

Web-Suche

Benutzer-Schnittstelle

1

Websuche: Benutzer-Schnittstelle

- Web-Suchmaschinen brauchen natürlich eine web-basierte Benutzer-Schnittstelle.
- Die Suchseite muss einen Anfragestring entgegennehmen und diesen mittels eines HTML-`<form>`ulars übertragen.
- Das Programm auf dem Server muss Anfragen verarbeiten und eine HTML-Seite für die am höchsten gelisteten Dokumente (mit Links zu den ursprünglichen und/oder zwischengespeicherten Webseiten) erzeugen.

2

- HTML unterstützt verschiedene Arten der Programmeingabe in Formularen einschließlich:
 - Textbox
 - Menüs
 - Prüfbox
 - Auswahlbuttons
- Wenn ein Anwender ein Formular abschickt, werden Stringwerte für verschiedene *Parameter* zur Verarbeitung an den Server übertragen.
- Der Server nutzt diese Werte, um eine geeignete HTML-Antwortseite zu berechnen.

3

Ein einfaches Suchformular

```
<form action="http://titan.cs.utexas.edu:8082/servlet/irs.Search"
method="POST">
<p>
<b> Gebe Anfrage ein: </b> <input type="text" name="Anfrage" size=40>
<p>
<b>Suchdatenbank: </b>
<select name="Verzeichnis">
  <option selected value="/corpora/UniK/">
    Universität Kassel
  <option value="/corpora/KDE/">
    FG Wissensverarbeitung
</select>
<input type="hidden" name="Start" Wert="0">
<br>
<br>
<input type="submit" value="Anfrage ausführen">
<input type="reset" value="Formular zurücksetzen">
</form>
```

4

Web-Anwendungen

- Web-Anwendungen sind Anwendungen, die über das Web benutzt werden, d.h.
 - HTTP als Transportprotokoll
 - HTML als Benutzer-Schnittstelle.
- Verknüpfung von HTML und Businesslogik
 - dynamisch generierte HTML-Seiten
- Beispiele:
 - CGI-Skripte (Perl, Python, ...)
 - PHP
 - Content-Management-Systeme: Zope, Typo3, ...
 - Java Servlets

5

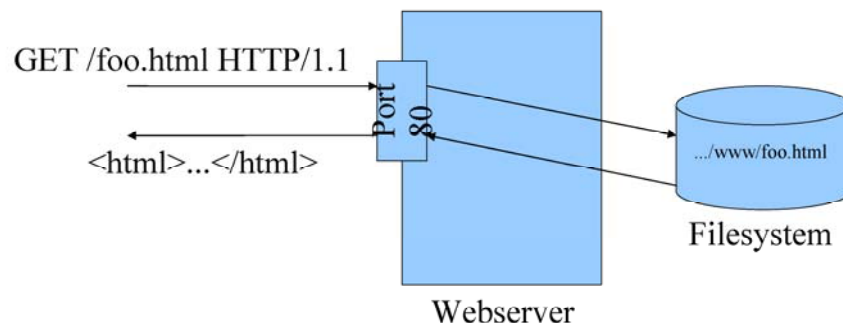
Web-Applikationsserver

- Webserver + Business-Logik
 - Reaktion auf Benutzereingaben
 - Datenbankzugriffe
- Mischen von HTML und Logik
 - Logik in HTML (vgl. PHP)
 - HTML wird von Programm erzeugt (vgl. CGI)
 - Mischformen
- beinhaltet oft weitere Dienste
 - Datenbankanbindung
 - Lastverteilung
 - Hochverfügbarkeit

7

Webserver

- Dienst zur Auslieferung von Webseiten über das HTTP-Protokoll.
- Einfachste Variante:
 - statisches HTML im Dateisystem



6

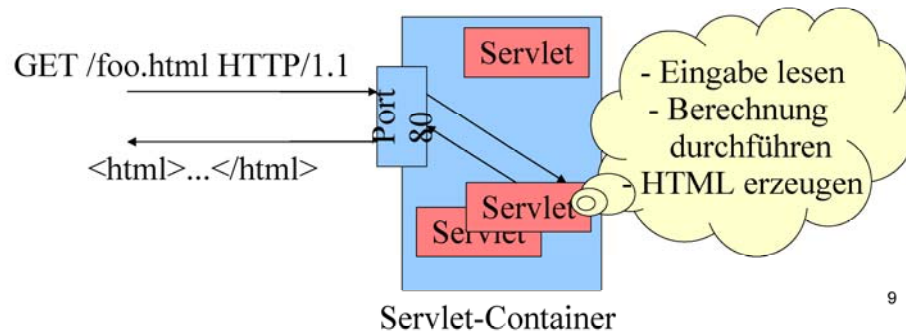
Was ist ein Servlet?

- Javas Antwort auf CGI-Programmierung zur Verarbeitung von Webformular-Anfragen.
- Das Programm läuft auf Webservern und erzeugt dynamische/angepasste Seiten.
- Wann verwendet man Servlets?
 - Seite basiert auf vom Benutzer eingegebenen Daten, z.B. Suchmaschinen.
 - Daten ändern sich oft, z.B. Wetterberichte.
 - Seite verwendet Informationen von einer Datenbank, z.B. Online-Speicher.
- Erfordert das Betreiben eines Webserver, der Servlets unterstützt.

8

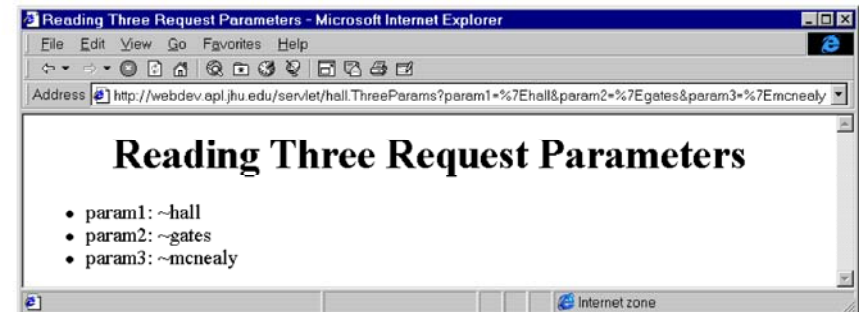
Servlet-Container

- Webserver, der Servlets beheimatet
 - übernimmt Netzwerkverkehr
 - bietet Infrastruktur
 - Konfiguration, Skalierbarkeit, ...



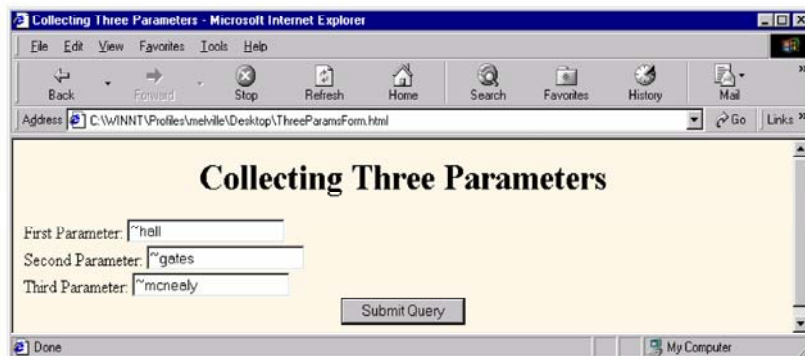
9

Servlet-Ausgabe



11

Formularbeispiele



10

HTML-Post-Form

```
<FORM ACTION="/servlet/hall.ThreeParams"
      METHOD="POST">
  First Parameter: <INPUT TYPE="TEXT"
NAME="param1"><BR>
  Second Parameter: <INPUT TYPE="TEXT"
NAME="param2"><BR>
  Third Parameter: <INPUT TYPE="TEXT"
NAME="param3"><BR>
  <CENTER>
    <INPUT TYPE="SUBMIT">
  </CENTER>
</FORM>
```

12

Reading Parameters

```
public class ThreeParams extends HttpServlet {
    public void doGet(HttpServletRequest request,
        HttpServletResponse response) throws ServletException,
        IOException {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println(... + "<UL>\n" +
            "<LI>param1: " + request.getParameter("param1") + "\n" +
            "<LI>param2: " + request.getParameter("param2") + "\n" +
            "<LI>param3: " + request.getParameter("param3") + "\n" +
            "</UL>\n" + ...);
    }
    public void doPost(HttpServletRequest request,
        HttpServletResponse response) throws ServletException,
        IOException {
        doGet(request, response);
    }
}
```

13

Grundsätzliche Servlet-Struktur

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

public class SomeServlet extends HttpServlet {
    // Handle get request
    public void doGet(HttpServletRequest request, HttpServletResponse
        response) throws ServletException, IOException {
        // request - access incoming HTTP headers and HTML form data
        // response - specify the HTTP response line and headers
        // (e.g. specifying the content type, setting cookies).
        PrintWriter out = response.getWriter(); //out - send content to
        browser
    }
}
```

14

Lesen aller Parameter

- Liste aller Parameternamen, die Werte haben:

```
Enumeration paramNames = request.getParameterNames();
```

- Parameternamen in unspezifizierter Reihenfolge.

- Parameter können viele Werte haben:

```
String[] paramVals =
    request.getParameterValues(paramName);
```

- Aufstellung von Parameterwerten, die mit *paramName* verbunden sind.

15

Einfaches Such-Servlet

- Erzeugt oder wählt einen vorhandenen InvertedIndex für den entsprechenden Korpus aus (z.B. **Verzeichnis/Korpus** Parameter).
- Verarbeitet die Anfrage (**query** Parameter) mit KSM, um eine gerankte Ergebnismenge zu ermitteln.
- Gibt in HTML eine geordnete Liste von 10 Ergebnissen aus, beginnend mit der Rangstelle des **start** Parameters.
- Jede Position enthält:
 - Ursprüngliche Basis-URL, die vom Spider am Anfang des Dokuments im BASE-Tag gespeichert wurde.
 - Seiten-<TITLE>, der aus der Datei extrahiert wurde.
 - Zusätzlicher Verweis zur lokal zwischengespeicherten Datei.
- Erstellt ein Formular, um **“Weitere Ergebnisse”** beginnend mit der nächsten gelisteten Position anfordern zu können, falls noch nicht alle Ergebnisse angezeigt wurden.

16

Schnittstellenverbesserungen für die einfache Suche

- Verarbeitet gegenwärtig die Anfragen erneut, auch wenn nach **“Weitere Ergebnisse”** gefragt wird.
 - Könnte die gegenwärtig gerankte Liste zusammen mit der Anwendersitzung speichern.
- Könnte weitere Interaktionen durch Relevance Feedback integrieren.
- Könnte Anfrage: **“Bestimme ähnliche Seiten”** für jedes gefundene Dokument (wie in Google) liefern.
 - Verwende einfach gegebenen Dokumententext als eine Anfrage.

17

Andere Verbesserungen der Suchschnittstelle

- Gruppensuchergebnisse in kohärenten **“Clustern”**:
 - **“Mikrowelle”**
 - Eine Gruppe über Ernährungsrezepte und Küchengeschirr.
 - Eine andere Gruppe über den Empfang von Satellitenfernsehen.
 - **“Huskies”**
 - Eine Gruppe über Hunde.
 - Eine Gruppe über das Kasseler Eishockey-Team.
- Northern Light gruppiert Ergebnisse in **“Ordern”**, die auf einer voretablierten Kategorisierung von Seiten basieren (wie Yahoo- oder DMOZ-Kategorien).
- Eine Alternative ist, Suchergebnisse dynamisch in Gruppen ähnlicher Dokumente zusammenzufassen.

19

Andere Verbesserungen der Suchschnittstelle

- Hebe Suchbedingungen in dem angezeigten Dokument hervor.
 - Wie bei gecachter Datei in Google.
- ermögliche **“erweiterte”** Suche:
 - Phrasensuche (“..”)
 - Verbindliche Terme (+)
 - Negierte Terme (-)
 - Sprachpräferenzen
 - Rückwärts-Link
 - Datenpräferenzen
- Maschinelle Übersetzung von Seiten.

18

Meta-Suchmaschinen

- Suchmaschine, die eine Anfrage an einige andere Suchmaschinen weitergibt und die Ergebnisse integriert.
 - Unterbreite Anfrage an verschiedene Hostsites.
 - Parse die resultierenden HTML-Seiten, um die Suchergebnisse zu extrahieren.
 - Integriere die unterschiedlichen Klassifizierungen in einer **“Konsens”**-Klassifikation.
 - Präsentiere dem Anwender das integrierte Ergebnis.
- Beispiele:
 - [Metacrawler](#)
 - [SavvySearch](#)
 - [Dogpile](#)

Anwenderverhalten

- Anwender neigen dazu, kurze Anfragen einzugeben.
 - Eine Studie in 1998 ergab eine Durchschnittslänge von 2.35 Worten.
- Anwender neigen dazu, keine erweiterten Suchoptionen zu verwenden.
- Anwender müssen bei der Verwendung komplexerer Anfragen eingewiesen werden.