



openHPI: **Web-Technologien** Serverseitige Web-Programmierung

Prof. Dr. Christoph Meinel

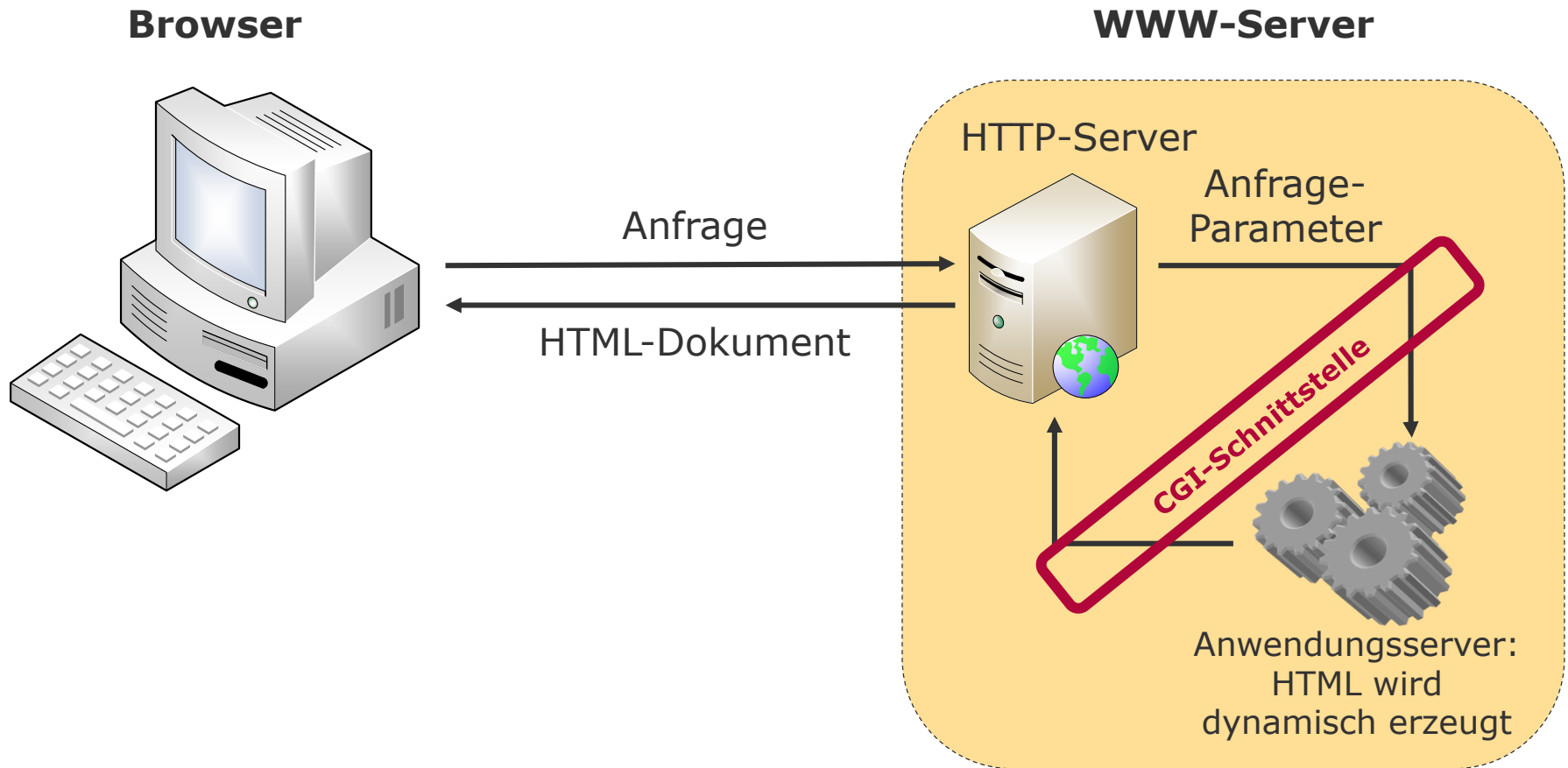
Hasso-Plattner-Institut

Universität Potsdam

Standard-Schnittstelle für serverseitige Programme zur dynamischen Erzeugung von HTML-Dokumenten: **CGI – Common Gateway Interface**

- Zunächst nur Laufzeitumgebung für externe Programme
 - z.B. für C, C++
 - dann Einsatz von Perl-Skripten, Aufruf über Kommandozeileninterpreter
- Zur schnelleren Ausführung, Entwicklung von Skriptinterpretern als Module im Webserver, z.B.
 - für Apache: mod_perl, mod_python, mod_php
 - für Internet Information Services: ASP, ASP.NET
- Parallel dazu: Aufkommen von Java Application Servern (für Servlets)
- **Heute:** Neben Interpreter-Modulen auch neue CGI-Standards, z.B.
 - WSGI (Python), FastCGI, SCGI

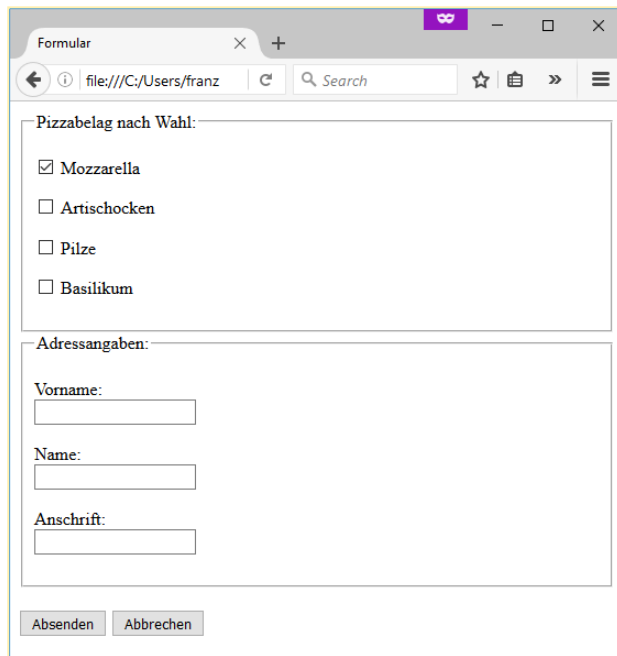
Erzeugung dynamischer Webseiten



CGI-Schnittstelle – Parameterübergabe (1/3)

Zur Steuerung der Anwendungsprogramme auf Seiten des WWW-Servers müssen Parameter übergeben werden können

- **Beispiel:** Ausfüllen und Abschicken eines HTML-Formulars



```
<form action="bestellen.php" method="POST">

<fieldset>
  <legend>Pizzabelag nach Wahl:</legend>
  <p>
    <input type="checkbox"
      name="zutat" value="mozzarella" checked>
      Mozzarella<br>
    <input type="ch...
    ...
  </fieldset>
</form>
```

CGI-Schnittstelle – Parameterübergabe (2/3)

GET

- Formulareingaben werden an die URL, die im action-Attribut angegeben ist, hinter „?“ als **Key-Value-Paare** angehängt
- HTTP-Server übergibt die Formulardaten an das über die angegebene URL aufgerufene CGI-Programm in der Umgebungsvariable **QUERY_STRING**
 - Beispiel: Google-Suche:
<https://www.google.de/?q=deutsche+online-kurse>
- Umgebungsvariablen:
 - Parameter, die beim Start an Programm übergeben werden
 - sind im gesamten Kontext des Programms definiert

CGI-Schnittstelle – Parameterübergabe (3/3)

POST, PUT, PATCH, DELETE

- Es wird ein HTTP-Request an die im action-Attribut angegebene URL initiiert
- Formulardaten werden im Body des HTTP-Requests übergeben
- Body des HTTP-Requests wird dem in der URL aufgerufenen CGI-Programm über die Standard-Eingabe zur Verfügung gestellt
 - Beispiel: Verfassen eines Beitrags in einem Online-Forum

CGI-Schnittstelle – Umgebungsvariable (1/2)

Umgebungsvariablen des HTTP-Servers

- Server belegt vor Aufruf des durch den Client initiierten CGI-Programms bestimmte Umgebungsvariablen, z.B.
 - CONTENT_LENGTH, SERVER_NAME, REQUEST_METHOD
 - PATH_INFO, SCRIPT_NAME, QUERY_STRING, ...
- Per CGI kann auf Umgebungsvariablen zugegriffen werden, z.B. für Auswertung
 - Beispiele - Portnummer:
 - Ruby: `ENV["SERVER_PORT"]`
 - PHP: `$_SERVER["SERVER_PORT"]`

CGI-Schnittstelle - Umgebungsvariable (2/2)

Umgebungsvariablen des HTTP-Servers

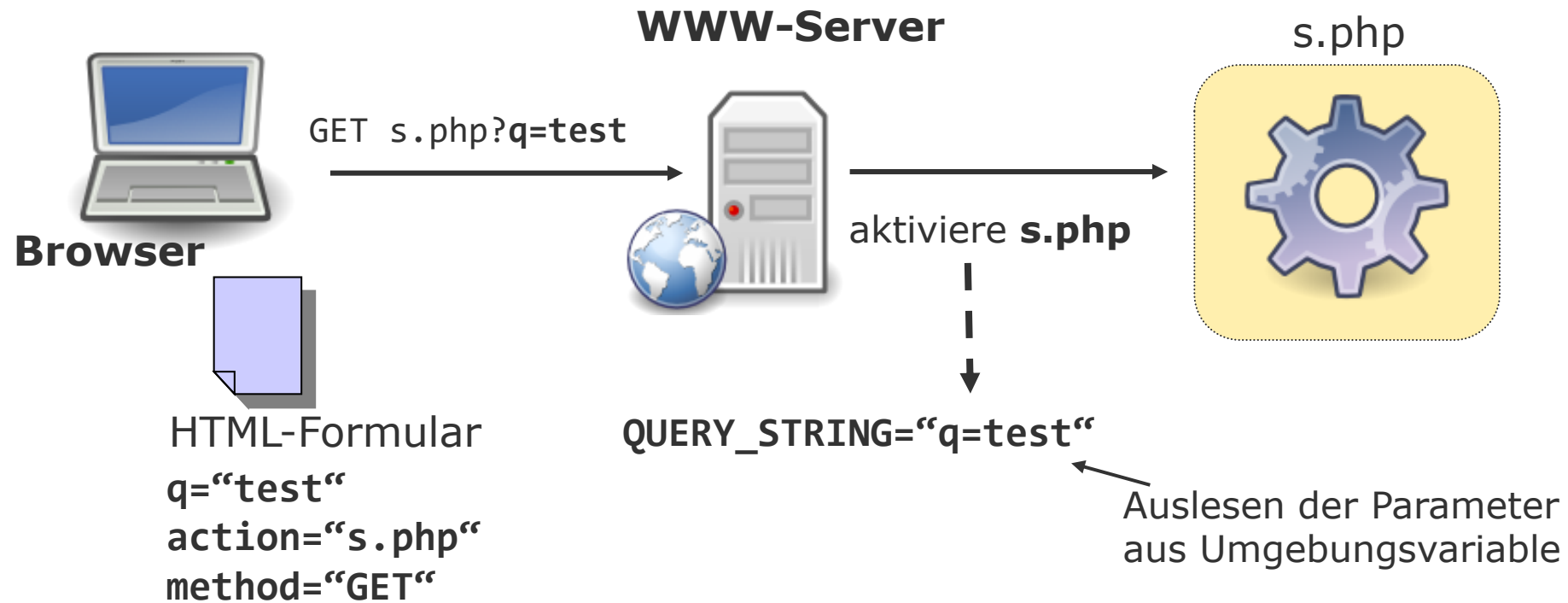
- Auch auf andere Teile des HTTP-Requests, z.B. per GET oder POST gesendete Parameter, kann zugegriffen werden
 - Beispiel PHP: `<?php echo $_POST['zutat'] ?>`

```
<form action="bestellen.php" method="POST">

<fieldset>
  <legend>Pizzabelag nach Wahl:</legend>
  <p>
    <input type="checkbox"
      name="zutat" value="mozarella" checked>
      Mozarella<br>
    <input type="ch...
  ...
</fieldset>
</form>
```

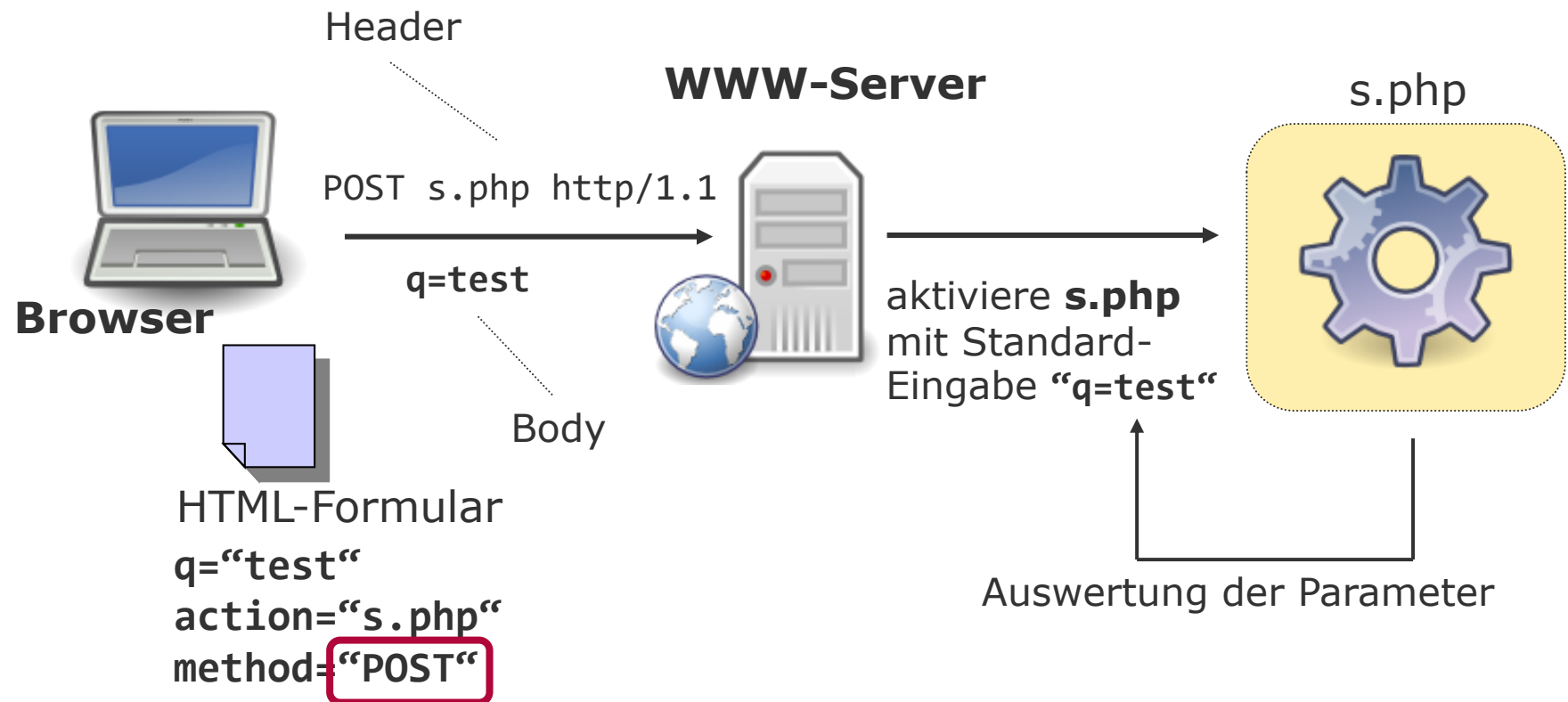
CGI-Schnittstelle – Parameterübergabe (1/2)

Parameterübergabe mit GET



CGI-Schnittstelle – Parameterübergabe (2/2)

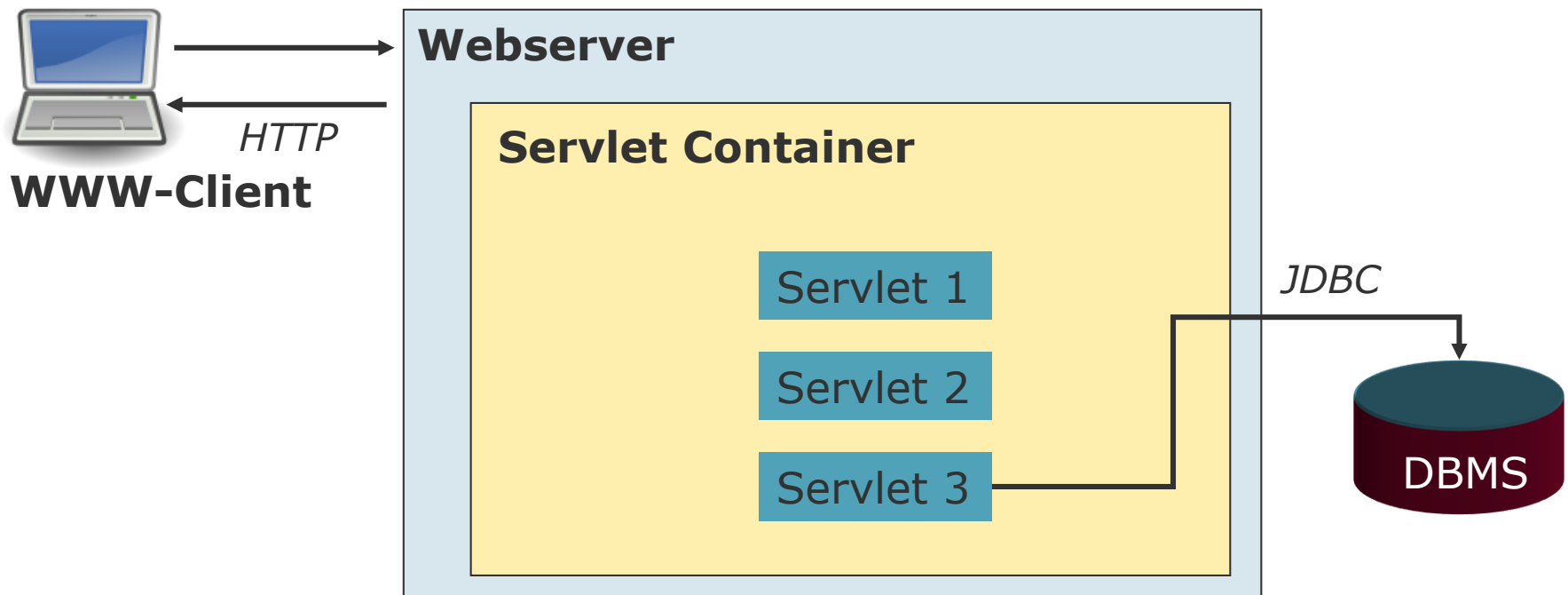
Parameterübergabe mit POST



Serverseitiges Java (1/2)

Ausführungsumgebung

- Webserver + Servlet-Container, z.B. Apache + Tomcat
- Application Server, z.B. Oracle Glassfish, WildFly



Serverseitiges Java (2/2)

■ **JSP Java Server Pages**

- ermöglichen Einbinden von Template-Texten in HTML-Dokumente
- Verwendung von Taglibs, z.B. JSTL (Java Standard Tag Library)

■ **Servlets**

- auf Application Server / im Servlet-Container ausgeführte Java-Programme
- reagieren auf von Webseite erstellten HTTP-Request mittels einem HTTP-Response

■ **JSF Java Server Faces**

- Framework zur Erstellung von Web-Applikationen

■ **Facelets**

- Ersetzen JSP im Zusammenspiel mit JSF

Webserver-Module

- **CGI:** Webserver-Prozess kommuniziert mit separatem Interpreter-Prozess
 - schlechte Performance
 - gute Sicherheit: Prozesse sind voneinander isoliert
- **Webserver-Module:** `mod_perl`, `mod_php`, `mod_python` ...
 - Interpreter-Prozess eingebettet in den Webserver-Prozess
 - Gute Performance für Requests, die dynamisch Code ausführen müssen
 - Aber: Zusätzlicher Speicherbedarf auch für Requests auf statische Ressourcen
 - Sicherheit der Skripte (und der Interpreter) beeinträchtigt die Sicherheit des Webserver-Prozesses

- Entwickelt als Nachfolger von CGI
- Übernimmt das Sicherheitskonzept von CGI; versucht, die Performance zu verbessern
 - Das Starten eines neuen Prozesses bei jedem Request fällt weg
 - (Beliebig viele) Anwendungsprozesse können mehrere Requests hintereinander verarbeiten: Overhead für Prozessstart fällt weg
- Kommunikation über **Sockets**
 - Vorteil: Anwendungsprozess kann auch auf separatem Server laufen
- Statische Inhalte werden direkt vom Web-Server ausgeliefert