Web programiranje

Vežbe 10 - Rad sa sesijama i *cookies*

Lako ugrađivanje PHP koda u HTML i ugrađena podrške za rad sa različitim bazama podataka nisu jedini razlozi zbog kojih je PHP postao popularan. Vrlo jednostavan XML API (*Application Programming Interface*), objektne mogućnosti i rukovanje izuzecima u PHP-u 5 i jednostavno rukovanje sesijama uvedeno još u PHP-u 4, samo su neke od dodatnih mogućnosti koje ova platforma nudi.

Defincija: "HTTP je protokol koji ne poseduje stanje, a Internet je razvojno okruženje u kome se ne pamti stanje klijenta".

HTTP je jednostavan protokol projektovan u vreme kada Internet nije imao tu dozu interaktivnosti koju poseduje danas. Na primer, logično je da se, primera radi, pri poseti portala web prodavnice funkcija punjenja "potrošačke korpe" pamti njen sadržaj u formi sesije. Prvi mehanizam koji pruža iluziju postojanja "stanja" u web programiranju je ***cookies***, o kome je već bilo reči. Dakle, *cookie* je jednostavno fajl koji čuva par atribut-vrednost, ima rok trajanja i vezan je za konkretan domen. Kada klijent u svom web pretraživaču zatraži određeni domen, taj domen može da proveri postojanje ili postavi *cookie*-je, najviše 20 po domenu. To je zgodan način da se prate posete određenog korisnika od jedne do druge.

Drugi uobičajen sistem perzistencije je **sesija**. Sesija je način da se tokom posete čuvaju podaci specifični za datu konekciju. Podaci iz sesije se čuvaju tokom posete na serveru, a zatim se brišu (obično kada korisnik zatvori svoj web pretraživač). Svaka sesija ima svoj ID koji se čuva na dva mesta: kod korisnika kao privremeni *cookie* i na serveru gde se automatski generiše od strane PHP-a, tako da programer o tome ne mora da vodi računa. Sesija takođe, slično kao *cookie*, čuva parove varijabla-vrednost, i može da čuva i tekstualne i numeričke podatke. Dakle, za podatke potrebne tokom jedne jedine posete web aplikaciji koristi se sesija, dok se za podatke koje treba sačuvati i za više poseta koristi *cookie*.

Sesije

Standardni prvi primer korišćenja sesija u PHP programiranju je brojač poseta. Svaki put kada se ovako napravljena stranica *refresh*-uje, parametar **brojac** se poveća za 1. Ukoliko se otvore 2 *tab*-a u web pretraživaču, oba će biti tretirani kao pripadnici iste sesije. Sesija se zatvara isključivanjem web pretraživača.

```
<?php
// inicijalizacija sesije
session_start();

// inkrementiranje brojaca sesije
if(isset($_SESSION['views']))
     $_SESSION['views']=$_SESSION['views']+1;
else
     $_SESSION['views']=1;

// stampaj vrednost
echo "Ova stranica je vidjena " . $_SESSION['views'] . " puta";
?>
```

Svaka sesija počinje pozivom **session_start()** koji proverava da li sesija već postoji, kada je učitava, ili, ukoliko ne postoji, kreira novu. Važno je napomenuti da se poziv **session_start()** MORA izvršiti na samom početku stranice, pre bilo kakvog izlaza u web pretraživač klijenta! Razlog tome je što mehanizam sesija interno koristi tzv. *in-memory cookies* za čuvanje podataka, a zaglavlja za njihovo kreiranje moraju biti poslata pre bilo kakvog izlaza. Greška koja se pojavljuje pri takvom pozivu je:

```
Warning: Cannot send session cache limiter - headers already sent (output started at ...)
```

Dalje, za čuvanje podataka sesije koristi se još jedan superglobalni asocijativni niz - $_SESSION. Ukoliko gore naveden kod ne radi iz prve, potrebno je proveriti vrednost promenljive **session.save_path** unutar **php.ini** fajla (na većini Linux sistema podrazumevan je **/tmp**).

Evo još jednog primera:

```
<?php
// inicijalizacija sesije
session_start();
?>
<html>
    <head></head>
    <body>

      <?php
        if (!isset($_SESSION['ime']) && !isset($_POST['ime'])) {
        // ako nema podataka, stampaj formu
      ?>
        <form action="<?php echo $_SERVER['PHP_SELF']?>" method="post">
            <input type="text" name="ime" />
            <input type="submit" name="submit" value="Unesite svoje ime" />
        </form>
      <?php
        }
        else if (!isset($_SESSION['ime']) && isset($_POST['ime'])) {
            // ako sesija ne postoji, a forma je submitovana
            // proveri da li forma sadrzi odgovarajuce vrednosti i
            // kreiraj novu sesiju
             if (trim($_POST['ime'])!= "") {
                 $_SESSION['ime'] = $_POST['ime'];
                 $_SESSION['start'] = time();
                 echo "Dobrodosli, " . $_POST['ime'] . ". Za Vas je aktivirana nova
sesija. Kliknite <a href=" . $_SERVER['PHP_SELF'] . ">ovde</a> za REFRESH stranice.";
                }
                else {
                    echo "GRESKA: Uneti Vase cenjeno ime!";
                }
            }
            else if (isset($_SESSION['ime'])) {
               // ako prethodna sesija postoji
               // izracunaj koliko je vremena proslo od njenog starta
                  echo "Dobrodosli nazad, " . $_SESSION['ime'] . ". Ova sesija
aktivirana je " . round((time() - $_SESSION['start']) / 60) . " minut(a) ranije.
Kliknuti <a href=" . $_SERVER['PHP_SELF'] . ">ovde</a> za REFRESH stranice.";
            }
        ?>
    </body>
</html>
```

U ovom primeru, prisustvo ili odsustvo sesije se koristi za odlučivanje koji će se od 3 različita korisnička ekrana prikazati. Takođe, data je i dodatna promenljiva sesije koja pamti vreme pokretanja sesije pomoću **time()** funkcije koja vraća trenutno vreme kao UNIX timestamp (broj sekundi od 01.01.1970.).

Kao što je već rečeno, svaka sesija ima svoj identifikator, koji se može dobiti kao povratna vrednost funkcije **session_id()**:

```
<?php
// inicijalizacija sesije
session_start();

// stampaj ID sesije
echo "Vodite se kao sesija ID " . session_id();
?>
```

Pored završetka sesije izlazom iz web pretraživača, sesija se može uništiti i eksplicitno, tj. pozivom **session_destroy()**, npr. kada se korisnik manuelno izloguje.

```
<?php
// inicijalizacija sesije
session_start();
// unistavanje sesije
session_destroy();
?>
```

Važno je napomenuti da sesija prvo mora da postoji na datoj stranici da bi mogla biti uništena, odatle poziv **session_start()** koji prethodi **session_destroy()**. Sledeći jednostavan primer prikazuje superglobal prirodu, tj. mogućnosti da se promeni vrednost članova asocijativnog niza bilo gde u kodu:

```
<?php
// inicijalizacija sesije
session_start();
// funkcija proverava vrednost session promenljive ime
// i vraca true ili false

function isAdmin() {
            if ($_SESSION['ime'] == 'admin') {
                return true;
            }
            else {
        return false;
    }
}

// setuj vrednost za $_SESSION['ime']
$_SESSION['ime'] = "bilo sta";
// ovde vraca false
echo isAdmin()."<br />";

// setuje novu vrednost $_SESSION['ime']
$_SESSION['ime'] = "admin";
// ovde vraca true
echo isAdmin()."<br />";
?>
```

Cookies

Rečeno je da sesije koriste tzv. *in-memory cookies*, tj. kao što sam termin nagoveštava, sesija je aktivna samo dok je aktivna aktuelna instanca web pretraživača. Ako je potrebno da nešto od podataka ostane perzistentno i u dužem vremenskom periodu, upotrebljavaju se klasični *cookies*, za koje PHP ima ugrađenu podršku.

Radi podsećanja, evo nekoliko njihovih karakteristika:

- Kako se *cookies* koriste za praćenje aktivnosti u okviru jednog jedinog domena, samo domen koji ih kreira može da ih čita,
- Jedan domen ne može setovati više od 20 *cookies*, svaki do 4KB veličine,
- Cookie obično ima 6 atributa, ali je samo prvi obavezan:
    1) **name** - ime
    2) **value** - vrednost
    3) **expires** - datum i vreme kada ističe
    4) **path** - top-level direktorijum domena odakle mu se može pristupiti, oblično "/"
    5) **domain** - domen za koji je *cookie* validan
    6) **secure** - Bulov flag koji određuje da li cookie može biti poslat samo preko sigurne konekcije HTTPS protokolom.

Treba imati u vidu da, pošto se *cookies* pohranjuju na korisnikov hard disk, programer web aplikacije ima veoma malo kontrole nad njima. Ukoliko ih korisnik iz nekog razloga obriše ili onemogući podršku za *cookies* u web pretraživaču, programer tu ne može ništa, osim da izbegava pisanje koda koji mnogo zavisi od njih.

PHP nudi jednu jedinu funkciju za manipulaciju kolačićima, **setcookie()**. Evo primera:

```
<?php
if (!isset($_COOKIE['posecen'])) {
    // ako cookie ne postoji, setuj ga
    setcookie("posecen", "1", time()+86400, "/") or die("Cookie se ne moze
setovati");
    echo "Ovo je prva Vasa poseta u toku danasnjeg dana.";
}
else {
    // ako cookie vec postoji
    echo "Vec ste bili danas ovde!";
}
?>
```

Sa navedenom stranicom treba pokušati pristup, a zatim *refresh*. Takođe, treba primetiti da *cookie* ostaje čak iako se web pretraživač restartuje (za razliku od sesije). Funkcija setcookie() prihvata 6 gore navedenih argumenta.

Vrednosti kolačića se PHP-u šalju automatski od strane klijenta i pretvaraju u parove ključ-vrednost u obliku novog superglobalnog niza - $_COOKIE.

Reference:

http://anti-virusi.blogspot.com/2009/05/sta-su-cookies.html

Evo sada nešto malo komplikovanijeg primera:

```
<?php
if (!isset($_POST['email'])) {
    // ako forma nije submitovana
    // prikazi formu
    // a ako cookie vec postoji, popuni polje za email adresu
?>
    <html>
    <head></head>
    <body>

    <form action="<?php echo $_SERVER['PHP_SELF']?>" method="post">
        Unesite svoju e-mail adresu:
        <input type="text" name="email"
        value="<?php if (isset($_COOKIE['zadnja_submisija'])) echo $_COOKIE['email'];
               ?>"
        >
        <input type="submit" name="submit" value="Posalji">
        <?php
        // takodje preracunaj vreme od poslednje submisije
        if (isset($_COOKIE['zadnja_submisija'])) {
            $dana = round((time() - $_COOKIE['zadnja_submisija']) / 86400);
            echo "<br /> $dana dan(a) od zadnje submisije";
        }
        ?>
    </form>

<?php
}
else {
    // ako je forma submitovana
    // setuj cookies za email i timestamp
    // oba isticu za 30 dana
    if (trim($_POST['email'])!= "") {
        setcookie("email", $_POST['email'], time()+(86400*30), "/");
        setcookie("zadnja_submisija", time(), time()+(86400*30), "/");
        echo "Vas email je sacuvan.";
    }
    else {
        echo "GRESKA: Unesite svoju email adresu!";
    }
}
?>
</body>
</html>
```

U ovom slučaju, vrednost email polja forme se pamti i kasnije koristi za automatsko popunjavanje polja za email. Ova tehnika se veoma često koristi za pomoć pri popunjavanju polja za login, email, itd.

Da bi se *cookie* uništio, dovoljno je setovati njegovo vreme isteka na neko vreme iz prošlosti:

```
<?php
// brisanje cookie-a
setcookie("zadnja_submisija", NULL, time() - 3600, "/");
?>
```

Dakle, sesije i kolačići se koriste za različite vidove perzistencije u web programiranju. Evo sada jednog primera koji koristi oba mehanizma uporedo. Aplikacija je jednostavan sistem za autentifikaciju, koji dozvoljava da se stranice vide samo ukoliko se korisnik na pravilan način loguje na sistem. Na primer, evo *MySQL* tabele koja sadrži korisnička imena i lozinke kriptovane SHA1 algoritmom:

```
mysql> select * from lozinke;
+----------+------------------------------------------+
| korisnik | lozinka                                  |
+----------+------------------------------------------+
| pera     | feb6891e2b23a798d6b1cfd09e3ba6da61c9b90a |
| mika     | 88b6bcb7a86c2b8eb2a8a015f0e9e1fe83ff12b3 |
| laza     | 478d5814276347ba628b537b6d2b2e57a320b32c |
+----------+------------------------------------------+
3 rows in set (0.00 sec)
```

Evo i primera u PHP-u:

```
<?php

if (isset($_POST['korisnik']) || isset($_POST['lozinka'])) {
// forma submitovana
// proveri zahtevane vrednosti
    if (empty($_POST['korisnik'])) {
        die ("GRESKA: Unesite ime!");
    }
    if (empty($_POST['lozinka'])) {
        die ("GRESKA: Unesite lozinku!");
    }

    $ime_hosta = "localhost";
    $korisnik = "branko";
    $sifra = "*********";
    $ime_baze = "branko";

    try {
        $konekcioni_string = "mysql:host=$ime_hosta;dbname=$ime_baze";
        $dbh = new PDO($konekcioni_string, $korisnik, $sifra);
        // Upit trazi kombinaciju imena i sha1() checksume lozinke
        $upit = "SELECT * FROM lozinke WHERE korisnik = '" . $_POST['korisnik'] . "'
                 AND lozinka = SHA1('" . $_POST['lozinka'] . "')";
        $pdo_izraz = $dbh->query($upit);
        $dbh = null;
    }
    catch(PDOException $e) {
        echo "GRESKA: ";
        echo $e->getMessage();
    }

    // da li je bilo sta vraceno?
    if ( $pdo_izraz->rowCount()==1 ) {
        // ako je vracena jedna vrsta
        // autorizacija je uspesna
        // kreiraj sesiju i setuj cookie username sa rokom trajanja 30 dana
        session_start();
        $_SESSION['autorizovan'] = 1;
        setcookie("username", $_POST['korisnik'], time()+(84600*30));
        echo "Pristup dozvoljen!";
    }
```

```
    else {
        // nema rezultata upita, autorizacija neuspela
        echo "Greska: Pogresna kombinacija korisnik/lozinka!";
    }

}
else {
// nije submitovano
// prikazi formu za login
    ?>
<html>
    <head></head>
    <body>
        <form style='text-align:center' method="post"
              action="<?php echo $_SERVER['PHP_SELF']; ?>">
          Korisnik <input type="text" name="korisnik"
             value="<?php if (isset($_COOKIE['username'])) echo $_COOKIE['username']; ?>">
          <p />
          Lozinka <input type="password" name="lozinka">
          <p />
            <input type="submit" name="submit" value="Logovanje">
        </form>
    </body>
</html>
<?php
}
?>
```

Kao što se vidi iz gornjeg koda, promenljiva sesije koja čuva logičku vrednost koja označava da li je korisnik autorizovan ili ne, je **$_SESSION['autorizovan']**. Proveru vrednosti ovog elementa $_SESSION niza, potrebno je implementirati i na svakoj stranici web lokacije, inače bi korisnik mogao da pređe na bilo koju stranu pozivom njenog URL-a. Sada svaka stranica zaštićenog web-a treba da ima formu sličnu sledećoj:

```
<?php
// start sesije
session_start();
if (!$_SESSION['autorizovan'] == 1) {   // proveriti da li je setovano zbog greške
    // proveri da li je izvrsena autorizacija
    // ako nije, zavrsi sa greskom
    die ("GRESKA: Neautorizovan pristup!");
}
else {
?>
    <html>
    <head></head>
    <body>
     Ovo je sigurna stranica. Moze joj se pristupiti samo ako je $_SESSION['auth'] = 1
    </body>
    </html>
<?php
}

?>
```