Web-Technologien kurz & knapp

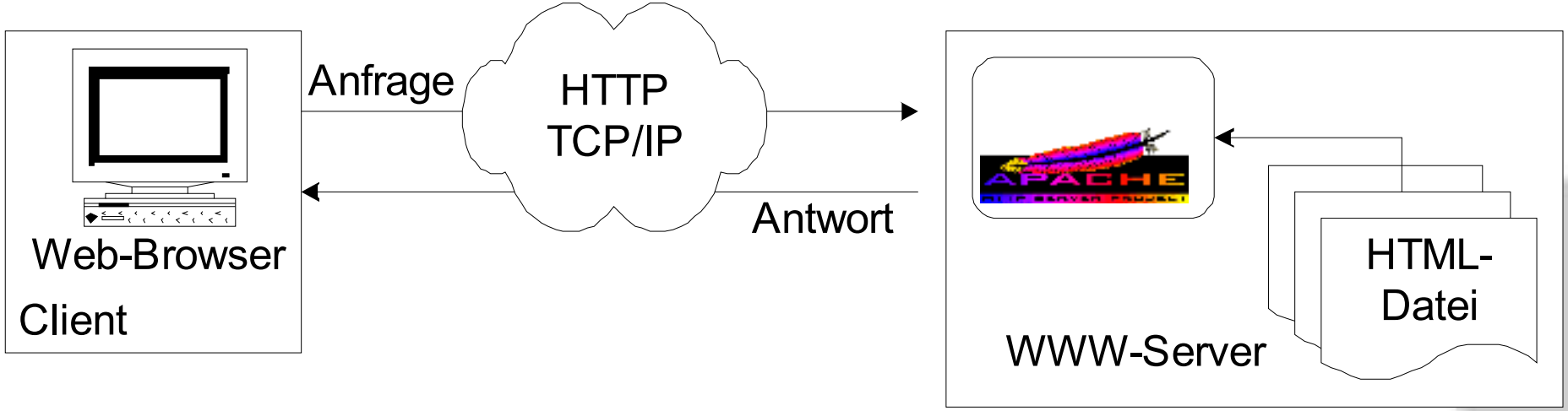
Software-Praktikum im Grundstudium WS 2004/2005

Dipl.-Inform. Michael Kirchhof
Dipl.-Inform. Bodo Kraft
Prof. Dr.-Ing. Manfred Nagl

Department of Computer Science III
Software Engineering
Ahornstr. 55
52074 Aachen, Germany

http://www-i3.informatik.rwth-aachen.de

World Wide Web



HTTP

- HyperText Transfer Protocol
- GET

```
GET /login.html?username=becker HTTP/1.0
User-Agent: Mozilla/4.51 [en] (WinNT; I)
Accept: image/gif, image/jpeg, */*
```

Informations-übertragung in der URL

Methode

Header

Body ist leer

- POST

```
POST /login.html HTTP/1.0
User-Agent: Mozilla/4.51 [en] (WinNT; I)
Accept: image/gif, image/jpeg, */*
Content-Length: 16
Content-Type: application/x-www-form-urlencoded

username=becker
```

Header

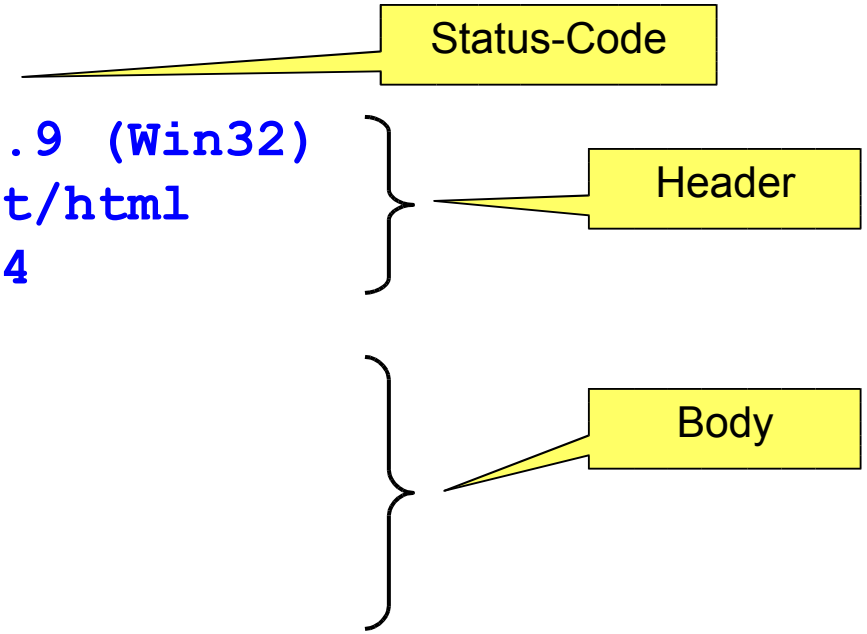
Body

- und andere

HTTP - Response

```
HTTP/1.0 200 OK
Server:Apache/1.3.9 (Win32)
Content-Type: text/html
Content-Length: 94

<HTML>
<HEAD>
<TITLE>...
</HTML>
```



HTML-Formulare

```
<FORM METHOD="POST" ACTION="./PerformEdit">
    Bezeichnung:
    <INPUT TYPE="TEXT" NAME="TerminBez"
           SIZE="35" value="erster Termin" ><BR>
   Datum:
    <INPUT TYPE="TEXT" NAME="TerminAm" SIZE="25"
           value="11.10.2001"><BR>
    <INPUT TYPE="SUBMIT" VALUE="OK">
   <INPUT TYPE="RESET" VALUE="Reset">
</FORM>
```

einzeiliges Textfeld

Button zum Abschicken des Formulars

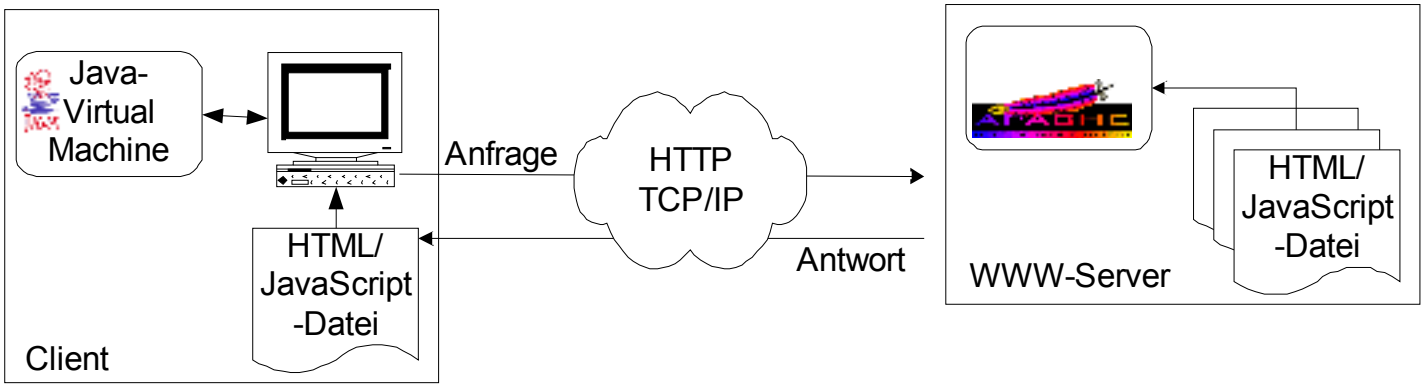
Button zum Rücksetzen des Formulars

Post: Daten werden für den Benutzer unsichtbar im HTTP-Request übertragen

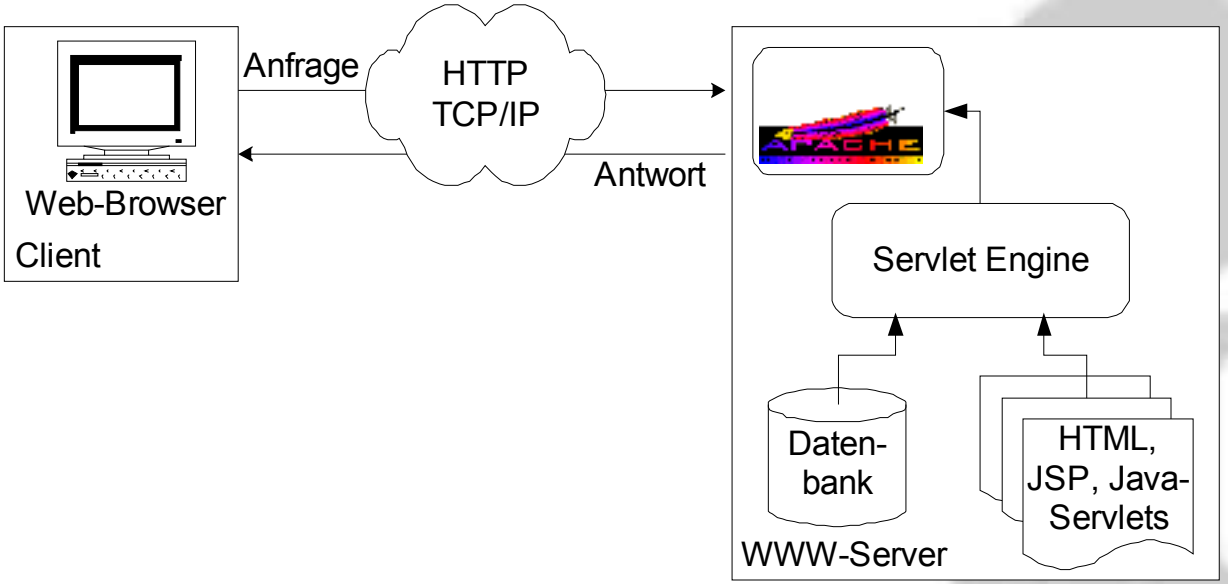
Get: Daten werden als Teil der URL übertragen (Längenbeschränkung!):
http://www.formular.com/form1?TerminBez=Squash&TerminAm=10.01.01

Dynamische Webinhalte

- Clientseitig
  - Java
  - Java-Script
  - ...



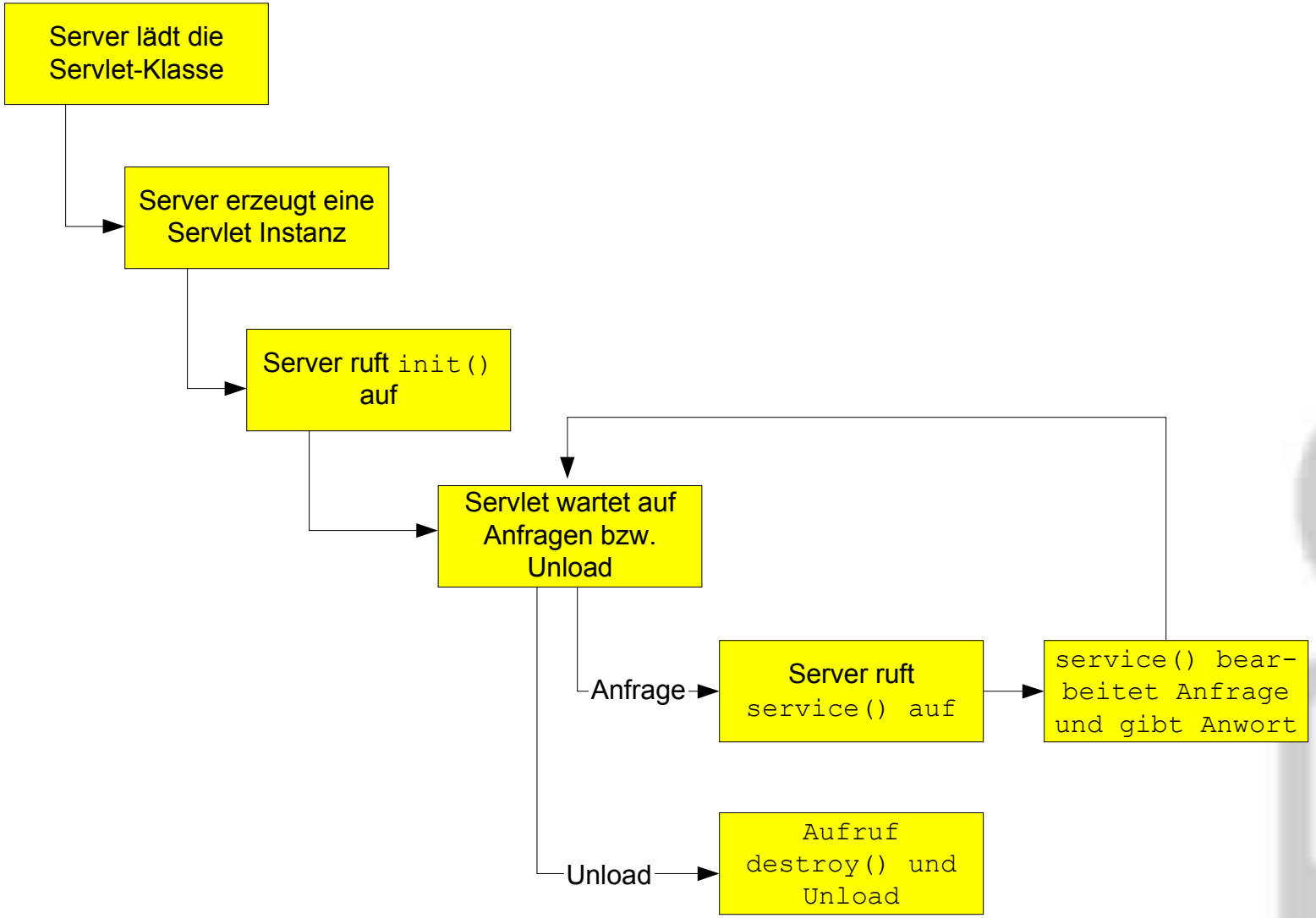
- Serverseitig
  - PHP
  - Perl
  - SSI
  - Java Server Pages
  - Java Servlets
  - ...



Java Servlets

- Idee: werden auf Server ausgeführt, nehmen Anfragen entgegen und beantworten diese
- Java Code
- Vorteile:
  - verbreitete Sprache, besser lesbar als Scriptsprachen
  - compilierte Ausführung
  - Ausführung im Server-Prozess, multithreaded

Lebenszyklus eines Servlets



Struktur eines Http-Servlets

```
import java.io.IOException;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class aServlet extends HttpServlet {

   public void init(){}

   public void service(HttpServletRequest request,
            HttpServletResponse response)
            throws IOException, ServletException {}

   public void destroy() { }
}
```

Importe

Wird beim Laden des Servlets ausgeführt

Wird für Anfragen (Get und Post) von Clients ausgeführt

Wird beim Entladen des Servlets ausgeführt

Get/Post

- in **HTTPServlets** ist **service** bereits vordefiniert, ruft je nach Anfrage z.B. eine der folgenden Methoden auf:

```
public void doGet(HttpServletRequest request,
                  HttpServletResponse response)
                  throws IOException,
                         ServletException
{}

public void doPost(HttpServletRequest request,
                   HttpServletResponse response)
                   throws IOException,
                          ServletException
{}
```

andere: **doPut**, **doDelete**, **doOptions**, **doTrace**, **getLastModified**

Beispiel: `doGet`-Methode

```
public void doGet(HttpServletRequest request,
                  HttpServletResponse response)
                  throws IOException, ServletException {

        response.setContentType("text/html");

        PrintWriter out = response.getWriter();

        out.println("<HTML>");
        out.println("<HEAD><TITLE>Sample Servlet</TITLE></HEAD>");
        out.println("<BODY>");
        out.println("<H1>Hello World!</H1>");
        out.println("<P>Datum: " + new java.util.Date() + "</p>");
        out.println("</BODY>");
        out.println("</HTML>");
}
```

Rückgabe ist HTML

Antwort in `out` schreiben

HTML ausgeben

Weiterleitung mit `RequestDispatcher`

```
public void service(HttpServletRequest request,
   HttpServletResponse response) throws IOException {
   ...
   ServletContext ctx=getServletContext();
   RequestDispatcher rdisp = ctx.getRequestDispatcher
                                        ("/Ziel.jsp");
   rdisp.forward(request, response);
}
```

Achtung: nie vorher **`getOutputStream()`** aufrufen!

Ziel der Umleitung, mit '/' relativ zur Applikation, ohne relativ zum aktuellen Pfad

Einige Klassen und Interfaces

- Pakete: **`javax.servlet`**, **`javax.servlet.http`**
- nur ausgewählte Methoden, siehe API-Dokumentation

`class HttpServlet`

`doGet, doPost, ...`	s. o.
`ServletContext getServletContext()`	Zugriff auf die Umgebung, in der das Servlet ausgeführt wird
`log(String msg [, Throwable t])`	erstellt Eintrag in Log-Datei

Einige Klassen und Interfaces

interface HttpServletRequest

`String getParameter(String name)`	liest mit dem Request übertragene Parameter aus
`String[] getParameterValues()`	gibt die Werte für einen Mengenwertigen Parameter zurück
`Enumeration getParameterNames()`	Liste aller Parameternamen ausgeben

Einige Klassen und Interfaces

interface HttpServletRequest (2)

`String getQueryString()`	Query String (URL ab '?')
`void setAttribute(String name, Object object)`	**`object`** im Request speichern
`Object getAttribute(String name)`	Attribut auslesen

Einige Klassen und Interfaces

interface HttpServletResponse

`ServletOutputStream getOutputStream()`	zum Schreiben von Binärdaten
`PrintWriter getWriter()`	zum Schreiben von ASCII-Daten
`void setContentType(String type)`	MIME-Typ der Ausgabe (text/html...)

Einige Klassen und Interfaces

interface ServletContext

`void setAttribute(String name, Object object)`	`object` im Kontext speichern
`Object getAttribute(String name)`	Attribut auslesen
`Enumeration getAttributeNames()`	alle Attributnamen
`void removeAttribute(String name)`	Attribut löschen
`RequestDispatcher getRequestDispatcher()`	liefert RequestDispatcher-Objekt

Java Server Pages (JSP)

- HTML mit "eingestricktem" Java Code
- Ausführung auf dem Server als generiertes Servlet

```
<HTML>
   <%! String getHello() {return "Hello";} %>
   <HEAD><TITLE>Simple JSP Page</TITLE></HEAD>
   <BODY>
         <H2>Request Origin</H2>
         Host Name:  <%= request.getRemoteHost() %> <BR/>
         IP Address: <%= request.getRemoteAddr() %> <BR/>
         <H2>Hello</H2>
         <% for (int i=1; i<=5; i++) { %>
               getHello returned: <%= getHello()%><BR/>
         <% } %>
   </BODY>
</HTML>
```

Scripting in JSP, Kommentare

Java Code

`<%! ... %>`	Deklarationen, ausserhalb der Methode **`service()`**
`<%= ... %>`	Ausdruck (liefert Ergebnis); ohne Semikolon!
`<% ... %>`	Scriptlet, Code der in der **`service()`** Methode ausgeführt wird

Kommentare

`<!-- ... -->`	normaler HTML-Kommentar, in generiertem HTML enthalten
`<%-- ... -->`	JSP-Kommentar, nicht in generiertem Servlet und HTML enthalten
`/* ... */` `//`	normale Java-Kommentare im Java-Code, // ist gefährlich (Zeilenumbrüche können verloren gehen)

Implizite Objekte

`application`	entspricht dem **`ServletContext`**
`pageContext`	Zugriff auf alle anderen impliziten Objekte
`request, response`	wie bei Servlets

Literatur

- http://www.teamone.de/selfhtml/
  - sehr gute HTML-Anleitung
- Dustin R. Callaway: Inside Servlets 2nd Edition, Server-Side Programming for the Java Platform. Addison Wesley 2001
  - behandelt HTTP, HTML, **Servlets**, **JSP**, JDBC
  - sehr empfehlenswert!
- Tomcat-Dokumentation: bei laufendem Tomcat unter http://localhost:8080/
  - Verwendung von Tomcat
  - Servlet-Entwicklung (Build, Deployment, ...)
  - Servlet-API-Dokumentation
- Java API-Dokumentation
  - ohne geht´s nicht