NTNU
Kunnskap for en bedre verden

Leksjon 6: Tilstandsbevaring med cookies og sessions

Svend Andreas Horgen, IDI Kalvskinnet, NTNU
Lærestoffet er utviklet for emnet IINI3003 Webprogrammering med PHP

Resymé: Denne leksjonen presenterer løsninger på et kjempestort problem når det kommer til utvikling av websteder: Hvordan bevare informasjon som brukeren har valgt eller oppgitt fra side til side og evt. over tid? Boka gjennomgår i kapittel 7 de grunnleggende teknikkene: bruk av URL og skjulte elementer. Leksjonen ser nærmere på praktisk bruk av URL og cookies. Med sessions er det mulig å bevare informasjon på en svært så enkel måte. De ulike metodene for tilstandsbevaring har ulike fordeler. Alle kan ofte (men ikke alltid) brukes for å løse samme problem. Som webprogrammerer bør du beherske alle metodene, selv om sessions nok er den metoden som oftest brukes i praksis.

1. Kort resymé av det som står i boka

Boka går nokså grundig igjennom problematikken knyttet til tilstandsbavaring i kapittel 7. Her er en kort oppsummering, før vi skal se på et konkret eksempel på praktisk bruk. I øvingen skal du få implementere din egen handlekurv-løsning. Det er lettere enn du tror! Følgende teknikker for tilstandsbevaring eksisterer kan brukes i PHP:

- Skjulte elementer (Hidden)
- URL (bevare informasjon i spørrestrengen)
- Informasjonskapsler (Cookies)
- Sessions

1.1. Klientsiden

Siden HTTP-protokollen ikke tar vare på informasjon mellom ulike forespørsler, er det utviklet teknikker for å bevare tilstandsinformasjon. Boka tar opp kreativ bruk av URL, skjulte elementer, bruk av cookies og sessions.

Med skjulte elementer kan informasjon fraktes ikke bare til neste skjema, men helt fram til målet via mange mellomliggende skjema. Fordelen sammenlignet med bruk av URL er at informasjonen kan skjules ved å bruke POST-metoden, men informasjonen vil likevel kunne endres av uærlige brukere og hackere ved å sjekke kildekoden og endre denne (hvordan, tror du?). Bevaring av informasjon direkte i URL kan passe i enkelte tilfeller, og er særlig hendig når vi ikke vil bruke skjema. Adressestrengen så vel som koden blir derimot fort uoversiktlig i og med at lenkene kan bli veldig lange. Cookies gir mulighet for å lagre informasjon på klientmaskinene, noe som gjør det mulig å tilpasse innholdet spesielt for hver enkelt person i tilfeller hvor ikke brukernavn og passord benyttes (hvilket igjen krever bruk av en database for å få skreddersydd informasjonen).

Felles for alle tre teknikkene er at det ikke er noen garanti for at brukeren endrer innholdet, uavhengig av om dette ligger i skjulte felt, cookies eller URL. Tenk litt på hvordan dette er mulig for hvert tilfelle.

1.2. Tjenersiden

Sessions og cookies er to teknikker for å huske informasjon om brukere som begge er lette å bruke. Måten som informasjonen lagres er ulik: Sessions bruker temporære filer på *tjenermaskinen* mens cookies lagrer små filer på *klientmaskinen*. Bruk av notasjonen med `$_SESSION` gjør ikke bare koden sikrere, men er også med å skille session-variabler fra vanlige variabler på en fin måte. I eldre versjoner av PHP (og derfor også mange eksisterende script til nedlasting fra web) vil du kunne se bruk av session-funksjoner, noe som baserer seg på at `register_globals=on`.

Boka gjennomgår et enkelt script som oppdaterer en teller for hver gang en side gjenoppfriskes. Denne telleren er knyttet til en og en besøkende, siden variablene som inngår i en session er knyttet til en bestemt bruker. Tellerens innhold blir bevart til tross for mange etterfølgende, uavhengige forespørsler, noe som boka forklarte med at den ble lagret i en spesiell fil på tjeneren, hvis filnavn inneholder verdien til en spesiell nøkkel (SID) som blir

sendt med en cookie, en URL eller et skjult felt for hver forespørsel. Puh! Logikken er ganske klar – uten SID er det umulig å si hvem som ”eier” de ulike settene av variabler.

Både cookies og session-variabler kan fjernes. Funksjonen `unset()` opp mot riktig element i den superglobale matrisen anbefales. I større problemstillinger er det fort gjort å få feil i programflyten, og da er bruk av funksjonen `print_r()` meget hendig for å sjekke innholdet i matrisene `$_POST`, `$_GET`, `$_SESSION` og `$_COOKIE`. Dette skal du få se et konkret eksempel på mot slutten av denne leksjonen.

2. Et nettsted som skreddersyr reklame

Mange utvikler internettsider for andre, og må i så måte tilfredsstille kundens behov. Med kunnskap om tilstandsbevaring kan du lage virkelig flotte løsninger som er brukervennlige. Tilstandsinformasjonen kan derimot også brukes på en mer kreativ måte, til vinning for nettstedet. Vi tar nå for oss et fiktivt ”konsulentoppdrag”, som er hentet fra kapittel 7.3.2 i boka.

2.1. Oppgaven

Redaktøren i portalen til et lite tettsted må skaffe seg mange sponsorer for å tjene penger. Innbyggerne besøker portalen daglig for å få med seg de siste oppdaterte sportsresultatene, arrangementene, konsertene, dronningbesøkene og så videre. Sponsorene legger vekt på at annonsene som legges ut på nett, skal føre til best mulig profilering for nettopp deres sak. De er klar over at portalen ikke kan legge ut all reklame på forsiden samtidig.

En stund har det vært utprøvd å la en sponsor få ha reklame på forsiden i en hel uke, men flere av sponsorene truer nå med å trekke seg fra portalen, i og med at de altfor sjelden får online-tid med innbyggerne. Nettredaktøren får en idé: Ulike folk har ulike interesser, og det ideelle hadde vært om ikke alle fikk se det samme innholdet. Reklame fra sportsbutikken vil tiltale noen, mens andre mer sannsynlig faller for informasjon om billig frukt på lokalbutikken. Tilsvarende vil nyheten fra kommunen om at det snart er kulturminnedager, være mer interessant for gamle enn unge, som på sin side kanskje heller vil lese nyheten om det at det planlegges et olabilløp.

2.2. Overordnet planlegging av løsningen

En løsning vil være å si at det brukeren i hovedsak gjør, er å lese nyheter. Mer nøyaktig – brukeren leser trolig de nyhetene som har interesse. Ved å kategorisere nyhetene, og så registrere hvilke kategorier som til enhver tid er forespurt, kan nettstedet bygge opp et bilde over hvem brukeren er. En slik profil vil kunne brukes på forskjellig måte, men mest nærliggende er å se på antall treff. Dersom kategorien *sport* har 5 treff (det vil si at brukeren har besøkt 5 nyheter som er sportsrelaterte), mens kategorien *kultur* bare har 1 treff, er det best å vise et reklamebilde som kommer fra den lokale sportsforretningen. En mer sofistikert analyse av dataene som lagres i profilen vil selvsagt gi en mer nøyaktig og rettferdig fordeling av reklamebilder. Slik kan en også ta hensyn til at en bruker som leser 100 sportsnyheter en dag, og deretter en og en kulturnyhet de neste fire ukene, trolig er mer interessert i kultur enn i sport, på tross av at antallet av kultur ikke overskrider sport.

Dersom disse opplysningene lagres i for eksempel cookies, vil portalen kunne samle opp informasjonen over tid. Det er ikke så farlig om cookien hos enkelte brukere slettes eller ikke tillates, da er det bare å presentere en standard utgave av nettstedet.

Vi begrenser implementasjonen til å vise reklamebilder basert på hvilke nyheter som er lest. I tillegg spesifiserer vi nærmere:

- Ett av fem mulige *reklamebilder* (av ulik kategori) skal vises på forsiden.
- Kategoriene er: sport, kultur, hjem, rampelys og data.
- Dette skal gjøres basert på *hvilke nyheter* en bruker tidligere har lest (trykket på).
- Systemet skal fungere slik at dersom det er registrert flest klikk på sportsnyheter for den aktuelle brukeren, skal reklame i form av sport vises på skjermen. Det er altså antallet som teller, og ingen nullstilling skjer.
- Systemet skal kunne endre inntrykket av brukeren over tid, ikke bare ved nåværende besøk.

Forsiden blir sentral. Fra forsiden er det en rekke lenker til nyheter, og der skal også reklamen vises. Forsiden kan se slik ut:

Figur 1: Reklamebildet på forsiden skal skreddersys avhengig av hva som er klikket på flest ganger.

Forsiden har nyheter til venstre og et reklamebilde til høyre. I tillegg har vi lagt til en lenke som viser en forklaring på hvorfor akkurat dette bildet blir vist. En slik lenke bør selvsagt ikke være med i den endelige løsningen.

2.3. Fra klikk på lenke til registrering

Den overordnede strategien er å vise et reklamebilde som tilsvarer den kategorien som har fått flest registreringer. Siden hver lenke skal kategoriseres, må disse ha informasjon om hvilken kategori de tilhører. En slik informasjon kan legges i spørrestrengen. Det vil si at i stedet for å la første lenke se slik ut:

```
Nyhet om kultur... <a href="kultur.php">Les hele nyheten</a>
```

lar vi heller lenken se slik ut:

```
Nyhet om kultur... <a href="kultur.php?kategori=kultur">Les hele
nyheten</a>
```

Spørrestrengen har nå informasjon om hvilken kategori nyheten har, og denne informasjonen sendes med til *kultur.php* i det lenken klikkes. Dette må vi utnytte:

```
<?php /* scriptet kultur.php */
      if (  ! isset($_COOKIE['kultur'])  ) $antall = 1;
      else $antall = ++$_COOKIE['kultur'];
                        //øker innholdet i cookien og
                        //legger i variabelen antall

      //Skal uansett lage en cookie
      setcookie('kultur', $antall, time()+3600*24*365);
      echo "Cookie med kategori 'kultur' er oppdatert med en ekstra";
?>
<h1>Kulturnyheten her</h1>
Det var en gang <br>
Det var en gang <br>
```

Det som skjer er dette:

- Husk at hver gang en forespørsel sendes til tjeneren, vil klienten legge ved eventuelle informasjonskapsler (cookies) som den har lagret for nettstedet.
- Hvis det ikke eksisterer en cookie fra før, vil antallet settes til 1. Det er da første gang at denne kategorien registreres.
- I motsatt fall har brukeren tidligere besøkt andre lenker som er av kategori kultur, og antallet må derfor oppdateres. Setningen `$antall = ++$_COOKIE['kultur'];` henter verdien til den cookien som har navn "kultur". Vi vet at dette er et tall, siden vi selv kontrollerer innholdet, og oppdaterer derfor tallet. `++_COOKIE[]` vil utføres først, og deretter skje tilordningen. Vi kunne også skrevet
  `$antall = 1 + $_COOKIE['kultur'];`
- Sil slutt oppdaterer vi cookien med funksjonen `setcookie()`. Det som skjer er at navnet og verdien lagres. Siden navnet fins fra før, overskrives cookien. Gyldigheten settes til et år frem i tid.

2.4. Optimalisering av løsningen

Slik det er nå er kategorien kultur hardkodet flere steder på siden. Det gir en pekepinn om at koden ikke er optimal. I tillegg må den limes inn på hver eneste nyhetsside. Det er tungvindt, og gir en krevende oppdatering hvis det blir aktuelt.

Det er bedre å legge koden som oppdaterer riktig cookie i en include-fil som kan brukes på hver eneste side. Det letter vedlikeholdet, men krever at kategorien ikke hardkodes. Her er koden:

```
<?php /* scriptet kategorier.inc.php */
      //Denne filen inkluderes i hver nyhetsfil og oppdaterer en cookie.
      //Riktig kategori er basert på mottatt kategori fra spørrestrengen.
      //Dermed får kategori og cookie-variabel samme navn.
      if(isset($_GET['kategori'])){
            $kat = $_GET['kategori'];
            if (  ! isset($_COOKIE[$kat])  ) $antall = 1;
            else $antall = ++$_COOKIE[$kat];  //øker antallet

            //Skal uansett lage en cookie
            setcookie($kat, $antall, time()+3600*24*365);
            echo "Cookie med kategori $kat er oppdatert med en ekstra";
      }
?>
```

Merk bruken av `$kat`. Ellers er det meste likt. `$_COOKIE[$kat]` har antallet av den kategorien som mottas fra spørrestrengen.

2.5. Forsiden må vise riktig bilde

Det er nå opp til forsiden å vise riktig bilde. I utgangspunktet skal et standard-bilde vises, det vil si dersom ingen cookie er satt eller cookies ikke aksepteres. Grunntrekkene i siden som vist i forrige figur, lages av denne koden:

```
<table width="600" align="center" cellpadding="15" style="border-width:1;
border-style:solid;border-color:black;">
<tr>
<td valign="top">
      <p>Nyhet om kultur... <a href="kultur.php?kategori=kultur">Les hele
nyheten</a></p>
      <p>Nyhet om sport. Fotballkamp mellom Norge og ... <a
href="sport.php?kategori=sport">Les hele nyheten</a></p>
      <p>Det var glede i fjernsynskjøkkenet lørdag, ... <a
href="espelid.php?kategori=hjem">Les hele nyheten</a> (om mat denne)</p>
      <p>Det var kongelig bryllup i helgen. <a
href="kjoler.php?kategori=rampelys">Se kjolene her!</a></p>
      <p>Magda (85) var med i prosessen med å lage brudekjolen til en av de
kongelige. <a href="magda.php?kategori=rampelys">Les den fantastiske
historien</a></p>
      <p>Spelemannslaget skal ha konsert. Lensmannen forventer fulle hus
(og folk). <a href="fulle_hus.php?kategori=kultur">Les mer i
kulturnytt</a></p>
</td>
<td valign="top" style="background-color:lightblue;">
      <h2>Reklamebildet kommer her</h2>
      <img src="bilder/<?php echo velgBilde()?>" height="200"><p>
      <a href="skrivUtCookie.php">Vis informasjon</a> om hvorfor akkurat
dette bildet ble vist fram...
</td>
```

```
</tr>
</table>
```

Legg spesielt merke til det som er uthevet i fet skrift. Kategoriene er lagt inn for hver lenke, og forutsatt at hver side inkluderer kategorier.inc.php, vil tjeneren forsøke å plassere ut en cookie med oppdatert antall.

Merk også bruken av stilsett (CSS) i tabellen. Slik kan kolonnen til høyre enkelt fargelegges annerledes, og rammen rundt tabellen får et mer tiltalende utseende.

Det viktigste er bildet som skal vises i kolonnen til høyre.

```
<img src="bilder/<?php echo velgBilde()?>" height="200">
```

I setningen over ser du at bildets filnavn er noe rart. Ved å hoppe til PHP-modus kan vi kalle funksjonen `velgBilde()`. Meningen er at den skal returnere navnet på riktig bilde basert på hvilken cookie-verdi som er størst. Alle bildene ligger i katalogen bilder/, så denne informasjonen kan hardkodes.

Funksjonen `velgBilde()` må nå lages, og legges øverst på forsiden.

```
//Må hente fram navn på den variabelen
//som har høyest verdi i cookien.
//Kunne brukt en innebygget array-funksjon til dette også,
//men tar det med for å illustrere
function velgBilde() {
      $maks = 0;

      //i tilfelle ingenting i cookie, eller cookies avslått,
      //vises et standard-bilde
      $kategori = "standard.jpg";

      foreach ($_COOKIE as $kat=>$verdi){
            if ($verdi > $maks) {
                  //oppdaterer slik at største bilde vises fram
                  $maks = $verdi;
                  $kategori =  $kat . ".jpg"; //alle bilder er av type jpg.
            }
      }
      return $kategori;  //bildet har samme navn som kategorien
}//function velgBilde
```

Det viktigste i funksjonen er i fet skrift. Kategorien som skal returneres settes i utgangspunktet til et standard-bilde. Deretter gjennomgås alle cookies. Hvis ikke cookies er slått på hos brukeren, eller ingen cookie-informasjon er satt enda, vil `$_COOKIE` ha tomt innhold, og foreach-løkken blir ikke utført.

I motsatt fall gjennomgås hver enkelt cookie. Verdien leses ut, og denne vet vi at er numerisk. I betingelsen til if-strukturen sammenliknes verdien med det som til nå er registrert som største verdi. Hvis gjeldende verdi er større, betyr det at bildet som skal vises ligger i en annen kategori. `$maks` får så denne verdien, og `$kategori` oppdateres til riktig bildenavn. Ved å la navnet til cookien være det samme som kategorinavnet, og igjen la bildet som vises ha dette kategorinavnet, blir behandlingen veldig enkel. Du kan prøve koden selv – den ligger fritt til nedlasting fra leksjonens ressursside.

Når brukeren har klikket på lenken til kultur flere ganger enn noen annen kategori, vil følgende forside vises:

Figur 2: Her vises et bilde av kategori ”kultur” siden det er cookien ”kultur” som har høyest antall.

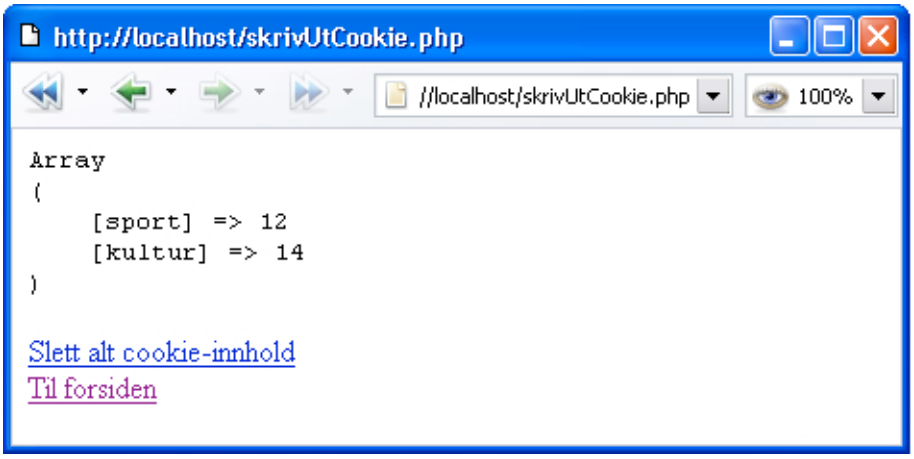
2.6. Se innholdet i alle cookier

Forskjellen mellom forrige og første figur, er at bildet har endret seg. Du ser lenken ”Vis informasjon” under bildet. Som nevnt er tanken bak denne bare å vise antall treff i utviklingsfasen. Hvis en på et slikt nettsted er helt åpen om at reklamen faktisk blir generert på denne måten, kan en gjerne ha en modifisert utgave av den scriptfilen du ser her:

```
<?php /* skrivUtCookie.php */
if(isset($_GET['slett'])) {
      //slett innholdet i cookien
      foreach ($_COOKIE as $navn=>$verdi)
            setcookie($navn, 0, time()-1000);  //Tilbake i tid = sletter
      echo "All informasjon er nå slettet";
}
elseif (!empty($_COOKIE)){
      //viser cookie-innholdet
      echo "<pre>";
      print_r($_COOKIE);
      echo "</pre>";
      //tilbyr sletting
```

```
      echo "<a href='" . $_SERVER['PHP_SELF'] . "?slett=ja'>
                  Slett alt cookie-innhold</a>";
}
else
      echo "Ingenting i cookien, derfor blir standard-bilde vist fram";
?>
<br><a href='portalMedReklame.php'>Til forsiden</a>
```

Normalt kjøres elseif-delen. Dersom det ble oversendt noen informasjonskapsler vil `$_COOKIE` ha innhold. Dette vises elegant ved hjelp av `print_r()`. Husk at `$_COOKIE` er en matrise. Skal du presentere informasjonen for dine brukere på et nettsted, er det bedre å formattere innholdet bedre. Bruk gjerne en foreach-løkke eller hent ut de ønskede elementer.



Figur 3: Utskrift av cookie-innhold viser hvor mange ganger lenkene i de ulike kategoriene er klikket på.

Du ser også at det er en lenke til å slette innholdet i cookien. Lenken har URL-en

```
<a href='" . $_SERVER['PHP_SELF'] . "?slett=ja'>Slett alt cookie-
innhold</a>
```

Samme side kalles opp på nytt i og med `$_SERVER['PHP_SELF']` og den totale lenken blir derfor

```
<a href='skrivUtCookie.php?slett=ja'>Slett alt ...</a>.
```

Trykkes lenken, vil if-delen utføres, og ved å sette levetiden til cookien til et tidspunkt tilbake i tid, slettes informasjonen. En beskjed om slettingen er også på sin plass.

```
      foreach ($_COOKIE as $navn=>$verdi)
            setcookie($navn, 0, time()-1000);  //Tilbake i tid = sletter
```

Om du prøver dette eksempelet, så husk at enkelte nettlesere vil mellomlagre (cache) sidene, og derfor kan det se ut som om cookie-matrisen ikke har endret seg. Oppdater siden skrivUtCookie.php eksplisitt, så er du sikker på at informasjonen er oppdatert.