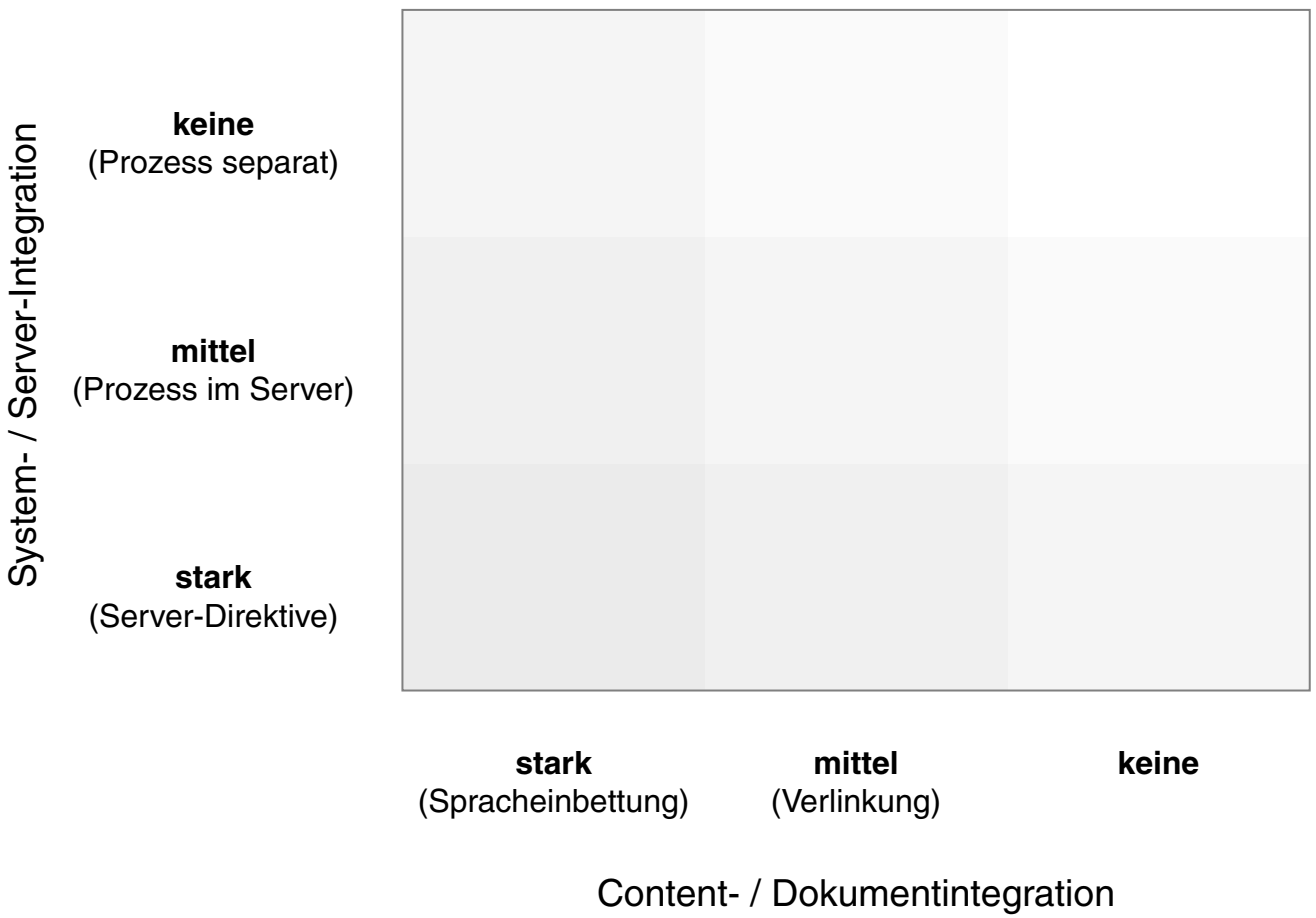
Kapitel WT:IV

Web-Server

Einordnung von Server-Technologien [Stein 2012]

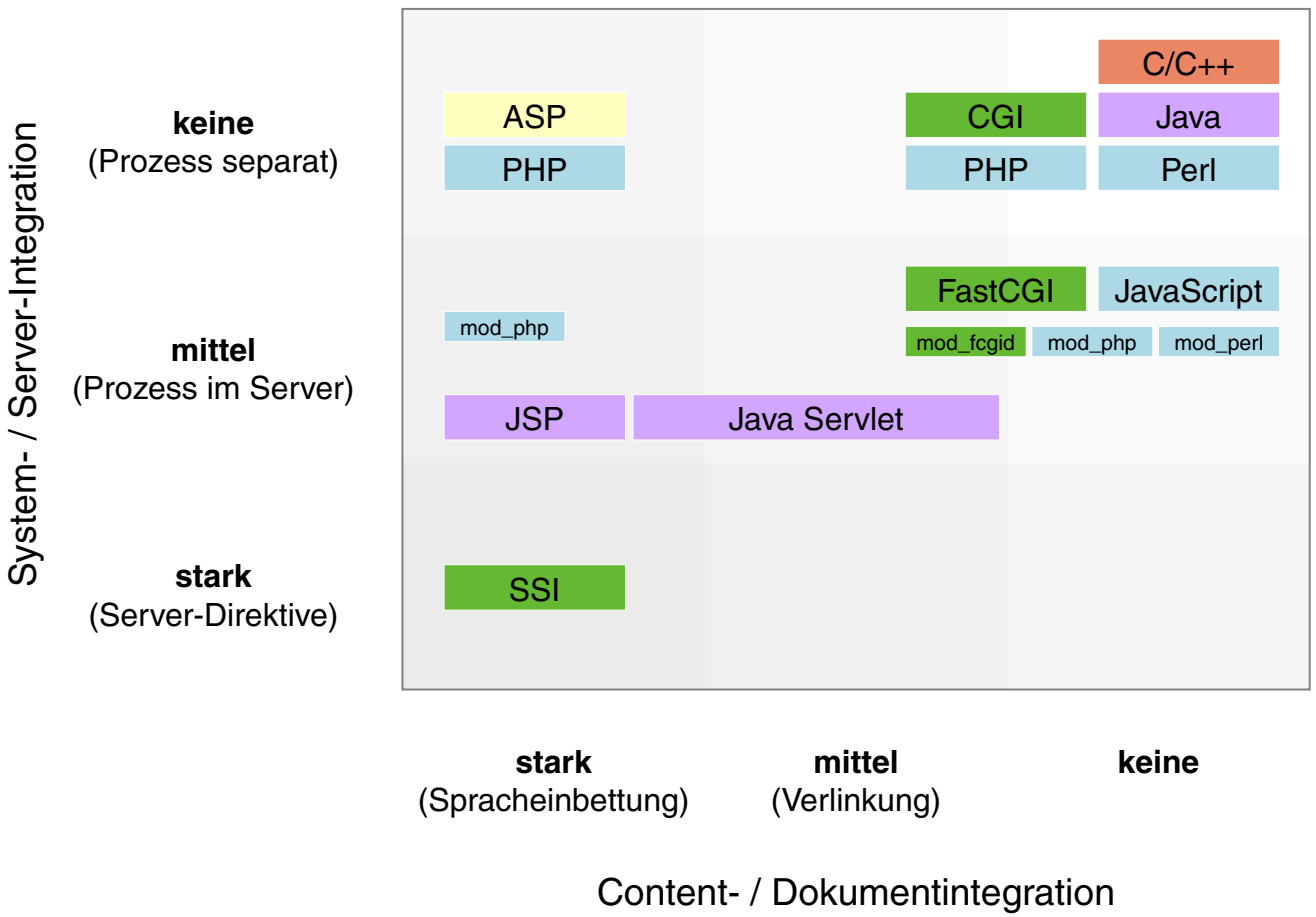
- x-Achse: Wie eng ist die Technologie mit dem auszuliefernden Content verwoben?
- y-Achse: Wie tief ist die Technologie in den Web-Server integriert?



Web-Server

Einordnung von Server-Technologien [Stein 2012]

- x-Achse: Wie eng ist die Technologie mit dem auszuliefernden Content verwoben?
- y-Achse: Wie tief ist die Technologie in den Web-Server integriert?



Bemerkungen:

- Ein Web-Server ist ein Server-Dienst, der Daten zur Verfügung stellt, auf die über das HTTP-Protokoll zugegriffen werden kann. Die Daten werden mit HTTP-URLs adressiert. [Wikipedia]
- Ein Web-Server wird auch als HTTP-Dämon bezeichnet; oft heißt das Programm, das den Web-Server implementiert, `httpd`.
- Separate Prozesse werden mittels CGI angebunden und direkt durch das Betriebsystem ausgeführt. Vorteil: einfache Anbindung des „Containers Betriebssystem“ in Form einer Shell. Nachteil (u.a.): zeitaufwändiger Start eines Prozesses.
- Prozesse *im* Web-Server sind durch Erweiterungsmodule (mod_xxx), zusätzliche Bibliotheken oder einen integrierten Container realisiert. Siehe Beispiele [Einordnung] :
  - FastCGI, mod_fcgid [apache.org, fastcgi.com]
  - mod_php [php.net]
  - mod_perl [apache.org]
  - JavaScript [nodejs.org]
  - JSP, Java Servlet [apache.org]

Web-Server

Wichtige Konfigurationseinstellungen

IP-Adresse, Hostname	Bei lokalem Betrieb die IP-Adresse 127.0.0.1 bzw. `localhost`.
Port	Üblicherweise lauscht der HTTP-Dämon auf Port 80 *(well-known Port)*; andere Einstellungen sind möglich.
ServerRoot	Verzeichnis im Dateisystem des Server-Rechners für Konfigurations-, Fehler-, und Log-Dateien der Server-Software.
DocumentRoot *(Web-Space)*	Verzeichnis im Dateisystem des Server-Rechners, in dem sich die auszuliefernden HTML-Dateien befinden.
Default-HTML-Dateiname	Spezifiziert die URL nur ein Verzeichns, wird nach einer Default-Datei gesucht. Üblich sind `index.html` oder `index.htm`.
CGI-Skripte	Angabe des physischen Verzeichnis und eines virtuellen Verzeichnisses (meist `/cgi-bin`) für CGI-Skripte. Aufruf mit `http://`*`Servername`*`/cgi-bin/`*`Skriptname`*.
Log-Dateien	Protokollierung der Zugriffe (`access.log`), Fehler (`error.log`)
Timeouts	Spezifiziert, wie lange der Web-Browser auf eine Antwort vom Server warten soll, und wie lange der Server versuchen soll, Daten an den Web-Browser zu schicken.
MIME-Typen	Dateiformate, die der Web-Server kennt.

Web-Server

Apache HTTP-Server: Historie

1995 Version 0.6.2. Sammlung von Bugfixes und Patches für den NCSA Web-Server (National Center for Supercomputing Applications) an der Universität von Illinois.

1998 Version 1.3. Grundstein für Apaches Erfolg. [apacheweek]

1999 Gründung der Apache Software Foundation. [apacheweek]

2002 Version 2.0. Modularisierung der Web-Server-Software. [apacheweek]

2005 Version 2.2. Features u.a.: Unterstützung von Dateien > 2 GB, überarbeitete Authentifizierung, verbessertes Caching.

2013 Aktuelle Version des Apache HTTP-Servers: [apache.org]

Bemerkungen:

- Statistiken zur Verbreitung von Web-Servern: [netcraft.com]
- Verfahren zum Zählen von aktiven Web-Servern: [netcraft.com]
- Dokumentation des Apache HTTP-Servers: [apache.org]
- Glossar mit Fachbegriffen im Zusammenhang mit dem Apache HTTP-Server und Web-Server im Allgemeinen: [apache.org]

Web-Server

Apache HTTP-Server: httpd.conf

Die zentrale Konfigurationsdatei httpd.conf bzw. apache2.conf liegt im Verzeichnis `/etc/httpd/` bzw. `/etc/apache2/`. Funktionsabschnitte:

1. Global Environment. Randbedingungen zur Arbeitsweise.
2. Main Server Configuration. Anweisungen zur Arbeitsweise.
3. Virtual Hosts. Einrichtung virtueller Hosts.

Konfigurationsanweisungen *(Directives)* sind in sogenannten Containern gruppiert. Die Syntax ist XML-ähnlich, hat aber nichts damit zu tun.

Container	**Beschreibung**
`<IfDefine>, <IfModule>`	werden bei Server-Start ausgewertet
`<Directory>, <DirectoryMatch>, <Files>, <FilesMatch>, <Location>, <LocationMatch>, <Proxy>, <ProxyMatch>,`	Alle anderen Container werden bei Anfragen ausgewertet. Sie enthalten Anweisungen, mit denen Verzeichnisse angesprochen und das Verhalten bei Zugriffen auf den Web-Space definiert wird.

Web-Server

Apache HTTP-Server: .htaccess

`.htaccess`-Dateien dienen zur Spezifikation von Zugriffen im Web-Space (Verzeichnisbaum unterhalb DocumentRoot) von Web-Servern. [apache.org]

Web-Server

Apache HTTP-Server: .htaccess

`.htaccess`-Dateien dienen zur Spezifikation von Zugriffen im Web-Space (Verzeichnisbaum unterhalb DocumentRoot) von Web-Servern. [apache.org]

Beispiel:

`.htaccess`-Datei:

```
# Kommentar
AuthType Basic
AuthName "Service-Bereich"
AuthUserFile /usr/verwaltung/web/.htusers
AuthGroupFile /usr/verwaltung/web/.htgroups
require user Werner Dieter Heidi
require group Servicetechniker
```

`.htusers`-Datei:

```
Werner:INY8m5KMwIc
Dieter:69gY8YPjQXeN6
Heidi:INw2mPEH.owe2
Anke:INh6DHvyejvf2
Bernd:INboWuvjjwQ7E
Karin:INwOXOz96UQOU
```

Direktive	Beschreibung
`AuthType`	Art der Authentifizierung, üblich ist `Basic`: Benutzer mit Passworten sind in einer anzugebenden Datei.
`AuthUserFile`	absoluter Pfad zur Datei von autorisierten Benutzern mit Passwort
`require {user|group}`	Liste der autorisierten Benutzer bzw. Gruppen

Bemerkungen:

- Die `.htaccess`-Dateien werden von Web-Servern ausgewertet, die zum NCSA-Server kompatibel sind.
- Das `.htaccess`-Konzept kann von dem Anwender, der Inhalte in dem Web-Space eines Web-Servers pflegt, eingesetzt werden. Aus Performanzgründen sollte grundsätzlich auf den Einsatz von `.htaccess`-Dateien verzichtet werden, falls auch die Möglichkeit besteht, Vorgaben in der `httpd.conf`-Datei machen zu können. [apache.org]
- Welche der globalen Vorgaben ein Anwender in der `.htaccess`-Datei überschreiben darf, wird mit der `allowoverride`-Direktive festgelegt. [apache.org]
- Standardmäßig gelten die Angaben einer `.htaccess`-Datei für das Verzeichnis, in dem die Datei gespeichert ist, einschließlich aller Unterverzeichnisse.
- Es ist sinnvoll, die sensiblen Dateien mit der Passwortinformation außerhalb des Web-Space des Web-Servers zu speichern.
- `.htaccess` ermöglicht viele Direktiven zur Zugriffsspezifikation:
  1. Optionen zum Verzeichnis-Browsing
  2. Optionen zum automatischen Weiterleiten
  3. Formulierung eigener Regeln zur Reaktion auf HTTP-Fehlermeldungen
  4. bedingte Auslieferung von Inhalten; z.B. können Web-Seiten abhängig von der Landessprache des benutzten Web-Browsers geliefert werden
  5. Optionen zur Komprimierung von Daten vor deren Übertragung zum Browser

Web-Server

Server Side Includes SSI [Einordnung] [SELFHTML] [apache.org]

Server Side Includes, SSI, sind die einfachste Möglichkeit, um HTML-Dokumente Server-seitig dynamisch zu verändern.

- SSI-Anweisungen sind Teil der HTML-Datei, maskiert als Kommentar:

  `<!--#`*Anweisung* *Parameter = "Wert"* `-->`

- HTML-Dateien, die SSI-Anweisungen enthalten, sind mit einer speziellen Dateiendung gekennzeichnet: `.shtml, .shtm, .sht`

Web-Server

Server Side Includes SSI [Einordnung] [SELFHTML] [apache.org]

Server Side Includes, SSI, sind die einfachste Möglichkeit, um HTML-Dokumente Server-seitig dynamisch zu verändern.

- SSI-Anweisungen sind Teil der HTML-Datei, maskiert als Kommentar:

  `<!--#`*Anweisung Parameter = "Wert"* `-->`

- HTML-Dateien, die SSI-Anweisungen enthalten, sind mit einer speziellen Dateiendung gekennzeichnet: `.shtml, .shtm, .sht`

Beispiel:

```
<h1>Dynamisches HTML mit Server Side Includes</h1>
Datum/Uhrzeit auf dem Server: <!--#config timefmt="%d.%m.%Y, %H.%M" -->
<!--#echo var="DATE_LOCAL" --> Uhr<br>
Name dieser HTML-Datei: <!--#echo var="DOCUMENT_NAME" --><br>
Installierte Server-Software: <!--#echo var="SERVER_SOFTWARE" --><br>
Ihr Web-Browser: <!--#echo var="HTTP_USER_AGENT" -->
<h3>Weitere Informationen:</h3>
<!--#exec cgi="/cgi-bin/perl-sample1.pl" -->
```

Web-Server

Server Side Includes SSI [Einordnung] [SELFHTML] [apache.org]

Server Side Includes, SSI, sind die einfachste Möglichkeit, um HTML-Dokumente Server-seitig dynamisch zu verändern.

- SSI-Anweisungen sind Teil der HTML-Datei, maskiert als Kommentar:

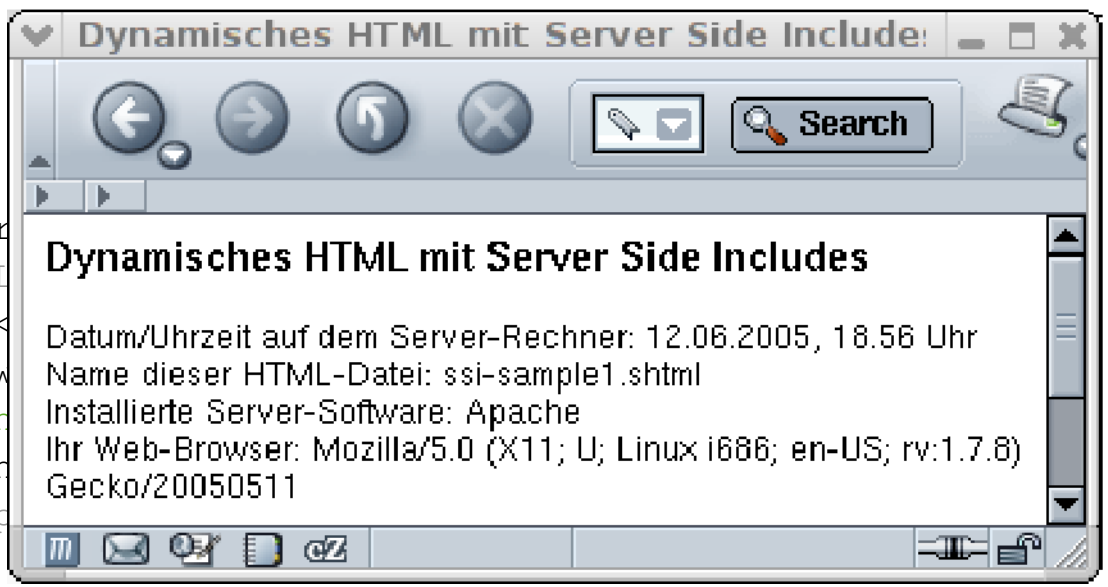
<!--#*Anweisung* *Parameter = "Wert"* -->

- HTML-Dateien, die SSI-Anweisungen enthalten, sind mit einer speziellen Dateiendung gekennzeichnet: `.shtml, .shtm, .sht`

Beispiel:

```
<h1>Dynamisches HTML mit
Datum/Uhrzeit auf dem Ser
<!--#echo var="DATE_LOCAL
Name dieser HTML-Datei: <
Installierte Server-Softw
Ihr Web-Browser: <!--#ech
<h3>Weitere Informationen
<!--#exec cgi="/cgi-bin/p
```

[Demo]



Web-Server

Server Side Includes SSI (Fortsetzung)

Anweisung	Parameter
`#config`	`errmsg="`*String*`", sizefmt="`*Formatstring*`", timefmt="`*Formatstring*`"`
`#echo`	`var="`*Name*`"` Es sind eigene, CGI-Umgebungsvariablen sowie folgende Variablen erlaubt: `DOCUMENT_NAME`: Name der HTML-Datei `DOCUMENT_URI`: Pfad der HTML-Datei `LAST_MODIFIED`: Zeitstempel der HTML-Datei `QUERY_STRING_UNESCAPED`: unmaskierter GET-Übergabestring `DATE_LOCAL`: Datum und Uhrzeit nach Server `DATE_GMT`: Datum und Uhrzeit nach Greenwich-Zeit
`#exec`	`cmd="`*Pfad/Programmdatei*`"` `cgi="`*CGI-Pfad/CGI-Skript*`"`
`#fsize`	`file="`*Pfad/Datei*`"` `virtual="`*Pfad/Datei*`"`
`#flastmod`	`file="`*Pfad/Datei*`"` `virtual="`*Pfad/Datei*`"`
`#include`	`file="`*Pfad/Datei*`"` `virtual="`*Pfad/Datei*`"`

Common Gateway Interface CGI

Prinzip: Ein Progamm außerhalb des Web-Servers stellt einen Zugang *(Gateway)* zu geschützten, für den Web-Server nicht erreichbaren Daten bereit. [SELFHTML]

Der hierfür standardisierte Kommunikationsmechanismus heißt CGI. [Einordnung]

Common Gateway Interface CGI

Prinzip: Ein Progamm außerhalb des Web-Servers stellt einen Zugang *(Gateway)* zu geschützten, für den Web-Server nicht erreichbaren Daten bereit. [SELFHTML]

Der hierfür standardisierte Kommunikationsmechanismus heißt CGI. [Einordnung]

Aufruf eines CGI-Skripts aus HTML-Datei mit Übergabe von Anwenderdaten:

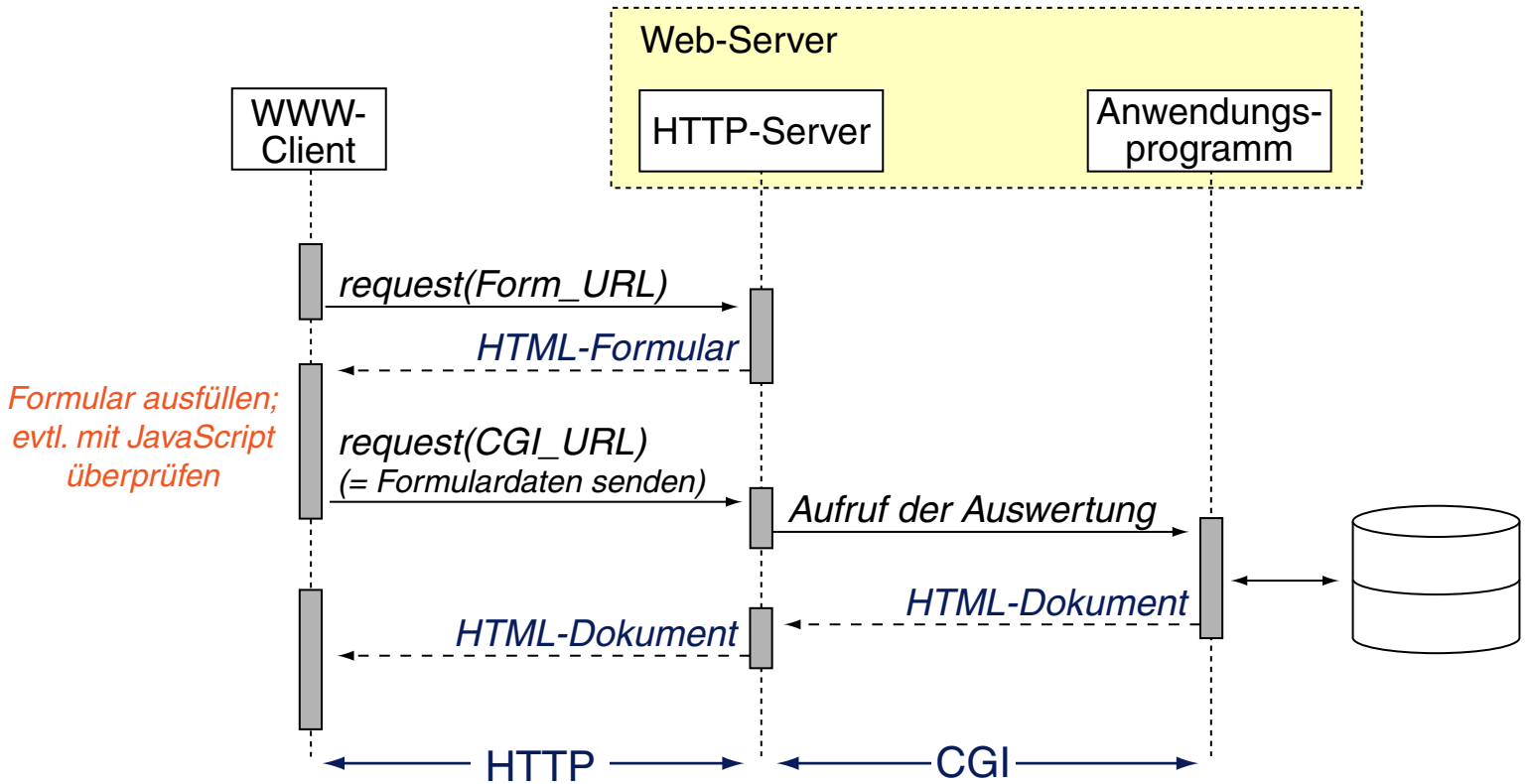
- Über ein Formular.
  ```
  <form action="/cgi-bin/sample.pl" method="get">
  ```

Typische Aufrufe aus HTML-Datei ohne Übergabe von Anwenderdaten:

- Über einen Verweis.
  ```
  <a href="/cgi-bin/statistik.pl">Statistik</a>
  ```
- Über eine Grafikreferenz.
  ```
  <img src="/cgi-bin/counter.pl">
  ```
- Über eine Server Side Include Anweisung.
  ```
  <!-#exec cgi="/cgi-bin/counter.pl" ->
  ```
- Über ein automatisches Laden / Weiterleiten.
  ```
  <meta http-equiv="REFRESH" content="0; URL=/cgi-bin/welcome.pl">
  ```

Common Gateway Interface CGI

Ablauf einer CGI-Interaktion



Vergleiche hierzu den Ablauf einer Servlet-Interaktion.

Bemerkungen:

- Anwendung von CGI: komplexe Berechnungen oder Datenverarbeitungsaufgaben, Anfragen und Updates bei Datenbanken.
- Jedes vom Betriebssystem (in einer Shell bzw. von einer Kommandozeile) ausführbare Programm kann als CGI-Programm dienen.
- Schritte bei der Kommunikation via CGI [Meinel/Sack 2004]:
  1. Der HTTP-Server erhält vom Client die Anforderung einer Informationsressource, die über ein (CGI-) Skript bzw. Programm bereitgestellt wird.
  2. Der HTTP-Server setzt auf Basis der Anforderung eine Reihe von Umgebungsvariablen.
  3. Der HTTP-Server startet das CGI-Skript bzw. das CGI-Programm.
     Geschieht der Aufruf via POST, so werden die Daten aus dem HTTP-Message-Body dem CGI-Skript über die Standardeingabe *(stdin)* zur Verfügung gestellt. [SELFHTML]
  4. Der HTTP-Server erhält die bei Ausführung des CGI-Skripts auf die Standardausgabe *(stdout)* geschriebenen Daten als Rückgabewert.
  5. Der HTTP-Server liefert die erhaltenen Daten an den Client aus.
- Die Ausgabe eines CGI-Skriptes enthält Entity- und Response-Header-Zeilen sowie den Body gemäß des HTTP-Protokolls. Die erste Zeile des Protokolls, die Status-Line, wird nicht vom CGI-Skript, sondern von dem Web-Server generiert; das CGI-Skript kann Angaben für den Status-Code generieren.

Common Gateway Interface CGI

Wichtige CGI-Umgebungsvariablen [SELFHTML]

Variable	Beschreibung
CONTENT_LENGTH	bei Aufruf via POST: Anzahl der übergebenen Zeichen
CONTENT_TYPE	bei Aufruf via POST: MIME-Typ der übergebenen Daten
DOCUMENT_ROOT	Pfad zu dem Web-Space des Web-Servers.
HTTP_ACCEPT	akzeptierte MIME-Typen des aufrufenden Browsers
HTTP_ACCEPT_LANGUAGE	Landessprache des aufrufenden Browsers
HTTP_CONNECTION	Informationen über den Status der HTTP-Verbindung
HTTP_COOKIE	Namen und Wert von Cookies, sofern vom Browser gesendet
HTTP_HOST	Domain-Namen bzw. IP-Adresse aus Adresszeile des Browsers
HTTP_USER_AGENT	Produkt- und Versionsinformationen zum Browser
QUERY_STRING	an die URL angehängte Daten, beginnend nach erstem „?“
REQUEST_METHOD	HTTP-Anfragemethode
REQUEST_URI	HTTP-Pfad des CGI-Skripts inklusive übergebenen Daten

Bemerkungen zur Codierung von Formulardaten:

- URL und Query sind durch ein „?“ voneinander getrennt.
- Formularelemente einschließlich ihrer Daten sind durch „&“ voneinander getrennt.
- Name und Daten eines Formularelements sind durch „=“ voneinander getrennt.
- Leerzeichen in den eingegebenen Daten sind durch ein „+“ ersetzt.
- Zeichen mit ASCII-Werten zwischen 128 bis 255 sind durch eine hexadezimal codiert, eingeleitet durch ein Prozentzeichen. Beispiel: „%F6“ für „ö“

Common Gateway Interface CGI

Beispiel: csh-Skript als CGI-Programm

In der HTML-Datei:

```
<a href="http://localhost/cgi-bin/cgi-sample1.cgi">CGI-Aufruf</a>
```

Das Shell-Script cgi-sample1.cgi :

```
#!/bin/csh

echo "content-type: text/html"
echo "<html>"

echo "<head>"
echo "<meta http-equiv="\""content-type"\" "content="\"text/html; ...
echo "<title>CGI-Sample</title>"
echo "</head>"

echo "<body>"
echo "<h3>Werte einiger CGI-Variablen</h3>"
echo "Installierte Server-Software: " $SERVER_SOFTWARE "<br>"
echo "Aufrufender Web-Browser: " $HTTP_USER_AGENT "<br>"
echo "Anfragemethode: " $REQUEST_METHOD "<br>"
echo "Query-String: " $QUERY_STRING "<br>"
echo "</body>"
echo "</html>"
```

Common Gateway Interface CGI

Beispiel: csh-Skript als CGI-Programm

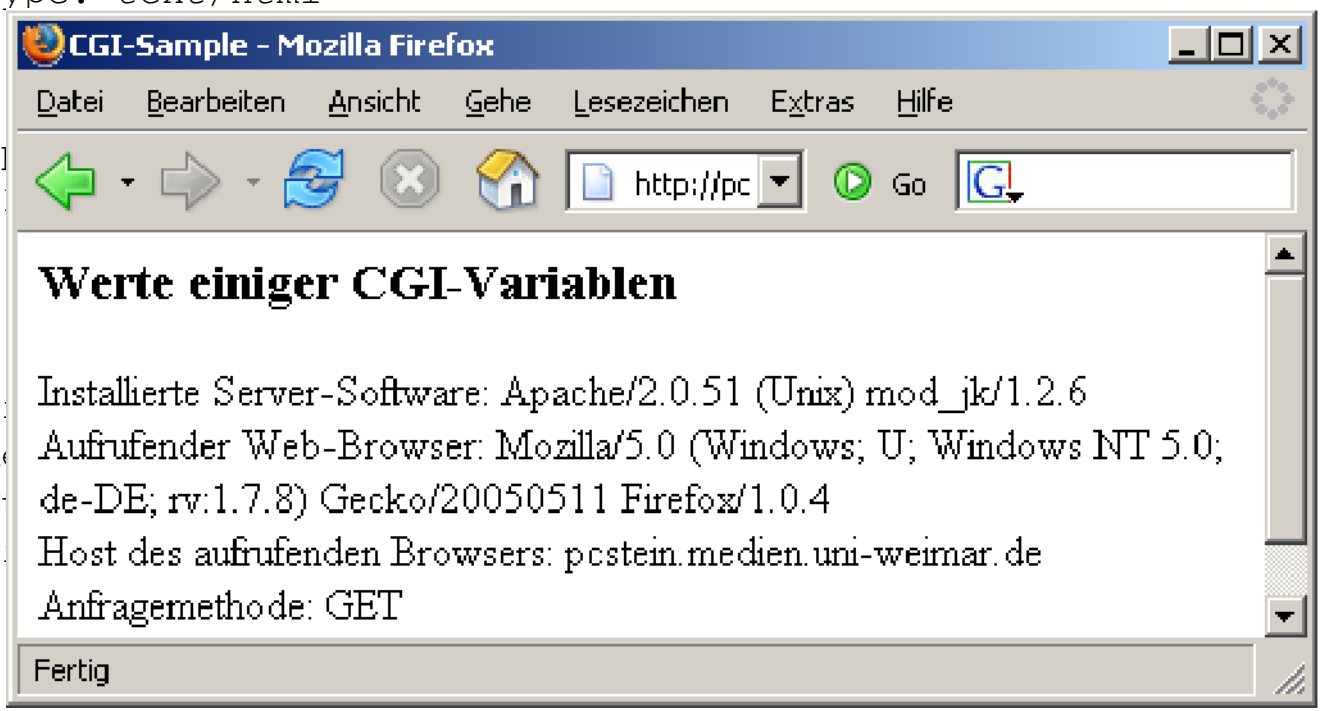
In der HTML-Datei:

```
<a href="http://localhost/cgi-bin/cgi-sample1.cgi">CGI-Aufruf</a>
```

Das Shell-Script cgi-sample1.cgi :

```
#!/bin/csh
echo "content-type: text/html"
echo "<html>"
echo "<head>"
echo "<meta http
echo "<title>CG
echo "</head>"
echo "<body>"
echo "<h3>Werte
echo "Installie
echo "Aufrufend
echo "Anfrageme
echo "Query-Str
echo "</body>"
echo "</html>"
```

[Demo]



Java Servlet

Einführung [Einordnung]

“A servlet is a Java programming language class used to extend the capabilities of servers that host applications accessed via a request-response programming model.”

[Oracle]

Java Servlet

Einführung [Einordnung]

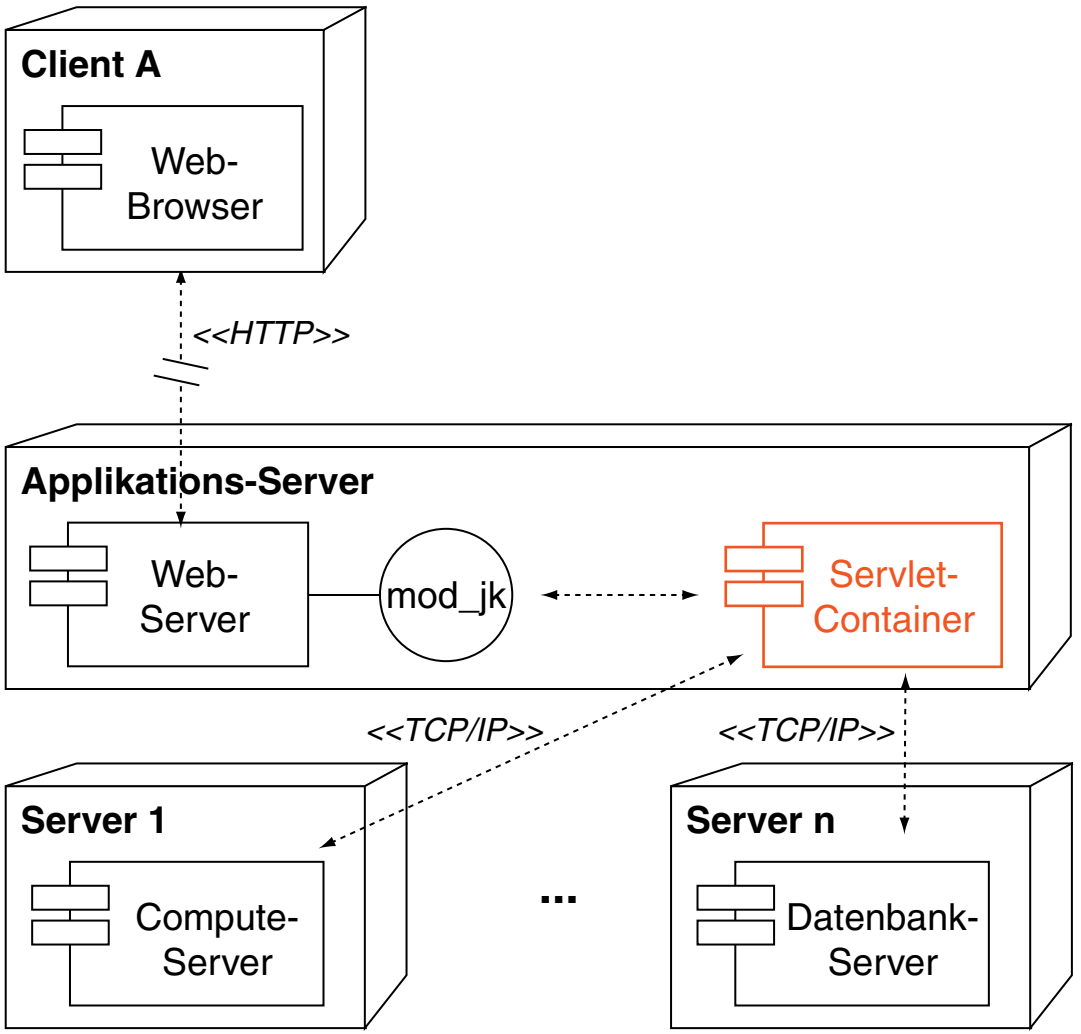
> *"A servlet is a Java programming language class used to extend the capabilities of servers that host applications accessed via a request-response programming model."*
>
> [Oracle]

- Servlets können jede Art von Anfrage beantworten; ihr Einsatz geschieht hauptsächlich im Zusammenhang mit Web-Servern.
- Die Java Servlet API besteht aus den `javax.servlet`-Packages. Diese gehören nicht zur J2SE, sondern zur Java Enterprise Edition J2EE. [Javadoc]
- Im Mittelpunkt der Servlet-Programmierung stehen:

Klasse bzw. Interface	**Konzepte**
`HttpServlet`	`service(), doGet(), doPost()`
`HttpServletRequest`	`get...()`-Methoden für CGI-Environment
`HttpServletResponse`	Streams als Ausgabekanal zum Client

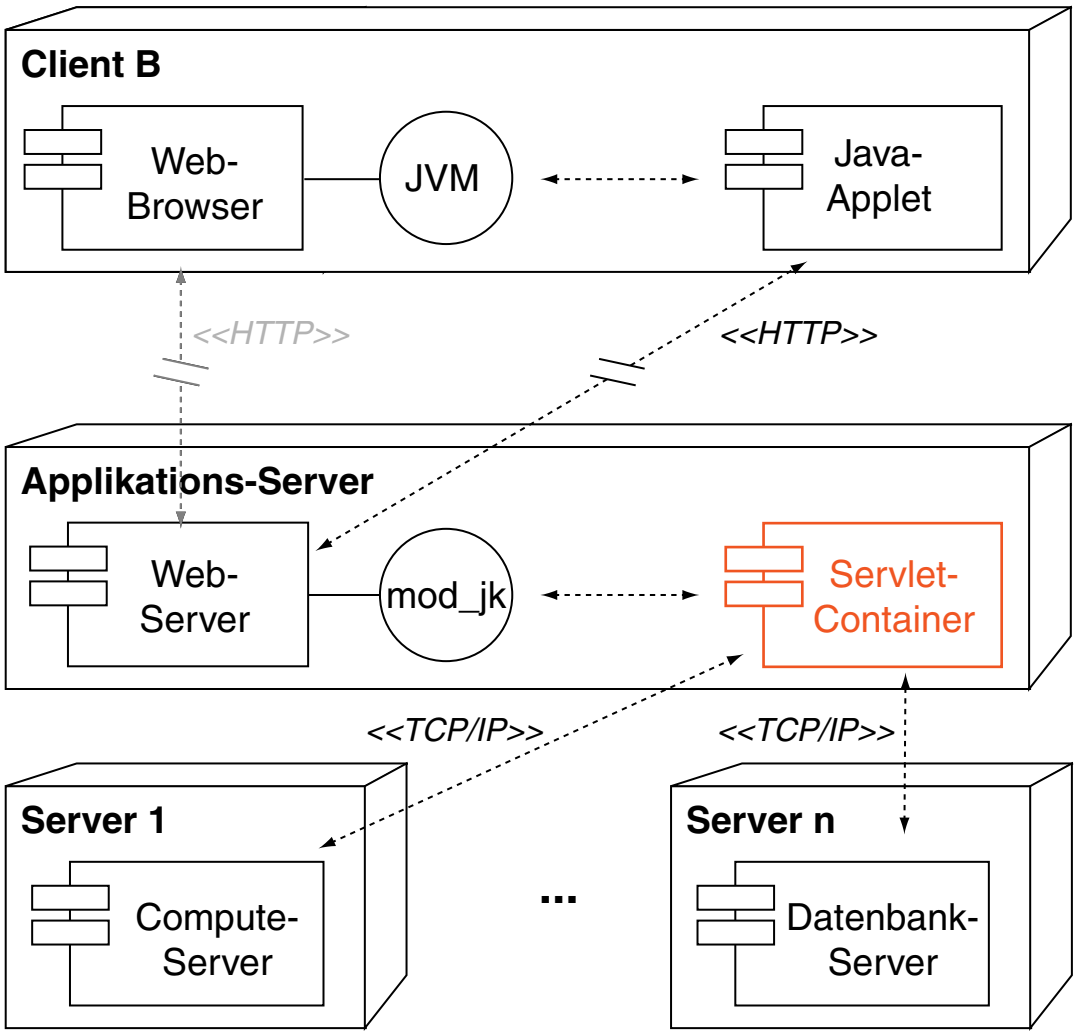
Java Servlet

Einführung (Fortsetzung)



Java Servlet

Einführung (Fortsetzung)



Bemerkungen:

- Die Deployment-Diagramme zeigen zwei Szenarien:
  1. Client A fragt beim Web-Server eine URL an, die evtl. auf ein Servlet verweist.
  2. Client B hat sich vom Web-Server ein Applet ausliefern lassen. Dieses Applet kann die Benutzerschnittstelle einer komplexen, mit Servlets realisierten Anwendung sein. Hierfür kommuniziert das Applet über den Web-Server mit einem oder mehreren Servlets.

Java Servlet

Servlet-Lebenszyklus

Der Lebenszyklus von Servlets wird von dem Container gesteuert, in dem das Servlet abgelegt ist.

Ein HTTP-Servlet wird durch einen HTTP-Request über eine URL angesprochen. Dann führt der Container folgende Schritte aus:

1. Überprüfung, ob eine Servlet-Instanz läuft. Falls nicht, wird
   (a) die Servlet-Klasse geladen,
   (b) eine Instanz der Servlet-Klasse erzeugt und
   (c) die Servlet-Instanz durch Aufruf der `init()`-Methode initialisiert.

2. Erzeugung eines `HttpServletRequest`-Objektes und eines `HttpServletResponse`-Objektes.

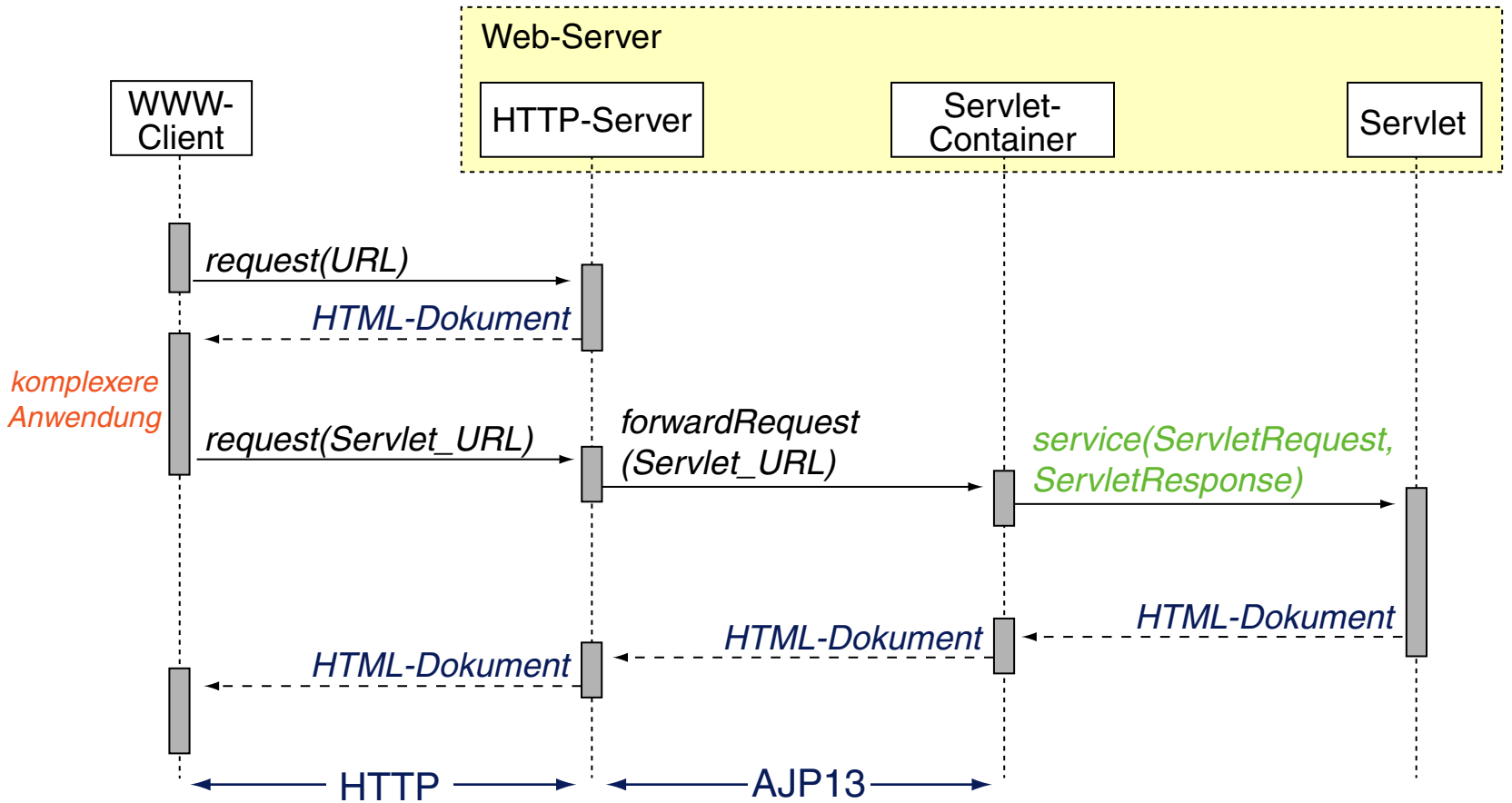
3. Aufruf der `service()`-Methode der Servlet-Instanz. Sie analysiert die Anfrage des Web-Servers und dispatched entsprechend; das Request-Objekt und das Response-Objekt werden mit übergeben.

Bemerkungen:

- Die `service()`-Methode erkennt die HTTP/1.1-Anfragen GET, POST, HEAD, PUT, DELETE, OPTIONS und TRACE. Die entsprechenden Java-Methoden heißen `doGet()`, `doPost()`, etc.
- Falls der Container eine Servlet-Instanz entladen soll, ruft er dessen `destroy()`-Methode auf.

Java Servlet

Ablauf einer Servlet-Interaktion



Vergleiche hierzu den Ablauf einer CGI-Interaktion.

Java Servlet

Charakteristika der Servlet-Technologie

- Portabilität
- Leistungsfähigkeit
- Effizienz
- Sicherheit
- Produktivität

Java Servlet

Charakteristika der Servlet-Technologie

- Portabilität

  Servlets sind über Betriebssysteme und Web-Server hinweg portabel.

- Leistungsfähigkeit

  Alle Konzepte von Java (Multithreading, Serialisierung, etc.) stehen zur Verfügung, einschließlich der gesamten Java-Bibliothek.

- Effizienz

  Eine Servlet-Instanz (ein Java-Objekt) wird nur einmal geladen und bleibt – mit seinem Zustand – im Speicher des Web-Servers.

- Sicherheit

  Java selbst ist sehr robust; zum Schutz des Web-Servers existieren darüberhinaus die Sicherheitsmechanismen des Java Security Managers. Stichwort: Servlet-Sandbox

- Produktivität

  Konzepte wie *Session Tracking, Cookie Handling* etc. erleichtern die Anwendungsentwicklung.

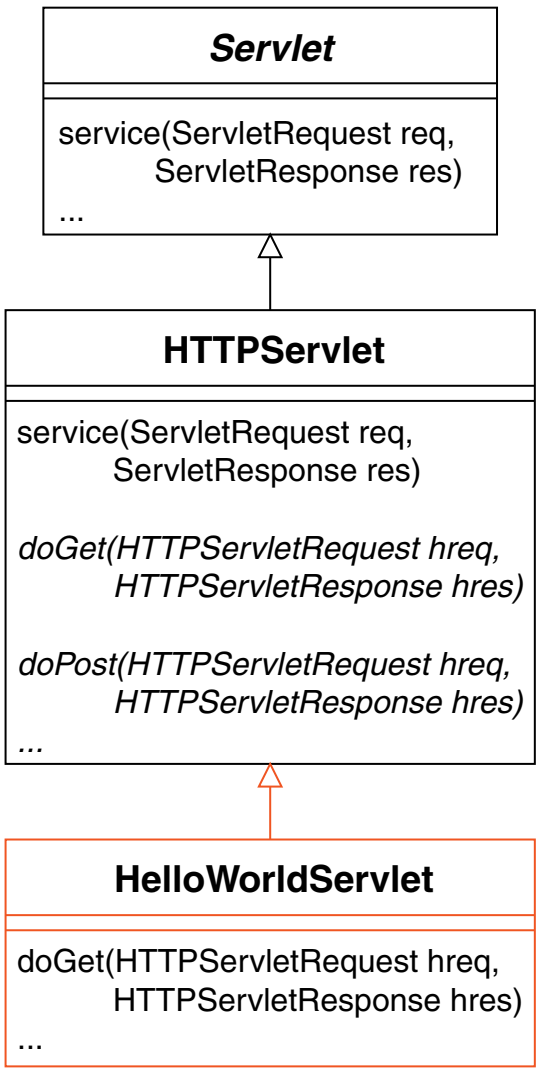
Java Servlet

Beispiel: HelloWorld [Demo]

```
package servlet;

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

public class ServletHelloWorld extends HttpServlet {

     public void init() throws ServletException {
          // Nothing to do here.
     }

     public void doGet(HttpServletRequest request,HttpServletResponse response)
               throws ServletException, IOException {
          // Use "response" to specify the HTTP response line and headers
          // (e.g. specifying the content type, setting cookies).
          response.setContentType("text/html");
          PrintWriter out = respons.getWriter();
          // Use "out" to send content to browser.
          out.println("<!DOCTYPE HTML>\n"
                    + "<html>\n"
                    + "<head><title>Hello World</title></head>\n" + "<body>\n"
                    + "<h3>Hello World!</h3>\n" + "</body></html>");
     }
}
```

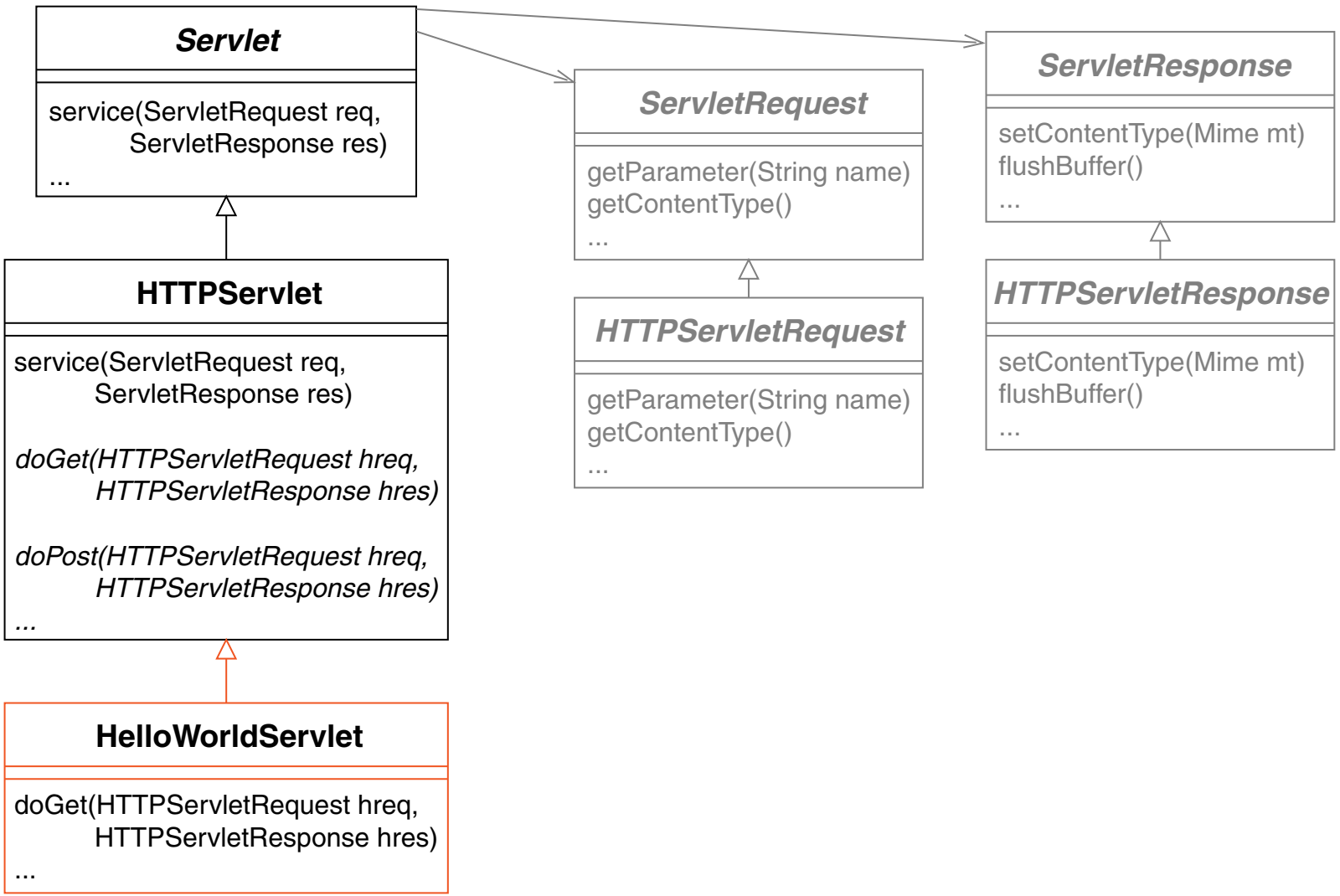
Java Servlet

Implementierung folgt dem „Template Method“-Pattern



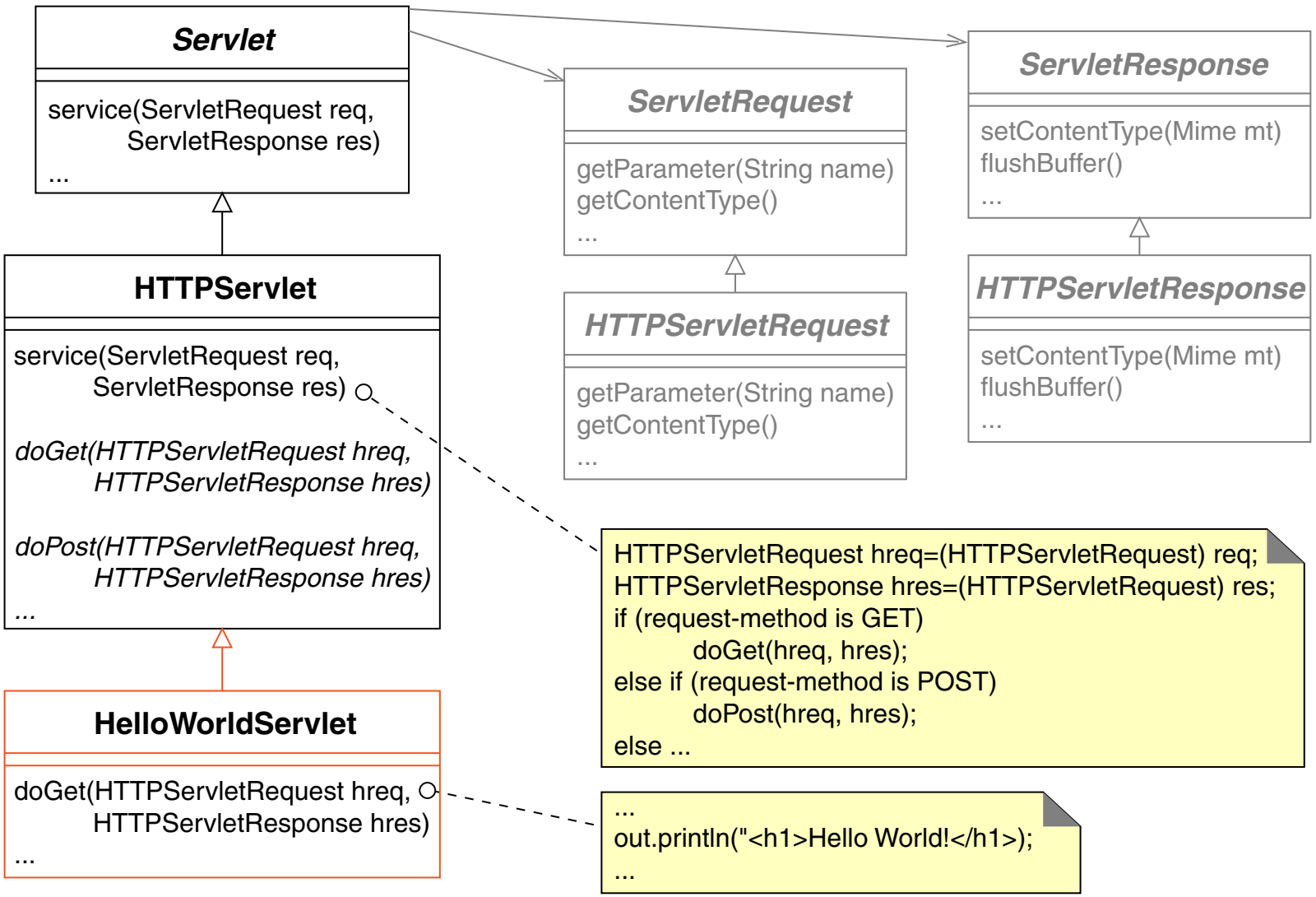
Java Servlet

Implementierung folgt dem „Template Method“-Pattern



Java Servlet

Implementierung folgt dem „Template Method“-Pattern



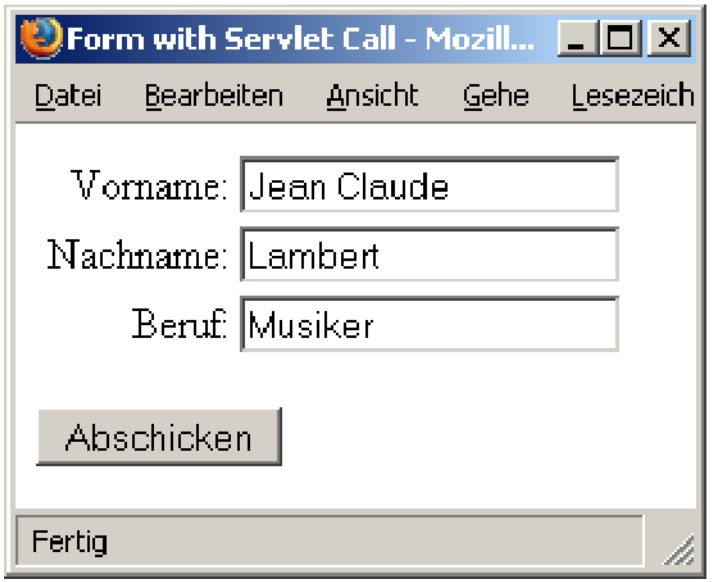
Java Servlet

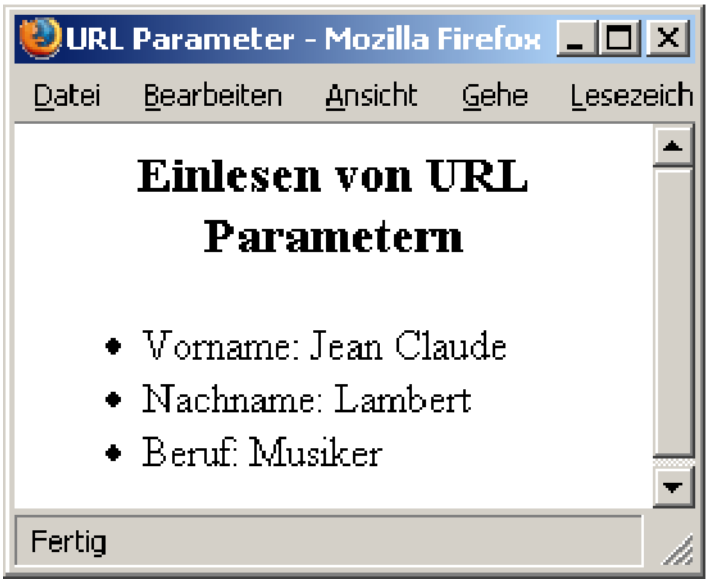
Beispiel: URL-Parameter einlesen

```
<html>
  <head>
    <meta http-equiv="content-type" content="text/html; ...">
    <title>Form with Servlet Call</title>
  </head>
  <body>
    <form action="http://.../ServletReadURLParam" method="GET">
      <table border="0" cellpadding="0" cellspacing="4">
        <tr>
          <td align="right">Vorname:</td>
          <td><input name="vorname" type="text" size="20" ...></td>
        </tr>
        ...
      </table>
      <p>
        <input type="submit" name="z" value="Abschicken"> <br>
      </p>
    </form>
  </body>
</html>
```

Java Servlet

Beispiel: URL-Parameter einlesen (Fortsetzung)





[Demo]

Java Servlet

Beispiel: URL-Parameter einlesen (Fortsetzung)

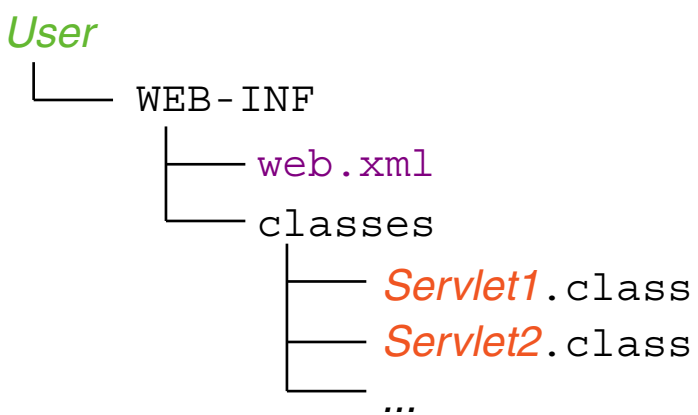
```
package servlet;

import java.io.*; ...

public class ServletReadURLParam extends HttpServlet {

    public void doGet(HttpServletRequest request,HttpServletResponse response)
            throws ServletException, IOException {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        String title = "Einlesen von URL Parametern";
        // Use "request" to read incoming HTTP headers (e.g. cookies)
        // and HTML form data (e.g. data the user entered and submitted)
        out.println("<head><title>URL Parameter</title></head>\n" + "<body>\n"
                + "<h3 align=center>" + title + "</h3>\n" + "<ul>\n"
                + " <li>Vorname: " + request.getParameter("vorname") + "\n"
                + " <li>Nachname: " + request.getParameter("nachname") + "\n"
                + " <li>Beruf: " + request.getParameter("beruf") + "\n"
                + "</ul>\n" + "</body></html>");
    }

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

Java Servlet

Deployment

Die Bereitstellung (*Deployment*) Servlet-basierter Web-Anwendungen ist standardisiert; der Java Community Process hat hierfür das `.war`-Archivformat spezifiziert.

Verzeichnisstruktur:



Bemerkungen:

- Das Deployment kann im laufenden Betrieb eines Web-Servers erfolgen.
- Liegt das `WEB-INF`-Verzeichnis im Web-Space von *User*, erfolgt der Aufruf von *Servlet* typischerweise mit `http://`*Web-Server*/*User*/*Servlet*.
- Die Datei `web.xml` ist der Deployment-Descriptor und enthält die URL-Mappings zu den Servlets. Folgende `web.xml`-Datei beschreibt das `ServletHelloWorld`-Servlet:

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<web-app xmlns="http://java.sun.com/xml/ns/j2ee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee ..."
    version="2.4">

  <servlet>
    <servlet-name>HelloWorld</servlet-name>
    <servlet-class>servlet.ServletHelloWorld</servlet-class>
  </servlet>

  <servlet-mapping>
    <servlet-name>HelloWorld</servlet-name>
    <url-pattern>/ServletHelloWorld</url-pattern>
  </servlet-mapping>

</web-app>
```

Java Server Pages JSP

Einführung [Einordnung]

Ziel: Pflege von statischen und dynamischen Inhalten in *einer* Datei.

Java Server Pages JSP

Einführung [Einordnung]

Ziel: Pflege von statischen und dynamischen Inhalten in *einer* Datei.

Charakteristika, Technologie:

- dokumentenzentrierte Programmierung – direkt über den Web-Client
- Einbettung von Java-Code in HTML
- Java für dynamische, HTML für statische Dokumentbestandteile
- automatische Code-Generierung und Code-Verwaltung mit Servlets
- Zugriff auf und Verwendung des JavaBeans-Komponentenmodells

Anwendung:

- mittelgroße bis sehr große Projekte, einfache bis komplexe Probleme
- bei Forderung nach Portabilität, skalierbarer Architekur, hoher Performanz

Java Server Pages JSP

Einführung (Fortsetzung)

Java-Code wird in die HTML-Datei eingebettet:

```
<!DOCTYPE HTML>
<html>
  <head> <title>JSP Sample</title> </head>
  <body>
    <h1> <% out.print("Hello " + "World!"); %> </h1>
  </body>
</html>
```

Java Server Pages JSP

Einführung (Fortsetzung)

Java-Code wird in die HTML-Datei eingebettet:

```
<!DOCTYPE HTML>
<html>
  <head> <title>JSP Sample</title> </head>
  <body>
    <h1> <% out.print("Hello " + "World!"); %> </h1>
  </body>
</html>
```

[Demo] [generiertes Servlet: 1, 2]

Java Server Pages JSP

Einführung (Fortsetzung) [generierte Web-Seite] [generiertes Servlet]

```
<!DOCTYPE HTML>
<html>
  <head> <title>Registration</title> </head>
  <body>

    <form action="registration.jsp" method="get">
      <table>
        <tr><td>Benutzername:</td>
            <td><input type="text" name="user"></td></tr>
        <tr><td><input type="submit" value="Anmelden"></td></tr>
      </table>
    </form>

  </body>
</html>
```

Java Server Pages JSP

Einführung (Fortsetzung) [generierte Web-Seite] [generiertes Servlet]

```
<!DOCTYPE HTML>
<html>
  <head> <title>Registration</title> </head>
  <body>

    Benutzername:

  </body>
</html>
```

Java Server Pages JSP

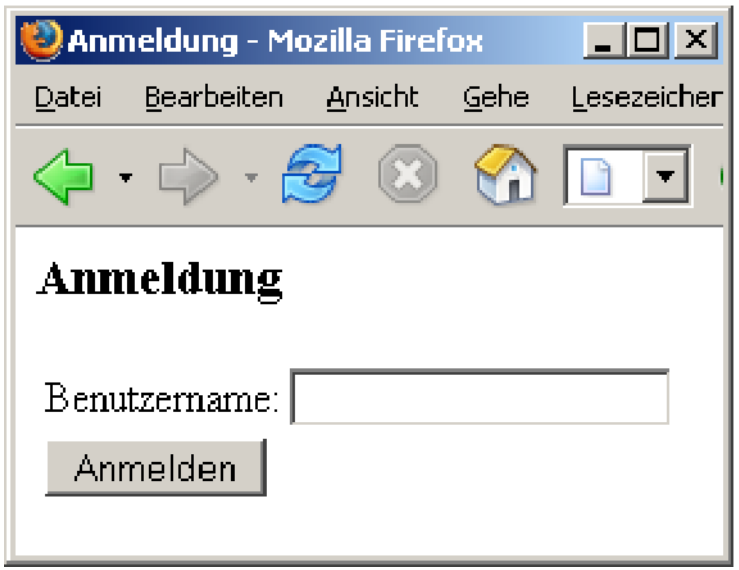
Einführung (Fortsetzung) [generierte Web-Seite] [generiertes Servlet]

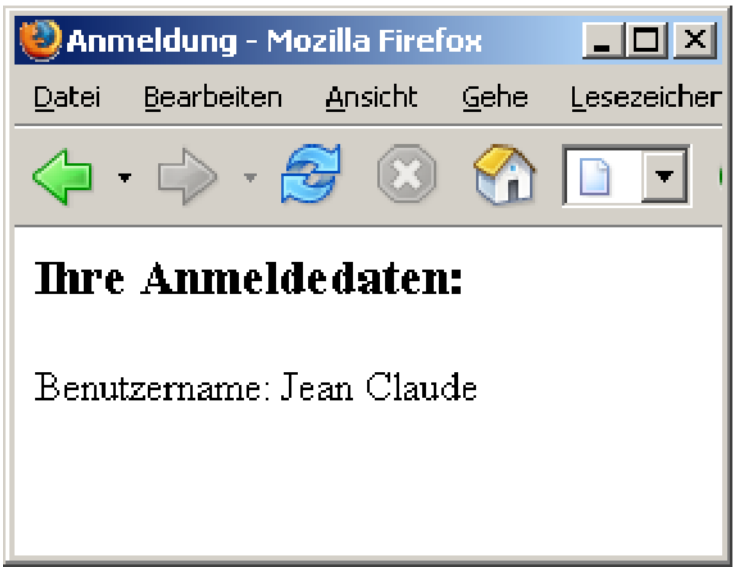
```
<!DOCTYPE HTML>
<html>
  <head> <title>Registration</title> </head>
  <body>
    <%
      String user = request.getParameter("user");
      if ( user == null || "".equals(user) ) {
    %>
    <form action="registration.jsp" method="get">
      <table>
        <tr><td>Benutzername:</td>
            <td><input type="text" name="user"></td></tr>
        <tr><td><input type="submit" value="Anmelden"></td></tr>
      </table>
    </form>
    <% } else { %>
    Benutzername: <%= user %>
    <% } %>
  </body>
</html>
```

Vergleiche hierzu die PHP-Realisierung.

Java Server Pages JSP

Einführung (Fortsetzung)

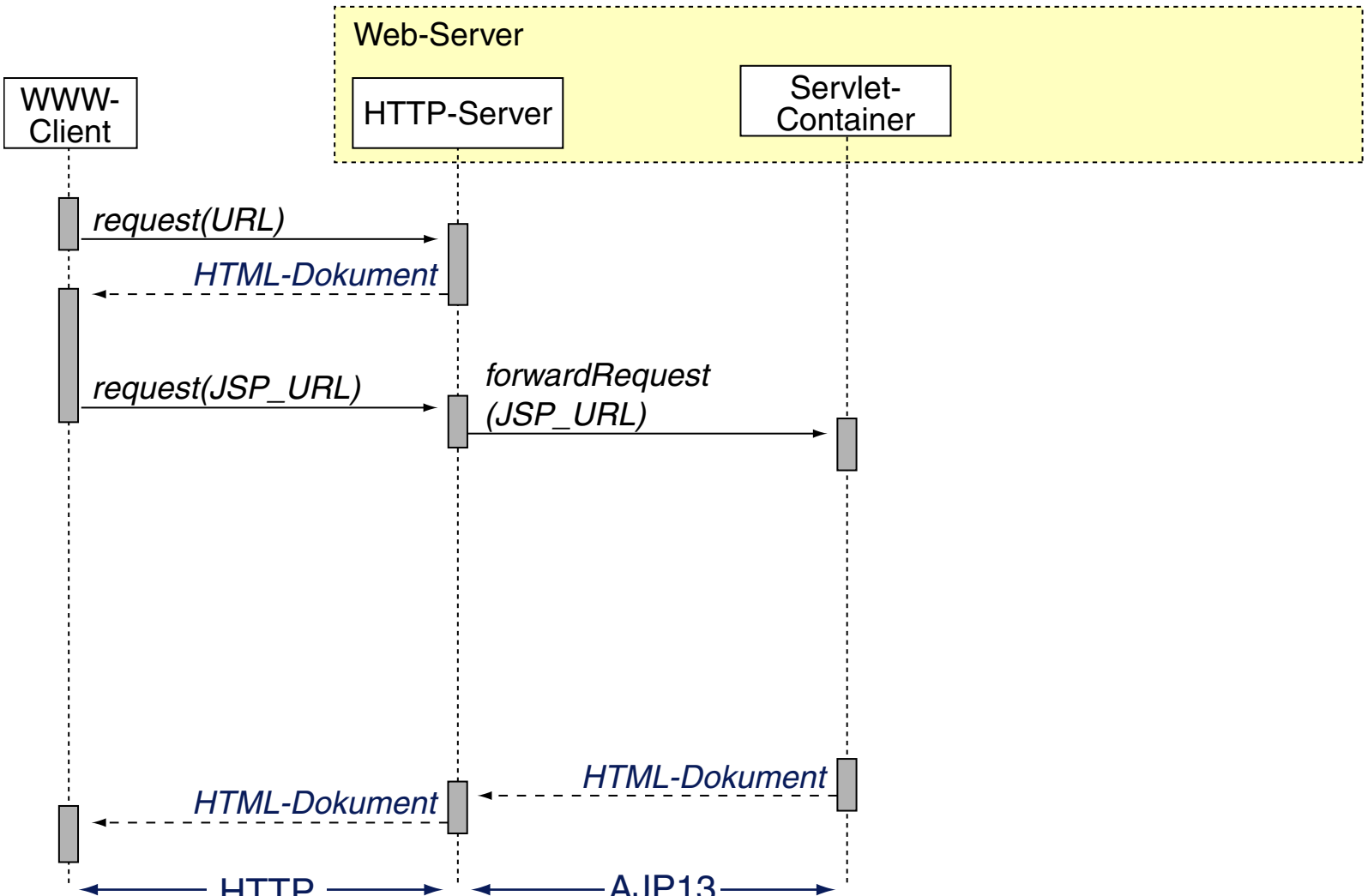




[Demo]

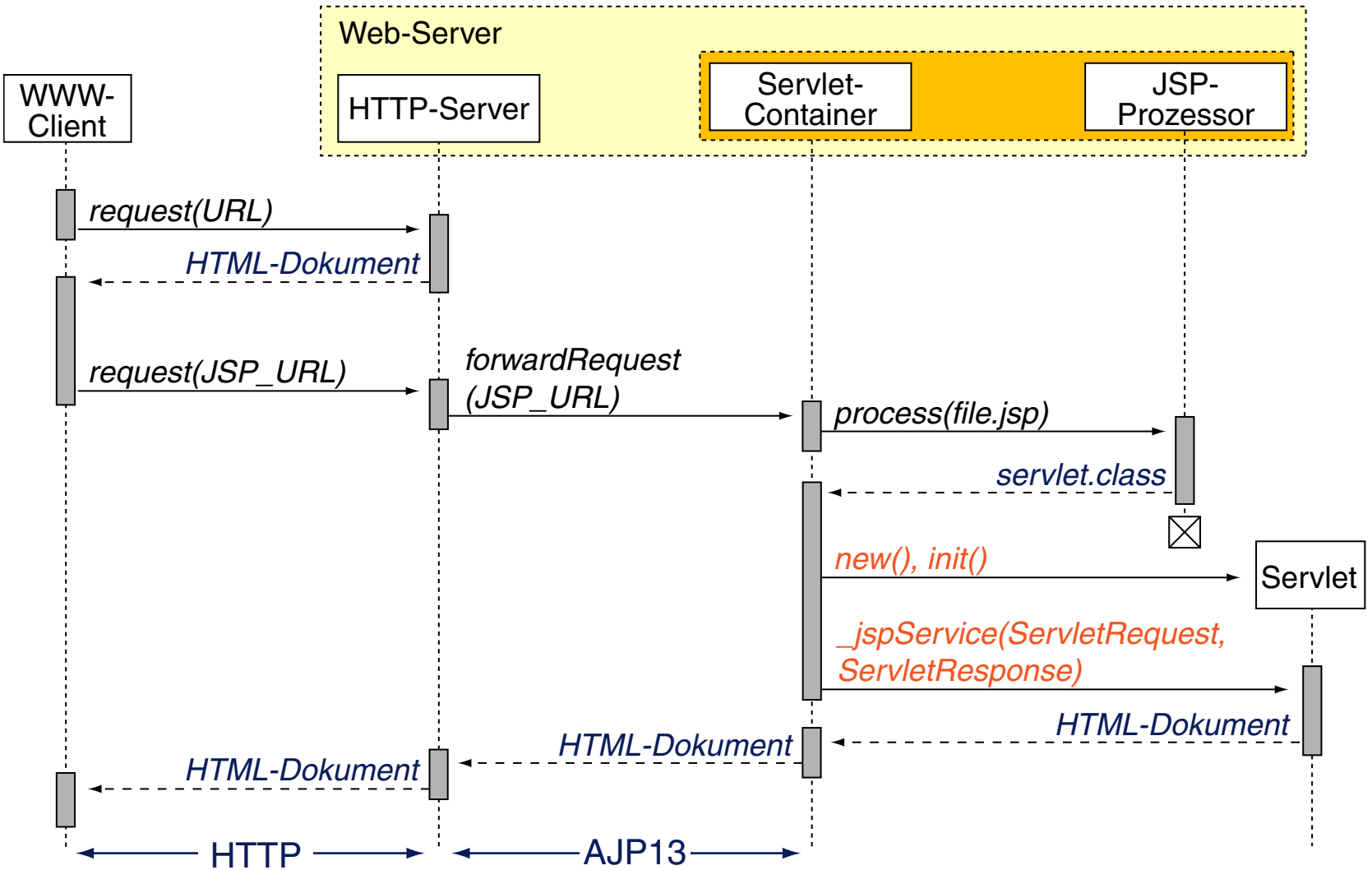
Java Server Pages JSP

Ablauf einer JSP-Interaktion



Java Server Pages JSP

Ablauf einer JSP-Interaktion



Bemerkungen:

- Die `.jsp`-Datei kombiniert HTML und Java-Code im Sinne eines Programms zur Erzeugung einer HTML-Seite.
- Die Generierung des Servlets aus der `.jsp`-Datei geschieht „on the fly“. Dabei wird auch erkannt, ob die `.jsp`-Datei zwischenzeitlich verändert wurde.
- Das Action-Attribut `<... action="registration.jsp">` ist laut Referenz obligatorisch. Falls es fehlt, wird als Default-Wert die Datei (hier: `registration.jsp`) selbst genommen.
- Es gibt kein isoliertes CGI-Programm mehr: HTML-Form, HTML-Antwort, Programm zur Verarbeitung – alles befindet sich in derselben Datei.

Java Server Pages JSP

Konzepte der JSP-Technologie

Der HTML-Code einer `.jsp`-Datei wird um `out.println()`-Anweisungen ergänzt und in die `_jspService()`-Methode des Servlets integriert. [Javadoc]

Java Server Pages JSP

Konzepte der JSP-Technologie

Der HTML-Code einer `.jsp`-Datei wird um `out.println()`-Anweisungen ergänzt und in die `_jspService()`-Methode des Servlets integriert. [Javadoc]

Zur Programmierung gibt es drei Konzepte:

1. Java.
   (a) Java-*Scriptlet*: beliebige Java-Anweisungen.
   Code wird Teil der `_jspService()`-Methode. Syntax: `<%` *Code* `%>`
   (b) Java-*Expression*: Wert, der zur Laufzeit ermittelt und als String ausgegeben wird.
   Code wird Teil der `_jspService()`-Methode. Syntax: `<%=` *Code* `%>`
   (c) Java-*Declaration*: beliebige Java-Anweisungen.
   Code ist außerhalb der `_jspService()`-Methode. Syntax: `<%!` *Code* `%>`.
2. JSP-Aktionen.
   Anbindung von JavaBeans-Komponenten; verwendet XML-Syntax.
3. JSP-Direktiven.
   Anweisungen, die direkt vom JSP-Prozessor verarbeitet werden.

Java Server Pages JSP

JSP-Prozessor

```
...
<html>
 <body>
  <h1>Hello World</h1>
  <% String user=request.getParameter("user"); %>
  <%= clock.getDayofMonth() %>
  <%! private int accessCount = 0; %>
  ...
 </body>
</html>
```

JSP-Dokument

```
public class SampleServlet extends HttpServlet {
  ...

  public void _jspService(...) throws ServletException {



    ...
  }
}
```

Java Server Pages JSP

JSP-Prozessor

```
...
<html>
 <body>
  <h1>Hello World</h1>
   <% String user=request.getParameter("user"); %>
   <%= clock.getDayofMonth() %>
   <%! private int accessCount = 0; %>
   ...
 </body>
</html>
```

JSP-Dokument

```
public class SampleServlet extends HttpServlet {
  ...

  public void _jspService(...) throws ServletException {
    out.println("<h1>Hello World</h1>");


    ...
  }
}
```

Java Server Pages JSP

JSP-Prozessor

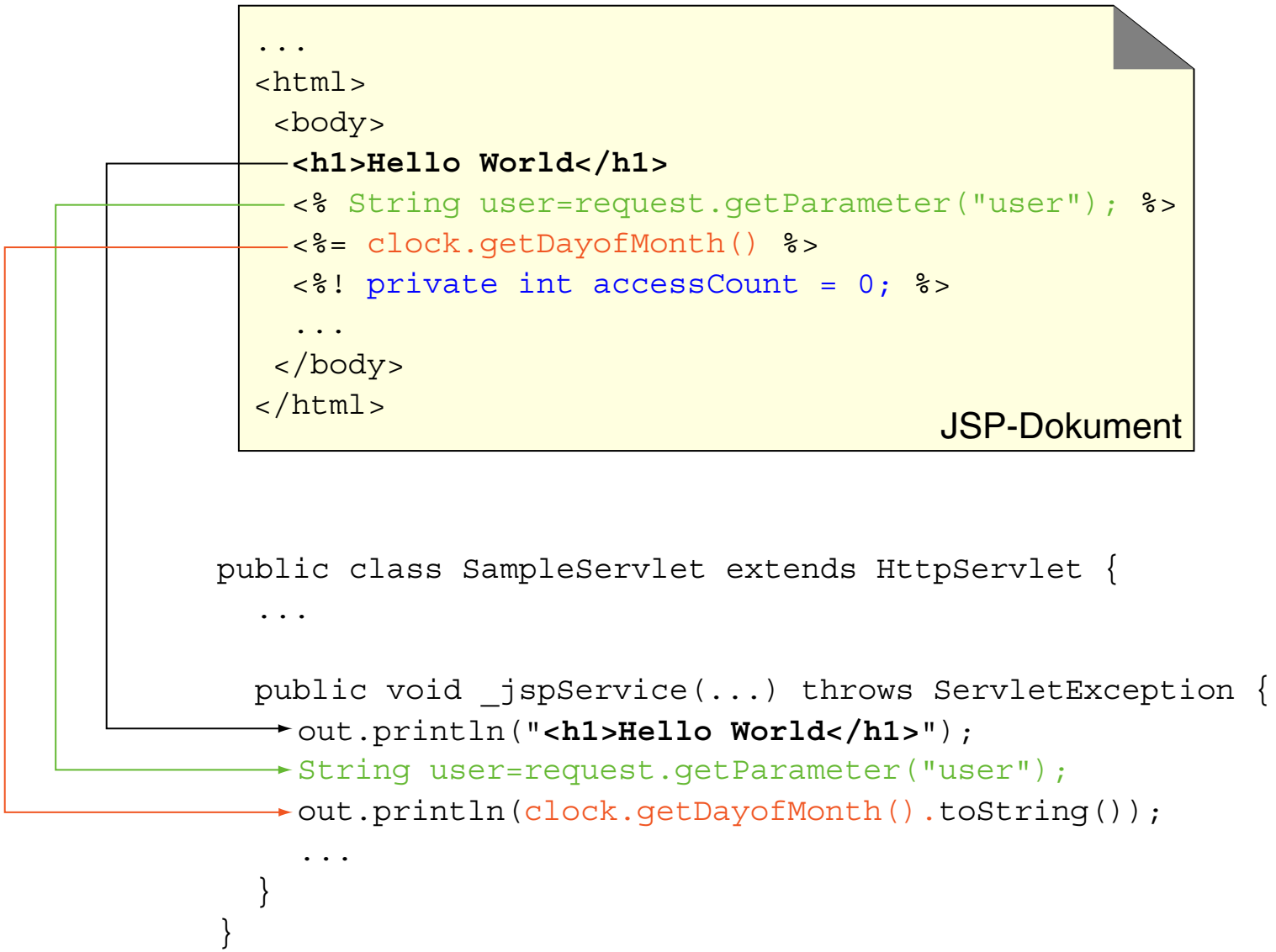
```
...
<html>
 <body>
  <h1>Hello World</h1>
  <% String user=request.getParameter("user"); %>
   <%= clock.getDayofMonth() %>
   <%! private int accessCount = 0; %>
   ...
 </body>
</html>
```

JSP-Dokument

```
public class SampleServlet extends HttpServlet {
   ...

   public void _jspService(...) throws ServletException {
      out.println("<h1>Hello World</h1>");
      String user=request.getParameter("user");

      ...
   }
}
```

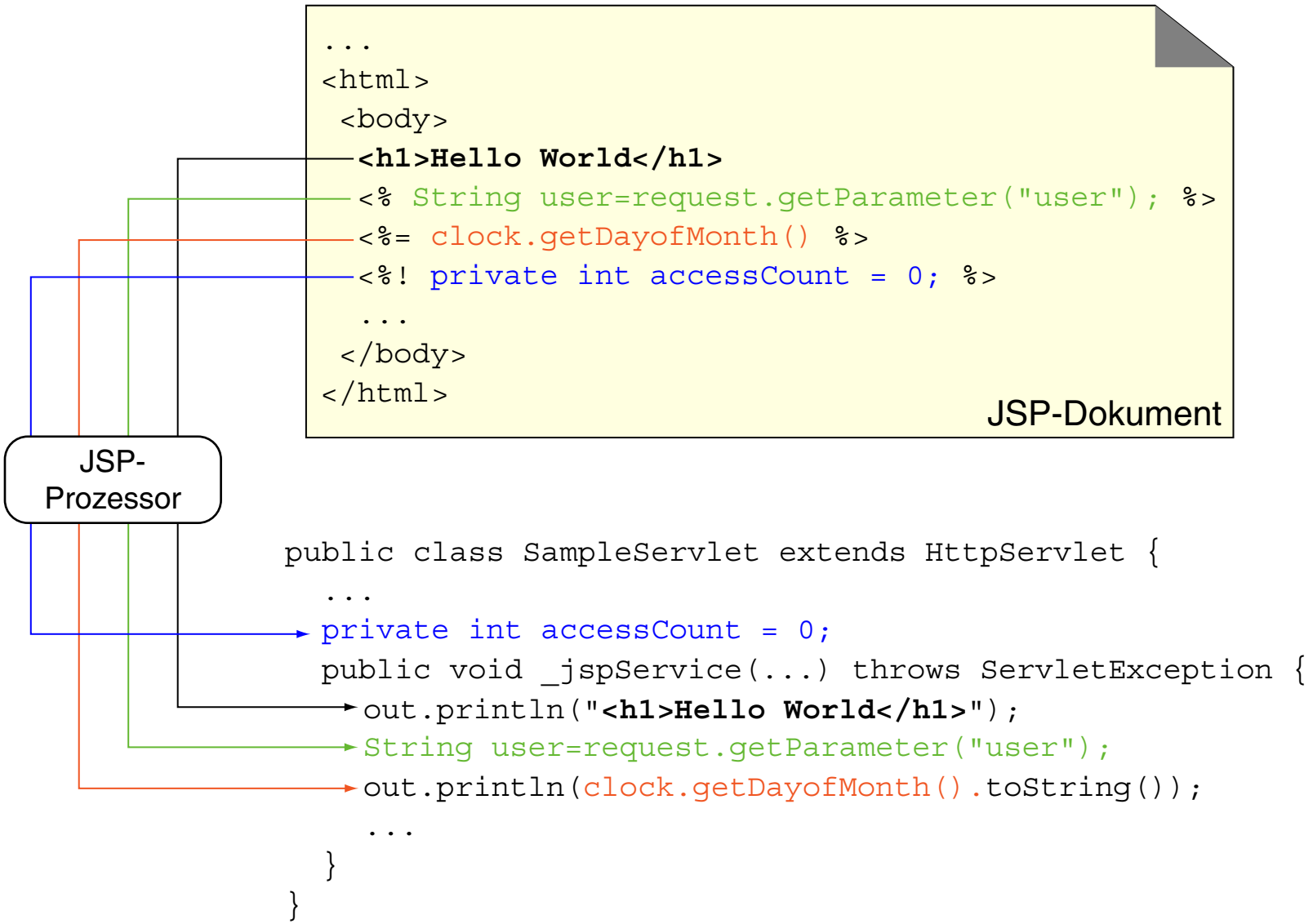
Java Server Pages JSP

JSP-Prozessor



Java Server Pages JSP

JSP-Prozessor



Java Server Pages JSP

JSP versus Active Server Pages ASP

Kriterium	Java Server Pages JSP	Active Server Pages ASP
Wartbarkeit	+ + + + +	+ + ++
Performance	+ + + + +	+ + + + +
Skalierbarkeit	+ + + + +	+ + ++
Verfügbarkeit	+ + + + +	+ + +
Plattformneutralität	+ + ++	
Abstraktionslevel	+ + + + +	+ + ++
Erweiterbarkeit	+ + + + +	+ + ++
Dokumentation	+ + +	+ + + + +
Support	+ + +	+ + +
Tool-Unterstützung	+ + ++	+ + + + +

[Wöhr 2004, S.390]

Server-Technologien

Quellen zum Nachlernen und Nachschlagen im Web

- Apache SF. *Apache HTTP Server Documentation.*
  httpd.apache.org/docs
- S. Münz. *SELFHTML: CGI.*
  de.selfhtml.org/servercgi
- ORACLE. *Java Servlet Tutorials.*
  download.oracle.com/javaee/1.4/tutorial/doc/Servlets2.html
  download.oracle.com/javaee/6tutorial/doc/bnafd.html
- ORACLE. *Java Servlet Technology.*
  www.oracle.com/technetwork/java/javaee/servlet/index.html
- M. Hall. *Servlets and JavaServer Pages Tutorial.*
  www.apl.jhu.edu/∼hall/java/Servlet-Tutorial

Kapitel WT:IV (Fortsetzung)

Exkurs: reguläre Ausdrücke

Grundlagen: Grammatik

- Alphabet Σ.
- Wort w.
- Sprache L.
- Grammatik G.

Exkurs: reguläre Ausdrücke

Grundlagen: Grammatik

- Alphabet Σ.
  Ein Alphabet Σ ist eine nicht-leere Menge von Zeichen bzw. Symbolen.

- Wort w.
  Ein Wort w ist eine endliche Folge von Symbolen aus Σ. Die Länge eines Wortes $|w|$ ist die Anzahl seiner Symbole.

  ε bezeichnet das leere Wort; es hat als einziges Wort die Länge 0.
  Σ^* bezeichnet die Menge aller Worte über Σ.

- Sprache L.
  Eine Sprache L ist eine Menge von Worten über einem Alphabet Σ.

- Grammatik G.
  Eine Grammatik G ist ein Kalkül, um eine Sprache zu definieren – also eine Menge von Regeln, mit denen man Worte ableiten kann. Die zu G gehörende Sprache besteht aus allen ableitbaren, terminalen Worten.

Bemerkungen:

- Bei der Definition von Spracheigenschaften unterscheidet man verschiedene Ebenen [Exkurs: Programmiersprachen]. Die Ebene 1 behandelt die Notation von Grundsymbolen, die Ebene 2 behandelt die syntaktische Struktur der Sprache. Zur Unterscheidung, auf welcher Ebene der Grammatikanwendung man sich befindet, werden auch folgende Begriffe verwendet:
  - Ebene 1: Alphabet, Zeichen, Wort, Sprache
  - Ebene 2: Vokabular, Symbol, Satz, Sprache
- Die Worte {Alphabet, Vokabular}, {Zeichen, Symbol} und {Wort, Satz} sind die jeweiligen Entsprechungen der Grundsymbolebene und der syntaktischen Ebene.

Exkurs: reguläre Ausdrücke

Grundlagen: Grammatik (Fortsetzung)

Definition 1 (Grammatik)

Eine Grammatik ist ein Viertupel $G = (N, \Sigma, P, S)$ mit

- N endliche Menge von Nichtterminalsymbolen
- Σ endliche Menge von Terminalsymbolen
- P endliche Menge von Produktionen bzw. Regeln

$$P \subset (N \cup \Sigma)^* \, N \, (N \cup \Sigma)^* \times (N \cup \Sigma)^*$$

- S Startsymbol, $S \in N$
- $N \cap \Sigma = \emptyset$

Bemerkungen:

- Eine Regel besteht aus einer linken Seite (Prämisse) und einer rechten Seite (Konklusion), die jeweils ein Wort bestehend aus Terminalen und Nichtterminalen sind. Die linke Seite muss mindestens ein Nichtterminal beinhalten und die rechte Seite kann dabei im Gegensatz zur linken Seite auch das leere Wort sein. [Wikipedia]
- Eine Regel kann auf ein Wort, bestehend aus Terminalen und Nichtterminalen, angewendet werden, wobei ein beliebiges Vorkommen der linken Seite der Regel im Wort durch die rechte Seite der Regel ersetzt wird: $w \rightarrow w'$
- Für w, w' gilt die sogenannte *Transitionsrelation*. Eine Folge von Anwendungen von Regeln bezeichnet man als *Ableitung*.

Exkurs: reguläre Ausdrücke

Grundlagen: Grammatik (Fortsetzung)

Definition 2 (erzeugte Sprache)

Die von einer Grammatik $G = (N, \Sigma, P, S)$ erzeugte Sprache $L(G)$ enthält genau die Worte, die nur aus Terminalsymbolen bestehen und vom Startsymbol aus mit einer endlichen Anzahl von Schritten abgeleitet werden können:

$$L(G) := \{w \in \Sigma^* \mid S \rightarrow_G^* w\}$$

$\rightarrow_G^*$ steht für die beliebige Anwendung der Produktionen in G, also die reflexiv-transitive Hülle der Transitionsrelation $\rightarrow_G$.

Exkurs: reguläre Ausdrücke

Grundlagen: Grammatik (Fortsetzung)

Definition 2 (erzeugte Sprache)

Die von einer Grammatik $G = (N, \Sigma, P, S)$ erzeugte Sprache $L(G)$ enthält genau die Worte, die nur aus Terminalsymbolen bestehen und vom Startsymbol aus mit einer endlichen Anzahl von Schritten abgeleitet werden können:

$$L(G) := \{w \in \Sigma^* \mid S \rightarrow_G^* w\}$$

$\rightarrow_G^*$ steht für die beliebige Anwendung der Produktionen in G, also die reflexiv-transitive Hülle der Transitionsrelation $\rightarrow_G$.

Beispiel:

$G = (N, \Sigma, P, S)$ mit $N = \{S, A, B\}$, $\Sigma = \{a, b\}$ und folgenden Produktionen:

$$\begin{array}{lcl} S & \rightarrow & ABS \\ S & \rightarrow & \varepsilon \end{array} \qquad \begin{array}{lcl} BA & \rightarrow & AB \\ BS & \rightarrow & b \\ Bb & \rightarrow & bb \\ Ab & \rightarrow & ab \\ Aa & \rightarrow & aa \end{array}$$

Bemerkungen:

- Es ist Konvention, die Nichtterminalsymbole mit Großbuchstaben und die Terminalsymbole mit Kleinbuchstaben zu bezeichnen.
- Zur Erzeugung einer Sprache existieren beliebig viele Grammatiken.
- Eine andere Grammatik, die die gleiche Sprache wie im Beispiel erzeugt, ist:
  $N = \{S, A, B\}$, $\Sigma = \{a, b\}$, $P = \{S \rightarrow aSb, S \rightarrow \varepsilon\}$

Exkurs: reguläre Ausdrücke

Grundlagen: Chomsky-Hierarchie

Grammatiken werden hinsichtlich der Komplexität der Sprachen, die sie erzeugen, in vier Klassen eingeteilt.

- Typ 0.
- Typ 1 $\sim$ kontextsensitiv.
- Typ 2 $\sim$ kontextfrei.
- Typ 3 $\sim$ regulär.

Exkurs: reguläre Ausdrücke

Grundlagen: Chomsky-Hierarchie

Grammatiken werden hinsichtlich der Komplexität der Sprachen, die sie erzeugen, in vier Klassen eingeteilt.

- Typ 0.
  Für die Regeln in P existieren keine Einschränkungen.

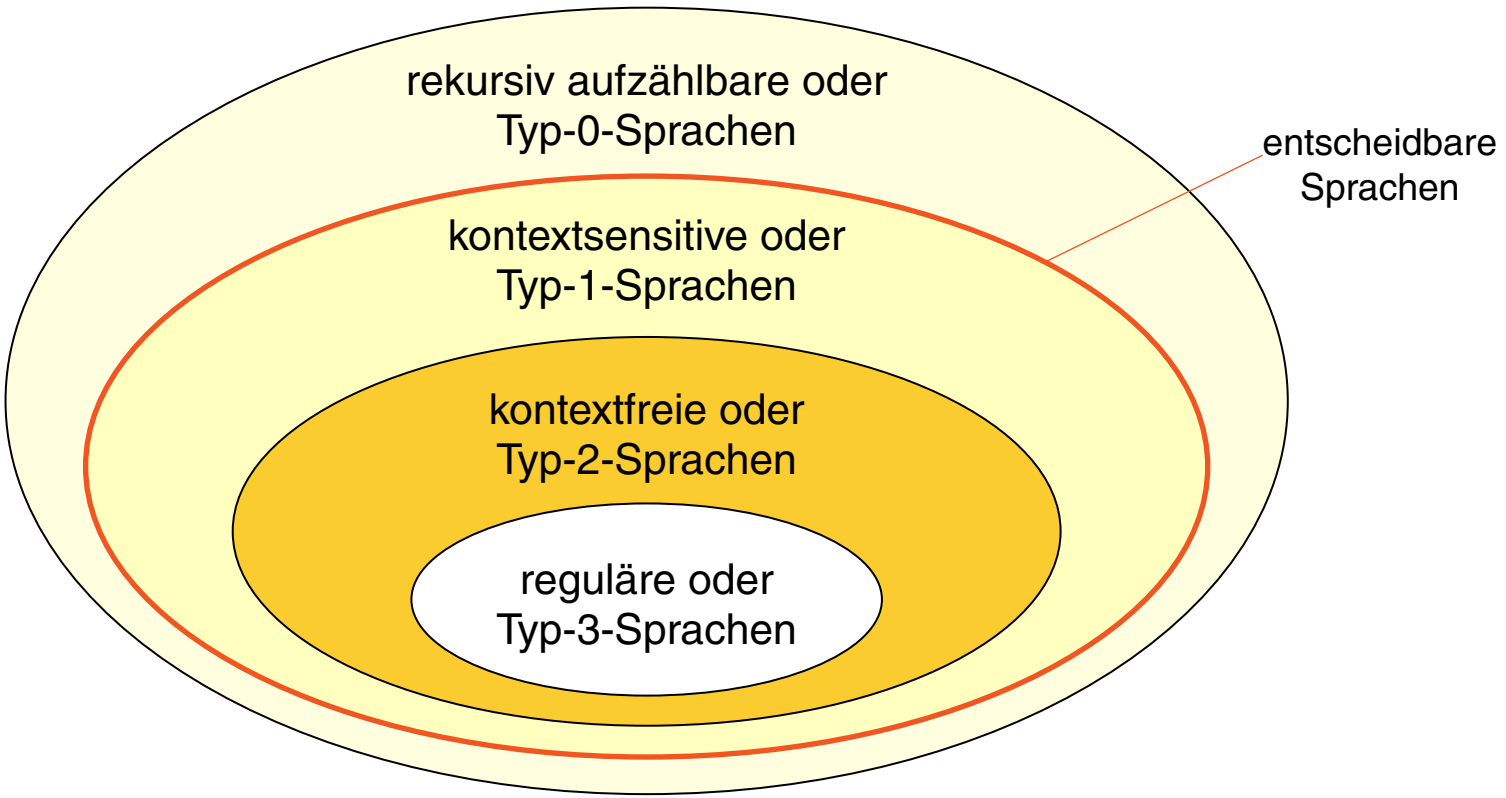
- Typ 1 $\sim$ kontextsensitiv.
  Für alle Regeln $w \rightarrow w' \in P$ gilt: $|w| \leq |w'|$

- Typ 2 $\sim$ kontextfrei.
  Für alle Regeln $w \rightarrow w' \in P$ gilt: w ist eine einzelne Variable; d.h., $w \in N$.

- Typ 3 $\sim$ regulär.
  Die Grammatik ist vom Typ 2 und zusätzlich gilt: $w' \in (\Sigma \cup \Sigma N)$, d.h., die rechten Seiten der Regeln sind entweder einzelne Terminalzeichen oder ein Terminalzeichen gefolgt von einem Nichtterminal.

Exkurs: reguläre Ausdrücke

Grundlagen: Chomsky-Hierarchie (Fortsetzung)



Definition 3 (Sprache vom Typ)

Eine Sprache $L \subseteq \Sigma^*$ wird Sprache vom Typ 0 (Typ 1, Typ 2, Typ 3) genannt, falls es eine Grammatik G vom Typ 0 (Typ 1, Typ 2, Typ 3) gibt, mit $L(G) = L$.

Bemerkungen:

- Die Chomsky-Hierarchie stellt eine Hierarchie mit echten Teilmengenbeziehungen dar:
  Typ 3 $\subset$ Typ 2 $\subset$ Typ 1 $\subset$ Typ 0
- Alle Sprachen vom Typ 1, 2 oder 3 sind entscheidbar:
  d.h., das Wortproblem für alle Sprachen vom Typ 1, 2 oder 3 ist entscheidbar;
  d.h., es gibt einen Algorithmus, der bei Eingabe einer Grammatik G und einem Wort w in endlicher Zeit feststellt, ob $w \in L(G)$ gilt oder nicht.
- Die Menge der Typ-0-Sprachen ist identisch mit der Menge der rekursiv aufzählbaren oder semi-entscheidbaren Sprachen. Daher gibt es Typ-0-Sprachen, die nicht entscheidbar sind.
- Im Übersetzerbau spielen Sprachen bzw. Grammatiken vom Typ 3 (lexikalische Analyse, Tokenisierung) und Typ 2 (syntaktische Strukturanalyse) die zentrale Rolle.

Exkurs: reguläre Ausdrücke

Kalküle für reguläre Sprachen

Verschiedene Kalküle zur Bildung von Worten einer regulären Sprache:

(a) endlichen Akzeptor bzw. Automat

(b) regulärer Ausdruck

(c) Typ-3-Grammatik

(d) Angabe endlich vieler Äquivalenzklassen

Exkurs: reguläre Ausdrücke

Kalküle für reguläre Sprachen

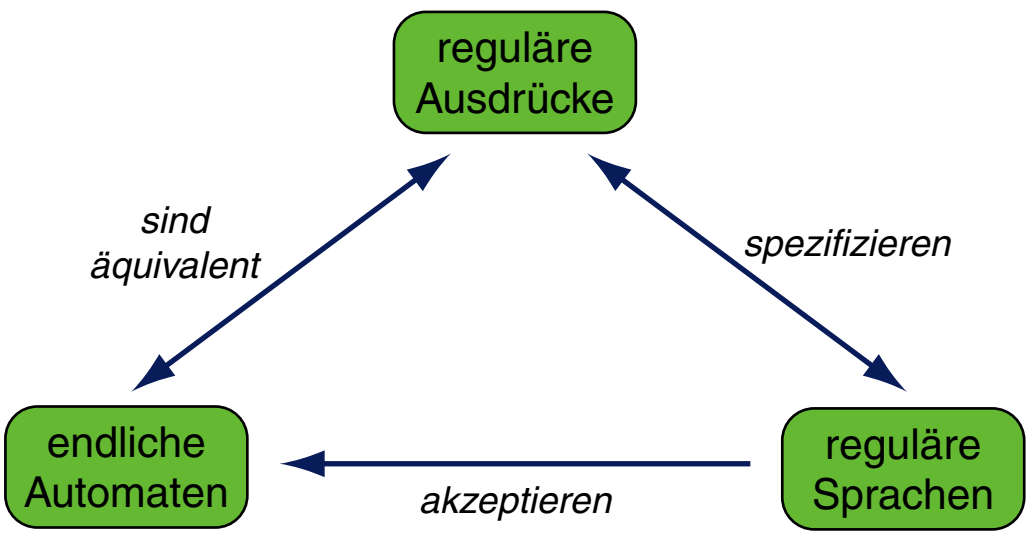
Verschiedene Kalküle zur Bildung von Worten einer regulären Sprache:

(a) endlichen Akzeptor bzw. Automat

(b) regulärer Ausdruck

(c) Typ-3-Grammatik

(d) Angabe endlich vieler Äquivalenzklassen



[vgl. Haenelt 2005]

Exkurs: reguläre Ausdrücke

Grundlagen: Zusammenfassung

	Kalküle zur Spracherzeugung
Typ 0	Typ-0-Grammatik allgemeine Turingmaschine
Typ 1	kontextsensitive Grammatik linear beschränkte Turingmaschine
Typ 2	kontextfreie Grammatik Kellerautomat
Typ 3	reguläre Grammatik (Typ-3-Grammatik) deterministischer/nicht-deterministischer endlicher Automat regulärer Ausdruck

Exkurs: reguläre Ausdrücke

Grundlagen: Zusammenfassung

	Kalküle zur Spracherzeugung
Typ 0	Typ-0-Grammatik allgemeine Turingmaschine
Typ 1	kontextsensitive Grammatik linear beschränkte Turingmaschine
Typ 2	kontextfreie Grammatik Kellerautomat
Typ 3	reguläre Grammatik (Typ-3-Grammatik) deterministischer/nicht-deterministischer endlicher Automat regulärer Ausdruck

	Komplexität des Wortproblems [Wikipedia]
Typ 0	unentscheidbar
Typ 1	exponentielle Komplexität, NP-hart
Typ 2	$O(n^3)$
Typ 3	lineare Komplexität

Exkurs: reguläre Ausdrücke

Konstruktion [Kastens 2005] [PHP]

Ein regulärer Ausdruck R kann wie folgt rekursiv zusammengesetzt sein. F und G bezeichnen gegebene reguläre Ausdrücke.

	R	**Erklärung**
1.	a	das Zeichen a
2.	FG	Zusammenfügen von zwei Worten
3.	$F \mid G$	Alternativen
4.	ε	das leere Wort
5.	(F)	Klammerung
6.	F^+	nicht-leere Folge von Worten aus $L(F)$
7.	F^*	beliebig lange Folge von Worten aus $L(F)$
8.	F^n	Folge von n Worten aus $L(F)$

Exkurs: reguläre Ausdrücke

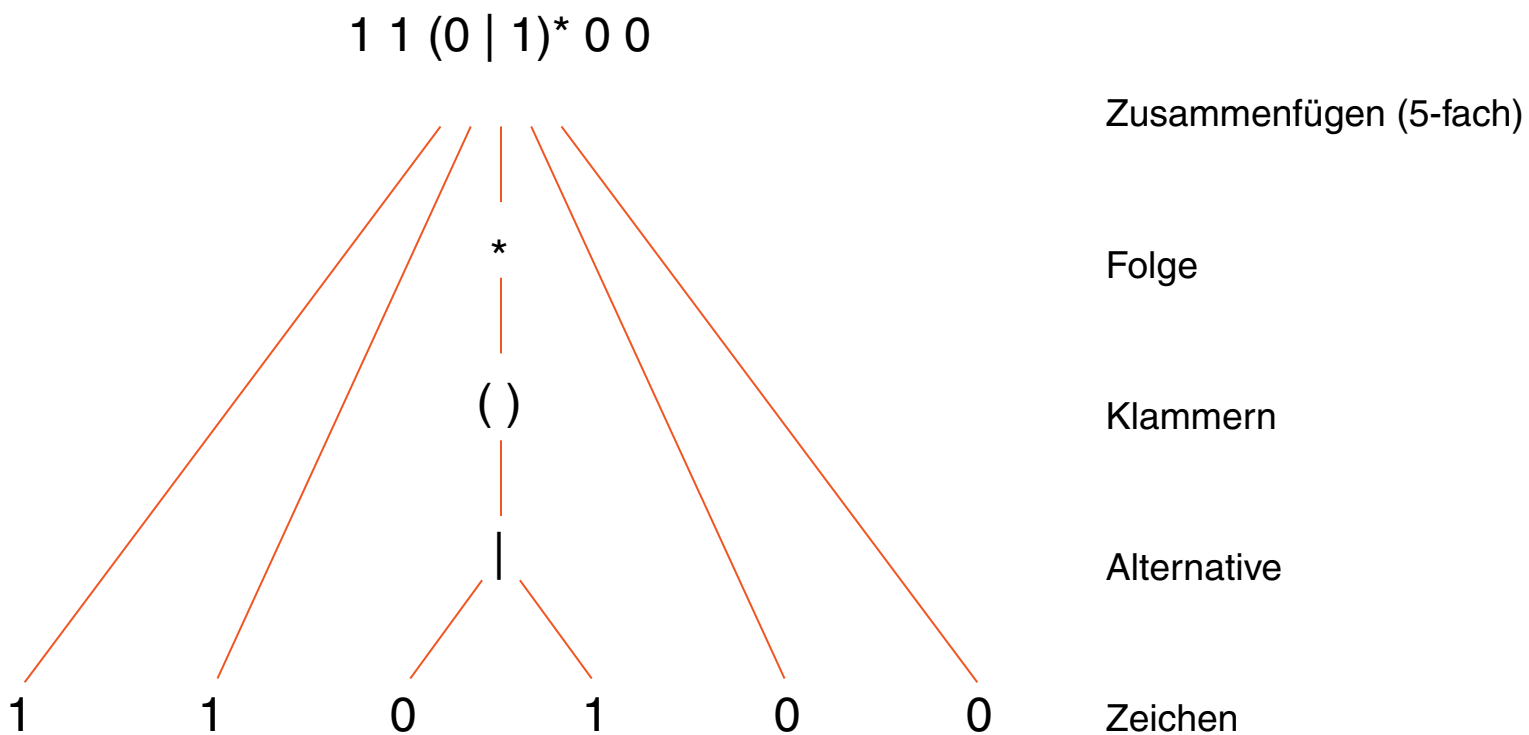
Konstruktion [Kastens 2005] [PHP]

Ein regulärer Ausdruck R kann wie folgt rekursiv zusammengesetzt sein. F und G bezeichnen gegebene reguläre Ausdrücke.

	R	Sprache $L(R)$	Erklärung
1.	a	$\{a\}$	das Zeichen a
2.	FG	$\{fg \mid f \in L(F),\ g \in L(G)\}$	Zusammenfügen von zwei Worten
3.	$F \mid G$	$\{f \mid f \in L(F)\} \cup \{g \mid g \in L(G)\}$	Alternativen
4.	ε	$\{\varepsilon\}$	das leere Wort
5.	(F)	$(L(F))$	Klammerung
6.	F^+	$\{f_1 f_2 \ldots f_n \mid f_i \in L(F),\ n \geq 1,\ i = 1, \ldots, n\}$	nicht-leere Folge von Worten aus $L(F)$
7.	F^*	$\{\varepsilon\} \cup L(F^+)$	beliebig lange Folge von Worten aus $L(F)$
8.	F^n	$\{f_1 f_2 \ldots f_n \mid f_i \in L(F),\ i = 1, \ldots, n\}$	Folge von n Worten aus $L(F)$

Exkurs: reguläre Ausdrücke

Illustration [Kastens 2005]



Jedes Wort aus der Sprache dieses regulären Ausdrucks besteht aus zwei Einsen, gefolgt von beliebig vielen Nullen oder Einsen, gefolgt von zwei Nullen.

Exkurs: reguläre Ausdrücke

Beispiele [Kastens 2005]

R	**Name der Sprache** $L(R)$	**Worte aus** $L(R)$
(a | b) (c | d | ε)	*Abc*	ac, bc, ad, bd, a, b
Sehr geehrte(r | ε) (Frau | Herr)	*Anrede*	Sehr geehrte Frau
0 | 1 | . . . | 9	*Digit*	7
a | b | . . . | z	*sLetter*	x
A | B | . . . | Z	*cLetter*	B
sLetter | *cLetter*	*Letter*	m, N
Letter (*Letter* | *Digit*)*	*Bezeichner*	Maximum, min7, a
Digit $^{+}$.*Digit* ^{2}	*GeldBetrag*	23.95, 0.50
(*cLetter* | *cLetter* ^{2} | *cLetter* ^{3})– (*cLetter* | *cLetter* ^{2})– (*Digit* | *Digit* ^{2} | *Digit* ^{3} | *Digit* ^{4})	*KFZ*	KR–AX–123
1^3 (1 | 0)* 0^3	*Dual*	1111000, 1111101010000

Exkurs: reguläre Ausdrücke

Spezifikation von Textmustern

Ein wichtiger Einsatz von regulären Ausdrücken in Sprachen, die zur Textverarbeitung eingesetzt werden, ist die Spezifikation von Textmustern.

Beispiel: Darstellung aller Dateinamen der Form

„webtecI(0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9)^3.html“

- Unix-Shell.

```
ls webtecI[0-9][0-9][0-9].html
```

- PHP.

```
$d = "[0-9]";
preg_match("/webtecI$d$d$d\.html/", $files)
```

Bemerkungen:

- Wenn Namen von regulären Ausdrücken in anderen regulären Ausdrücken verwendet werden, müssen sie als Teil der Meta-Sprache kenntlich gemacht werden.
  Hier: Verwendung der kursiven Schreibweise.
- Jede Skriptsprache zur Textverarbeitung verwendet eine andere Syntax zur Spezifikation regulärer Ausdrücke; die Konstruktionsprinzipien und die Mächtigkeit sind vergleichbar.
- Die Spezifikation regulärer Ausdrücke in PHP ist aus der Skriptsprache Perl übernommen.

PHP Hypertext Preprocessor

PHP Hypertext Preprocessor

Einführung [Einordnung]

Charakteristika:

- wie JSP: dokumentenzentrierte (HTML) Programmierung
- prozedurale Sprache mit objektorientierten Erweiterungen
- wenige einfache Typen, dynamisch typisiert
- Notation an C und Perl orientiert
- große Funktionsbibliothek
- umfassende Datenbankunterstützung
- Open Source

Anwendung:

- kleine private bis mittelgroße kommerzielle Projekte
- Schwerpunkt auf Datenbanken

PHP Hypertext Preprocessor

Einführung (Fortsetzung)

PHP-Code wird in die HTML-Datei eingebettet:

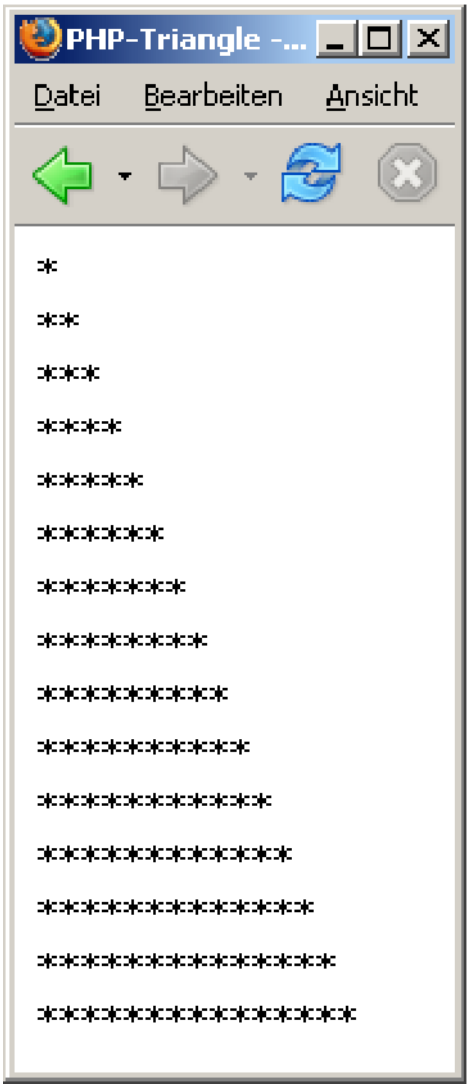
```
<!DOCTYPE HTML>
<html>
  <head> <title>PHP Dreieck</title> </head>
  <body>
    <?php
      $line = 1;
      while ($line < 16) {
        $col = 1;
        while ($col <= $line) {
          echo "*";
          $col = $col + 1;
        }
        echo "<br>\n";
        $line = $line + 1;
      }
    ?>
  </body>
</html>
```

PHP Hypertext Preprocessor

Einführung (Fortsetzung)

PHP-Code wird in die HTML-Datei eingebettet:

```
<!DOCTYPE HTML>
<html>
  <head> <title>PHP Dreieck</title> </head>
  <body>
    <?php
      $line = 1;
      while ($line < 16) {
        $col = 1;
        while ($col <= $line) {
          echo "*";
          $col = $col + 1;
        }
        echo "<br>\n";
        $line = $line + 1;
      }
    ?>
  </body>
</html>
```



[Demo]

PHP Hypertext Preprocessor

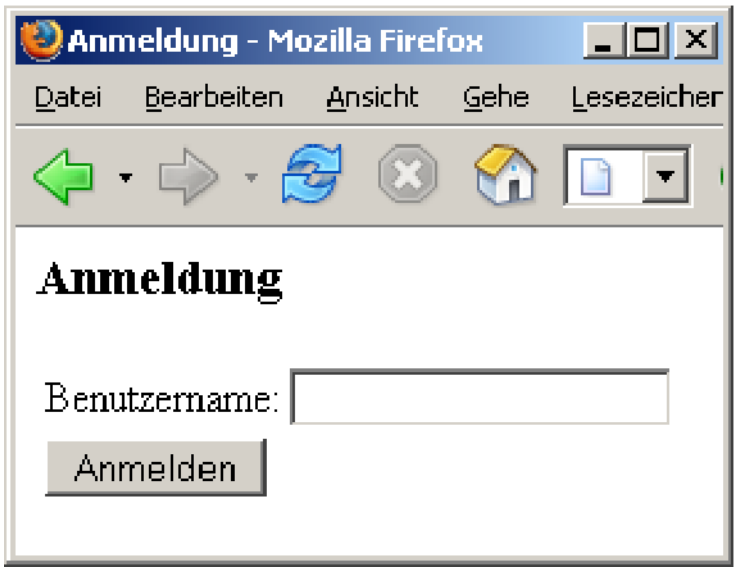
Einführung (Fortsetzung)

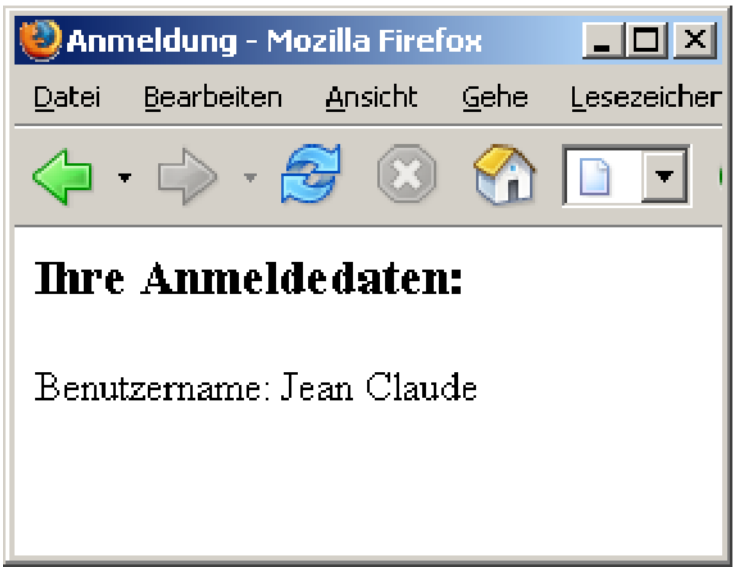
```
<!DOCTYPE HTML>
<html>
  <head> <title>Registration</title> </head>
  <body>
    <?php
      $user = $_REQUEST['user'];
      if(trim($user) == ""){
    ?>
    <form action="registration.php" method="get">
      <table>
        <tr><td>Benutzername:</td><td><input type="text"
            name="user"></td></tr>
        <tr><td><input type="submit" value="Anmelden"></td></tr>
      </table>
    </form>
    <?php } else { ?>
    Benutzername: <?php echo $user ?>
    <?php } ?>
  </body>
</html>
```

Vergleiche hierzu die JSP-Realisierung.

PHP Hypertext Preprocessor

Einführung (Fortsetzung)

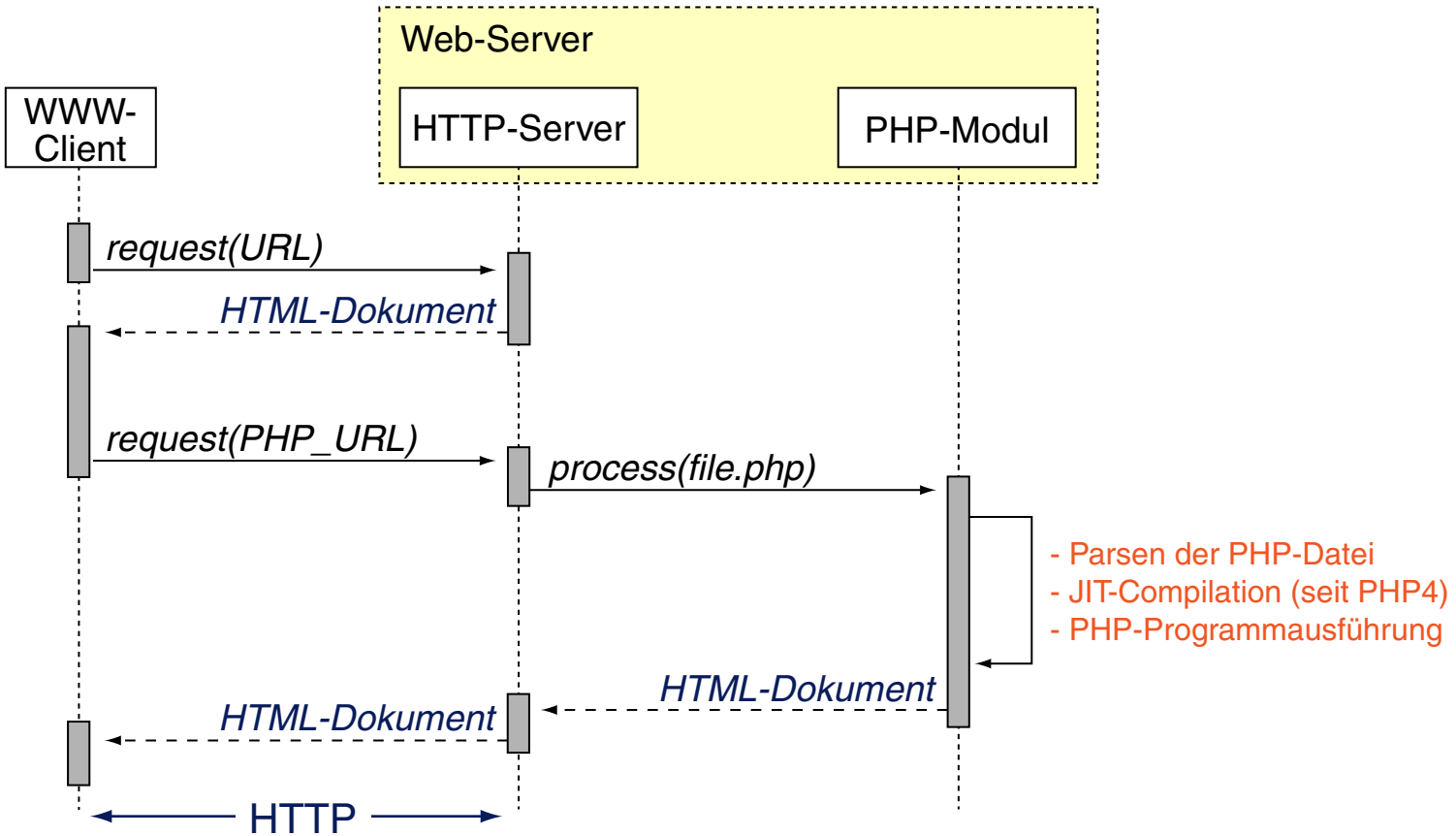




[Demo]

PHP Hypertext Preprocessor

Ablauf einer PHP-Interaktion



Vergleiche hierzu den Ablauf einer JSP-Interaktion.

Bemerkungen:

- Die `.php`-Datei kombiniert HTML und PHP-Code im Sinne eines Programms zur Erzeugung einer HTML-Seite.
- Das Action-Attribut `<... action="registration.php">` ist laut Referenz obligatorisch. Falls es fehlt, wird als Default-Wert die Datei (hier: `registration.php`) selbst genommen.
- Wie bei JSP gibt es auch hier kein isoliertes CGI-Programm mehr: HTML-Form, HTML-Antwort, Programm zur Verarbeitung – alles befindet sich in derselben Datei.

PHP Hypertext Preprocessor

Historie [php.net]

1994 Rasmus Lerdorf entwickelt erste Version der „Personal Home Page Tools“ PHP zur Erfassung der Zugriffe auf seine Web-Seiten.

1995 PHP/FI 1. Neuer Parser und Erweiterung auf HTML-Forms (FI = Form Interpreter).

1996 PHP/FI 2. Open Source Gemeinde steigt in die Weiterentwicklung ein.

1997 PHP 3. Neuentwicklung unter Leitung von Andi Gutmans und Zeev Suraski. Konsistente Syntax, objektorientierte Konzepte. Wird von mehr als 50.000 Entwicklern verwendet. Umbenennung in „PHP Hypertext Preprocessor“.

1999 Gründung von Zend ([Ze]ev Suraski und A[nd]i Gutmanns) Technologies. [zend.com]

2000 PHP 4. Leistungsfähiger Parser (Zend-Engine), modularer Basiscode, HTTP-Sessions, Ausgabepufferung, sicherere Benutzereingaben, umfangreiche Datenbankunterstützung.

2004 PHP 5. Zend Engine 2.0, neues Objektmodell mit public-/private-/protected-Modifizierer, deutlich verbesserte Unterstützung der XML-Konzepten DOM, SAX, XSLT.

2009 PHP 5.3. Namespaces, Late Static Bindings, Closures und Lambda-Kalkül.

2013 Aktuelle Version von PHP: [php.net]
Statistiken zur Verbreitung: [php.net] [tiobe.com]

PHP Hypertext Preprocessor

Einbindung in HTML-Dokumente

Der PHP-Prozessor erhält das gesamte HTML-Dokument, interpretiert aber nur die Anweisungen, die als PHP-Code ausgezeichnet sind. Der übrige Text wird unverändert zum Client gesendet.

Syntaxalternativen zur Auszeichnung:

1. Standard-Tags:

   ```
   <?php ... ?>
   ```

2. Sprachspezifische Script-Deklaration:

   ```
   <script language="php"> ... </script>
   ```

3. ASP-Stil:

   ```
   <% ... %>
   ```

4. Kurzschreibweise einer SGML-Verarbeitungsanweisung:

   ```
   <? ... ?>
   ```

Bemerkungen:

- Die Kurzschreibweisen mit „<%“ bzw. „<?“ müssen in der Konfiguration von PHP aktiviert sein. Aus Sicht der Portabilität von PHP-Dateien sind sie nicht sinnvoll; insbesondere wird „<?“ in Zukunft nicht mehr unterstützt. [php.net]
- Es ist eine Frage des Stils, ob PHP-Code in mehrere Abschnitte aufgeteilt wird, zwischen denen HTML-Code steht, oder ob HTML-Code durch die PHP-Funktion `echo()` ausgegeben wird. Die erste Variante ist performanter.
- `echo()` ist keine (Built-in-)Funktion sondern ein Element der Sprache und wird hinsichtlich ihrer Argumente besonders behandelt. [php.net]

PHP Hypertext Preprocessor

Grundlagen der Syntax

Bezeichner [php.net]

- der Grundaufbau von Namen folgt der Form `[a-zA-Z_][a-zA-Z_0-9]*`
- Variablennamen beginnen immer mit `$`.
  Groß-/Kleinschreibung wird unterschieden *(case sensitive)*.
- Konstantennamen werden ohne `$` geschrieben.
- Funktionsnamen sind *case insensitive*.

PHP Hypertext Preprocessor

Grundlagen der Syntax

Bezeichner [php.net]

- der Grundaufbau von Namen folgt der Form `[a-zA-Z_][a-zA-Z_0-9]*`
- Variablennamen beginnen immer mit `$`.
  Groß-/Kleinschreibung wird unterschieden *(case sensitive)*.
- Konstantennamen werden ohne `$` geschrieben.
- Funktionsnamen sind *case insensitive*.

Anweisungen [php.net]

- Jede Anweisung wird mit einem Semikolon beendet; auch der schließende Tag „`?>`“ beendet eine Anweisung.

```
<?php
  echo "Hello world!";   ≈   <?php echo "Hello word!" ?>
?>
```

- `//` kommentiert bis Zeilenende aus.
- Balancierte Kommentarklammerung: `/*` *Kommentar* `*/`

PHP Hypertext Preprocessor

Variablen [JavaScript]

- Variablen werden durch ihre Initialisierung definiert.
- Eine Variable kann Werte beliebigen Typs annehmen.
- Es wird zwischen lokalen, globalen, statischen und vordefinierten (superglobalen) Variablen unterschieden. [php.net]
- Eine Variable ist global, wenn Sie außerhalb des Bindungsbereiches einer Funktion steht.
- Globale und vordefinierte Variablen gelten im ganzen Programm. Im Bindungsbereich einer Funktion sind globale Variablen mit dem Schlüsselwort `global` sichtbar zu machen; vordefinierte Variablen sind immer sichtbar.
- Statische Variablen existieren nur im Bindungsbereich einer Funktion; ihr Wert geht beim Verlassen dieses Bereichs nicht verloren.
- Der Gültigkeitsbereich von Konstanten entspricht dem von vordefinierten Variablen. Konstanten werden durch Funktion `define()` definiert.

PHP Hypertext Preprocessor

Variablen: Illustration von Geltungsbereichen

```
<?php
  $a = 1;     // globaler Bereich

  function test() {
    echo $a; // Referenz auf den Bindungsbereich von test()
  }

  test();
?>
```

PHP Hypertext Preprocessor

Variablen: Illustration von Geltungsbereichen

```php
<?php
  $a = 1;     // globaler Bereich

  function test() {
    echo $a; // Referenz auf den Bindungsbereich von test()
  }

  test();
?>
```

```php
<?php
  $a = 1;
  $b = 2;

  function Summe() {
    global $a, $b;
    $b = $a + $b;
  }

  Summe();
  echo $b;
?>
```

PHP Hypertext Preprocessor

Variablen: Illustration statischer Variablen

```
<?php
  function fak($n) {
    static $m = 1; // Initialisierung der statischen Variable (einmalig)
    if ($n == 1) {
      echo $m;      // Ausgabe des Ergebnisses
      $m=1;         // Zurücksetzen der statischen Variable
    }
    else {
      $m *= $n;
      fak(--$n);
    }
  }

  fak(10);
?>
```

PHP Hypertext Preprocessor

Variablen: besondere Konzepte

- variable Variablen: `$$`*VarName*
  Der Inhalt von `$`*VarName* wird als Variablenname verwendet.

- Referenzen [php.net] : `$`*VarName2* `= &$`*VarName1*
  Neuer (Alias)Variablenname für den Inhalt von `$`*VarName1*.

- Überprüfung, ob eine Variable definiert ist:
  `boolean isset($`*VarName*`)`

- Löschen einer Variablen:
  `void unset($`*VarName*`)`

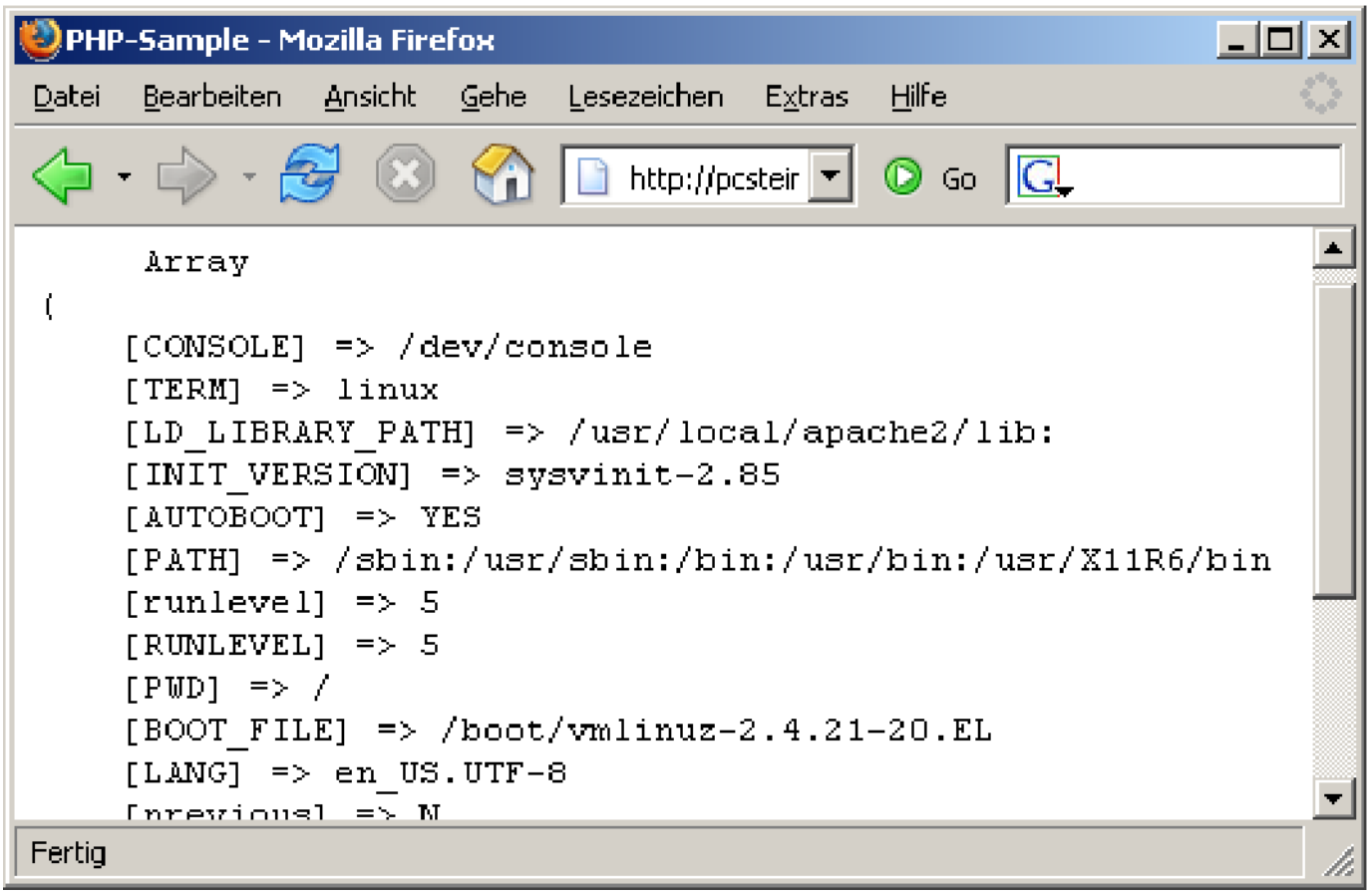
- Zeigt Informationen über eine Variable in lesbarer Form an:
  `boolean print_r($`*VarName*`)`

- Die vordefinierten Variablen (superglobale Arrays) [php.net] :
  `$GLOBALS, $_SERVER, $_GET, $_POST, $_FILES, $_COOKIE, $_SESSION, $_REQUEST, $_ENV`

PHP Hypertext Preprocessor

Variablen: besondere Konzepte (Fortsetzung)

```
<pre>
  <?php print_r($_REQUEST); ?>
  <?php print_r($_ENV); ?>
</pre>
```



[Demo]

Bemerkungen:

- Vordefinierte Variablen können nicht als variable Variablen verwendet werden. [php.net]
- Referenzen sind ein Mechanismus, um verschiedene Namen für den gleichen Inhalt von Variablen zu ermöglichen. Sie sind nicht mit Zeigern in C zu vergleichen, sondern Alias-Definitionen in der Symboltabelle: der gleiche Variableninhalt kann unterschiedliche Namen besitzen, ähnlich dem Konzept der Hardlinks im Unix-Dateisystem. [php.net]

PHP Hypertext Preprocessor

Datentypen: Primitive [JavaScript]

`integer, float`

- Ganzzahlen können dezimal, hexadezimal oder oktal notiert werden. Bei Überlauf findet eine Konvertierung nach `float` statt.

`string` [php.net]

- Einfache Anführungszeichen. Alle Zeichen stehen für sich selbst; nur `'` muss „escaped“ werden: `\'`
- Doppelte Anführungszeichen. Der String wird geparst und eventuell vorkommende Variablen und Escape-Folgen ersetzt. Beispiel:

  ```
  "Summe $jahr = \t${Betrag}EUR"
  ```

- Konkatenation mit Punkt: `"Hello" . "world!"`
- Ausgabe von Ausdrücken, deren Rückgabewert eine Zeichenkette ist:

`echo`	durch Kommata getrennte Folge von Ausdrücken
`print`	einzelner Ausdruck

PHP Hypertext Preprocessor

Datentypen: Primitive (Fortsetzung)

`boolean`

- Literale: `true` und `false`.
  Groß-/Kleinschreibung wird nicht unterschieden.
- Operatoren: Konjunktion `&&` bzw. `and`, Disjunktion `||` bzw. `or`, Negation `!` und Exklusiv-Oder `xor`.

`NULL`

- Der spezielle Wert `NULL` steht dafür, dass eine Variable keinen Wert hat.
- `NULL` ist der einzig mögliche Wert des Typs `NULL`. Groß-/Kleinschreibung wird nicht unterschieden.
- Eine Variable wird als `NULL` interpretiert, wenn

  (a) ihr die Konstannte `NULL` als Wert zugewiesen wurde,
  (b) ihr bis jetzt kein Wert zugewiesen wurde, oder
  (c) sie mit `unset()` gelöscht wurde.

PHP Hypertext Preprocessor

Datentypen: Arrays [JavaScript]

Ein Array ist eine Abbildung von Indizes auf Werte. Jedes Element eines Arrays ist ein Paar bestehend aus numerischem oder String-Index und zugeordnetem Wert.

Erzeugung von Arrays:

(a) Mit der `array()`-Funktion als Liste von Werten, indiziert von 0 an:

(b) Durch explizite Indizierung:

(c) Mit der `array()`-Funktion als Liste von Index-Wert-Paaren oder als assoziatives Array:

PHP Hypertext Preprocessor

Datentypen: Arrays [JavaScript]

Ein Array ist eine Abbildung von Indizes auf Werte. Jedes Element eines Arrays ist ein Paar bestehend aus numerischem oder String-Index und zugeordnetem Wert.

Erzeugung von Arrays:

(a) Mit der `array()`-Funktion als Liste von Werten, indiziert von 0 an:

```
$monatsName = array("", "Jan", ..., "Dez");
```

(b) Durch explizite Indizierung:

```
$monatsName[1]= "Jan"; monatsName[2]= "Feb"; ...
```

(c) Mit der `array()`-Funktion als Liste von Index-Wert-Paaren oder als assoziatives Array:

```
$monatsName = array(1 => "Jan", ..., 12 => "Dez");
$monatsName = array("Jan" => 1, ..., "Dez" => 12);
```

PHP Hypertext Preprocessor

Datentypen: Arrays [JavaScript]

Ein Array ist eine Abbildung von Indizes auf Werte. Jedes Element eines Arrays ist ein Paar bestehend aus numerischem oder String-Index und zugeordnetem Wert.

Erzeugung von Arrays:

(a) Mit der `array()`-Funktion als Liste von Werten, indiziert von 0 an:

```
$monatsName = array("", "Jan", ..., "Dez");
```

(b) Durch explizite Indizierung:

```
$monatsName[1]= "Jan"; monatsName[2]= "Feb"; ...
```

(c) Mit der `array()`-Funktion als Liste von Index-Wert-Paaren oder als assoziatives Array:

```
$monatsName = array(1 => "Jan", ..., 12 => "Dez");
$monatsName = array("Jan" => 1, ..., "Dez" => 12);
```

Aufzählung aller Elemente mit Schlüssel:

```
foreach ($monatsName as $key => $value) {
  echo "Schlüssel-Wert-Paar: " . $key . "=>" . $value . "<br/>";
}
```

PHP Hypertext Preprocessor

Datentypen: Konversion

Ein Wert eines Typs wird in einen „entsprechenden“ Wert eines anderen Typs umgewandelt.

- explizite Konversion *(Type Cast)*
  Der Zieltyp, in den der Wert eines Ausdruckes umgewandelt werden soll, wird explizit angegeben. Beispiel:

  `(string)(5+1)` liefert die Zeichenreihe `"6"`

- implizite Konversion *(Coercion)*
  Wenn der Typ eines Wertes nicht zu der darauf angewandten Operation passt, wird versucht, den Typ anzupassen. Beispiel:

  ```
  $sum = 42;
  print "Summe = " . $sum;
  ```

  Die ganze Zahl `42` wird in die Zeichenreihe `"42"` konvertiert. Der Wert der Variablen `$sum` bleibt unverändert.

PHP Hypertext Preprocessor

Kontrollstrukturen [JavaScript]

- Anweisungsfolge:
- Bedingte Anweisung:
- `while`-Schleife:
- `do-while`-Schleife:
- `for`-Schleife:

PHP Hypertext Preprocessor

Kontrollstrukturen [JavaScript]

- Anweisungsfolge:

```
{ $i = $i+1; print "Hello world!"; }
```

- Bedingte Anweisung:

```
if ($a < $b) {$min = $a;} else {$min = $b;}
```

  Bei einzelnen Anweisungen sind die {}-Klammern optional.

- `while`-Schleife:

- `do-while`-Schleife:

- `for`-Schleife:

PHP Hypertext Preprocessor

Kontrollstrukturen [JavaScript]

- Anweisungsfolge:

```
{ $i = $i+1; print "Hello world!"; }
```

- Bedingte Anweisung:

```
if ($a < $b) {$min = $a;} else {$min = $b;}
```

Bei einzelnen Anweisungen sind die {}-Klammern optional.

- `while`-Schleife:

```
$i = 0; while ($i < 42) {$stars = $stars . "*"; $i = $i+1;}
```

- `do-while`-Schleife:

```
$i = 0; do {$stars = $stars . "*"; $i = $i+1;} while ($i < 42);
```

- `for`-Schleife:

```
for ($i = 0; $i < 12; $i++) {echo $i, $monatsName[$i], "\n";}
```

PHP Hypertext Preprocessor

Kontrollstrukturen [JavaScript]

- Anweisungsfolge:

```
{ $i = $i+1; print "Hello world!"; }
```

- Bedingte Anweisung:

```
if ($a < $b) {$min = $a;} else {$min = $b;}
```

Bei einzelnen Anweisungen sind die {}-Klammern optional.

- `while`-Schleife:

```
$i = 0; while ($i < 42) {$stars = $stars . "*"; $i = $i+1;}
```

- `do-while`-Schleife:

```
$i = 0; do {$stars = $stars . "*"; $i = $i+1;} while ($i < 42);
```

- `for`-Schleife:

```
for ($i = 0; $i < 12; $i++) {echo $i, $monatsName[$i], "\n";}
```

- Parameterübergabe standardmäßig mittels call-by-value.

Bemerkungen:

- call-by-value: Der formale Parameter (Funktionsdefinition) ist eine Variable, die mit dem Wert des aktuellen Parameters (Funktionsaufruf) initialisiert wird.
- Notiert man ein „`&`“ vor dem formalen Parameter (Funktionsdefinition) oder vor dem aktuellen Parameter (Funktionsaufruf), geschieht die Übergabe für diesen Parameter durch call-by-reference. [php.net]

PHP Hypertext Preprocessor

Funktionsbibliothek [JavaScript]

Es existiert eine große, ausgereifte Funktionsbibliothek (> 700 Funktionen), die in jedem PHP-Programm zur Verfügung steht. Beispiele:

- Arrays
- Protokolle
- Datenbanken
- Datum/Uhrzeit
- Dateiverzeichnisse
- Dateien
- Grafik
- HTTP
- IMAP
- LDAP
- Mathematik
- MCAL
- Mcrypt
- Mhash
- PDF
- POSIX
- reguläre Ausdrücke
- Strings
- Variablenmanipulation
- XML

PHP Hypertext Preprocessor

Funktionsbibliothek: reguläre Ausdrücke [php.net]

Es gibt zwei „Engines“ zum Verarbeiten regulärer Ausdrücke:

1. POSIX-Engine.
2. PCRE-Engine. Perl-kompatibel und 200x schneller als POSIX-Engine.

Aufbau des Perl-kompatiblen Strings zur Definition regulärer Ausdrücke [php.net] :

"*Delimiter Regular_Expression Delimiter* [*Modifiers*]"

- Der Delimiter muss ein nicht-alphanumerisches Zeichen sein.
- Optionale Modifizierer beeinflussen die Match-Strategie. [php.net]

Beispiele:

```
echo preg_match("/def/", "defabcdef");

echo preg_match("=def=", "defabcdef");
```

PHP Hypertext Preprocessor

Funktionsbibliothek: reguläre Ausdrücke (Fortsetzung)

- `int preg_match(string` *Pattern*`, string` *String*
  [, `array` *Matches* [, `PREG_OFFSET_CAPTURE` [, `int` *Offset*]]])
  Durchsucht *String* nach der ersten Übereinstimmung mit dem durch *Pattern* definierten regulären Ausdruck. [php.net]

- `int preg_match_all(...)`
  Sucht nach allen Übereinstimmungen. [php.net]

- `mixed preg_replace(...)`
  Suchen und Ersetzen von Übereinstimmungen. [php.net]

- `mixed preg_replace_callback(...)`
  Suchen und Ersetzen von Übereinstimmungen, wobei der Return-Wert einer Callback-Funktion den Ersetzungstext bestimmt. [php.net]

PHP Hypertext Preprocessor

Funktionsbibliothek: reguläre Ausdrücke (Fortsetzung) [Exkurs]

Ein regulärer Ausdruck R kann wie folgt rekursiv zusammengesetzt sein. F, G bzw. $L(F)$, $L(G)$ bezeichnen reguläre Ausdrücke sowie die definierten Sprachen.

R	**Erklärung**
a	das Zeichen a
FG	Zusammenfügen von zwei Worten
$F \mid G$	Alternativen
$F?$	F ist optional (gleiche Semantik wie $F \mid \varepsilon$)
(F)	Klammerung
$F+$	nicht-leere Folge von Worten aus $L(F)$
$F*$	beliebig lange Folge von Worten aus $L(F)$
$F\{n\}$	Folge von n Worten aus $L(F)$
$F\{m,n\}$	Folge mit mindestens m und höchstens n von Worten aus $L(F)$
$[abc]$	alternativ ein Zeichen aus der Klammer
$[\hat{}abc]$	alternativ ein anderes Zeichen als die in der Klammer
$[a-zA-Z]$	alternativ ein Zeichen aus Zeichenbereichen
.	beliebiges Zeichen
^	Anfang der Zeichenfolge (nichts darf vorangehen)
$	Ende der Zeichenfolge (nichts darf darauf folgen)

PHP Hypertext Preprocessor

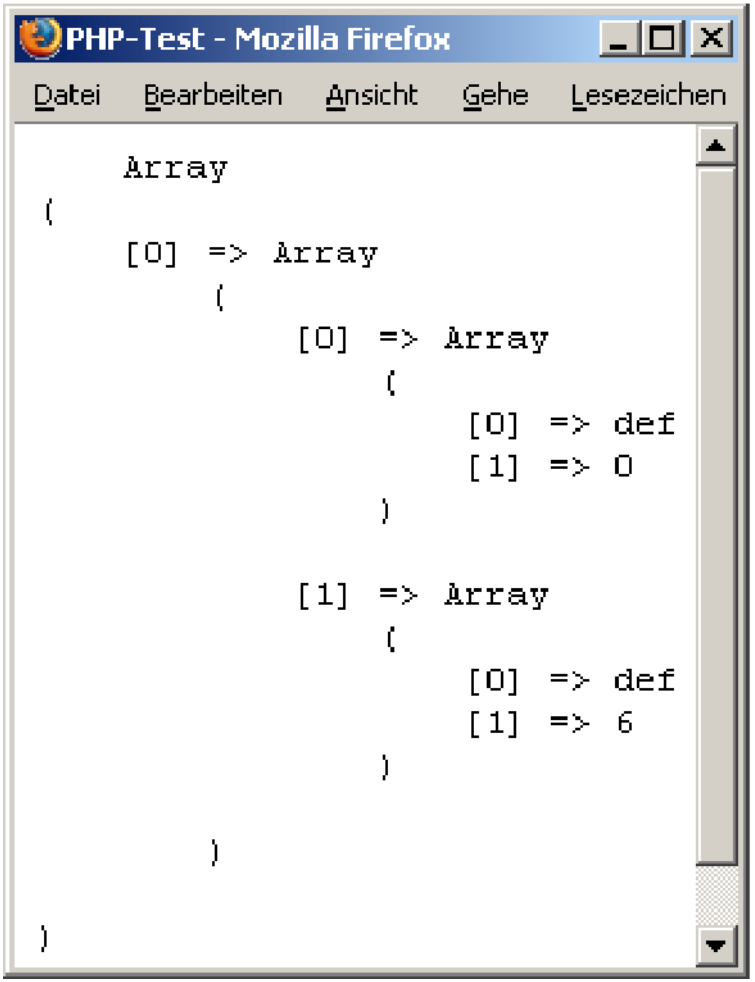
Funktionsbibliothek: reguläre Ausdrücke (Fortsetzung)

```
<?php
  $pattern = "/d[a-z]f/";
  $str = "defabcdef";
  $num = preg_match_all($pattern, $str,
    $matches, PREG_OFFSET_CAPTURE);
?>

<pre>
  <?php print_r($matches); ?>
</pre>
```

PHP Hypertext Preprocessor

Funktionsbibliothek: reguläre Ausdrücke (Fortsetzung)

```
<?php
  $pattern = "/d[a-z]f/";
  $str = "defabcdef";
  $num = preg_match_all($pattern, $str,
    $matches, PREG_OFFSET_CAPTURE);
?>

<pre>
  <?php print_r($matches); ?>
</pre>
```

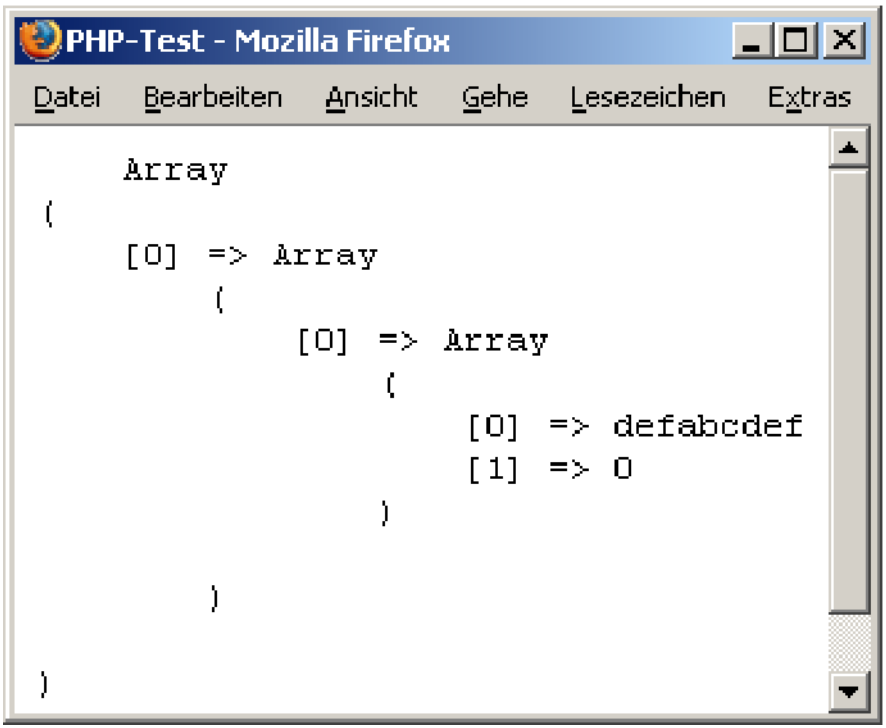


[Demo]

PHP Hypertext Preprocessor

Funktionsbibliothek: reguläre Ausdrücke (Fortsetzung)

```
<?php
  $pattern = "/d[a-z]*f/";
  $str = "defabcdef";
  $num = preg_match_all($pattern, $str,
    $matches, PREG_OFFSET_CAPTURE);
?>

<pre>
  <?php print_r($matches); ?>
</pre>
```

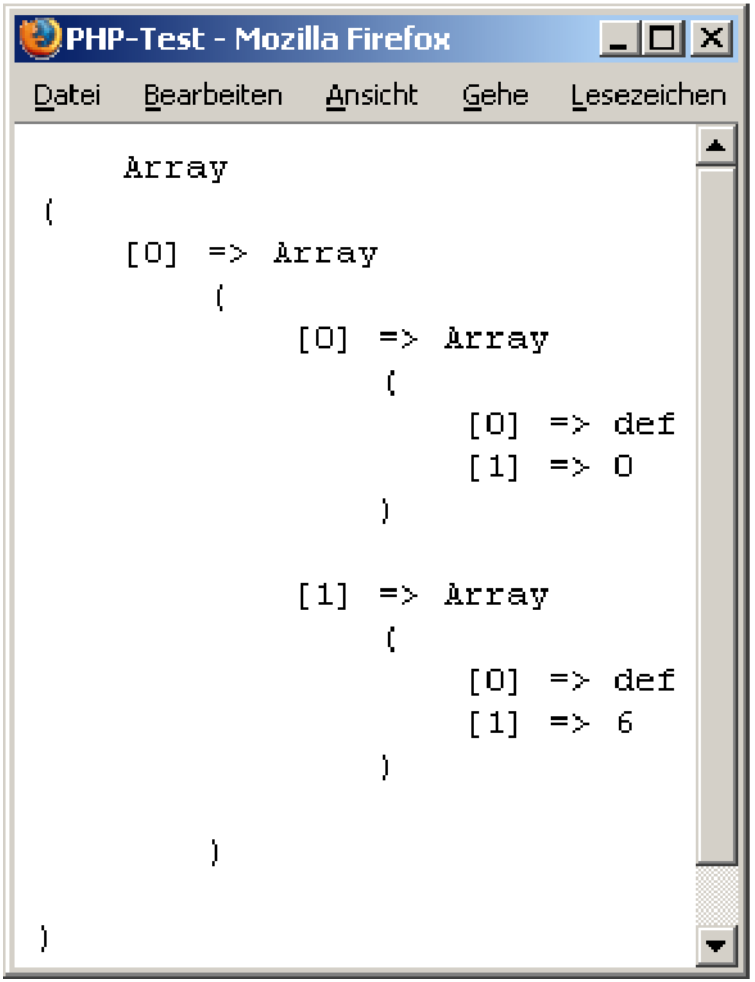


[Demo]

PHP Hypertext Preprocessor

Funktionsbibliothek: reguläre Ausdrücke (Fortsetzung)

```
<?php
  $pattern = "/d[a-z]*f/U";
  $str = "defabcdef";
  $num = preg_match_all($pattern, $str,
    $matches, PREG_OFFSET_CAPTURE);
?>

<pre>
  <?php print_r($matches); ?>
</pre>
```



[Demo]

Bemerkungen:

- Innerhalb eines regulären Ausdrucks fungiert „\“ als Escape-Zeichen.
- Die Standardeinstellung für die Match-Bildung ist „gierig“ (*greedy*); hierbei wird der längste Match gesucht. Mit dem Ungreedy-Modifizierer „U“ wird in den Modus „nicht gierig“ umgeschaltet. [php.net]
- Unter Linux können reguläre Ausdrücke mit dem Shell-Programm `/usr/bin/pcretest` interaktiv getestet werden.

PHP Hypertext Preprocessor

Funktionsbibliothek: Datenbanken

Die PHP-Datenbankschnittstellen ermöglichen es, dynamisch HTML-Seiten basierend auf Datenbankinhalten zu generieren. Beispiele:

- Auswahl und Anzeige von Inhalten aus einer Produktdatenbank
- Benutzerverwaltung wie Abgleich von Benutzernamen und Passworten

Funktionalität der PHP-Datenbankschnittstellen:

- Erstellung und Verwaltung von Verbindungen zu Datenbank-Servern
- Auswahl von Datenbanken
- Generierung von SQL-Ausdrücken
- Zugriff auf die Ergebnisse von SQL-Anfragen

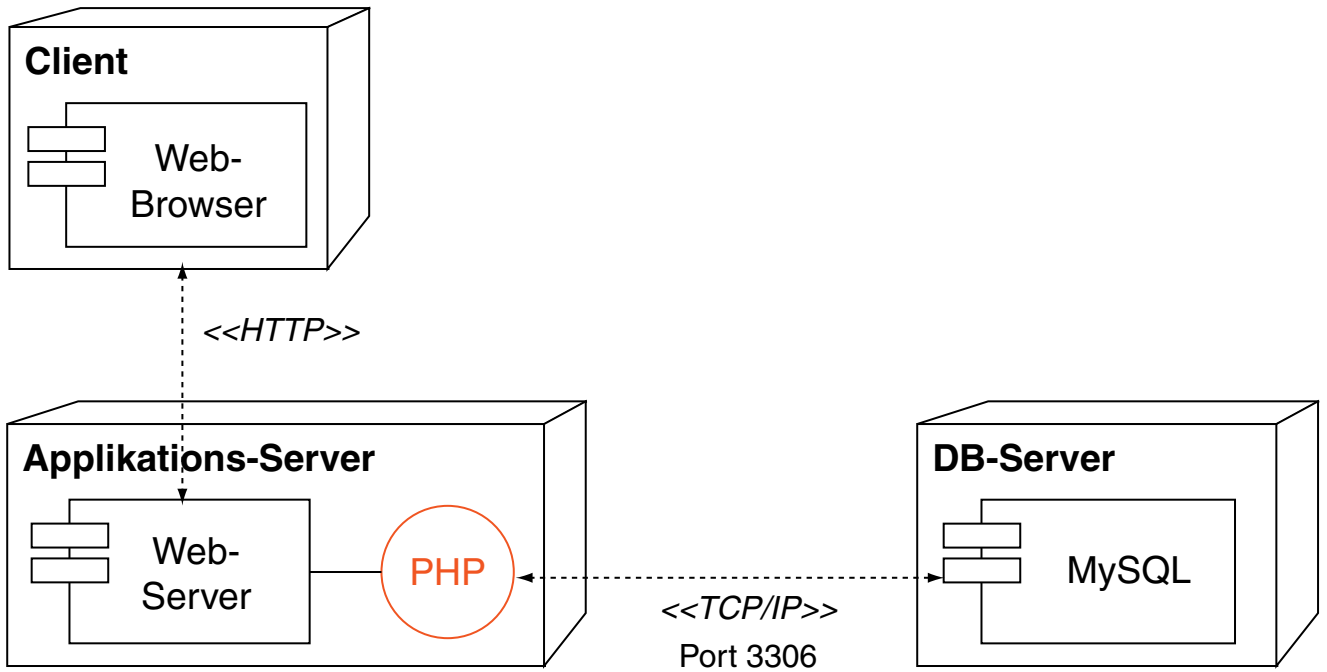
Zahlreiche Datenbanken werden unterstüzt, u.a.: dBase-kompatible Formate, Berkeley-DB-Formate, Oracle, Sybase, MySQL, ODBC-Datenquellen. [php.net]

Bemerkungen:

- Eine weit verbreitete Software-Kombination zur Erstellung Web-basierter Datenbankanwendungen ist unter dem Schlagwort „Lamp“ bzw. „Wamp“ bekannt: Linux bzw. Windows + Apache + MySQL + PHP.

PHP Hypertext Preprocessor

Funktionsbibliothek: Datenbanken (Fortsetzung)



Beispielszenario:

- Auf einem Web-Server liegt eine Kundendatenbank, u.a. mit der `addresses`-Relation.
- Aufgabe: Realisierung eines Web-Interfaces zur Suche, Sortierung und Auflistung von Kundendaten.

PHP Hypertext Preprocessor

Funktionsbibliothek: Datenbanken (Fortsetzung)

1. Mit Datenbank-Server verbinden:

```
$dbhost = "webis2.uni-weimar.de";
$dbuser ="peter";
$dbpassword = "secret";
$connection = mysql_connect($dbhost, $dbuser, $dbpassword)
    or die("Could not connect: " . mysql_error());
```

2. Datenbank auswählen:

```
$dbname = "customers";
mysql_select_db($dbname, $connection)
    or die("Could not select database");
```

PHP Hypertext Preprocessor

Funktionsbibliothek: Datenbanken (Fortsetzung)

1. Mit Datenbank-Server verbinden:

```
$dbhost = "webis2.uni-weimar.de";
$dbuser ="peter";
$dbpassword = "secret";
$connection = mysql_connect($dbhost, $dbuser, $dbpassword)
   or die("Could not connect: " . mysql_error());
```

2. Datenbank auswählen:

```
$dbname = "customers";
mysql_select_db($dbname, $connection)
   or die("Could not select database");
```

3. SQL-Ausdruck konstruieren und auswerten:

```
$table = "addresses";
$searchstring = $_REQUEST["searchstring"];

$query = "SELECT lastname, firstname, phone FROM $table
          WHERE lastname LIKE '%$searchstring%'
          ORDER BY lastname, firstname";

$result = mysql_query($query, $connection)
   or die("Query failed: " . mysql_error());
```

PHP Hypertext Preprocessor

Funktionsbibliothek: Datenbanken (Fortsetzung)

4. Anzahl von Ergebniszeilen prüfen:

```
if (mysql_num_rows($result) == 0) {
  echo "<h1>Kein Kunde mit dem Namen $searchstring
vorhanden.</h1>";
}
```

5. Datensätze aus assoziativem Array auslesen:

```
echo "<ol>";
while ($row = mysql_fetch_assoc($result)) {
  echo "<li>";
  echo $row["lastname"] . ", ";
  echo $row["firstname"] . " ";
  echo $row["phone"];
  echo "</li>";
}
echo "</ol>";
```

6. Speicher freigeben:

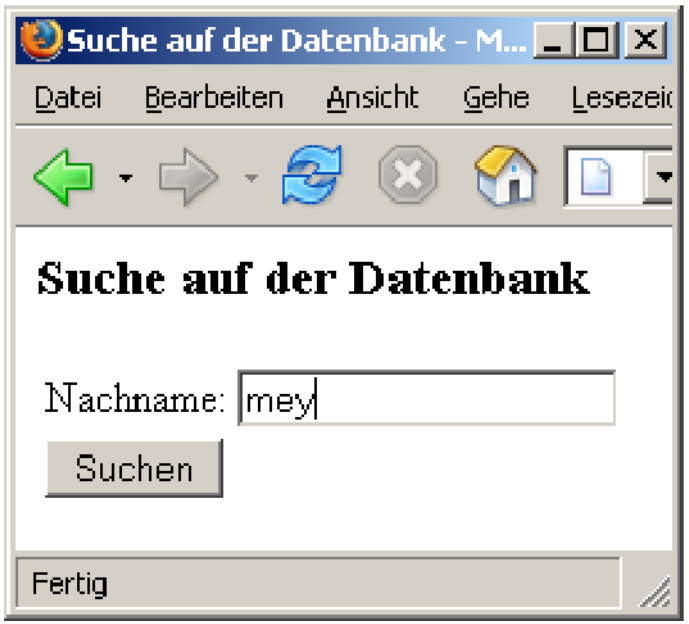
```
mysql_free_result($result);
```

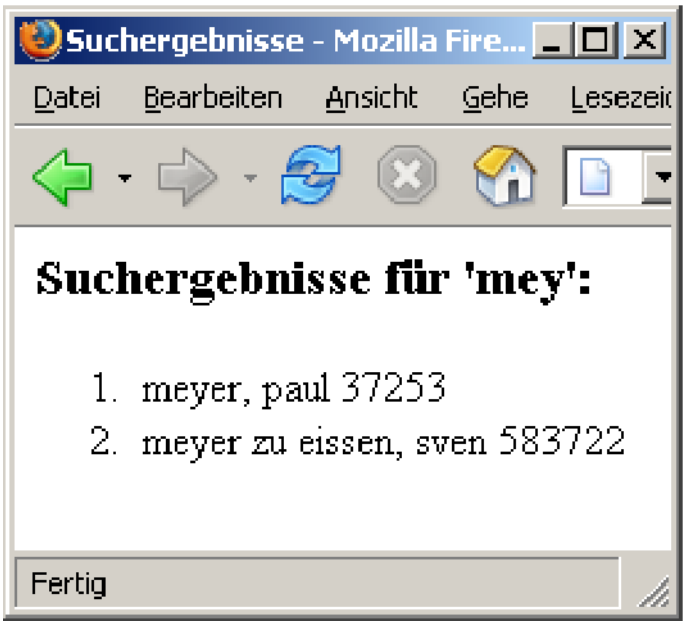
Bemerkungen:

- Wird `mysql_select_db` ohne `$connection`-Parameter aufgerufen, wird sich auf die zuletzt geöffnete Verbindung bezogen.
- `mysql_pconnect()`
  Variante zum Aufbau einer persistenten Verbindung.
- `die(string` *Message*`)`
  Ausgabe von *Message* und Beendung des aktuellen Skripts.
- `mysql_fetch_assoc(resource` *Result*`)`
  Gibt ein assoziatives Array zurück, das den nächsten Datensatz aus *Result* enthält. Sind keine weiteren Datensätze vorhanden, wird `FALSE` zurückgegeben.

PHP Hypertext Preprocessor

Funktionsbibliothek: Datenbanken (Fortsetzung)





[Demo]

Server-Technologien

Vergleich der Technologien [Einordnung]

	Programmier-aufwand	Anwendungen	Abhängigkeit vom Server	Abhängigkeit von Progammiersprache
CGI	mittel	allgemeiner Zugriff auf Informationsquellen	keine	klein
Web-Server-Bibliothek	hoch	komplexere, Performanz-kritische Anwendungen	hoch	meist C/C++
Servlet	mittel	"	mittel	nur Java
SSI	sehr gering	Wiederverwendung von kleinen Dokumententeilen	gering	keine
PHP	mittel	allgemeiner Zugriff auf Informationsquellen	hoch	klein

[Turau 1999]

Server-Technologien

Vergleich der Technologien [Einordnung]

	Programmier-aufwand	Anwendungen	Abhängigkeit vom Server	Abhängigkeit von Progammiersprache
CGI	mittel	allgemeiner Zugriff auf Informationsquellen	keine	klein
Web-Server-Bibliothek	hoch	komplexere, Performanz-kritische Anwendungen	hoch	meist C/C++
Servlet	mittel	"	mittel	nur Java
SSI	sehr gering	Wiederverwendung von kleinen Dokumentteilen	gering	keine
PHP	mittel	allgemeiner Zugriff auf Informationsquellen	hoch	klein

[Turau 1999]

Server-Technologien

Vergleich der Technologien [Einordnung]

	Programmier-aufwand	Anwendungen	Abhängigkeit vom Server	Abhängigkeit von Progammiersprache
CGI	mittel	allgemeiner Zugriff auf Informationsquellen	keine	klein
Web-Server-Bibliothek	hoch	komplexere, Performanz-kritische Anwendungen	hoch	meist C/C++
Servlet	mittel	"	mittel	nur Java
SSI	sehr gering	Wiederverwendung von kleinen Dokumentteilen	gering	keine
PHP	mittel	allgemeiner Zugriff auf Informationsquellen	hoch	klein

[Turau 1999]

Server-Technologien

Vergleich der Technologien [Einordnung]

	Programmier-aufwand	Anwendungen	Abhängigkeit vom Server	Abhängigkeit von Progammiersprache
CGI	mittel	allgemeiner Zugriff auf Informationsquellen	keine	klein
Web-Server-Bibliothek	hoch	komplexere, Performanz-kritische Anwendungen	hoch	meist C/C++
Servlet	mittel	"	mittel	nur Java
SSI	sehr gering	Wiederverwendung von kleinen Dokumentteilen	gering	keine
PHP	mittel	allgemeiner Zugriff auf Informationsquellen	hoch	klein

[Turau 1999]

Server-Technologien

Vergleich der Technologien [Einordnung]

	Programmier-aufwand	Anwendungen	Abhängigkeit vom Server	Abhängigkeit von Progammiersprache
CGI	mittel	allgemeiner Zugriff auf Informationsquellen	keine	klein
Web-Server-Bibliothek	hoch	komplexere, Performanz-kritische Anwendungen	hoch	meist C/C++
Servlet	mittel	"	mittel	nur Java
SSI	sehr gering	Wiederverwendung von kleinen Dokumentteilen	gering	keine
PHP	mittel	allgemeiner Zugriff auf Informationsquellen	hoch	klein

[Turau 1999]

Server-Technologien

Vergleich der Technologien (Fortsetzung)

Kriterium	CGI, Perl	PHP	Servlet
Wartbarkeit	+	++	+++++
Performance	+	+++	+++++
Skalierbarkeit	+	+++	+++++
Verfügbarkeit	+++++	+++++	+++++
Plattformneutralität	+++++++	+++	++++
Abstraktionslevel	+	+++	+++++
Erweiterbarkeit	+++	++	+++++
Dokumentation	++++	++++	+++
Support	+++++	+++++	+++
Tool-Unterstützung	+	+	++++

[Wöhr 2004, p.390]

Server-Technologien

Quellen zum Nachlernen und Nachschlagen im Web

- Leibniz-Rechenzentrum. *Reguläre Sprachen, reguläre Ausdrücke.*
  www.lrz-muenchen.de/services/schulung/unterlagen/regul

- D. Enseleit. *SELFPHP.*
  www.selfphp.info

- PHP-Dokumentationsgruppe. *PHP Documentation.*
  www.php.net/docs.php

- J. Stärk. *PHP Tutorial.*
  www.usegroup.de/software/phptutorial

- W3 Schools. *PHP Tutorial.*
  www.w3schools.com/php

- V. Vaswani. *PHP 101: PHP For the Absolute Beginner.*
  devzone.zend.com

- Zend. *Eclipse PHP Development Tools (PDT).*
  www.zend.com/community/pdt