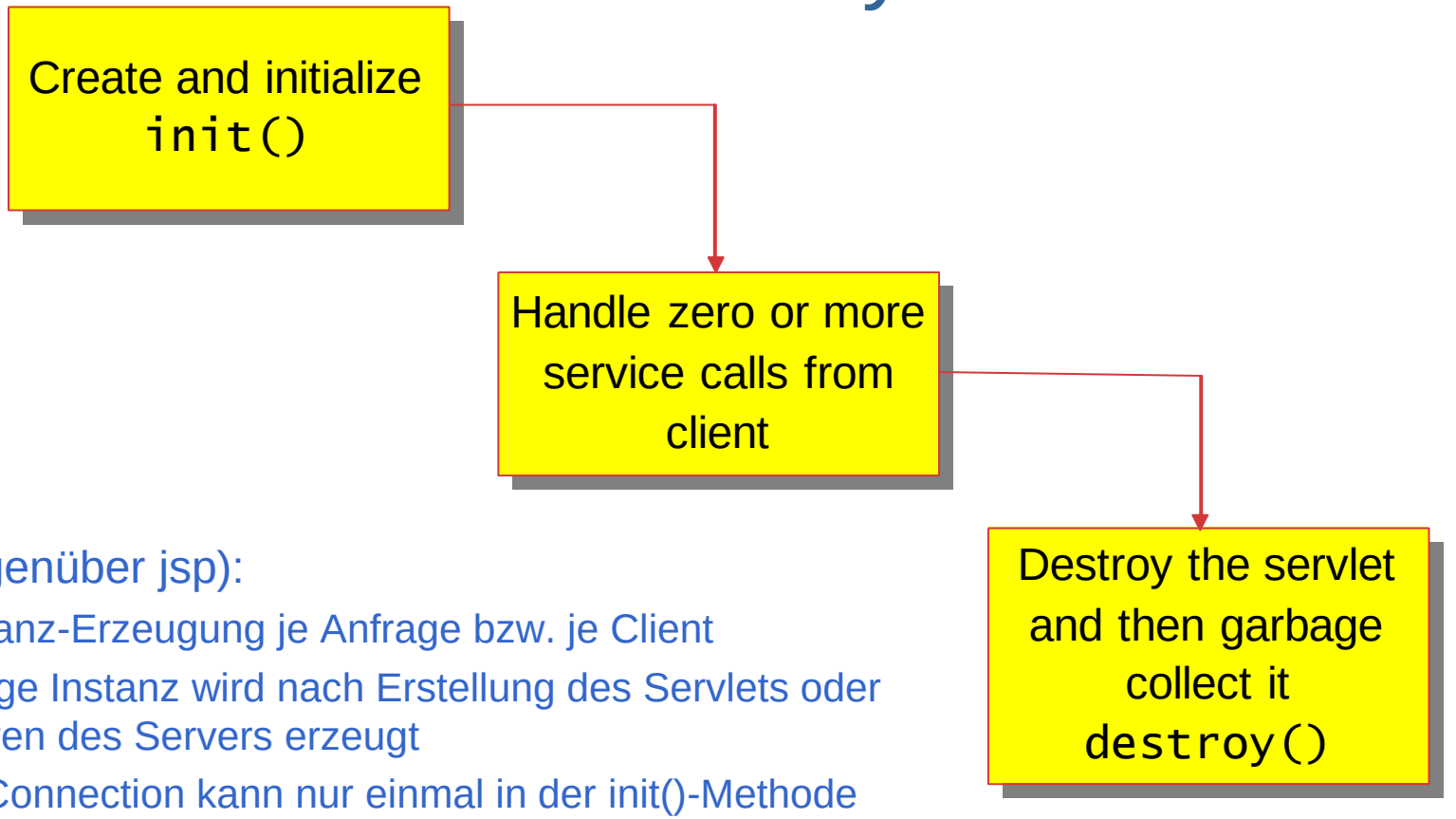
```
<html>
<head>
<title> Introductions </title>
</head>

<body>
  <form method=GET action="/HelloWorld/name">
        If you don't mind me asking, what is your name?
<input type=text name= "name"><P>
<input type=submit>
</form>
</body>
</html>
```

Verwendung von Formulardaten: „Hallo Name“

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
Public class HelloWorld extends HttpServlet {
  public void doGet(HttpServletRequest req, HttpServletResponse res)   throws
   ServletException, IOException {
  res.setContentType(“text/html“);
  PrintWriter out = res.getWriter();
  String name = req.getParameter(“name“);
  out.println(“<HTML>“);
  out.println(“<HEAD><TITLE>Hello World</TITLE></HEAD>“);
  out.println(“<BODY>“);
  out.println(“<BIG>Hello, </BIG>“ + name);
  out.println(“</BODY></HTML>“);
  out.close()
  }
}
```

Servlet-Lebenszyklus



Vorteile (gegenüber jsp):

- keine Instanz-Erzeugung je Anfrage bzw. je Client
- Eine einzige Instanz wird nach Erstellung des Servlets oder Hochfahren des Servers erzeugt
- z.B.: DB-Connection kann nur einmal in der init()-Methode erzeugt, und in destroy() geschlossen werden

GET vs. POST

	GET	POST
Service-Methode	`doGet()`	`doPost()`
Verwendungszweck	Daten vom Server holen	Daten an dem Server übermitteln und Antwort abwarten
Formulardaten	Werden in Form von Name-Value-Paaren an den URL angefügt	Werden im Request-Body übermittelt und sind kein Bestandteil des URL
Lände des Requests	Ist limitiert auf ~ 1KB	Ist nicht limitiert
Daten	Benutzer sieht Formulardaten in der Adresszeile des Browsers	Werden vor dem Benutzer verborgen
Bookmarks	Können einfach gesetzt werden	Können weniger gut gesetzt werden
Verwendung	Häufiger	Weniger häufig

Ein einfacher Zähler

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

Public class SimpleCounter extends HttpServlet {

   int count = 0;

   public void doGet(HttpServletRequest req ,HttpServletResponse res) throws ServletException,
      IOException {

         res.setContentType(“text/html“);
         PrintWriter out = res.getWriter();
         count++;
         out.println(“Since loading, this servlet has been accessed“ + count + “times.“);
    }
}
```

Session Tracking

- Die “Shopping Cart” (Einkaufswagen) Applikation ist ein klassisches Beispiel.
- Der Client soll die Einkäufe in den virtuellen Einkaufswagen legen können
- Der Server muss die Einkäufe speichern bis der Client mit dem Einkauf fertig ist.
- HTTP liefert keine Möglichkeit um festzustellen, dass eine Sequenz von Anfragen von einem und demselben Kunden stammt.
- Cookies
- URL-Rewriting
- usw.

Session Tracking (Servlets)

Servlets unterstützen Session Tracking

Jeder Benutzer der Seite wird mit dem *javax.servlet.http.HttpSession* Objekt assoziiert

In einem Session-Objekt kann eine Menge der Java-Objekte abgelegt werden

z.B.: In einem Session-Objekt werden die Einkäufe eines Kunden abgelegt.

public HttpSession HttpServletRequest.getSession()

liefert das Session-Objekt, dass mit dem anfragenden Benutzer assoziiert wird

falls kein Session-Objekt vorhanden: wird ein neues erstellt

Session Tracking (Servlets)

Durch die Methode `setAttribute()` werden Objekte an einen *HttpSession* Objekt angehängt

public void HttpSession.setAttribute(String name, Object value)

Diese Methode bindet ein Objekt an einen spezifizierten Namen

Die Methode `getAttribute()` liefert die gespeicherten Objekten in einem Session-Objekt.

public Object HttpSession.getAttribute(String name)

Die Aufzählung von allen gespeicherten Objekten erhält man durch die Methode `getAttributeNames()`.

public Enumeration HttpSession.getAttributeNames()

Session Tracking (Servlets)

Ein Objekt wird durch die Methode removeAttribute() aus einem Session-Objekt entfernt.

public void HttpSession.removeAttribute(String name)

“Objekt” immer: [Name][Wert] →

Session-Objekt ([Name,Wert],[Name,Wert],[Name,Wert],...)

Hitcounter mit Session Tracking realisiert

```
import java.io.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;
Public class SimpleCounter extends HttpServlet {
   public void doGet(HttpServletRequest req ,HttpServletResponse res)
      throws ServletException, IOException {
   res.setContentType("text/html");
   PrintWriter out = res.getWriter();
    // Get the current session object, create one if necessary
    HttpSession session = req.getSession();
```

Hitcounter mit Session Tracking realisiert

```
    // Increment the hit count for this page. The value is saved
    // in this client‘s session under the name „tracker.count“.
    Integer count = (Integer)session.getAttribute(“tracker.count“);
    if (count == null) { count = new Integer(1); }
    else { count = new Integer(count.intValue() + 1); }
    session.setAttribute(“tracker.count“, count);
    out.println(“You‘ve visited this page“ + count + “times.“);
  }
}
```

Einbinden weiterer, Lokalisierung von Servlets

Einbinden weiterer Servlets:

Mit Hilfe von RequestDispatcher

Lokalisierung:

Abfragen bzw. Setzen eines *Locale* Objektes

Behandeln von Sprache, Land in einer Ein-/Ausgabe

Abfragen der Einstellungen über Request Objekt

```
Locale  getLocale();        // Lies lokale Sprache aus der Anfrage aus
```

Setzen der Einstellungen über das Response Objekt

```
void setLocale();           // Stell lokale Sprache in der Antwort ein
```

Referenzen

Java ist auch eine Insel

Das J2EE Codebook