

10 World-Wide Web

10.3 Dynamische Ressourcen (fortgesetzt)	
10.3.4 PHP	2
10.3.5 JSP	11
10.3.6 ASP	15
10.4 Sitzungen	17
Zusammenfassung	24

10.3.4 PHP

Dokumentorientierung statt Programmorientierung

mit Skriptsprache PHP

Gothic Club revisited: Fragment aus einem möglichen Servlet:

```
...
res.setContentType("text/html");
PrintWriter out = res.getWriter();
out.println("<html>");
out.println("<head><title>Reply</title></head>");
out.println("<body>");
if (!eligible) {
    out.println("<br><i>Sorry</i>, ");
    out.println(firstname.split(" ")[0]);
    out.println(", you are not eligible.");
}
else .....
```

Lästige `out.println`-Anweisungen für einen Großteil
des HTML-Textes - den statischen Teil!
Ähnlich mit CGI und anderen Sprachen!

*Umgekehrter Ansatz:
statt HTML in Code einzubetten
wird Code in HTML eingebettet.*

```
...  
println("<br><i>Sorry</i>, " );  
firstname.split(" ")[0]);  
println(", you are not eligible.");}  
...  
  
<br><i>Sorry</i>,  
<%= firstname.split(" ")[0])%>  
, you are not eligible.
```

Programmorientierung

Dokumentorientierung

Dokumentorientierung mit PHP

- PHP ist eine leistungsfähige *Skriptsprache*, die speziell für in HTML eingebetteten Code entwickelt wurde.
- PHP ist ursprünglich das Akronym für „*personal home pages*“.
- PHP ist heute weit verbreitet und wird von den meisten Web Servern unterstützt.
- Eine PHP-Seite `page.php` wird interpretativ verarbeitet; der Interpretierer (*PHP processor*) wird entweder in einem separaten Prozess mit *CGI*-Umgebung, vorzugsweise aber innerhalb des Web Servers ausgeführt (z.B. als *Apache „module“*).

PHP-Code in einem Dokument wird wie folgt in Begrenzer (delimiters, wie Tags) eingeschlossen:

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

oder <?php ... code ... ?>

oder <? ... code ... ?>

Gothic Club re-revisited: gothic.php

```
<html><head><title>Reply</title></head><body>
<?      $sex = $_GET['sex'];
        $age = $_GET['age'];
        $firstname = $_GET['firstname'];
        $eligible =
                $sex=='Male'    && $age>'70'    ||
                $sex=='Female' && $age>'75';
        if (!$eligible) {
                $names = explode(' ', $firstname); ?>
                <br><i>Sorry</i>,
<?=            $names[0].', '                      ?>
                you are not eligible.
<?      } else echo '<br>Ok.';                      ?>
</body></html>
```



Che

Firs Sorry, Mary, you are not eligible.

Age

ema

Sex:

Sen

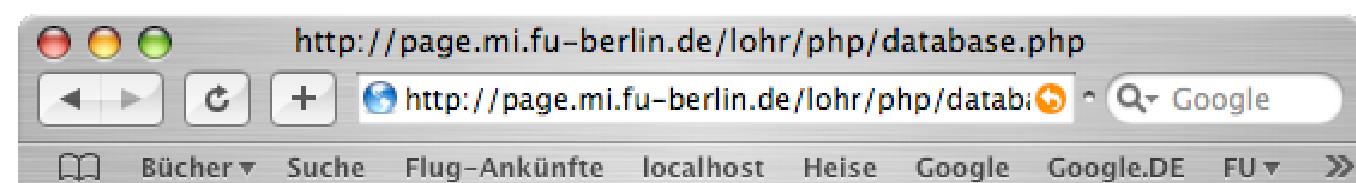
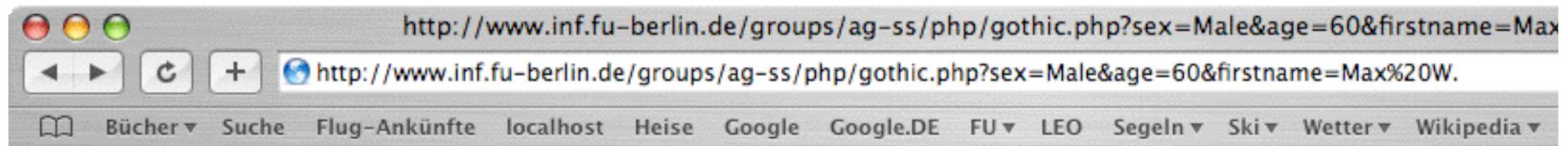
Üblich ist die Verwendung eines PHP-Dokuments .php, das die verschiedenen benötigten Seiten generieren kann, abhängig von den Anfragedaten - z.B. so:

```
... <body>
<? if($_SERVER[QUERY_STRING] == "") { ?> ... (Formular) ...
<? } else ?> ... (Antwort) .......
```

(Alternative: POST statt GET verwenden - „Postback“)

Erfreulich:

wegen der engen Anbindung des PHP-Interpretierers an den Web Server ist dieser in der Lage, im Fehlerfall mit spezifischen Fehlermeldungen zu reagieren. 2 Beispiele:



Start ...

- Aktuelle PHP-Version ist **PHP 5**
- *Objektorientierung seit PHP 3:*
Klassen (`class`) mit Vererbung (`extends`), keine Schnittstellen.
- (seit PHP 4:) Variable können Verweise auf „resource“-Objekte enthalten; das sind *externe Objekte* wie Dateiverbindungen, Datenbankverbindungen, Fenster etc.
(Beispiel: `$out = fopen("myfile", "w");`)
- **PEAR - PHP Extension and Application Repository:**
Bibliothek von Code-Paketen für verschiedene Anwendungsgebiete
(<http://pear.php.net/manual/en>)

10.3.5 JSP

(Java Server Pages)

- JSP ist das Java-Analogon zu PHP.
- Aus einem HTML-Text mit eingebettetem Java-Code wird dynamisch ein *Servlet* erzeugt (*JSP Compiler*), übersetzt und ausgeführt.
- Web Server muss dies unterstützen (vgl. *Servlets*, 10.3.2)!

PHP

page.php

--

<? PHP script ?>

<?= PHP expression ?>

--

JSP

page.jsp

<%! Java declarations %>

<% Java statements %>

<%= Java expression %>

<%@ JSP directive %>

Vorvereinbarte Objekte in JSP

HttpServletRequest request
mit Methoden getParameter, getMethod, ...

HttpServletResponse response
mit Methoden setHeader, getWriter, ...

JspWriter out
mit Methoden println, flush, ...

HttpSession session (siehe unten unter „Sitzungen“)
mit Methoden getSession, invalidate, ...

... weitere

Gothic Club revisited: gothic.jsp

```
<html><!-- aufgerufen aus gothic.html - vgl. PHP(S.7) -->
<head><title>Reply</title></head>
<body>
<%    String sex = request.getParameter("sex");
      String age = request.getParameter("age");
      String firstname = request.getParameter("firstname");
      boolean eligible =
          sex=="Male"    && age.compareTo("70")>0    ||
          sex=="Female" && age.compareTo("75")>0;
      if (!eligible) {                                %>
          <br><i>Sorry</i>,
<%=        firstname.split(" ")[0]                %>
          , you are not eligible.
<%    } else out.println("<br>Ok.");             %>
</body>
</html>
```

10.3.6 ASP

(Active Server Pages - Microsoft)

- Ähnlich PHP und JSP
- aber verschiedene (Skript-)Sprachen einsetzbar!
- gilt als überholt, obsolet
- wird nicht mehr unterstützt
- Nachfolger ASP.NET hat inhaltlich nichts mehr mit ASP gemein

PHP: page.php

<? PHPscript ?>

<?= PHPexpression ?> <%= expression %>

ASP: page.asp

<% script %>

<%= expression %>

<%@ ASPdirective %>

insbesondere:

<%@ language="JScript" %>

(Voreinstellung ist "VBScript")

JSP: page.jsp

<%! Java declarations %>

<% Java statements %>

<%= Java expression %>

<%@ JSPdirective %>

10.4 Sitzungen

- Eine HTTP-basierte Interaktion zwischen dem Web Server und einem Klienten ist *kein längerer Dialog*, sondern eine kurze *Anfrage/Antwort-Interaktion*.
- Häufig stehen aber aufeinanderfolgende Interaktionen eines Benutzers in Beziehung zueinander: sie bilden eine **Sitzung** (*session*). Typisches Beispiel: Warenkorb zusammenstellen.
- Der Server muss **Zustandsdaten** zu einer solchen Sitzung führen.
- Wie kann der Server erkennen, dass eine Anfrage zu einer bestimmten Sitzung gehört?

- **Sitzungskennung (session identifier)** verwenden, die bei jeder Anfrage mitgeliefert wird (große Zufallszahl).
- Bei der ersten Interaktion einer Sitzung wird beim Server durch den Code der dynamischen Seite eine **Sitzungskennung** generiert und in die Antwort eingebaut.
- Jede weitere Anfrage der Sitzung muss die Sitzungskennung tragen.
- 3 Alternativen:
 - *Unsichtbares Textfeld* im Formular wird mit Kennung vorbelegt.
 - *URL rewriting*: alle Links in einer ausgelieferten Seite werden *ad hoc* um die Kennung erweitert (!).
 - *Cookies* als Sitzungskennungen.

- **Cookie** = kleiner Datensatz (Name, Wert), der vom Server generiert und beim Klienten gespeichert wird.
- Cookies werden durch HTTP unterstützt (!): Kopfzeile
 - Set-Cookie:** Name=Wert [; Max-Age=Sekunden ;] wird in der Antwort des Servers an den Klienten geschickt.
Server und Browser speichern das Cookie. Bei einer weiteren Anfrage an den Server schickt der Browser alle Cookies dieses Servers mit - unter Verwendung der Kopfzeile
 - Cookie:** Name1=Wert1 ; Name2=Wert2 ;
- Wenn **Max-Age** angegeben war, sollte der Browser das Cookie nach Ablauf der angegebenen Zeitspanne löschen. Ansonsten hängt es von den Einstellungen des Browsers ab, wann Cookies gelöscht werden.

Sitzungskennungen in Cookies

- Bei der ersten Anfrage (auf die Startseite einer Sitzung) generiert der Server eine große Zufallszahl als neue Sitzungskennung, z.B. "977e5d8ae8500c456ab1fca6cbaa12af",
- ... und fügt der Antwort ein Cookie mit Namen `sessionID` bei
 - mit der Kennung als Wert:
`Set-Cookie: sessionID=977e5d8ae8500c456ab1fca6cbaa12af`
- Bei der nächsten Anfrage in der Sitzung schickt der Browser dieses Cookie mit und versorgt so den Server mit der Sitzungskennung:
`Cookie: sessionID=977e5d8ae8500c456ab1fca6cbaa12af.`
- Bei jeder Anfrage und Antwort wird das Cookie mitgeschickt. Der Server beendet die Sitzung mit
`Set-Cookie: sessionID=977e5d.....a12af; Max-Age=0.`

- *Achtung:* aus Datenschutzgründen sind Cookies umstritten:
- Längerlebige Cookies spiegeln - in gewissem Umfang - das Verhalten des Benutzers im Web wieder; in welchem Umfang, hängt von den Informationen ab, die die Server in den Cookies ablegen.
- Web-Seiten mit *klientenseitig* ausgeführtem Code (→ 10.5) können auf Cookies zugreifen, ohne dass der Benutzer das merkt.
- → Einstellungen im Browser, Warnungen des Browsers, regelmäßiges Löschen von Cookies!

Cookies und Sitzungen werden von den verschiedenen Techniken für dynamische Ressourcen gut unterstützt.

Beispiel: **Cookies** in JSP: `javax.servlet.http.Cookie`

`httpServletResponse.addCookie(cookie)`
hängt Cookie-Kopfzeile an den Antwort-Kopf an.

`httpServletRequest.getCookies()`
liefert die vom Browser übermittelten Cookies
als Feld von `Cookie`-Objekten.

Beispiel: Sitzungen in JSP: `javax.servlet.http.HttpSession`

`httpServletRequest.getSession()`

liefert Objekt vom Typ `HttpSession`: dies *abstrahiert* von der für die Sitzungskennung verwendeten Technik (konfigurierbar: entweder *URL rewriting* oder *Cookies*).

Wenn die Anfrage bereits zu einer Sitzung gehört, wird diese geliefert, wenn nicht, wird eine neue Sitzung erzeugt. Methoden von `HttpSession`:

- `getId()` liefert die Sitzungskennung (Zeichenkette)
- `setAttribute(name, object)` assoziiert benanntes Objekt mit der Sitzung
- `getAttribute(name)` liefert benanntes Objekt
- `invalidate()` beendet die Sitzung

Zusammenfassung

- *Programmorientierung (CGI, Servlets)* ist gut geeignet für „sehr dynamische“ Ressourcen. CGI ist „alte Technologie“.
- *Dokumentorientierung (PHP, JSP, ASP)* ist gut geeignet für „wenig dynamische“ HTML-Dokumente. (Allerdings wird JSP-Text intern in ein Servlet umgewandelt.)
- JSP/Servlets sind Java-Technologien - mit allen Vor- und Nachteilen. (Java-basierter Server nötig.)

- PHP ist populäre Skriptsprache, die von allen Web Servern unterstützt wird und für kleine Projekte geeignet ist.
- ASP ähnelt PHP und JSP, gilt aber als veraltet.
Microsoft propagiert das *Web Framework ASP.NET* .
- Das Konzept *Sitzungen* ist unverzichtbar für komplexere Web-Anwendungen. Es wird in den meisten Technologien für dynamische Ressourcen gut unterstützt.
- *Cookies* gehören zu den Konzepten des HTTP-Protokolls.
Sie sind vielseitig verwendbar - und missbrauchbar (*privacy!*!).

Quellen

PHP: <http://www.php.net/>

<http://php.oregonstate.edu/manual/en/>

JSP: <http://www.oracle.com/technetwork/java/javaee/jsp/index.html>

<http://www.apl.jhu.edu/~hall/java/Servlet-Tutorial>

ASP: <http://msdn.microsoft.com/en-us/library/aa286483.aspx>

ASP.NET: <http://www.asp.net>

Cookies - RFC 2109: <http://www.ietf.org/rfc/rfc2109.txt>

RFC 2965: <http://www.ietf.org/rfc/rfc2965.txt>