Kapitel WT:IV

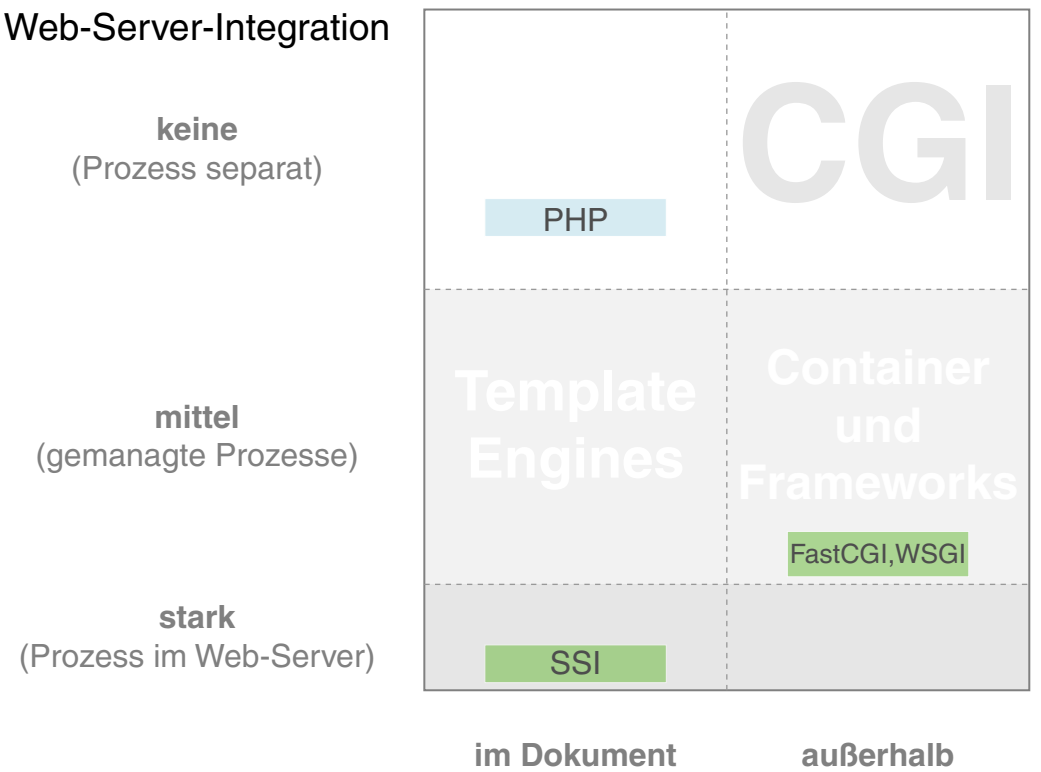
IV. Server-Technologien

- Web-Server
- Common Gateway Interface CGI
- Web-Container und -frameworks
- Web-Template-Engines
- Exkurs: reguläre Ausdrücke
- PHP Hypertext Preprocessor
- PHP Funktionsbibliotheken

©STEIN 2022

Web-Server

Einordnung von Server-Technologien



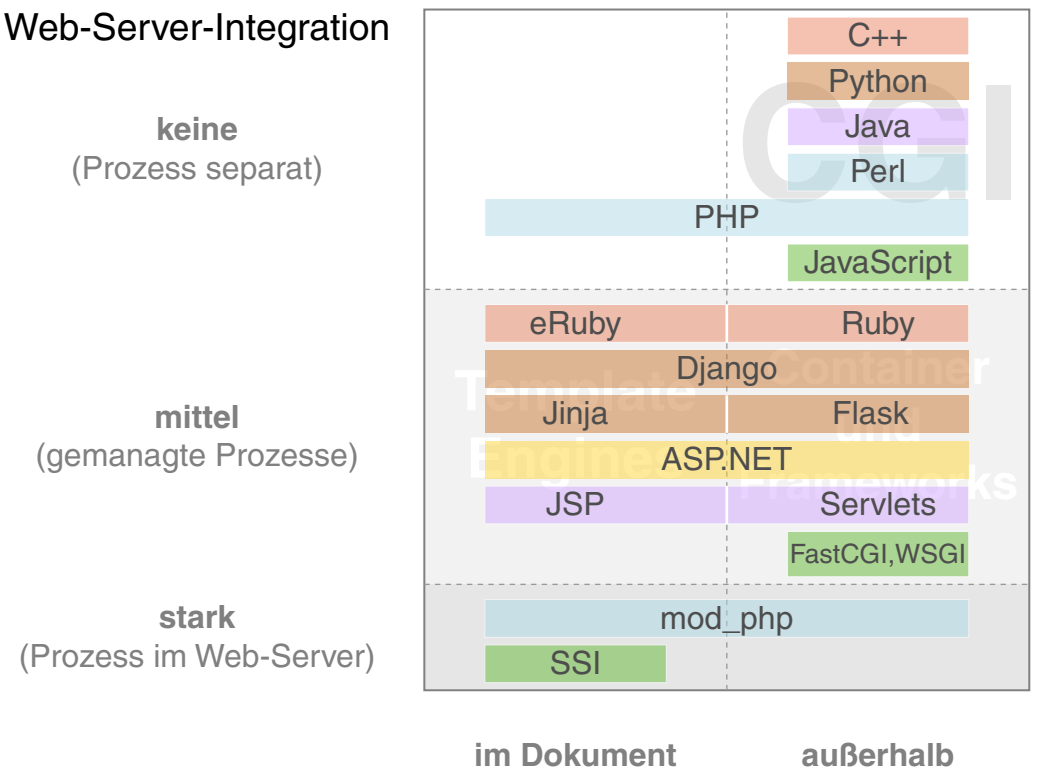
[Stein 2015-2022]

- x-Achse: Wo befindet sich der Code zur Dokumenterzeugung?
- y-Achse: Wie stark ist die Technologie in den Web-Server integriert?

©STEIN 2022

Web-Server

Einordnung von Server-Technologien (Fortsetzung)



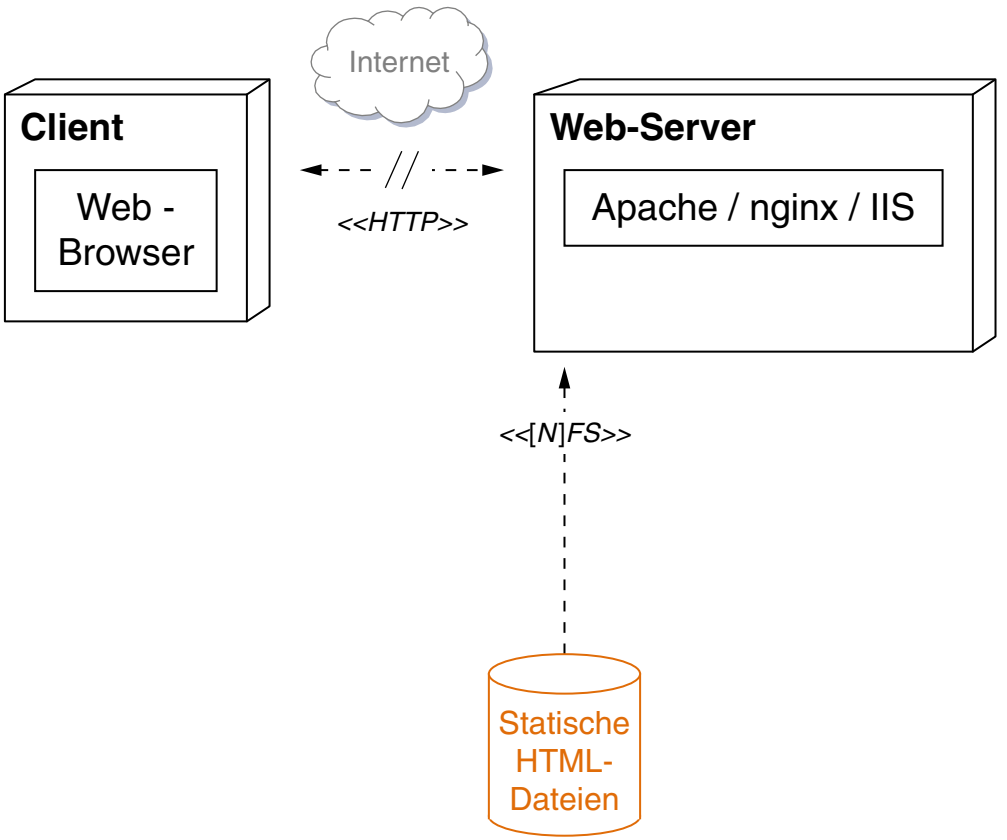
[Stein 2015-2022]

- x-Achse: Wo befindet sich der Code zur Dokumenterzeugung?
- y-Achse: Wie stark ist die Technologie in den Web-Server integriert?

©STEIN 2022

Web-Server

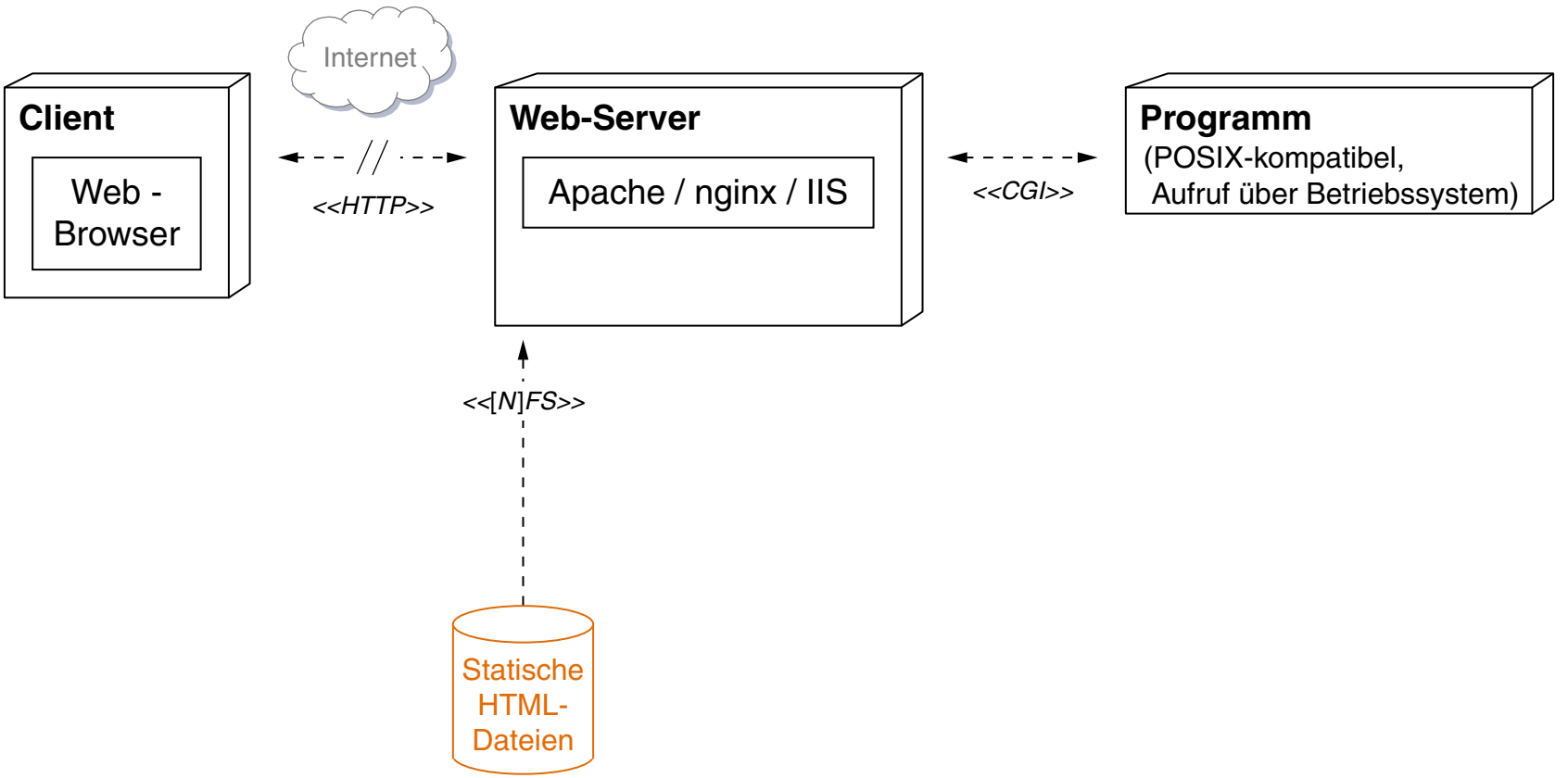
Deployment von Server-Technologien



©STEIN 2022

Web-Server

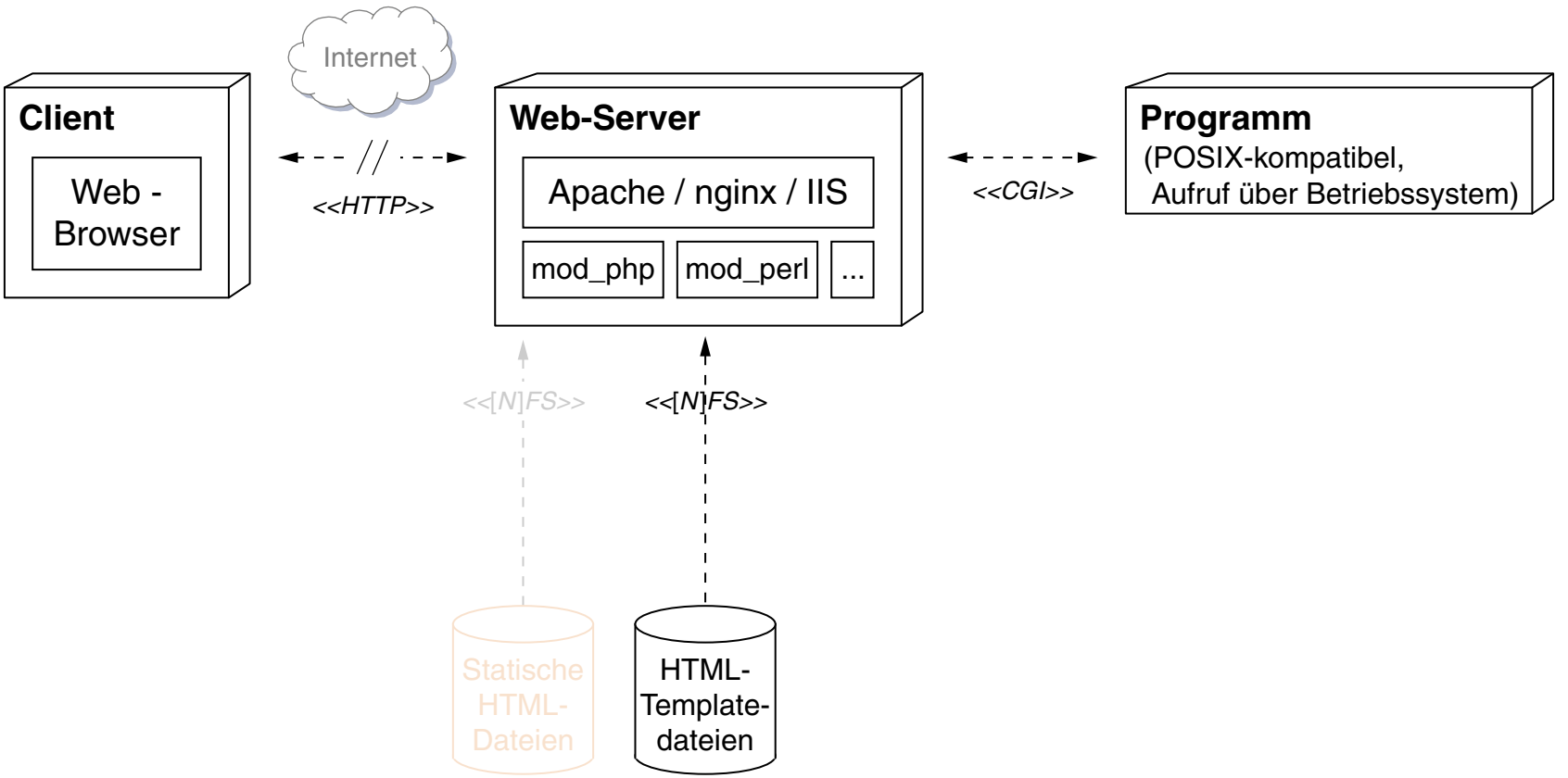
Deployment von Server-Technologien (Fortsetzung)



©STEIN 2022

Web-Server

Deployment von Server-Technologien (Fortsetzung)

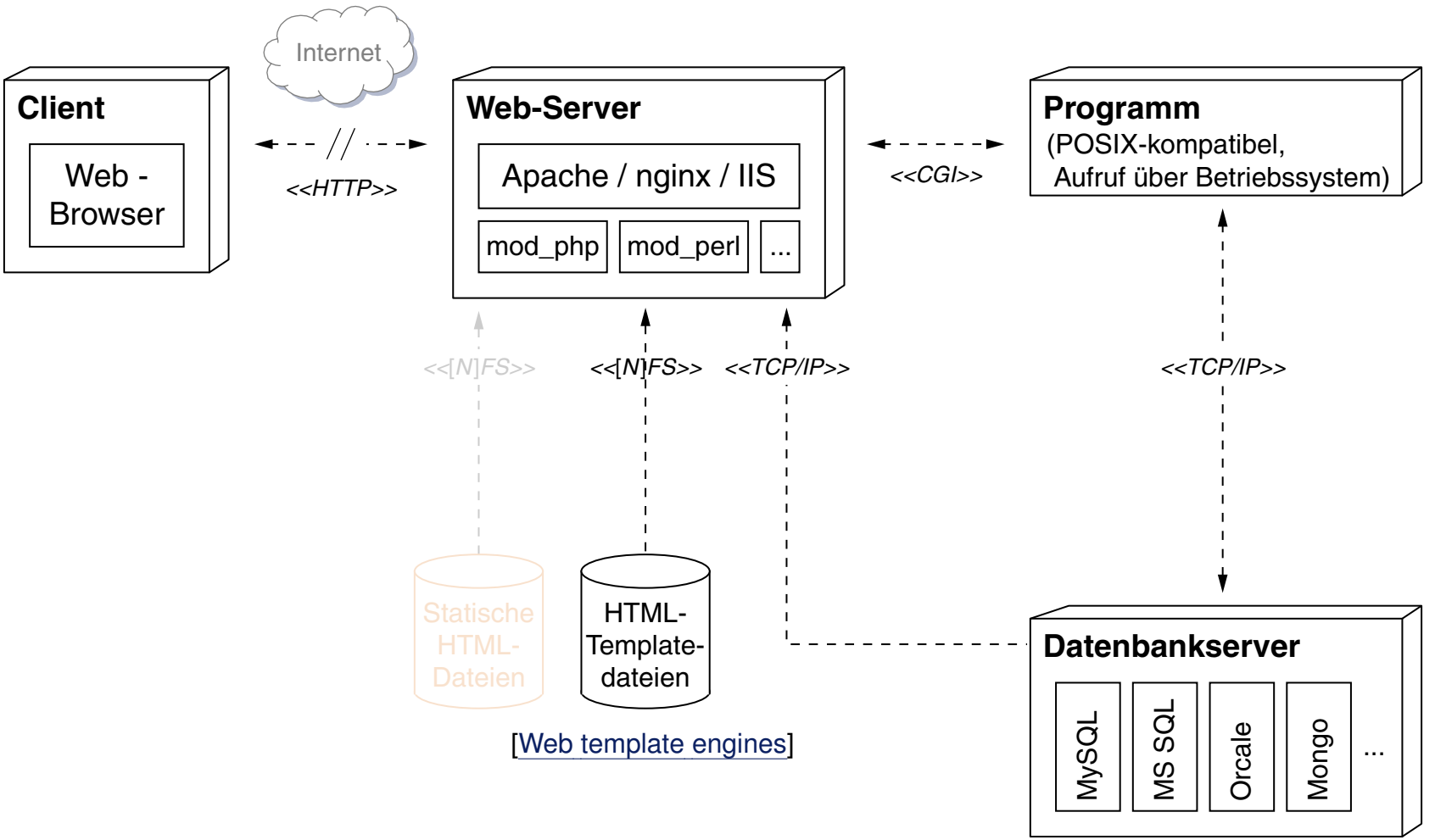


[Web template engines]

©STEIN 2022

Web-Server

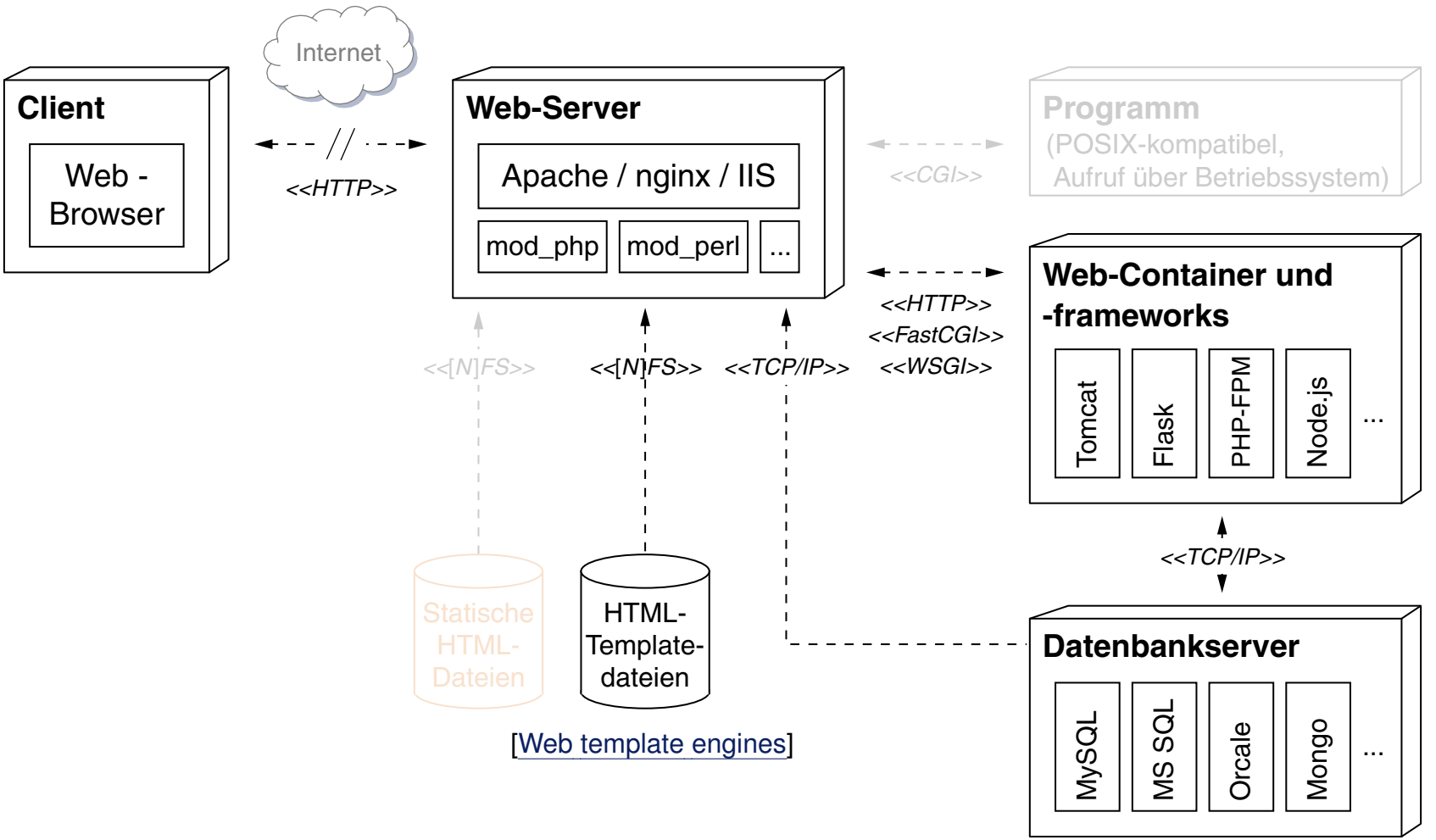
Deployment von Server-Technologien (Fortsetzung)



©STEIN 2022

Web-Server

Deployment von Server-Technologien (Fortsetzung)



[Web template engines]

©STEIN 2022

Bemerkungen:

- Ein Web-Server ist ein Computersystem, das Anfragen verarbeitet, die gemäß des HTTP-Protokolls übermittelt werden. Der Begriff „Web-Server“ kann sich auf das gesamte Computersystem oder auf die Software beziehen, die HTTP-Anfragen beantwortet. Ein Web-Server wird auch als „HTTP-Dämon“ bezeichnet; oft heißt das Programm, das den Web-Server implementiert, `httpd`.
- Beispiele für Prozesse *im* Web-Server:
  - SSI (mod_include) [apache.org]
  - mod_php [apache.org]
  - mod_perl [apache.org]
- Überblick über Apache-Module [apache.org, Wikipedia, SELFHTML]

©STEIN 2022

Bemerkungen: (Fortsetzung)

- *Separate* Prozesse (POSIX-kompatibel) können mittels CGI angebunden und direkt durch das Betriebsystem ausgeführt werden. Vorteil: einfache Anbindung des Containers „Betriebssystem“ in Form von Shell-Aufrufen. Nachteil (u.a.): zeitaufwändiger Start eines Prozesses.
- Das Portable Operating System Interface, POSIX, ist eine gemeinsam vom IEEE und der Open Group für Unix entwickelte standardisierte Programmierschnittstelle, welche die Schnittstelle zwischen Anwendungssoftware und Betriebssystem darstellt.
- Zur effizienten Verwaltung und Ausführung von Programmen mit Web-Funktionalität können spezialisierte Container und Webframeworks persistente, *gemanagte* Prozesspools vorhalten. Beispiele:
  - Jakarta Servlet, Jakarta JSP [apache.org]
  - FastCGI, mod_fcgid [apache.org]
  - JavaScript [nodejs.org]

©STEIN 2022

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, der HTTPS-Dämon auf Port 443 (*well-known Ports*).
ServerRoot	Verzeichnis für Konfigurations-, Fehler-, und Log-Dateien.
DocumentRoot	Verzeichnis für statische HTML-Dateien.
Default-HTML-Dateiname	Spezifiziert die URL nur ein Verzeichns, wird nach einer Default-Datei gesucht. Üblich sind `index.html` oder `index.htm`.
CGI-Skripte	Physische und virtuelle Verzeichnisse für CGI-Skripte.
Log-Dateien	Protokollierung der Zugriffe (`access.log`) und 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.

©STEIN 2022

Web-Server

Wichtige Konfigurationseinstellungen

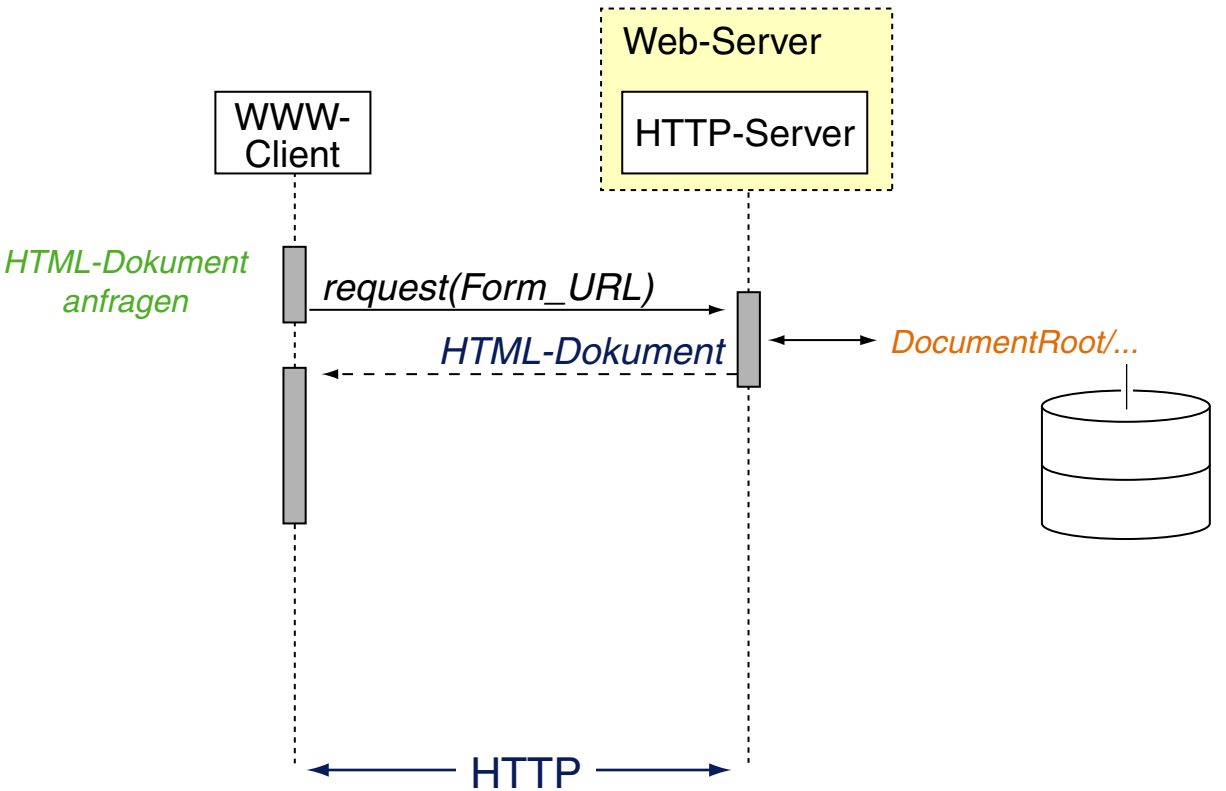
IP-Adresse, Hostname	Bei lokalem Betrieb die IP-Adresse 127.0.0.1 bzw. `localhost`.
Port	Üblicherweise lauscht der HTTP-Dämon auf Port 80, der HTTPS-Dämon auf Port 443 (*well-known Ports*).
ServerRoot	Verzeichnis für Konfigurations-, Fehler-, und Log-Dateien.
DocumentRoot	Verzeichnis für statische HTML-Dateien.
Default-HTML-Dateiname	Spezifiziert die URL nur ein Verzeichns, wird nach einer Default-Datei gesucht. Üblich sind `index.html` oder `index.htm`.
CGI-Skripte	Physische und virtuelle Verzeichnisse für CGI-Skripte.
Log-Dateien	Protokollierung der Zugriffe (`access.log`) und 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.

©STEIN 2022

Web-Server

Statisch: Sequenzdiagramm Seitenauslieferung

[statisch, CGI, Servlet, JSP, PHP]

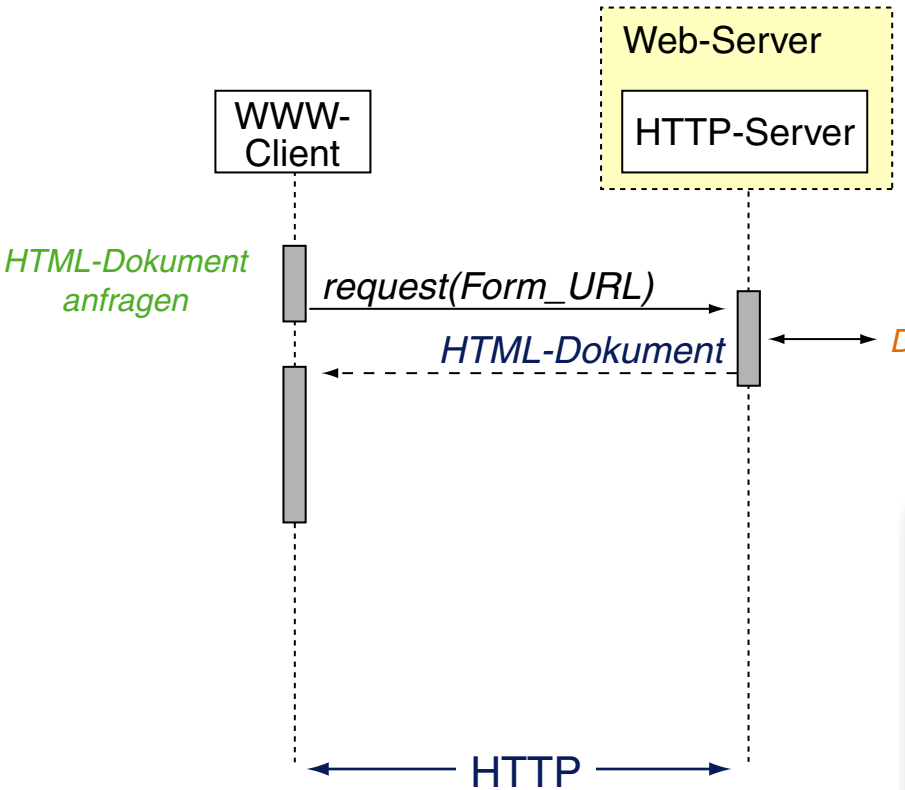


©STEIN 2022

Web-Server

Statisch: Sequenzdiagramm Seitenauslieferung

[statisch, CGI, Servlet, JSP, PHP]



[statisch: HTML, Aufruf]

©STEIN 2022

Web-Server

Apache HTTP-Server: Historie [Wikipedia]

1995 Version 0.6.2. Sammlung von 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. Unterstützung von Dateien > 2 GB, überarbeitete Authentifizierung, verbessertes Caching.

2012 Version 2.4. Deutlich performanter, geringerer Resourcenverbrauch.

2022 Aktuelle Version des Apache HTTP-Servers: [apache.org]

©STEIN 2022

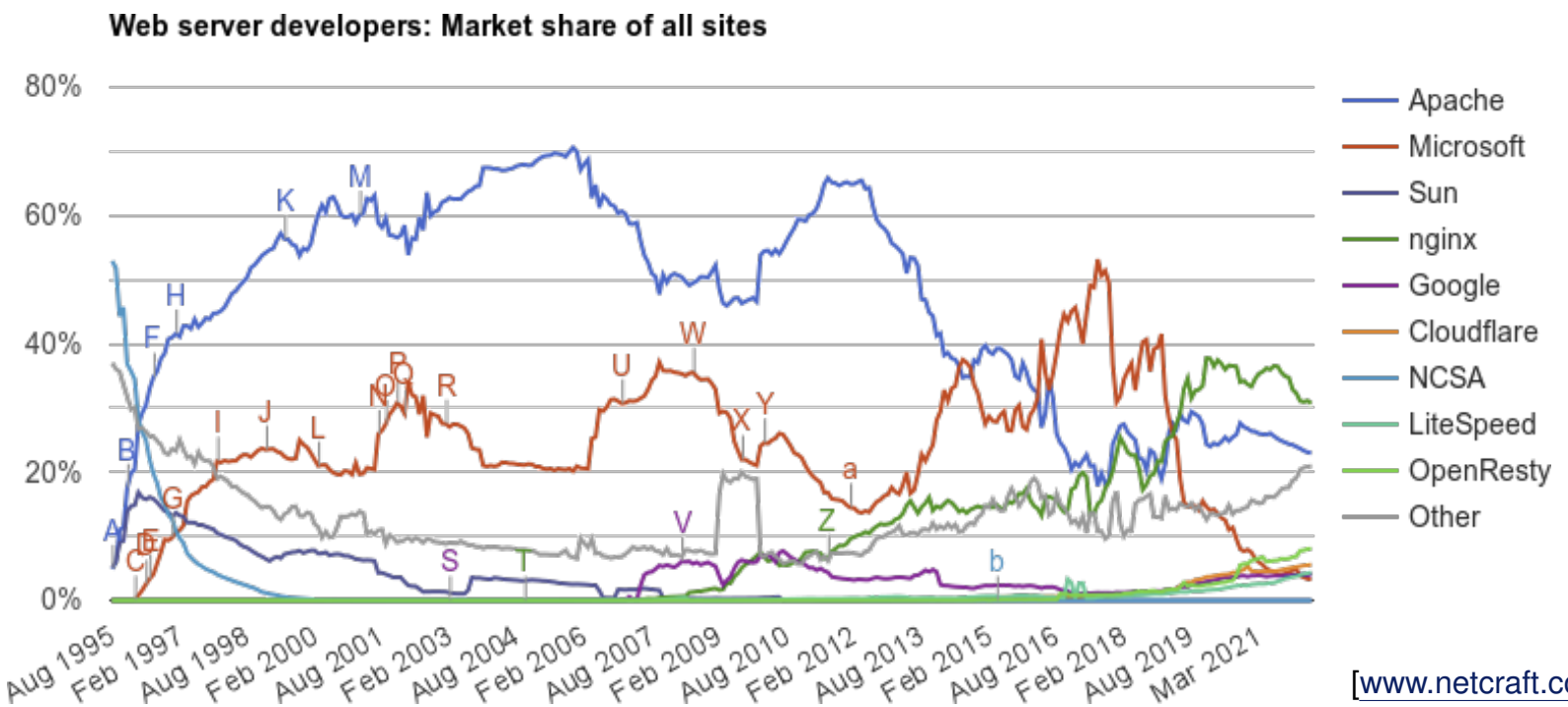
Web-Server

Apache HTTP-Server: Verbreitung

2022 1.155.729.496 Websites (= unique hostnames) [internetlivestats: live statistics]

197.960.959 active Websites

12.069.814 Web-Servers (aka web-facing computers)



[www.netcraft.com]

©STEIN 2022

Bemerkungen:

- Der Marktanteil bezieht sich auf den Anteil an allen Websites. [netcraft.com]
- Zählen von 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-facing application = Web application that is visible or accessible from the Internet. [treyford]
- “active” Websites: The web includes [. . .] a considerable quantity of sites that are untouched by human hand, produced automatically at the point of customer acquisition by domain registration or hosting service companies, advertising providers or speculative domain registrants, or search-engine optimisation companies. [www.netcraft.com]

©STEIN 2022

Web-Server

Apache HTTP-Server: httpd.conf

Die Konfigurationsdatei `httpd.conf` liegt typischerweise im Verzeichnis `/etc/httpd/`. Funktionsabschnitte:

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

©STEIN 2022

Web-Server

Apache HTTP-Server: httpd.conf

Die Konfigurationsdatei `httpd.conf` liegt typischerweise im Verzeichnis `/etc/httpd/`. 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 mit XML zu tun.

Container	**Beschreibung**
<IfDefine>, <IfModule>	Bedingte Ausführung von Direktiven.
`<Directory>`, <DirectoryMatch> `<Files>`, <FilesMatch> `<Location>`, <LocationMatch>	Spezifikation von Direktiven für Verzeichnisse, Dateien und URLs. Die `<Match>`-Variante ermöglicht die Angabe regulärer Ausdrücke.

©STEIN 2022

Web-Server

Apache HTTP-Server: .htaccess

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

©STEIN 2022

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"
AuthUserFile /usr/maintenance/web/.htpasswd
AuthGroupFile /usr/maintenance/web/.htgroups
Require user Alice Bob Eve
Require group Support
```

`.htpasswd`-Datei:

```
Alice:INY8m5KMwIc
Bob:69gY8YPjQXeN6
Eve:INw2mPEH.owe2
Frank:INh6DHvyejvf2
Greg:INboWuvjjwQ7E
```


©STEIN 2022

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"
AuthUserFile /usr/maintenance/web/.htpasswd
AuthGroupFile /usr/maintenance/web/.htgroups
Require user Alice Bob Eve
Require group Support
```

`.htpasswd`-Datei:

```
Alice:INY8m5KMwIc
Bob:69gY8YPjQXeN6
Eve:INw2mPEH.owe2
Frank:INh6DHvyejvf2
Greg:INboWuvjjwQ7E
```

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

©STEIN 2022

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 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

©STEIN 2022

Web-Server

Server Side Includes SSI [Einordnung, Deployment]

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`

©STEIN 2022

Web-Server

Server Side Includes SSI [Einordnung, Deployment]

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:

```
<h3>Dynamisches HTML mit Server Side Includes</h3>
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>
Aufrufender Web-Browser:        <!--#echo var="HTTP_USER_AGENT" -->
<h3>Weitere Informationen:</h3> <!--#exec cmd="free" -->
```


©STEIN 2022

Web-Server

Server Side Includes SSI [Einordnung, Deployment]

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-
  Dateiendung gekennzei

Beispiel:

```
<h3>Dynamisches HTML mit S
Datum/Uhrzeit auf dem Serv

Name dieser HTML-Datei:
Installierte Server-Softwa
Aufrufender Web-Browser:
<h3>Weitere Informationen:</h3> <!--#exec cmd="free" -->
```

[SSI: SHTML, Aufruf]

©STEIN 2022

Web-Server

Server Side Includes SSI (Fortsetzung) [apache.org] [SELFHTML]

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*`"`

©STEIN 2022

Kapitel WT:IV

IV. Server-Technologien

- Web-Server
- Common Gateway Interface CGI
- Web-Container und -frameworks
- Web-Template-Engines
- Exkurs: reguläre Ausdrücke
- PHP Hypertext Preprocessor
- PHP Funktionsbibliotheken

©STEIN 2022

Common Gateway Interface CGI

Einführung [Einordnung, Deployment]

Prinzip: Ein Programm außerhalb des Web-Servers stellt einen Zugang (*Gateway*) zu geschützten, für den Web-Server nicht erreichbaren Daten bereit. Der hierfür standardisierte Kommunikationsmechanismus heißt CGI.

©STEIN 2022

Common Gateway Interface CGI

Einführung [Einordnung, Deployment]

Prinzip: Ein Programm außerhalb des Web-Servers stellt einen Zugang (*Gateway*) zu geschützten, für den Web-Server nicht erreichbaren Daten bereit. Der hierfür standardisierte Kommunikationsmechanismus heißt CGI.

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

- Über ein Formular.
  ```
  <form action="/cgi-bin/sample.sh" method="get">
  ```
- Über einen Verweis.
  ```
  <a href="/cgi-bin/statistik.py?page=42">Statistik</a>
  ```

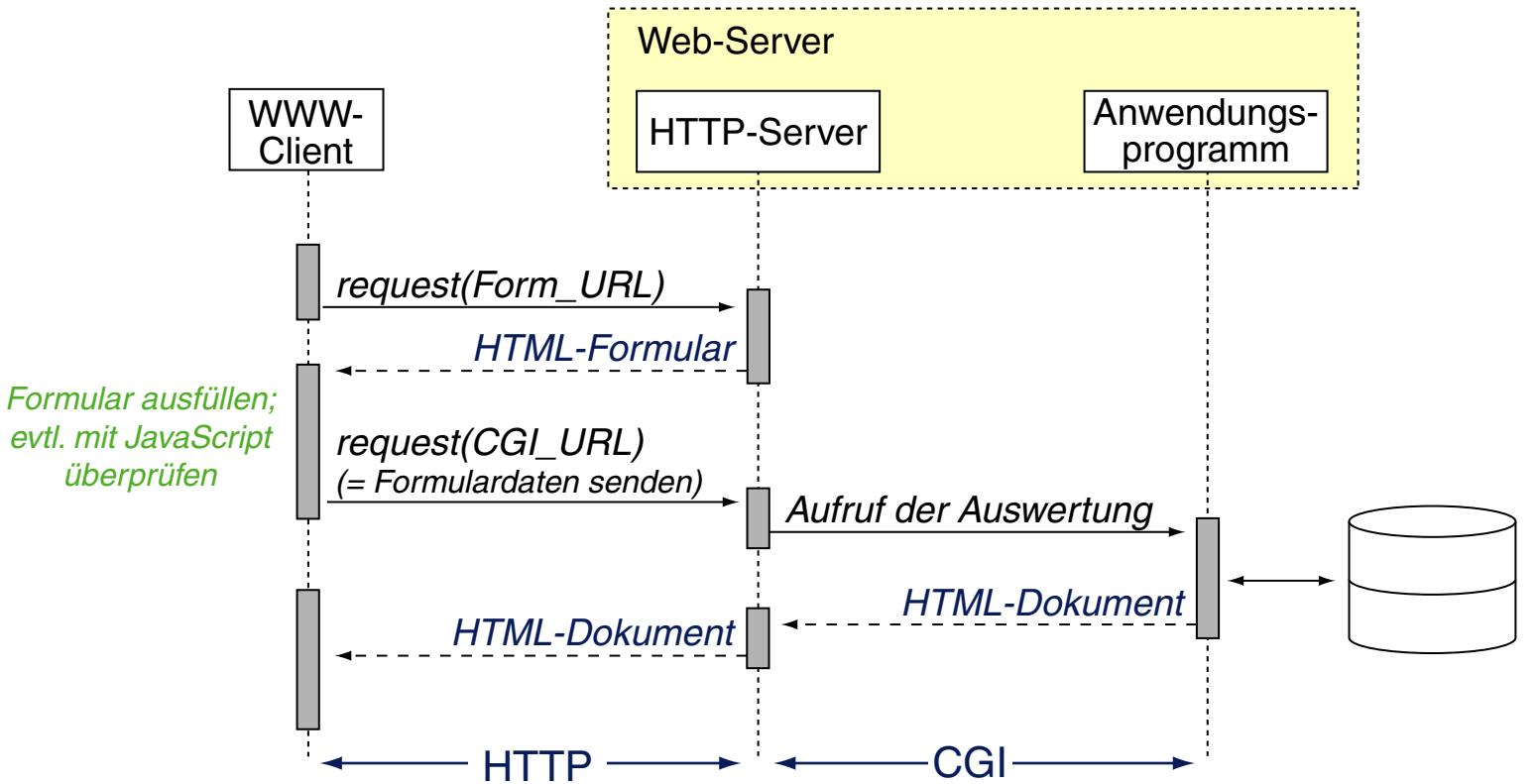
Typische Aufrufe aus einer HTML-Datei ohne Übergabe von Anwenderdaten:

- 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.sh">
  ```


©STEIN 2022

Common Gateway Interface CGI

CGI: Sequenzdiagramm Seitenauslieferung [statisch, CGI, Servlet, JSP, PHP]



©STEIN 2022

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 [RFC 3875] [Meinel/Sack]:
  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 standardisierten 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.
  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.

©STEIN 2022

Common Gateway Interface CGI

Wichtige CGI-Umgebungsvariablen

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

©STEIN 2022

Bemerkungen zur Codierung von Formulardaten:

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

©STEIN 2022

Common Gateway Interface CGI

Beispiel: Shell-Skripte als CGI-Programm

In der HTML-Datei:

```
<a href="https://webtec.webis.de/cgi-bin/cgi-sample1.cgi">CGI-Aufruf</a>
```

Die Shell-Skript-Datei:

```
#!/bin/bash
echo "content-type: text/html"
echo ""   #Leerzeile gemäß HTTP-Protokoll.
echo "<!DOCTYPE html>"
echo "<html>"
echo "<head>"
echo "<meta http-equiv=\"content-type\" content=\"text/html; ...\">"
echo "<title>cgi-sample1</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>"
```


©STEIN 2022

Common Gateway Interface CGI

Beispiel: Shell-Skripte als CGI-Programm

In der HTML-Datei:

```
<a href="https://webtec.webis.de/cgi-bin/cgi-sample1.cgi">CGI-Aufruf</a>
```

Die Shell-Skript-Datei:

```
#!/bin/bash
echo "content-type: text/html"
echo ""   #Leerzeile gemäß HTTP-Protokoll.
echo "<!DOCTYPE html>"
echo "<html>"
echo "<head>"
echo "<meta http-equiv=\"content-typ
echo "<title>cgi-sample1</title>"
echo "</head>"
echo "<body>"
echo "<h3>Werte einiger CGI-Variable
echo "Installierte Server-Software:
echo "Aufrufender Web-Browser: " $HT
echo "Anfragemethode: " $REQUEST_MET
echo "Query-String: " $QUERY_STRING
echo "</body>"
echo "</html>"
```

[CGI: Script, Aufruf]

©STEIN 2022

Kapitel WT:IV

IV. Server-Technologien

- Web-Server
- Common Gateway Interface CGI
- Web-Container und -frameworks
- Web-Template-Engines
- Exkurs: reguläre Ausdrücke
- PHP Hypertext Preprocessor
- PHP Funktionsbibliotheken

©STEIN 2022

Web-Container und -frameworks

(1) Jakarta-Servlets bzw. Servlets [Einordnung, Deployment]

> *“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 1, 2]

©STEIN 2022

Web-Container und -frameworks

(1) Jakarta-Servlets bzw. Servlets [Einordnung, Deployment]

> *"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 1, 2]

- Servlets können jede Art von Anfrage beantworten; ihr Einsatz geschieht hauptsächlich im Zusammenhang mit Web-Servern.
- Die Servlet API besteht aus den `jakarta.servlet.*`-Packages. [Javadoc]
  Diese gehören zu Jakarta EE (bzw. vor 2019 zu Java EE), nicht zu Java SE.
- 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

©STEIN 2022

Web-Container und -frameworks

(1) Servlets: Lebenszyklus

Der Lebenszyklus eines Servlets wird von dem Container gesteuert, der das Servlet verwaltet.

Ein HTTP-Servlet wird durch einen HTTP-Request über eine URL angesprochen. Darauf hin 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.

©STEIN 2022

Bemerkungen (Servlet-Container) :

- Servlets werden in einem Servlet-Container verwaltet, der u.a. für die persistente Speicherung der Zustände und die Verfügbarkeit der Servlets zuständig ist.

  Ein Servlet-Container stellt eine Laufzeitumgebung für Servlets dar und besteht im Wesentlichen aus einer Java-Plattform. Servlet-Container gehören zur Vermittlungsschicht zwischen (Web-)Client und (Web-)Server und zählen somit zur Middleware.

- Servlet-Container können direkt oder über einen vorgeschalteten Web-Server angefragt werden. Der letztere Fall benötigt eine Schnittstelle, die zwischen den URLs unterscheidet, die der Web-Server bzw. der Servlet-Container bedienen soll. Im Apache-Web-Server stehen hierfür die alternativen Module “mod_jk” und “mod_proxy” zur Verfügung. [apache.org]

- In der „klassischen“ Konfiguration sind Servlet-Container ein Service, der zusätzlich oder anstelle eines Web-Servers installiert wird. Alternativ kann jede Servlet-basierte Web-Anwendungen ihren eigenen Servlet-Container zur Installation mit bringen.
  Stichwort: *Embedded Servlet Container* [Tomcat, Jetty]

- Bei embedded Servlet-Containern steht die Container-Funktionalität im Hintergrund: der Container dient nur *einer* Anwendung. Sie sind eine elegante Möglichkeit, eine Java-Anwendung verteilt – mit allen Vorteilen von standardisierten Web-Technologien – zur Verfügung zu stellen: weltweiter Zugriff via URL und HTTP, Benutzeroberfläche via Browser, etc.

©STEIN 2022

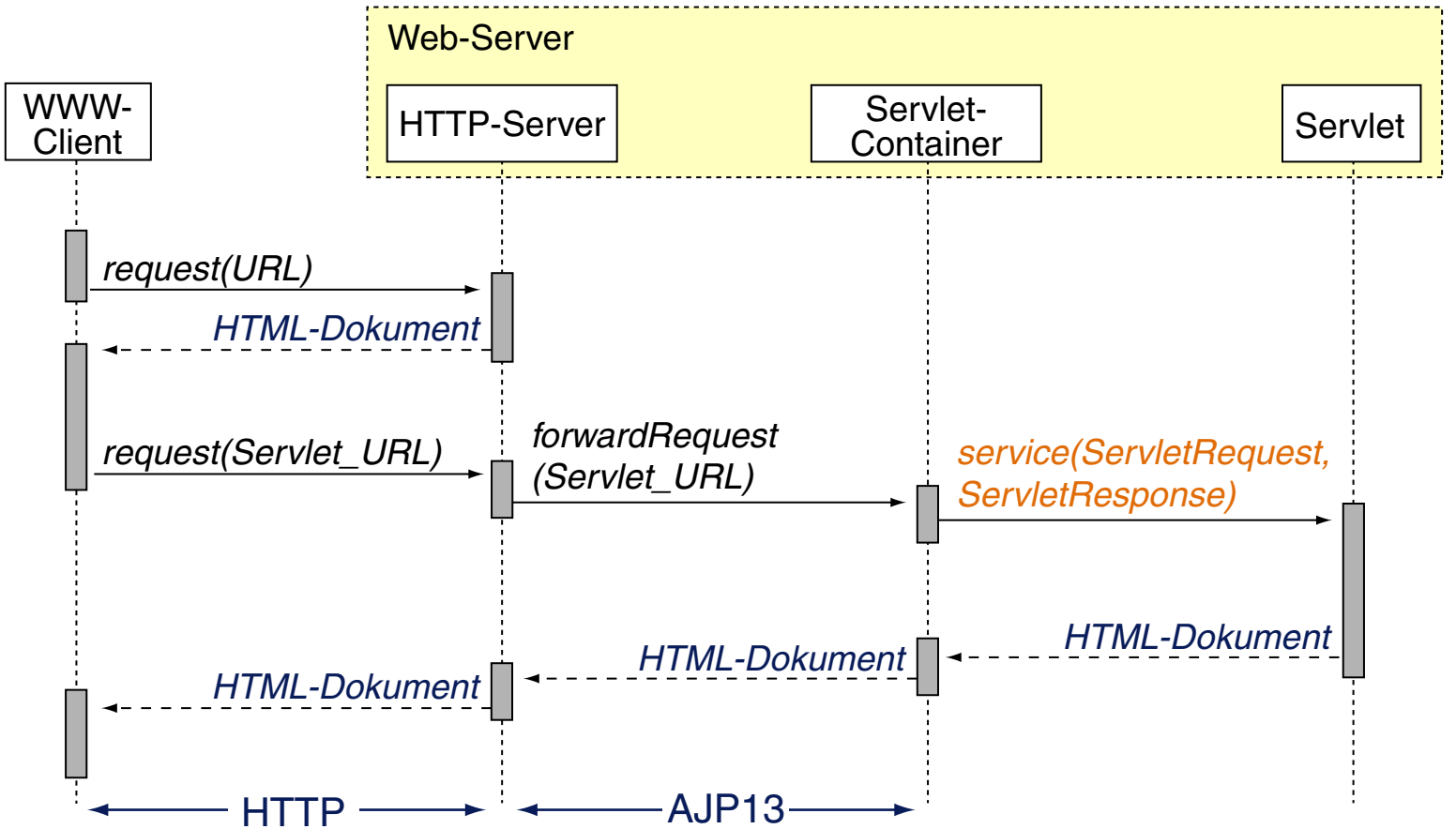
Bemerkungen (Servlet-Funktionen) :

- Die `service()`-Methode erkennt die HTTP-Anfragen GET, POST, HEAD, PUT, DELETE, OPTIONS bzw. TRACE und ruft die entsprechenden Java-Methoden `doGet()`, `doPost()`, etc. auf.
- Durch den Aufruf der `destroy()`-Methode entlädt der Container eine Servlet-Instanz.

©STEIN 2022

Web-Container und -frameworks

(1) Servlets: Sequenzdiagramm Seitenauslieferung [statisch, CGI, Servlet, JSP, PHP]



©STEIN 2022

Web-Container und -frameworks

(1) Servlets: Charakteristika

- 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 – – 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.

©STEIN 2022

Web-Container und -frameworks

(1) Servlets: Charakteristika

- 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.

©STEIN 2022

Web-Container und -frameworks

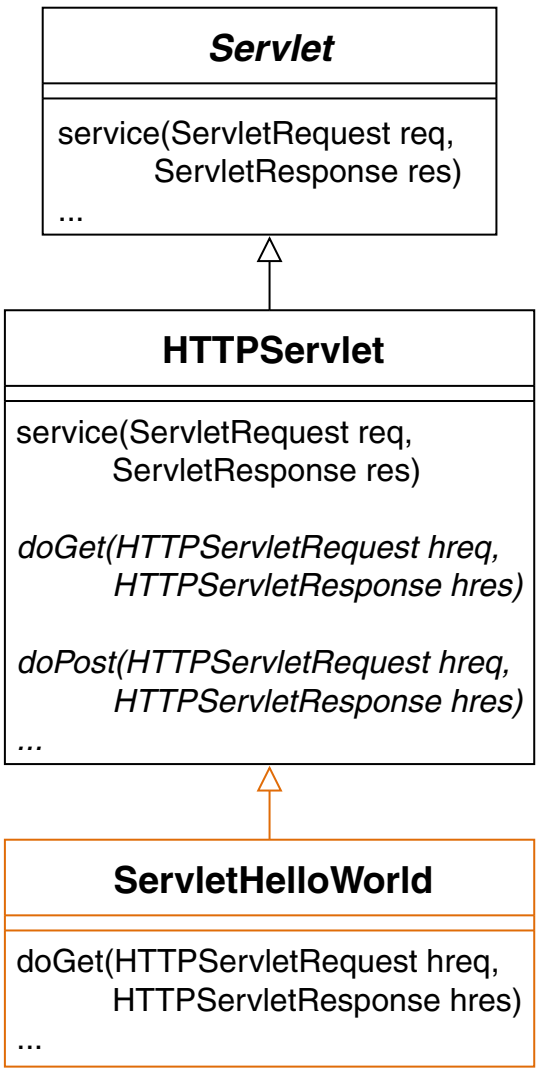
(1) Servlets: Hello World [Servlet: Aufruf] [Javadoc: doGet]

```
package servlet;

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

public class ServletHelloWorld extends HttpServlet {

    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 = response.getWriter();
        // Use "out" to send content to browser.
        out.print("<!DOCTYPE html>"
                + "<html lang=\"en\">"
                + "<head><title>Hello World</title>"
                + "<meta charset=\"utf-8\"></head>"
                + "<body><h1>Hello World!</h1></body></html>");
    }
}
```


©STEIN 2022

Web-Container und -frameworks

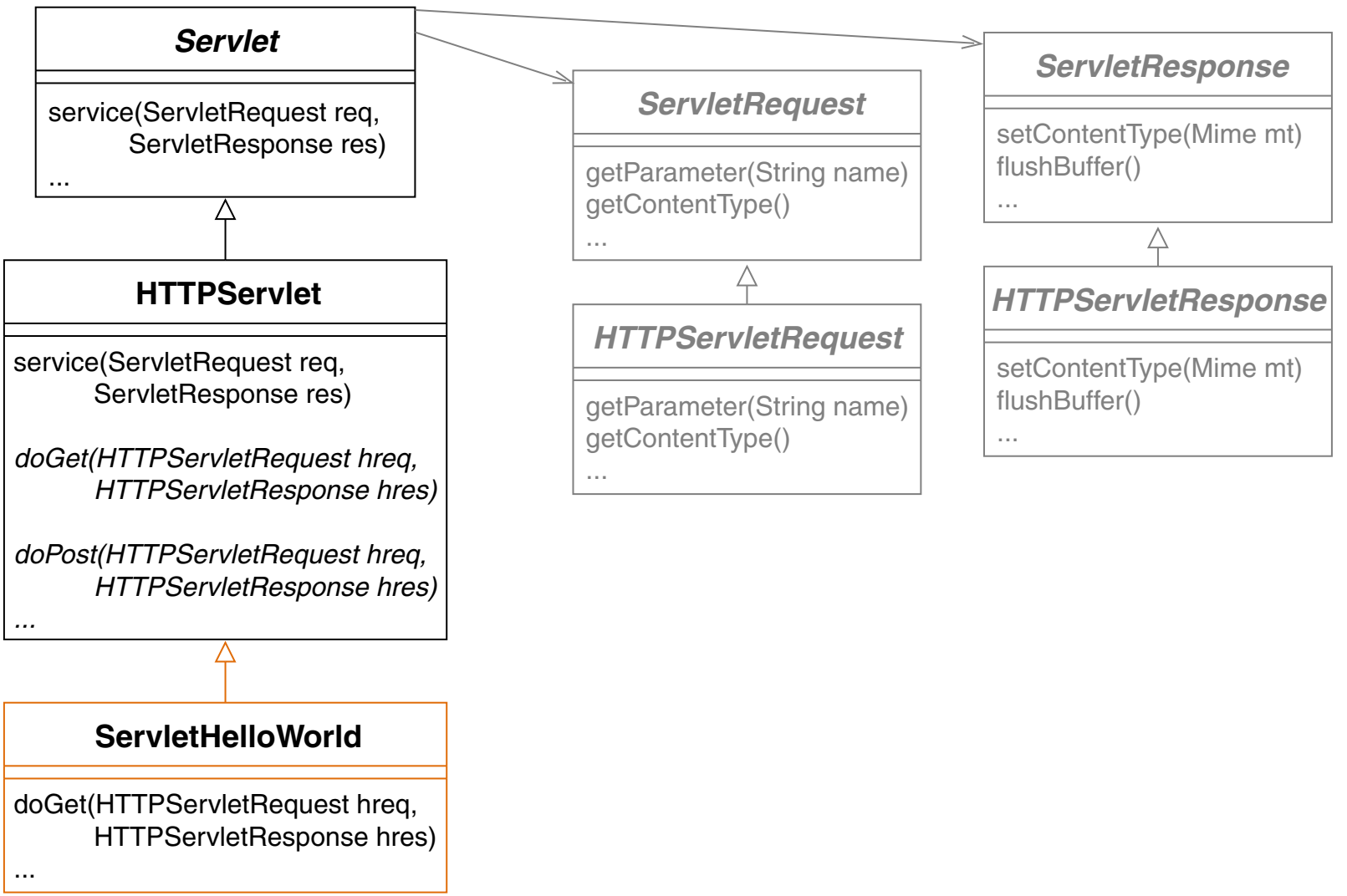
(1) Servlets: Implementierung via Template-Method-Pattern



©STEIN 2022

Web-Container und -frameworks

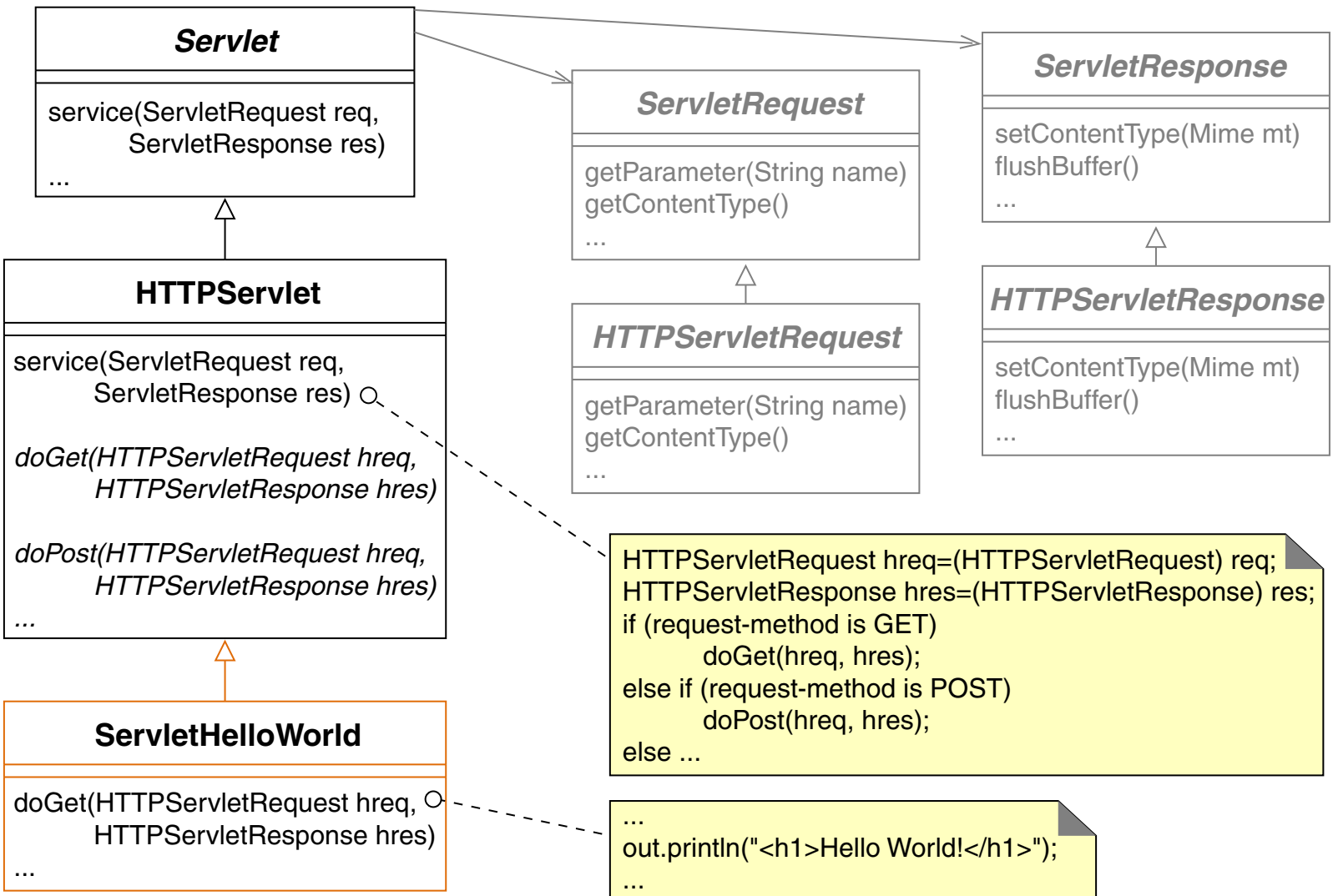
(1) Servlets: Implementierung via Template-Method-Pattern



©STEIN 2022

Web-Container und -frameworks

(1) Servlets: Implementierung via Template-Method-Pattern



©STEIN 2022

Web-Container und -frameworks

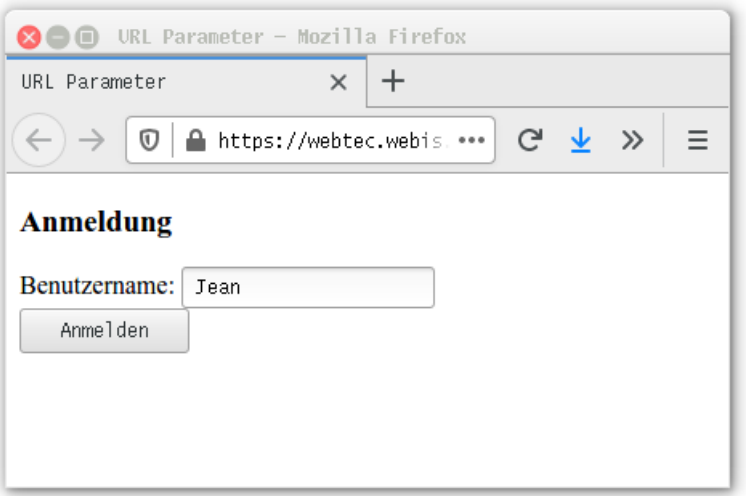
(1) Servlets: URL-Parameter einlesen [HTML: Servlet, Flask]

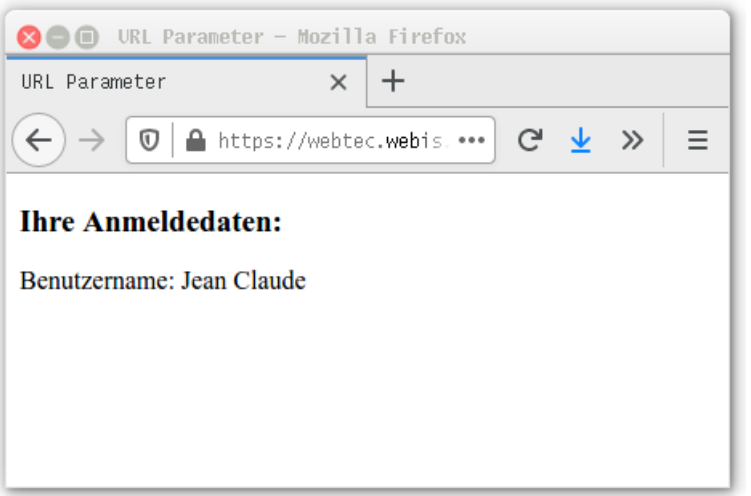
```
<!DOCTYPE html>
<html lang="de">
  <head>
    <meta charset="utf-8">
    <title>URL Parameter</title>
  </head>
  <body>
    <h3>Anmeldung</h3>
    <form action="ReadURLParam" method="GET">
      <div>
        <label for="user">Benutzername:</label>
        <input id="user" name="user" type="text">
      </div>
      <input type="submit" value="Anmelden"> <br>
    </form>
  </body>
</html>
```


©STEIN 2022

Web-Container und -frameworks

(1) Servlets: URL-Parameter einlesen (Fortsetzung)





[Servlet: Java, Aufruf]

©STEIN 2022

Web-Container und -frameworks

(1) Servlets: URL-Parameter einlesen (Fortsetzung) [Code: Servlet, Flask]

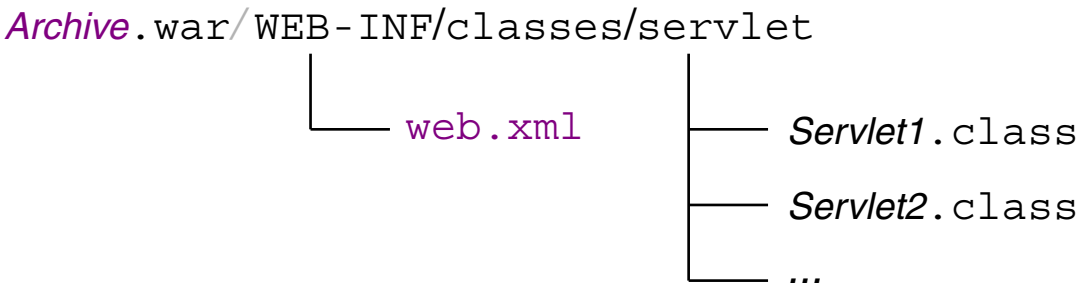
```
package servlet;

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();
        // Use "request" to read incoming HTTP headers (e.g. cookies)
        // and HTML form data (e.g. data the user entered and submitted)
        out.println("<!DOCTYPE html><html lang=\"de\">"
             + "<head><title>URL Parameter</title>
             + "<meta charset=\"utf-8\"></head>"
             + "<body><h3>Ihre Anmeldedaten:</h3>"
             + "Benutzername: " + request.getParameter("user")
             + "</body></html>");
    }

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


©STEIN 2022

Web-Container und -frameworks

(1) Servlets: Deployment [Deployment: Servlet, Flask]

Die Bereitstellung (*Deployment*) Servlet-basierter Web-Anwendungen ist mittels des `war`-Archivformats standardisiert. Ein `war`-Archiv besitzt folgende Struktur:

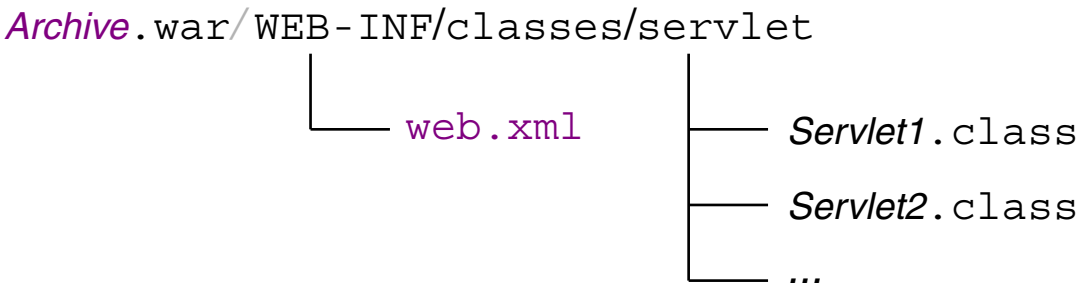


©STEIN 2022

Web-Container und -frameworks

(1) Servlets: Deployment (Fortsetzung) [Deployment: Servlet, Flask]

Die Bereitstellung (*Deployment*) Servlet-basierter Web-Anwendungen ist mittels des `war`-Archivformats standardisiert. Ein `war`-Archiv besitzt folgende Struktur:



Die Abbildung einer URL auf das entsprechende Servlet ist definiert durch

1. den DocumentRoot des Servlet-Containers,
2. ein *Filter-Pattern* zur Auswahl des Web-Containers bzw. -frameworks,
3. den `war`-Archivnamen *Archive* sowie
4. das in der Datei `web.xml` definierte *Servlet-Mapping* für die Java-Klasse:

`https://` *WebServerDomainName* [`:port`] / *Filter-Pattern*/ *Archive*/ *Servlet-Mapping*

©STEIN 2022

Bemerkungen:

- Die Datei `web.xml` ist der Deployment-Descriptor und enthält die Mappings zu den Servlets. Folgende web.xml-Datei beschreibt das Hello-World-Servlet:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<web-app xmlns="https://jakarta.ee/xml/ns/jakartaee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="https://jakarta.ee/xml/ns/jakartaee.../web-app_5_0.xsd"
    version="5.0">
  <servlet>
    <servlet-name>HelloWorld</servlet-name>
    <servlet-class>servlet.ServletHelloWorld</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>HelloWorld</servlet-name>
    <url-pattern>/HelloWorld</url-pattern>
  </servlet-mapping>
</web-app>
```

  Schema-Datei für `<web-app>`-Elemente.

- Die Hello-World-Anwendung wurde in einem `war`-Archiv mit dem speziellen Namen `ROOT.war` bereitgestellt. Dadurch entfällt in der URL die Pfadkomponente *Archive*`/`.

- Das Servlet-Deployment kann im laufenden Betrieb eines Web-Servers erfolgen.

©STEIN 2022

Web-Container und -frameworks

(2) Flask: URL-Parameter einlesen

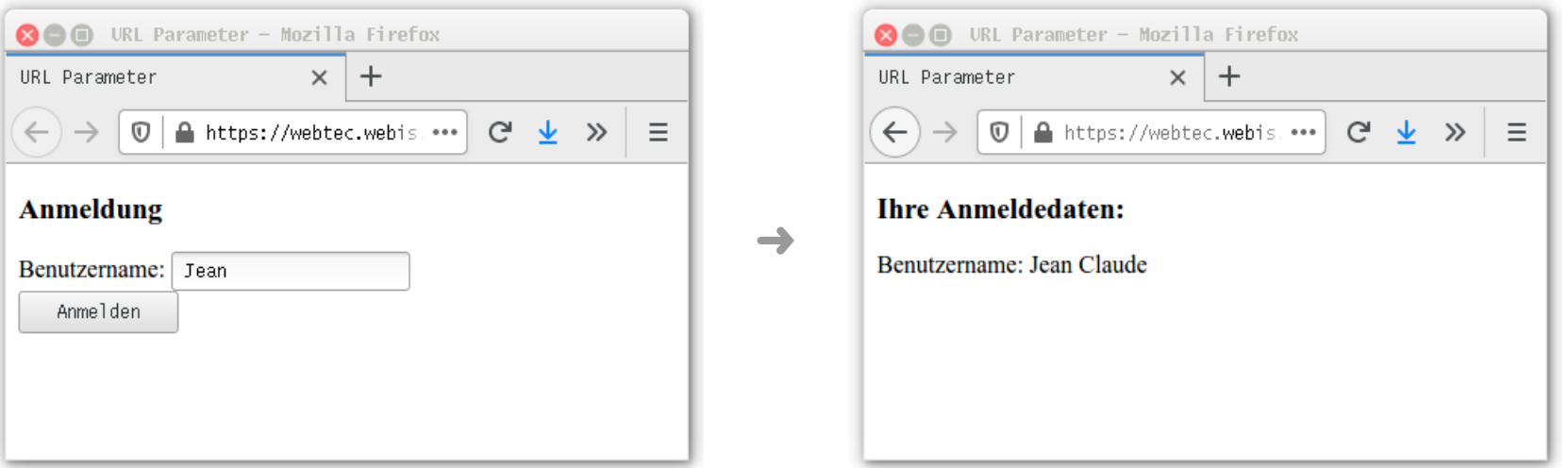
[Einordnung, Deployment] [HTML: Servlet, Flask]

```
<!DOCTYPE html>
<html lang="de">
  <head>
    <meta charset="utf-8">
    <title>URL Parameter</title>
  </head>
  <body>
    <h3>Anmeldung</h3>
    <form action="ReadURLParam" method="GET">
      <div>
        <label for="user">Benutzername:</label>
        <input id="user" name="user" type="text">
      </div>
      <input type="submit" value="Anmelden"> <br>
    </form>
  </body>
</html>
```


©STEIN 2022

Web-Container und -frameworks

(2) Flask: URL-Parameter einlesen (Fortsetzung)



[Flask: Python, Aufruf]

©STEIN 2022

Web-Container und -frameworks

(2) Flask: URL-Parameter einlesen (Fortsetzung) [Code: Servlet, Flask]

```
from flask import Flask, request

app = Flask(__name__, static_folder="web/static", static_url_path="/")

@app.route("/ReadURLParam")
def read_url_param():
    return "<!DOCTYPE html><html lang=\"de\">" \
           "<head><title>URL Parameter</title>" \
           "<meta charset=\"utf-8\"></head>" \
           "<body><h3>Ihre Anmeldedaten:</h3>" \
           f"Benutzername: {request.args.get('user')}" \
           "</body></html>"

def main():
    app.run(host="0.0.0.0", port=80,
            debug=True, threaded=False)

if __name__ == '__main__':
    main()
```


©STEIN 2022

Web-Container und -frameworks

(2) Flask: Deployment [Deployment: Servlet, Flask]

Bei der Bereitstellung (*Deployment*) Flask-basierter Web-Anwendungen wird die Abbildung einer URL auf die entsprechenden Funktionsaufrufe der Flask-Anwendung wie folgt spezifiziert:

1. den DocumentRoot der Flask-Anwendung,
2. ein *Filter-Pattern* zur Auswahl des Web-Containers bzw. -frameworks,
3. das in der Python-Datei mittels `@app.route()` definierte *Flask-Mapping* der Python-Funktion:

`https://` *WebServerDomainName* [`:port`] / *Filter-Pattern*/ *Flask-Mapping*

©STEIN 2022

Kapitel WT:IV

IV. Server-Technologien

- Web-Server
- Common Gateway Interface CGI
- Web-Container und -frameworks
- Web-Template-Engines
- Exkurs: reguläre Ausdrücke
- PHP Hypertext Preprocessor
- PHP Funktionsbibliotheken

©STEIN 2022

Web-Template-Engines

(1) Jakarta Server Pages JSP [Einordnung, Deployment]

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
- Anbindung von Jakarta EE Komponenten

Anwendung:

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

©STEIN 2022

Web-Template-Engines

(1) JSP: Hello World

Java-Code wird in den HTML-Code einer `jsp`-Datei eingebettet:

```
<!DOCTYPE html>
<html lang="en">
  <head> <title>Hello World</title> ...</head>
  <body>
    <h1> <% out.print("Hello World!"); %> </h1>
  </body>
</html>
```


©STEIN 2022

Web-Template-Engines

(1) JSP: Hello World

Java-Code wird in den HTML-Code einer `jsp`-Datei eingebettet:

```
<!DOCTYPE html>
<html lang="en">
  <head> <title>Hello World</title> ...</head>
  <body>
    <h1> <% out.print("Hello World!"); %> </h1>
  </body>
</html>
```

[JSP: jsp-Datei, Aufruf] [Java: generiert, manuell]

©STEIN 2022

Web-Template-Engines

(1) JSP: URL-Parameter einlesen

[Java: generiert] [jsp-Datei, jinja-Datei, php-Datei]

```
<!DOCTYPE html>
<html>
  <head> <title>URL Parameter</title> ... </head>
  <body>

    <h3>Anmeldung</h3>
    <form action="readURLParam.jsp" method="get">
      <div>
        <label for="user">Benutzername:</label>
        <input type="text" name="user" id="user">
      </div>
      <input type="submit" value="Anmelden">
    </form>

  </body>
</html>
```


©STEIN 2022

Web-Template-Engines

(1) JSP: URL-Parameter einlesen

[Java: generiert] [jsp-Datei, jinja-Datei, php-Datei]

```
<!DOCTYPE html>
<html>
  <head> <title>URL Parameter</title> ... </head>
  <body>

    <h3>Ihre Anmeldedaten:</h3>
    Benutzername:

  </body>
</html>
```


©STEIN 2022

Web-Template-Engines

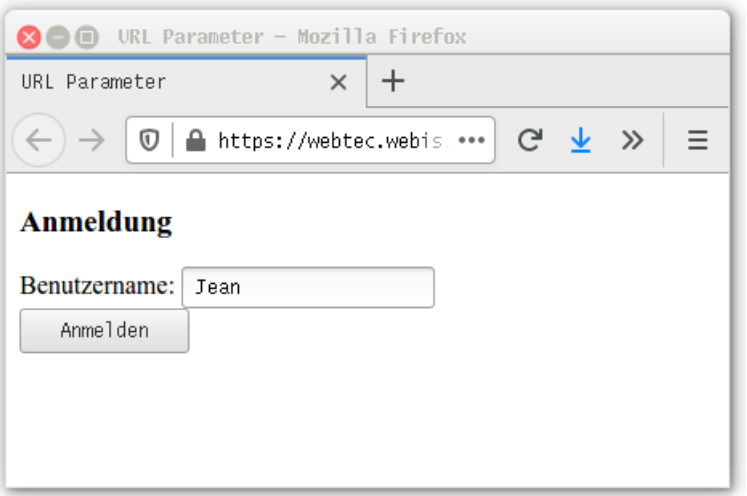
(1) JSP: URL-Parameter einlesen [Java: generiert] [jsp-Datei, jinja-Datei, php-Datei]

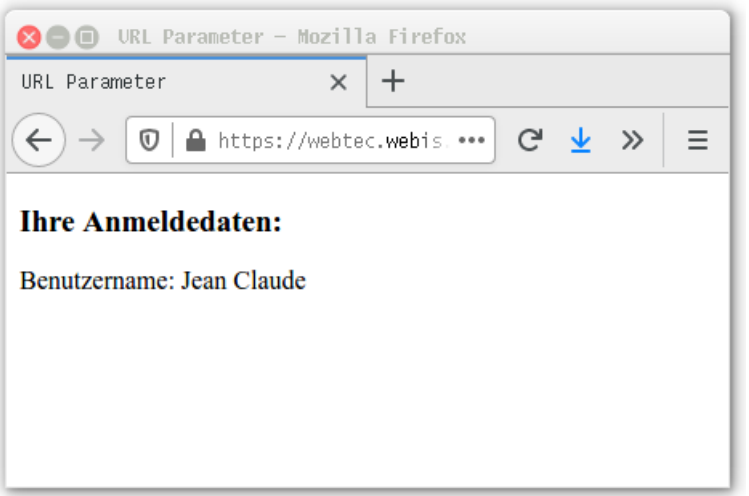
```
<!DOCTYPE html>
<html>
  <head> <title>URL Parameter</title> ... </head>
  <body>
    <%
      String user = request.getParameter("user");
      if ( user == null || "".equals(user) ) {
    %>
    <h3>Anmeldung</h3>
    <form action="readURLParam.jsp" method="get">
      <div>
        <label for="user">Benutzername:</label>
        <input type="text" name="user" id="user">
      </div>
      <input type="submit" value="Anmelden">
    </form>
    <% } else { %>
    <h3>Ihre Anmeldedaten:</h3>
    Benutzername: <%= user %>
    <% } %>
  </body>
</html>
```


©STEIN 2022

Web-Template-Engines

(1) JSP: URL-Parameter einlesen (Fortsetzung)





[JSP: jsp-Datei, Aufruf]

©STEIN 2022

Bemerkungen:

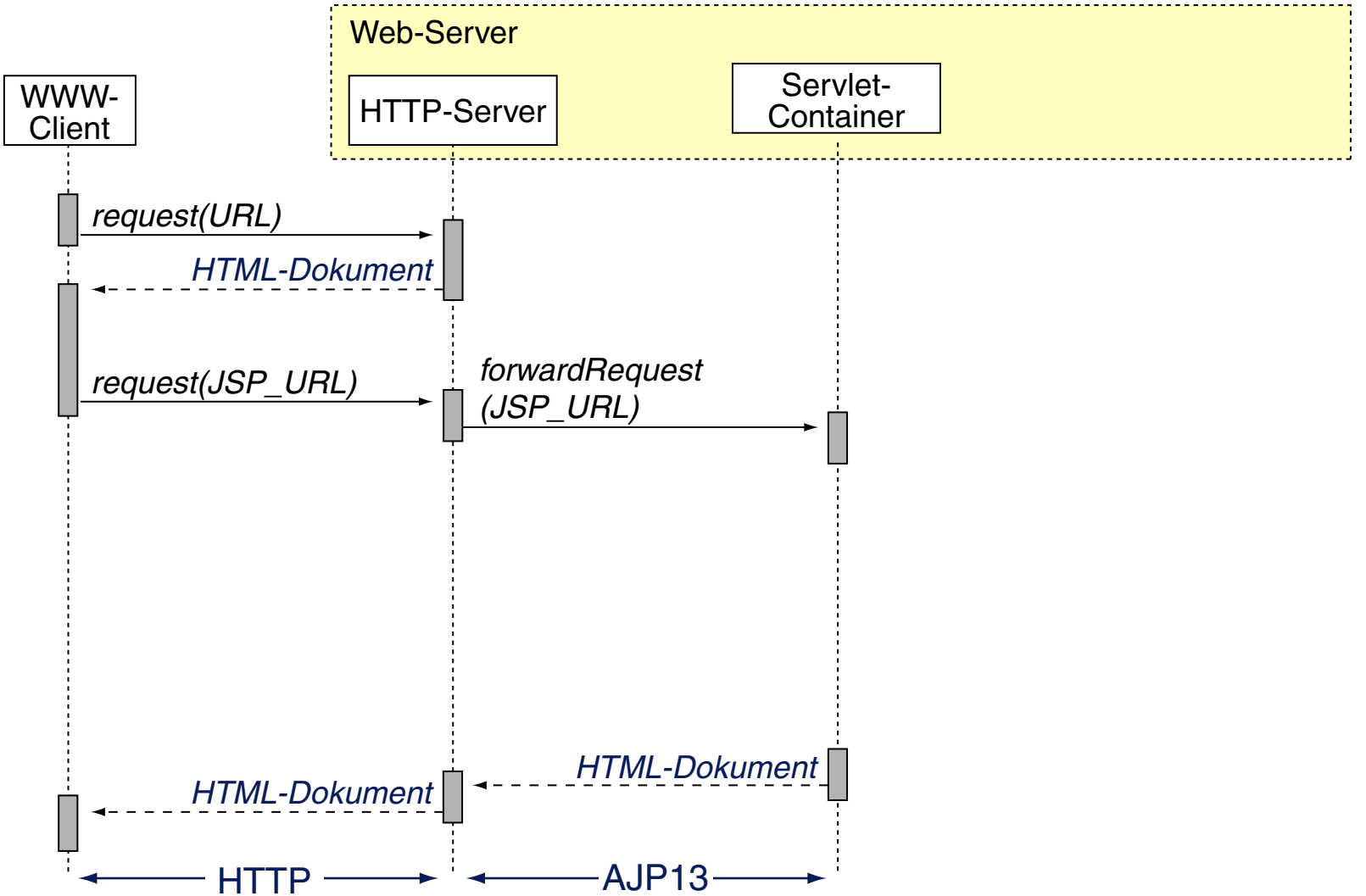
- Die `jsp`-Datei kombiniert HTML-Code und Java-Code zur Erzeugung einer HTML-Seite.
- Die Generierung des Servlets aus der `jsp`-Datei geschieht „on the fly“. Dabei wird berücksichtigt, ob die `jsp`-Datei zwischenzeitlich verändert wurde.
- Das Action-Attribut `<... action="readURLParam.jsp">` ist seit HTML5 optional. Falls es fehlt, wird als Default-Wert der aktuelle Pfad (hier: `readURLParam.jsp`) genommen.

©STEIN 2022

Web-Template-Engines

(1) JSP: Sequenzdiagramm Seitenauslieferung

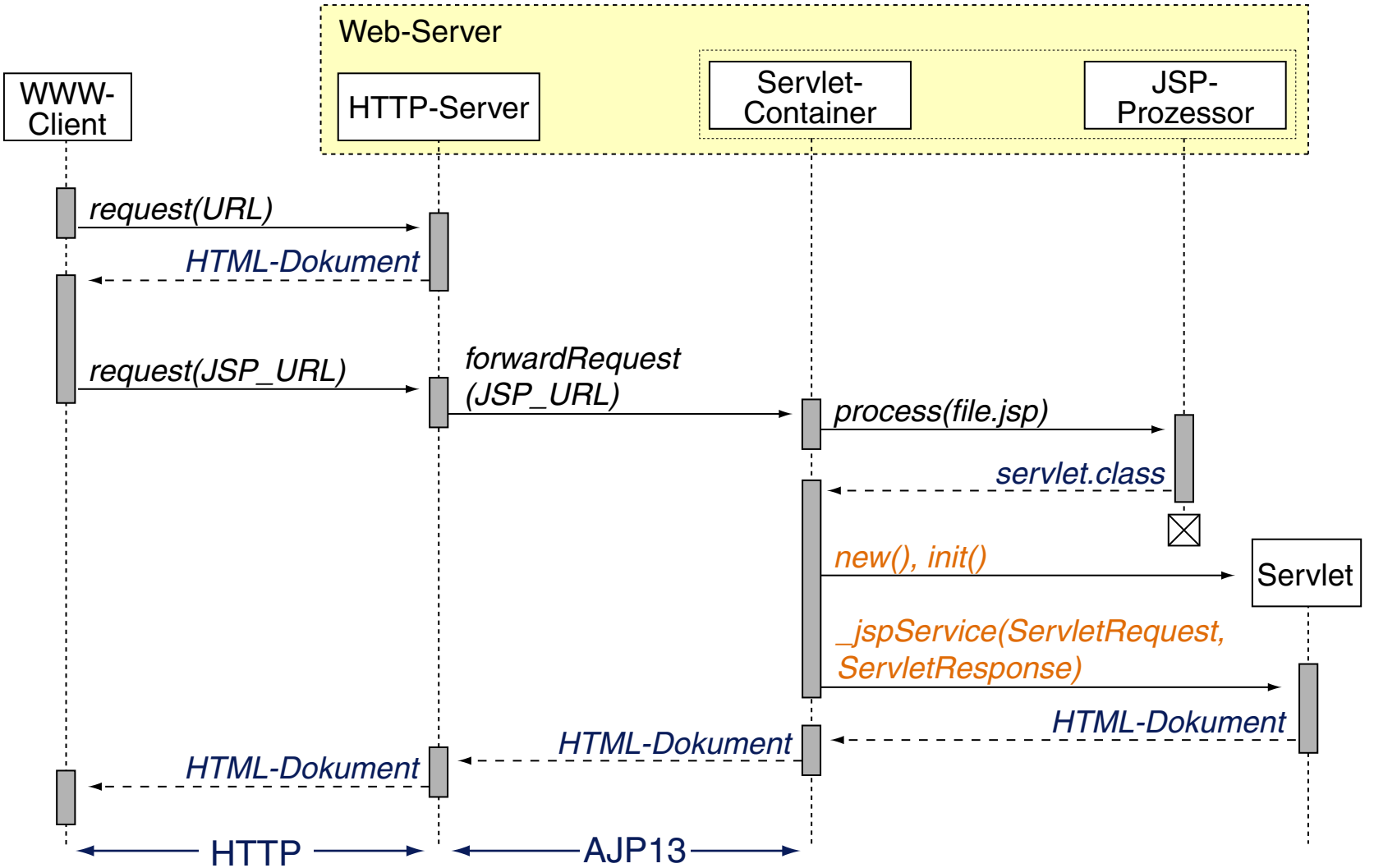
[statisch, CGI, Servlet, JSP, PHP]



©STEIN 2022

Web-Template-Engines

(1) JSP: Sequenzdiagramm Seitenauslieferung [statisch, CGI, Servlet, JSP, PHP]



©STEIN 2022

Web-Template-Engines

(1) JSP: Programmierkonzepte

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

©STEIN 2022

Web-Template-Engines

(1) JSP: Programmierkonzepte

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

Zur Programmierung gibt es drei Konzepte:

1. Java.
   (a) Java-*Scriptlet*: beliebige Java-Anweisungen.
   HTML-Syntax: `<%` *Code* `%>` Code wird Teil der `_jspService()`-Methode.
   (b) Java-*Expression*: Wert, der zur Laufzeit ermittelt und als String ausgegeben wird.
   HTML-Syntax: `<%=` *Code* `%>` Code wird Teil der `_jspService()`-Methode.
   (c) Java-*Declaration*: beliebige Java-Anweisungen.
   HTML-Syntax: `<%!` *Code* `%>` Code ist außerhalb der `_jspService()`-Methode.

2. JSP-Aktionen.
   Anbindung von Jakarta EE Komponenten.

3. JSP-Direktiven.
   Anweisungen, die direkt vom JSP-Prozessor verarbeitet werden.

©STEIN 2022

Web-Template-Engines

(1) JSP: Programmierkonzepte (Fortsetzung)

```
...
<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 {



    ...
  }
}
```


©STEIN 2022

Web-Template-Engines

(1) JSP: Programmierkonzepte (Fortsetzung)

```
...
<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>");


    ...
  }
}
```


©STEIN 2022

Web-Template-Engines

(1) JSP: Programmierkonzepte (Fortsetzung)

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

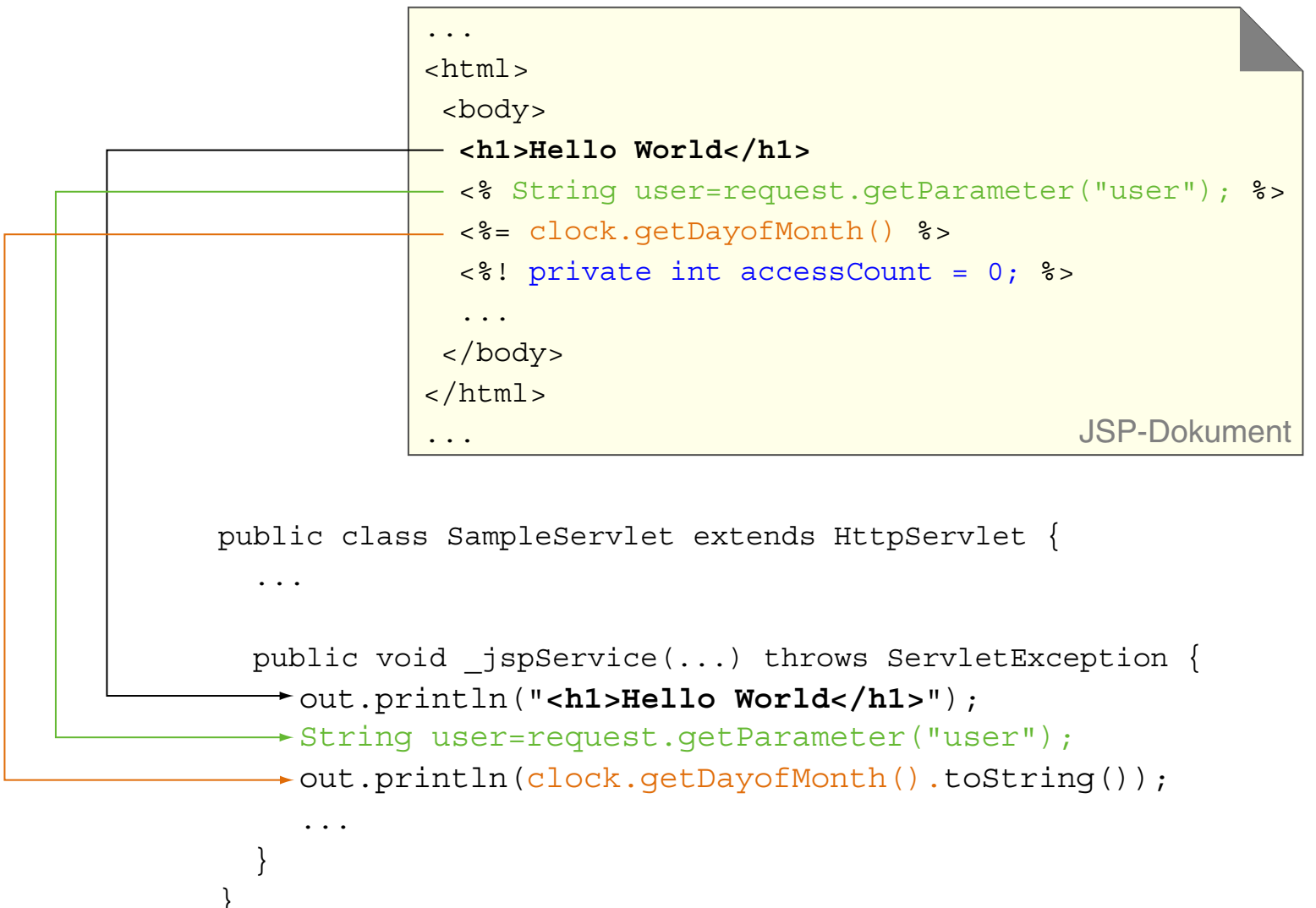
JSP-Dokument

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

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

    ...
  }
}
```


©STEIN 2022

Web-Template-Engines

(1) JSP: Programmierkonzepte (Fortsetzung)



©STEIN 2022

Web-Template-Engines

(1) JSP: Programmierkonzepte (Fortsetzung)

©STEIN 2022

Web-Template-Engines

(2) Jinja: URL-Parameter einlesen

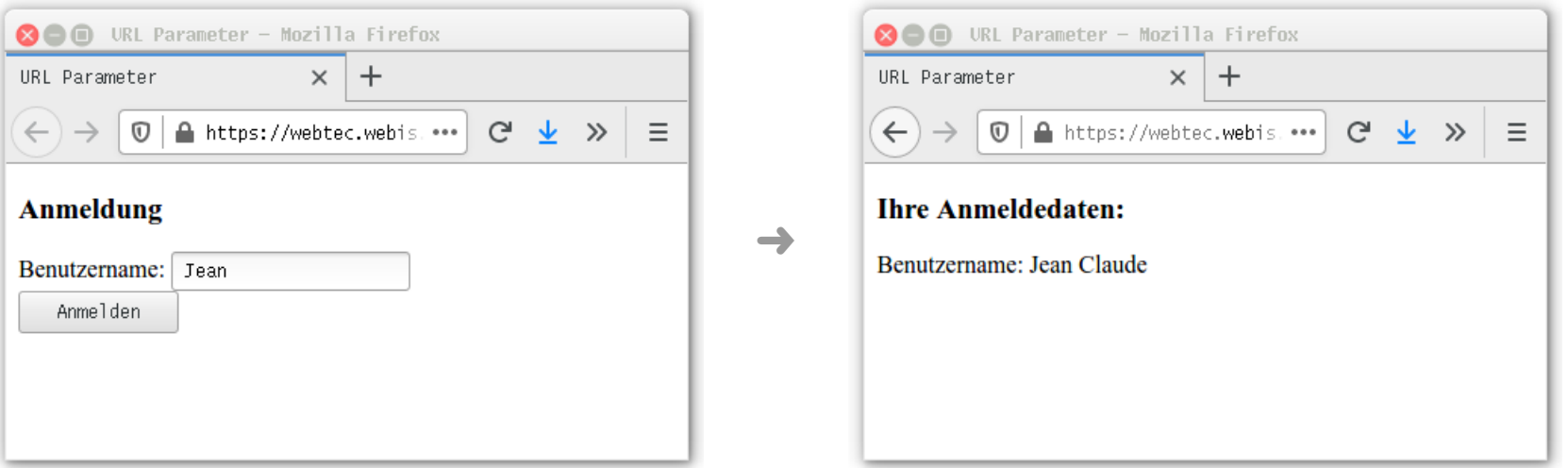
[Einordnung, Deployment] [jsp-Datei, jinja-Datei, php-Datei]

```
<!DOCTYPE html>
<html>
  <head> <title>URL Parameter</title> ... </head>
  <body>
    {%
      if request.args.get('user') is none
    %}
    <h3>Anmeldung</h3>
    <form action="ReadURLParam.jinja" method="get">
      <div>
        <label for="user">Benutzername:</label>
        <input type="text" name="user" id="user">
      </div>
      <input type="submit" value="Anmelden">
    </form>
    {% else %}
    <h3>Ihre Anmeldedaten:</h3>
    Benutzername: {{ request.args.get('user') }}
    {% endif %}
  </body>
</html>
```


©STEIN 2022

Web-Template-Engines

(2) Jinja: URL-Parameter einlesen (Fortsetzung)



[Jinja: jinja-Datei, Aufruf]

©STEIN 2022

Web-Template-Engines

(2) Jinja: URL-Parameter einlesen (Fortsetzung)

```
from flask import Flask, request, render_template

app = Flask(__name__, static_folder="web/static", static_url_path="/",
            template_folder="web/templates")

@app.route("/ReadURLParam.jinja")
def jinja2_read_url_param():
    return render_template("ReadURLParam.html.jinja", request=request)

def main():
    app.run(host="0.0.0.0", port=80,
            debug=True, threaded=False)

if __name__ == '__main__':
    main()
```


©STEIN 2022

Bemerkungen:

- Die `jinja`-Datei kombiniert HTML-Code und Python-Code zur Erzeugung einer HTML-Seite.
- Die Flask-Anwendung, die die HTML-Datei aus der `jinja`-Template-Datei generiert und ausliefert, wird nicht automatisch generiert (wie bei JSP), sondern ist manuell zu erstellen.
- Das Action-Attribut `<... action="ReadURLParam.jinja">` ist seit HTML5 optional. Falls es fehlt, wird als Default-Wert der aktuelle Pfad (hier: `ReadURLParam.jinja`) genommen.

©STEIN 2022

Web-Template-Engines

(3) PHP: URL-Parameter einlesen

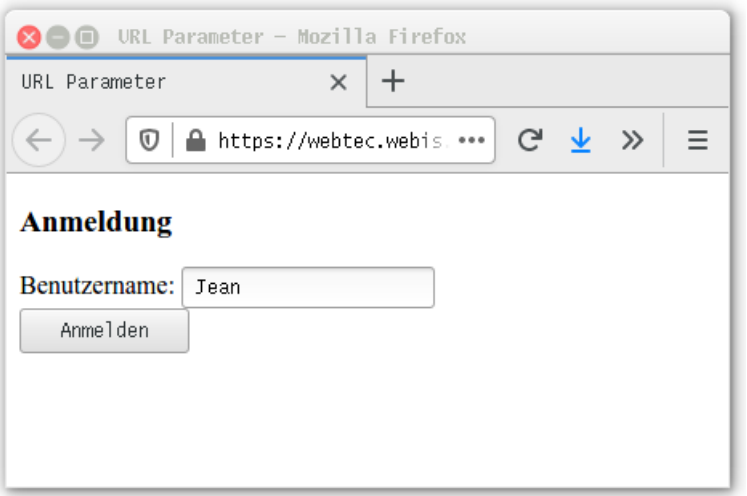
[Einordnung, Deployment] [jsp-Datei, jinja-Datei, php-Datei]

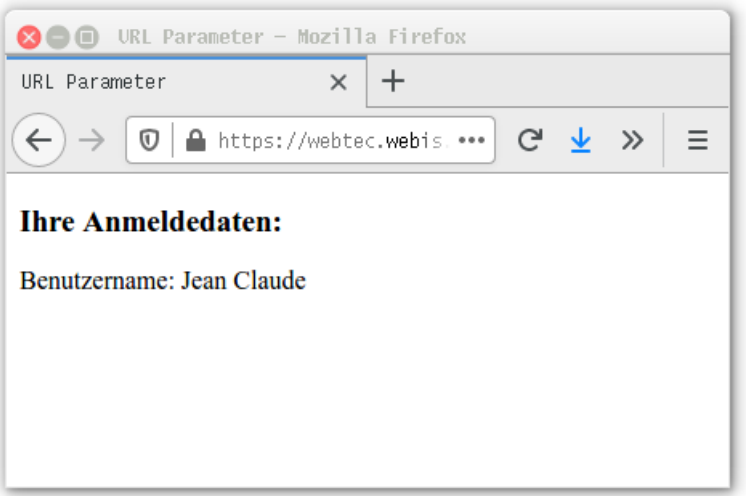
```
<!DOCTYPE html>
<html>
  <head> <title>URL Parameter</title> ... </head>
  <body>
    <?php
      $user = $_REQUEST['user'];
      if ( trim($user) == "" ) {
    ?>
    <h3>Anmeldung</h3>
    <form action="read-url-param.php" method="get">
      <div>
        <label for="user">Benutzername:</label>
        <input type="text" name="user" id="user">
      </div>
      <input type="submit" value="Anmelden"></td></tr>
    </form>
    <?php } else { ?>
    <h3>Ihre Anmeldedaten:</h3>
    Benutzername: <?php echo $user ?>
    <?php } ?>
  </body>
</html>
```


©STEIN 2022

Web-Template-Engines

(3) PHP: URL-Parameter einlesen (Fortsetzung)





[PHP: php-Datei, Aufruf]

©STEIN 2022

Bemerkungen:

- Die `php`-Datei kombiniert HTML-Code und PHP-Code zur Erzeugung einer HTML-Seite.
- Das Action-Attribut `<... action="read-url-param.php">` ist seit HTML5 optional. Falls es fehlt, wird als Default-Wert der aktuelle Pfad (hier: `read-url-param.php`) genommen.
- Keine getrennten (CGI-)Programme zur Gestaltung eines Web-Dialogs: HTML-Form, HTML-Antwort sowie der Programmcode zur Verarbeitung sind in derselben Datei.

©STEIN 2022

Server-Technologien

Quellen zum Nachlernen und Nachschlagen im Web

- Apache. *Apache HTTP Server Documentation.*
  httpd.apache.org/docs
- Eclipse. *Jakarta EE.*
  Jakarta EE Home
  Javadoc: Jakarta EE Platform API
- Hall. *Servlets and JavaServer Pages Tutorial Series.*
  Publications
- Oracle. *Java Servlet Tutorials.*
  javaee.github.io/tutorial/servlets.html
  www.oracle.com/java/technologies/servlet-technology.html
- Vogel. *Apache Tomcat - Tutorial.*
  www.vogella.com/tutorials/ApacheTomcat/article.html

©STEIN 2022