Netzprogrammierung
12. Serverseitige Verarbeitung im Web

Prof. Dr.-Ing. Robert Tolksdorf
Freie Universität Berlin
Institut für Informatik
Netzbasierte Informationssysteme
mailto: tolk@inf.fu-berlin.de
http://www.robert-tolksdorf.de

Überblick

- CGI
- SSI
- JSP
- Servlets

Dynamische Webinhalte

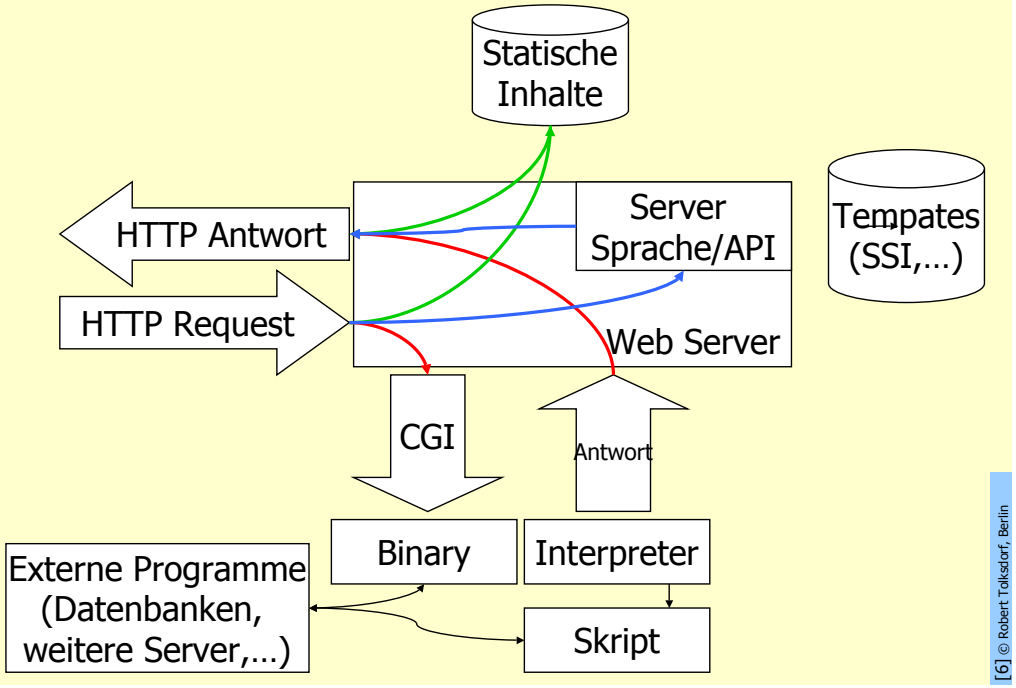
Dynamische Web Seiten

- Seiten können
  - *Statische Inhalte* liegen vorgefertigt auf dem Server und werden unverändert ausgeliefert und dargestellt
  - *Dynamische Inhalte* bewirken vor und bei der Auslieferung oder der Darstellung Effekte auf Inhalt oder Darstellung durch Ausführung von Programmen
- Aufgaben
  - Serverseitig
    - Vereinfachung der Inhaltserstellung
    - Erzeugung/Konvertierung von Inhalt
    - Anpassung/Personalisierung von Inhalt
    - Ergebnis anderer serverseitiger Transaktionen
  - Clientenseitig
    - Erzeugung einer Darstellung
    - Anpassung von Darstellung
    - GUI Interaktion mit Nutzer

Serverseitige Verarbeitung

- Templatesprachen
  - SSI, PHP,…
  - Ähnl. SSI Anweisungen in HTML eingestreut, mächtigere Sprache
- CGI/Skriptsprachen
  - /bin/sh, Perl, Phython,…
  - Imperativer Programmcode, oft mit Stärken in Stringverarbeitung
- CGI/Reguläre Programme
  - Binaries in beliebiger Programmiersprache
  - Geschwindigkeit
- Sprache/API in Server eingebunden
  - Servererweiterungen (NSAPI, ISAPI, Apache), Servlets/JSP, ASP,
  - Kein externer Interpreter sondern Teil des Servers
- Servlets
- SSI
- JSP

Im Überblick



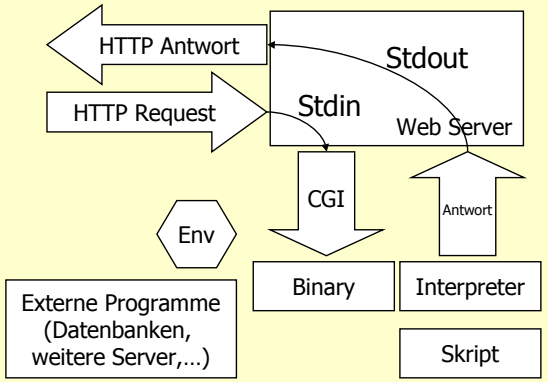
Common Gateway Interface CGI

CGI

- CGI ist eine Schnittstelle zwischen Web-Server und Programm
  - URL bezeichnet CGI-Programm
  - Server startet Programm als Prozess
  - Server übergibt Parameter der Anfrage und anderes in Umgebungsvariablen
  - Programm verarbeitet Anfrage
  - Kann Seiteneffekte haben, teilweise sehr große
  - Programm erzeugt eine HTTP-Antwort (mit HTML-Seite oder anderem)
  - Server leitet Antwort an Klienten weiter
- CGI-Programm ist etwas Ausführbares
  - Interpreter (der Skript interpretiert)
  - Compiliertes Programm

Ablauf

1. Web Server erkennt, dass URL ein Skript bezeichnet
2. Prozess wird gestartet
3. Prozess werden Eingaben mitgeteilt
   1. Initialisierte Umgebungsvariablen
   2. Standardeingabe
4. Standardausgabe des Prozesses wird zum Web-Server gelenkt
5. Skript wird ausgeführt
6. Ausgaben werden an Klienten weitergeleitet



Einfaches Beispiel

- Erzeugung einer einfachen Seite:

```
#!/usr/perl/bin/perl
$temperatur=&messe_temperatur;
print "Content-type: text/html\n";   # Header
print "\n";                          # Leerzeile
print "<html><head>\n";              # Inhalt...
print "<title>Temperatur in Berlin</title>";
print "</head>\n";
print "<body>Temperatur: $temperatur</body>\n";
print "</html>\n";
```


Umgebungsvariable

- Kommunikation Web-Server und CGI Programm über festen Satz von Umgebungsvariablen

```
#!/usr/perl/bin/perl
print "Content-type: text/html\n";   # Header
print "\n";                          # Leerzeile
print "<html><head>\n";              # Inhalt...
print "<title>Ich erkenne Sie!</title></head>\n";
print "<body>Sie benutzen einen ";
print $ENV{'HTTP_USER_AGENT'};
print "</body>\n</html>\n";
```

- $ENV ist in perl ein assoziatives Feld der Umgebungsvariablen

Umgebungsvariablen

- SERVER_SOFTWARE
  Server, der das Skript startet, als metaName/Version. Beispiel: NCSA/1.4.2.
- SERVER_NAME
  Der Name des Servers, wie er in der URL des Skripts auftritt.
- GATEWAY_INTERFACE
  Version der CGI-Spezifikation, der der Server folgt, als CGI/Version. Z.B.: CGI/1.1.
- SERVER_PROTOCOL
  Das Protokoll, mit dem die Anfrage geschickt wurde, als Zeichenkette der Form Protokoll/Version. Z.B. HTTP/1.1
- SERVER_PORT
  Die Nummer des (Unix-)Ports, über den die Anfrage geschickt wurde.
- REQUEST_METHOD
  Methode, die bei der Anfrage verwendet wurde also zumeist get oder post.
- PATH_INFO
  Enthält den URL-Teil nach der eigentlichen Identifikation des Skripts.
  - CGI-Skript www.info.berlin/cgi-bin/info
  - Aufruf über http://www.info.berlin/cgi-bin/info/tempelhof
  - In der Umgebungsvariablen: /tempelhof
- PATH_TRANSLATED
  PATH_INFO mit vorangestelltem lokalen Pfadnamen des Dokumentenverzeichnis: /usr/local/etc/httpd/htdocs/tempelhof

Umgebungsvariablen

- SCRIPT_NAME
  Der Name des Skripts in der URL, z.B. /cgi-bin/info.
- QUERY STRING
  Die Anfrage
- REMOTE_HOST
  Der Name des Rechners, von dem die Anfrage kam.
- REMOTE ADDR
  Die Internetadresse des Rechners, von dem die Anfrage kam.
- REMOTE_USER
- Der Benutzername, für den das Passwort eingegeben wurde.
- REMOTE_IDENT
  Nutzernamen des anfordernden Users, falls ermittelbar
- AUTH_TYPE
  Name des Verfahrens, mit dem das Passwort kodiert ist, z.B. Basic
- CONTENT_TYPE
  Bei Verwendung der POST-Methode steht hier der Medientyp des Inhalts, der an der Standardeingabe gelesen werden kann, z.B. application/x-www-form-encoded.
- CONTENT_LENGTH
  Die Länge des Inhalts bei der POST-Methode.
- HTTP_ACCEPT
  Die Medienarten, die der Browser akzeptieren will

Umgebungsvariablen

- Real je nach Server auch mehr Variablen
- Script

```
#!/usr/bin/perl
print "Content-Type: text/plain\n\n";
foreach $var (sort keys %ENV) {
  $val = $ENV{$var};
  $val =~ s|\n| |g;
  print "$var=\"$val\"\n";
}
```

- Ergibt für http://www.inf.fu-berlin.de/~tolk/cgi-bin/env.cgi:

```
DOCUMENT_ROOT="/export/local-1/www/page/htdocs"
GATEWAY_INTERFACE="CGI/1.1"
HTTP_ACCEPT="text/xml,application/xml,application/xhtml+xml,text/html
   ;q=0.9,text/plain;q=0.8,video/x-mng,image/png,image/jpeg,
   image/gif;q=0.2,*/*;q=0.1"
HTTP_ACCEPT_CHARSET="ISO-8859-1,utf-8;q=0.7,*;q=0.7"
HTTP_ACCEPT_ENCODING="gzip,deflate"
HTTP_ACCEPT_LANGUAGE="de,en-us;q=0.7,en;q=0.3"
```


Umgebungsvariablen

```
HTTP_CONNECTION="keep-alive"
HTTP_HOST="page.mi.fu-berlin.de"
HTTP_KEEP_ALIVE="300"
HTTP_USER_AGENT="Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.4)
   Gecko/20030624"
PATH="/usr/local/bin:/usr/bin:/bin"
QUERY_STRING=""
REMOTE_ADDR="160.45.114.204"
REMOTE_PORT="1559"
REQUEST_METHOD="GET"
REQUEST_URI="/~tolk/cgi-bin/env.cgi"
SCRIPT_FILENAME="/export/local-1/www/userpages/tolk/
                 public_html/cgi-bin/env.cgi"
SCRIPT_NAME="/~tolk/cgi-bin/env.cgi"
SERVER_ADDR="160.45.117.11"
SERVER_ADMIN="webmaster@inf.fu-berlin.de"
SERVER_NAME="page.mi.fu-berlin.de"
SERVER_PORT="80"
SERVER_PROTOCOL="HTTP/1.1"
SERVER_SOFTWARE="Apache/1.3.33 Ben-SSL/1.55 (Debian GNU/Linux)
   PHP/4.3.10-2 mod_perl/1.29"
UNIQUE_ID="QgeKzaAtdQsAABj1HC4"
```


Standardeingabe

- Auf der Standardeingabe steht der Inhalt der Anfrage-Mitteilung an
- GET/POST Unterscheidung, wo der Query-String erhältlich ist:

```
if ($ENV{'REQUEST_METHOD'} eq 'POST') {
  read(STDIN, $buffer, $ENV{'CONTENT_LENGTH'});
} elsif ($ENV{'REQUEST_METHOD'} eq 'GET') {
  $buffer=$ENV{'QUERY_STRING'};
}
```

Standardausgabe

- Ausgaben des CGI-Scripts werden als Antwort an den Klienten geschickt
- Notwendig: Content-Type-Header
- Möglich: Beliebige Medienarten als Antwort, z.B. auch Bilder

```
#!/usr/bin/perl
print("Content-Type: application/postscript\n\n");
print <<EO_POSTSCRIPT
%!PS-Adobe-3.0
%% see http://www.ntecs.de/old-hp/uu9r/lang/html/postscript.de.html
/Courier findfont
28 scalefont
setfont
0 0 moveto
(Hallo!) show
showpage
EO_POSTSCRIPT
```


Standardausgabe

- Setzen des Statuscodes:

```
#!/usr/bin/perl
print "Content-Type: text/plain\n";
print "Status: 405 Not Allowed\n\n";
```

- Ergebnis:

```
>head http://www.inf.fu-berlin.de/~tolk/cgi-bin/405.cgi
head http://www.inf.fu-berlin.de/~tolk/cgi-bin/405.cgi
405 Not Allowed
Connection: close
Date: Mon, 07 Feb 2005 16:22:26 GMT
Server: Apache/1.3.33 Ben-SSL/1.55 (Debian GNU/Linux) PHP/4.3.10-2
   mod_perl/1.29
Content-Type: text/plain; charset=iso-8859-1
```

- Umleitung

```
#!/usr/bin/perl
@uni=("http://www.fu-berlin.de",
      "http://www.tu-berlin.de",
      "http://www.hu-berlin.de");
print("Location: $uni[rand(3)]\n\n");
```


Java Servlets

Servlets

- Servlets:
  - CGI Programme könnten auch in Java geschrieben werden
  - Als „Interpreter" müsste eine JVM gestartet werden
  - Wenn der Webserver selber auf einer JVM läuft, könnte er eine „CGI-Komponente" nachladen und ausführen
  - Java Servlets sind solche Komponenten
- Unterschiede zu CGI und Applets
  - Parameterkommunikation läuft nicht über Umgebung und stdin/stdout sondern über Java-Schnittstelle
  - Applets als Komponente in JVM eines Browsers geladen
  - Servlets als Komponente in JVM eines Servers geladen
- Schnittstelle:
  - Erweitern von HttpServlet
  - Überschreiben von doGet und doPost

Beispiel

```
// HelloServlet.java
package buch;
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class HelloServlet extends HttpServlet {
  public void doGet( HttpServletRequest req, HttpServletResponse res )
                  throws ServletException, IOException {
    res.setContentType( "text/html" );
    PrintWriter out = res.getWriter();
    out.println("<HTML>");
    out.println("<HEAD><TITLE>HelloServlet</TITLE></HEAD>");
    out.println("<BODY>");
    out.println("<BIG>Hello</BIG>");
    out.println("</BODY>");
    out.println("</HTML>");
  }
}
```


Eingaben

- Über das HttpServletRequest Objekt kann man die auch in CGI vorhandenen Informationen erfragen:
  - getServerName()
  - getServerPort()
  - getRemoteAddr()
  - getRemoteHost()
  - getQueryString()
  - getMethod()
  - getServletPath()
- Header
  - getHeaderNames()
  - getHeader(String name)
- Query-String
  - getParameter(String name)
  - getParameterValues(String name)

Beispiel

```
// AnmeldenServlet.java
package buch;
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class AnmeldenServlet extends HttpServlet {
  public void doGet( HttpServletRequest req, HttpServletResponse res )
                  throws ServletException, IOException {
    res.setContentType( "text/html" );
    PrintWriter out = res.getWriter();
    String user = req.getParameter( "user" );
    String password = req.getParameter( "password" );
    out.println("<html><head><title>Ihre
     Anmeldedaten</title></head><body>");
    out.println("<h3>Ihre Anmeldedaten</h3><br><table>");
    out.println(" <tr><td>Benutzername:</td><td>" + user + "</td></tr>");
    out.println(" <tr><td>Passwort:</td><td>" + password + "</td></tr>");
    out.println("</table></body></html>");
  }
}
```


Ausgaben

- Statuscode und Header sind setzbar
  - setStatus(int no)
  - setHeader(String name, String value)

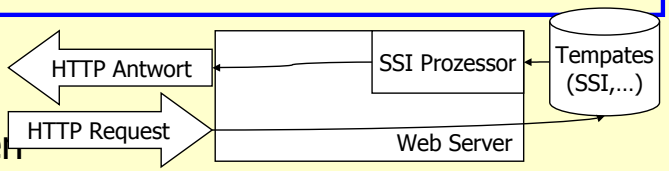
Vergleich CGI / Servlets

- Portabilität
  - CGI führt weitere Sprache ein, oft extra Bibliothek nötig
  - Servlets sind innerhalb des Java Rahmens und sind einfacher strukturiert
- Leistungspotential
  - CGI ist eine Teillösung für Server Skripte
  - Servlets haben Anschluss an die gesamte Java Welt, einschließlich Sicherheitsarchitektur
- Effizient
  - CGIs werden jedesmal als Prozess gestartet (Ausnahme: Fast CGI Mechanismus)
  - Servlets bleiben in JVM geladen

Server Side Includes

Server Side Includes

- Server Side Includes (SSI) sind Anweisungen an den Web Server, die im HTML Code eingebunden sind



- Werden vom Server bei Auslieferung ausgeführt und zu HTML Code expandiert
- SSI Anweisung im statischen HTML-Code:

```
<div align="right">
 Letzte &Auml;nderung: <!--#flastmod file=""-->.
</div>
```

- Ergebnis nach Auslieferung

```
<div align="right">
Letzte &Auml;nderung: Wednesday, 08-Jan-2003
09:30:15 CET.
</div>
```


SSI Anweisungen

- Als HTML-Kommentar eingebettet:
  `<!--#direktive Attribut1="Wert1" Attribut2="Wert2"…-->`
- Direktiven:
  - `flastmod`: Fügt Änderungsdatum der im Attribut `file` genannten Datei ein
  - `fsize`: Fügt Größe der im Attribut `file` genannten Datei ein
  - `include`: Fügt Datei ein. Bezeichnung je nach Attribut:
    - `virtual`: Pfad im Web-Verzeichnisbaum
      `<!--#include virtual="/inst/ag-nbi/include/nbiheader" -->`
    - `file`: Pfad im (nur „unter" aktueller Datei)
  - `exec`: Führt Kommando oder
    - `cgi`: Pfad im Web-Verzeichnisbaum (Apache 2.0: `include virtual`)
    - `cmd`: Ausführung im Datei-Verzeichnisbaum

```
<pre>
<!--#exec cmd="/usr/bin/ls -l"-->
</pre>
```

SSI

- `echo`: Gibt Inhalt der im Attribut `var` bezeichneten Variablen aus
- SSI-Variablen
  - `DOCUMENT_NAME`: Dateiname des Dokuments im Datei-Verzeichnisbaum
  - `DOCUMENT_URI`: Dateiname des Dokuments im Web-Verzeichnisbaum
  - `QUERY_STRING_UNESCAPED`: Suchanfrage
  - `DATE_LOCAL`: Aktuelles lokales Datum
  - `DATE_GMT`: Aktuelles lokales Datum als GMT
  - `LAST_MODIFIED`: Veränderungsdatum
- Beispiel:

```
Hier ist es <!--#echo var="DATE_LOCAL"-->.
```


SSI

- CGI-Variablen

`SERVER_SOFTWARE`	`SERVER_NAME`	
`GATEWAY_INTERFACE`		
`SERVER_PROTOCOL`	`SERVER_PORT`	`REQUEST_METHOD`
`PATH_INFO`	`PATH_TRANSLATED`	`SCRIPT_NAME`
`QUERY_STRING`	`REMOTE_HOST`	`REMOTE_ADDR`
`AUTH_TYPE`	`REMOTE_USER`	`REMOTE_IDENT`
`CONTENT_TYPE`	`CONTENT_LENGTH`	

- HTTP-Header der Form `HTTP_`*`Headername`*:
  - `HTTP_ACCEPT`, `HTTP_USER_AGENT`, ...
- Beispiel:

```
Wie geht es Ihnen bei
<!--#echo var="REMOTE_ADDR"--> mit Ihrem
<!--#echo var="HTTP_USER_AGENT"-->?
```


SSI

- set: Setzt den Inhalt der im Attribut `var` bezeichneten Variablen auf den im Attribut `value` angegebenen Wert
- Einfachste Ausdrücke bildbar

```
<!--#set var="salutation" value="Guten Tag"-->
<!--#set var="remote"
  value="${REMOTE_ADDR} (${REMOTE_HOST})"-->

<!--#echo var="salutation"--> at
<!--#echo var="remote"-->!
```


SSI

- Bedingte Abschnitte

```
<!--#if expr="Bedingung" -->
<!--#elif expr="Bedingung" -->
<!--#else -->
<!--#endif -->
```

- Bedingungen und Ausdrücke
  - String
  - $String_1$!= $String_2$
  - $String_1$ < $String_2$
  - $String_1$ <= $String_2$
  - $String_1$ > $String_2$
  - $String_1$ >= $String_2$
  - (Bedingung)
  - ! Bedingung
  - $Bedingung_1$ && $Bedingung_2$
  - $Bedingung_1$ || $Bedingung_2$

Java Server Pages

Java Server Pages

- Java Server Pages
  - In SSI gibt es rudimentäre Ausdrücke
  - Man könnte auch komplexere Programmfragmente mit HTML-Code mischen
  - Java Server Pages erlauben die Mischung von HTML-Code mit Java Fragmenten
  - Aus ihnen werden automatisch Servlets generiert und ausgeführt
    - HTML-Code nach Ausgabe schreiben
    - Java-Fragmente in Servlet Rahmen einbetten
- JSP bestehen aus
  - Scriptlets
  - JSP Ausdrücken
  - Deklarationen
  - JSP Anweisungen
  - HTML

Beispiel für Scriptlets und JSP Ausdrücken

```
<html>
<head><title>Anmeldung</title></head>
<body>
<%
String us = request.getParameter( "user" );
String pw = request.getParameter( "password" );

if( us == null || "".equals( us )
 || pw == null || "".equals( pw ) ) {
%>
<h3>Anmeldung</h3>
<br>
<form method="get">
 <table border="0" cellspacing=0 cellpadding=0>
  <tr>
   <td><b>Benutzername:</b>  </td>
   <td><input type="text" name="user" size=12
            maxlength=50 value="<%out.print( us != null ? us : "" );%>"></td>
  </tr>
```


Beispiel für Scriptlets und JSP Ausdrücken

```
  <tr>
   <td><b>Passwort:</b>  </td>
   <td><input type="password" name="password" size=12
            maxlength=50 value="<%out.print( pw != null ? pw : "" );%>"></td>
  </tr>
  <tr>
   <td> </td>
   <td><input type="submit" value="Anmelden"></td>
  </tr>
 </table>
</form>
<br>
<%
  }
  else {
%>
```

Beispiel für Scriptlets und JSP Ausdrücken

```
<h3>Ihre Anmeldedaten</h3>
<br>
<table>
 <tr><td>Benutzername:</td><td> <%=request.getParameter( "user" )%>
    </td></tr>
 <tr><td>Passwort:</td><td> <%=request.getParameter( "password" )%>
    </td></tr>
</table>
<%
  }
%>
</body>
</html>
```


Beispiel Deklarationen

```
<html>
<head><title>Anmeldung</title></head>
<body>
<%!
  private String[] obligatList = { "user", "password" };
  private boolean validate( HttpServletRequest request ) {
    for( int i = 0; i < obligatList.length; i++ ) {
      String param = request.getParameter( obligatList[i] );
      if( param == null || "".equals( param ) )
        return false;
    }
    return true;
  }
%>
<%
boolean valid = validate( request );
if( !valid ) {
%>
<h3>Anmeldung</h3>
<br><form method="get"> <table border="0" cellspacing=0 cellpadding=0>...
```


Beispiel JSP Anweisungen

```
<html>
<head><title>Include.jsp</title></head>
<body>
<h2>Include.jsp</h2>
<p>Hier steht der spezifische Seiteninhalt.
<br><br><br><br><br><br><br><br><br><br>
<%@ include file="foot.jsp" %>
</body>
</html>
```


Scriptlets und JSP Ausdrücke

- Scriptlets
  - Syntax <% Java code %>
  - Beliebiger Java Code, wird in generierte Methode _jspService() entsprechend eincompiliert
- JSP Ausdrücke
  - Syntax <%= Java Ausdruck %>
  - Wird zur Laufzeit evaluiert
  - Ergebnis wird als Zeichenkette in die Ausgabe geschrieben
- Deklarationen
  - Syntax <%! Java Code %>
  - Wird in Klasse eincompiliert
- JSP Anweisungen
  - Werden bei Übersetzungszeit interpretiert

Vergleich SSI / JSP

- Ausführungsort
  - SSI vom Server interpretiert
  - JSP zu Servlet compiliert und in JVM ausgeführt
- Ausdrücksfähigkeit
  - SSI sehr rudimentär
  - JSP volle Java-Mächtigkeit
- Erstellung
  - Beide sind eigentlich HTML Erweiterungen
  - JSP ist aber eigentlich Programmierung

Zusammenfassung

Zusammenfassung

- CGI
  - Verarbeitung von Eingabe
  - Beliebige Sprache
  - Kommunikation vom Server-Prozess mit Umgebungsvariablen, stdin/ stdout
- Servlets
  - Rahmenwerk für „CGI-Komponenten" in Java
  - Kommunikation über Schnittstellen
- SSI
  - Vom Web-Server evaluierte zusätzliche Tags
  - „Makros"
- JSP
  - Mischung von HTML- und Java-Code
  - Compiliert zu Servlets

Referenzen

- CGI Specification. http://hoohoo.ncsa.uiuc.edu/cgi/
- NCSA HTTPd Development Team. *Server Side Includes (SSI)*. http://hoohoo.ncsa.uiuc.edu/docs/tutorials/includes.html
- Apache HTTP Server Documentation Project. *Apache Tutorial: Introduction to Server Side Includes*. http://httpd.apache.org/docs/howto/ssi.html http://httpd.apache.org/docs-2.0/howto/ssi.html
- Sun Microsystems, Inc. *JavaServer Pages.* http://java.sun.com/products/jsp/
- Heiko Wöhr. Web-Technologien, Konzepte - Programmiermodelle – Architekturen. dpunkt.verlag 2004.