Rad sa sesijama u PHP-u

www.sasastamenkovic.com/blog

Autor: Stamenković Saša

E-knjiga: Rad sa sesijama u PHP-u sadrži praktična uputstva koja se mogu smatrati korisnim izvorom informacija pri radu sa sesijama u PHP-u. U ovom E-resursu se mogu naći važnije teorijske osnove koje bliže definišu mehanizame za formiranje sesija pri čemu su obuhvaćeni svi važniji procesi koji se odvijaju na klijenskom sloju, kao što je proces skladištenja jedinstvenog identifikoatora sesije ali i drugih važnijih principa.

Nakon usvajanja koncepata iznetih u ovom e-resursu čitalac će biti osposobljen da samostalno formira jednostavan mehanizam za autentifikaciju korisnika primenom troslojnog modela arhitekture pomoću koncepta formiranja promenljive sesije.

Autor e-knjige: Stamenković Saša, *E-mail: office@sasastamenkovic.com,.URL: http://www.sasastamenkovic.com/blog*

- Sadržaj -

Uvodna razmatranja

Verovatno ste dosad koristili veliki broj Web aplikacija koje primenjuju princip rada sa sesijama. Reč je, naime, o onim aplikacijama koje verovatno koristite svakodnevno kada proveravate elektronsku poštu na Vašem Mail serveru, kada objavljujete post na Twitter-u ili gubite vreme na Facebook-u. Moglo bi se reći da Vam osnovna korist od primene sesija omogućava da "pratite" određenog korisnika od početka do kraja njegove sesije.

Zašto bismo bilo koga pratili?

Ako pretpostavimo da je određeni korisnik došao na Vašu Web lokaciju i tada uputio jedan HTTP zahtev klikom na neki interni link u okviru Vašeg sajta, a odmah zatim uputio još jedan HTTP zahtev zahtevajući potom drugu Web stranicu, u tom slučaju ne biste bili u mogućnosti da razlučite da li su oba zahteva došla od istog korisnika!

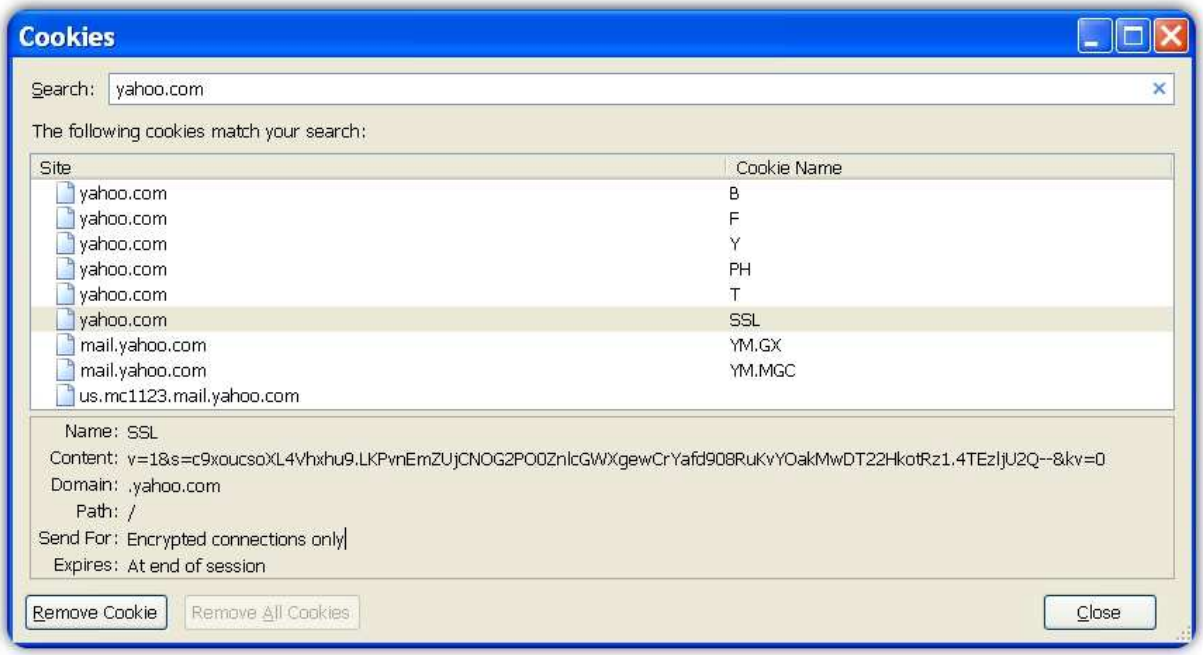
Šta ovo znači?

Pod pretpostavkom da želite da na Vašem Web sajtu formirate sistem za autentifikaciju korisnika, kako biste inače znali da ste autentifokavili pravog korisnika, odnosno, da odredite njegova prava, ako ne znate ko je taj korisnik? Dakle, Vi biste, sa dosadašnjim znanjem o PHP-u istaknutom u prošlim delovima ovog kursa, mogli sa više ili manje poteškoća i da formirate bazu korisnika, pa čak i aplikacionu logiku za deo provere autentifikovanosti korisnika, ali ne biste bili u mogućnosti da obeležite svakog korisnika i pratite sve do trenutka dok ne dođe do kraja njegove sesije. Dakle, važnost razumevanja rada sa sesijama je velika, budući da sam HTTP protokol ne čuva stanje sesije. Dakle, trenutno ne postoji efikasniji sistem za obeležavanje korisnika od rada sa sesijama.

Osnovni pojmovi

Vema je važno približiti princip rada sa sesijama, ali je još važnije razumeti kako se i šta odvija prilikom formiranja sesije. Moglo bi se reći da u svakoj sesiji dolazi do dodeljivanja jedinstvenog identifikatora sesije. Ovaj identifikator treba razumeti kao jednu vrstu podatka koji služi za obeležavanje korisnika. Ovaj podatak se čuva na klijenteskom sloju (korisnika) kao kolačić o kome će kasnije biti reči ili čak prosleđivanjem unutar url-a.

Sada je preporučljivo da se upoznate i sa praktičnim delom tako što ćete pomoću svog Web čitača odabrati Vaš omiljeni Mail server, a potom pristupiti u sistem (*autentifikovati se*). Nakon uspešno izvršene autentifikacije odaberite opciju Tools -> Page info, ako koristite Mozzillu. Ukoliko odaberete opciju Security, bićete u stanju i da zaista razumete šta je to sesija i kako se ona odvija. U slučaju predstavljenom na slici 1 dat je primer u kome je formirana sesija sa više kolačića različite namene. No, bez obzira na višestrukost koja bi trebalo da bude u službi višeg nivoa zaštite, što je u ovom slučaju dobro, treba istaći da je važno da sa slike uočite red koji nosi naziv Content. Naime, u ovom redu se beleži maločas pomenuti identifikator sesije. Kao što vidite, reč je o podatku određene dužine koji je nedvosmisleno odredio mene ili, tačnije, moj računar.



Slika 1. Provera tekućih sesija

Na osnovu ovog i drugih podataka, ja mogu videti stranice namenjene registrovanim i autentifikovanim članovima. Kao potvrda ovoj tvrdnji, a i u cilju boljeg razumevanja sesija, pristupite sada Vašem Mail serveru i autentifikujte se, a potom otvorite ovaj dijaloški okvir sa slike prema uputstvu. Videćete da je, kao na slici 1, došlo do formiranja jedinstvenog identifikatora, dakle, došlo je do formalnog početka trajanja sesije za mene kao korisnika (ja sam na ovaj način obeležen!). Sada, upotrebite opciju za uklanjanje kolačića sa slike 1, "Remove Cookie", a potom se vratite u okruženje Vašeg Mail servera. Slobodno uputite jedan HTTP zahtev, recimo, klikom na opciju "Inbox" na Vašem Mail serveru. Videćete da je umesto prikazivanja sadržaja Vašeg poštanskog sandučeta, zapravo, došlo vraćanja na stranu za autentifikaciju ili poruke da niste prijavljeni u sistem u zavisnosti od toga koji Mail server koristite. Šta ovo znači? To znači da ste manuelno ili "naglo" prekinuli trajanje sesije. Isti efekat možete postići i ukoliko kliknete na opciju "Sign out" ili "odjavi se". Dakle, sesija će biti prekinuta samim tim što je identifikator obrisan. Ako je identifikator obrisan, to znači da Vi, identifikovani kao korisnik v=1&s=c9xoucsoXL4... više niste taj korisnik budući da o tome nema nikakvog podatka koji bi to potvrdio!

Ovo je sasvim dovoljno da biste razumeli važnost upotrebe sesija prilikom Vašeg budućeg razvoja Web apliakcija. Možda bi u ovom trenutku trebalo ukazati i na sigurnost ovakve metode.

Svakako, nema sumnje da je koncept primene sesija široko primenjivan metod budući da ga koriste sve one složenije Web lokacije, međutim, da li je on sam po sebi dovoljan? Moglo bi se reći da primena sesija u potpunosti zadovoljava ono za šta je i projektovana, a sama bezbednost zavisi od velikog broja parametara. Najvažniji među njima je sama struktura Vašeg koda (kvalitet aplikacije), zatim, da li je na Vašem Web serveru podešen SSL, da li funkcioniše, da li Vi i Vaši korisnci imate digitalne sertifikate itd. Sve ovo nije

toliko važno ukoliko je Vaš cilj da formirate neku jednostavniju aplikciju kao što je Mail server, forum, i sl., ali suprotno tome, ako su Vaši ciljevi usmereni ka razvoju elektronskih prodavnica koje će raditi sa realnim novcem i to, u najgorem slučaju, primenom metode "tunelovanja", onda Vam je potrebna maksimalna sigurnost Web aplikacije i to primenom svih mogućih sigurnosnih mehanizama koje možete da

primenite, počev od kriptografskih metoda, pa do SSL-a i dobro osiguranog servera i same aplikacije.

Treba da znate je moguće da dođe do "krađe sesije", krađa sesije označava krađu identiteta, a postoje brojni mehanizmi koje zlonamerni korisnici pritom primenjuju. Da

biste bili sigurniji, i da biste raspolagali većim brojem informacija na ovu temu, u članku "Sigurnost Moolde LCMS-a" treba da potražite neke dodatne informacije o svemu ovome, bez obzira na to što je ovde mahom obuhvaćena analiza sigursnosti sistema za elektronsko učenje. Tu su ipak obrađeni i drugi važni segmenti koji mogu biti veoma dobar polazni osnov za obezbeđivanje viših sigurnosnih nivoa. Ipak, težište ovog segmenta kursa je prvenstveno usmereno ka interperetaciji modela rada i formiranja sesija.

Gde se skladište identifikatori sesije?

Na početku ovog dela kursa, pomenuta su dva načina uskladištenja podataka o sesijama. Tada smo rekli da se prvi odnosi na generisanje identifikatora koji se potom beleži u kolačić. Delom smo čak i videli kako se oni fizički beleže u određenu vrednost. Kasnije je pomenut i drugi metod beleženja identifikatora pri kraju URL adrese. Ovaj drugi metod je takođe interesantan princip, ali on zahteva malo podešavanje, Vama već dobro poznate skripte php.ini iz priručnika o pokretanju Web servera. U oviru ove skripte potrebno je proveriti vrednost direktive `session.use_trans_sid`! Vrednost bi trebalo da bude podešena na ON.

NAPOMENA: drugi metod spada u grupu visoko rizičnih poteza budući da na taj način omogućavate url koji obeležava sesiju. Tada se može desiti da se taj URL sa identifikatorom, recimo, zadrži u istoriji poseta na korisnikovom Web čitaču, tako da bi se korisnik sa ubeleženim identifikatorom mogao direktno povezati na Vašu Web lokaciju, a da se prethodno nije ni autentifikovao. Savet je, dakle, vrednost ove direktive ipak ostavite na off. Razmislite samo zašto su razvojni programeri prema podrazumevanim podešavanjima onemogućili ovu direktivu.

Konačno, još jedan način koji omogućava manuelni unos sesije je tzv. upotreba indentifikatora u obliku konstante **SID**. Ovakva metoda zahteva da ručno, pri kraju URL-a, unesete identifikator :

<a href="adresa.php?<?php echo strip_tags(SID); ?>"> - da biste umanjili rizik trebalo bi da uvek primenite funkciju strip_tags() koja blokira sve što počinje znakom <. Na ovaj način ćete sprečiti umetanje programskog koda koji bi mogao da naškodi Vašoj Web lokaciji, serveru. Ono što bi ostalo primenom ove funkcije bi mogao biti samo plain text bez specijalnih karaktera i suvišnih oznaki, itd. **Ovaj savet praktikujte uvek kada želite da omogućite korisniku da nešto unese u Vaš sistem**. Postoje, naravno, i druge, dodatne metode zaštite, a uputstva treba potražiti u drugim e-reseurima na ovu temu.

Kolačići

Postoji razlika između kolačića i sesija čije trajanje po iniciranju teče sve dok se ista i formalno ne okonča, a rekli smo već da postoje dva načina: u prvom se sesija prirodno prekida izvršavanjem PHP skripte sve do reda na kome je napisana funkcija session_destroy; i drugi način, koji je zapravo manuelni prekid sesije koji obično korisnik

na klijentskoj strani može izazvati i to zatvaranjem Web čitača. Nasuprot tome, kolačići predstavljaju sistem koji **traje neograničeno, ukoliko drugačije nije zadano**. Ovo konkretno znači da, ukoliko eksplicitno ne definišete unutar skripte vremensku dužinu trajanja kolačića, on će imati tendenciju da bude vežeći na klijenstkom Web čitaču, sve

dok sam klijent ne obriše taj kolačić. Upravo je to razlog što se većina autora u ovoj oblasti slaže da se sesijama može više verovati nego kolačićima.

U daljem toku kursa biće interpretiran metod definisanja kolačića, ograničavanja njegove vremenske dužine trajanja i, konačno, prikazivanja njegove vrednosti primenom funkcije print_r();

```
<?php

$duzina_trajanja = time()+86400;
setcookie("ime","korisnik-x",$duzina_trajanja);
setcookie("godiste", 26, $duzina_trajanja);
echo '<a href="prikazi_kolacice.php"<p>Kolacici</a></p>'

?>
```

Ukoliko formiramo promenljivu $duzina_trajanja i operatorom dodele iskopiramo vrednost funkcije time() za koju smo definisali ograničenje na 86400 sekundi, koliko ima u jednom danu, onda ćemo biti u mogućnosti da upravljamo dužinom trajanja kolačića koji su definisani u nastavku. Primećujete da se za definisanje kolačića koristi naredba setcookie u okviru koje se unose vrednosti kolačića. Na našem primeru, vrednosti su ime -> koje u slučaju ovog kolačića odgovara korisniku x, ali takođe možete primetiti da možemo upotrebiti i promeljive prilikom građenja kolačića, što omogućava veliki stepen fleksibilnosti.

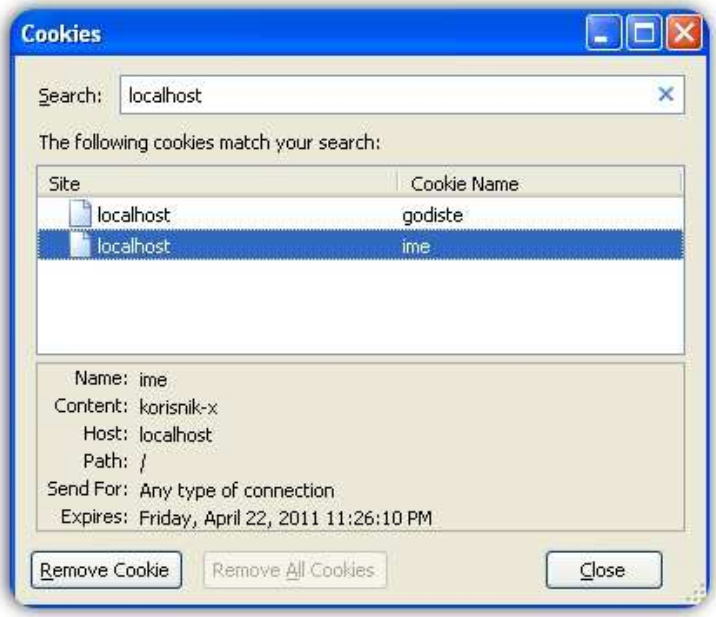
U daljem toku skripte upotrebom jezičkog konstrukta echo, postavljen je link ka PHP skripti, čiji je primarni zadatak da prikaže vrednosti definisanih kolačića u obliku niza:

```
<?php
print_r($_COOKIE)
?>
```

Prvu skriptu snimite pod proizvoljnim imenom, a drugoj dodelite ime koje je istaknuto u okviru echo konstrukta ili modifikujte link prema potrebi. Nakon izvršavanja ovih dveju skripti izveštaj koji ćete dobiti na klijentskom sloju je sledeće sadržine:

```
Array ( [ime] => korisnik-x [godiste] => 26 )
```

Ono što smo definisali kao vrednost pojaviće se u obliku niza, a dužina trajanja biće ubeležena u Vašem Web čitaču. Sve to takođe možete proveriti odabirom opcija Tools, Page info, View Cookies. Pri dnu dijaloškog okvira, videćete trenutak isteka važenja kolačića koji je ekvivalentan definisanim prethodno vrednostima.



Slika 2. Vrednosti kolačića i dužina trajanja

Formiranje sesije u praksi

Kada želite da formirate sesiju, možete primeniti gotovu PHP-ovu funkciju session_start ();, ili da omogućite pristup superglobalnom nizu $_SESSION. Funkciju session_start (); uvek postavljate na vrhu svakog PHP skripta koji će biti zaštićen sesijom. Mi ćemo se u toku ovog kursa baviti prvenstveno ovim principom rada, pored toga što je moguće primeniti i drugi maločas pomenuti metod session.auto_start.

Kraj sesije takođe morate manuelno uneti u svaku Vašu PHP skriptu u trenutku kada smatrate da ona treba da se okonča. Zato postoji PHP-ova funkcija session_destroy ();.

Prateći dosadašnju praksu, naš cilj će u toku ovog kursa biti usmeren prvenstveno na usvajanje praktičnih znanja koja bi trebalo da posluže kao dobar oslonac za dalji razvoj Web aplikacija. Da bismo ostvarili ovaj cilj, u daljem toku kursa radićemo na praktičnom primeru koji će ilustrovati princip rada sa sesijama.

Cilj je da projektujemo sistem za autentifikaciju korisnika pomoću baze i aplikativnog srednjeg sloja uz primenu promenljive seije.

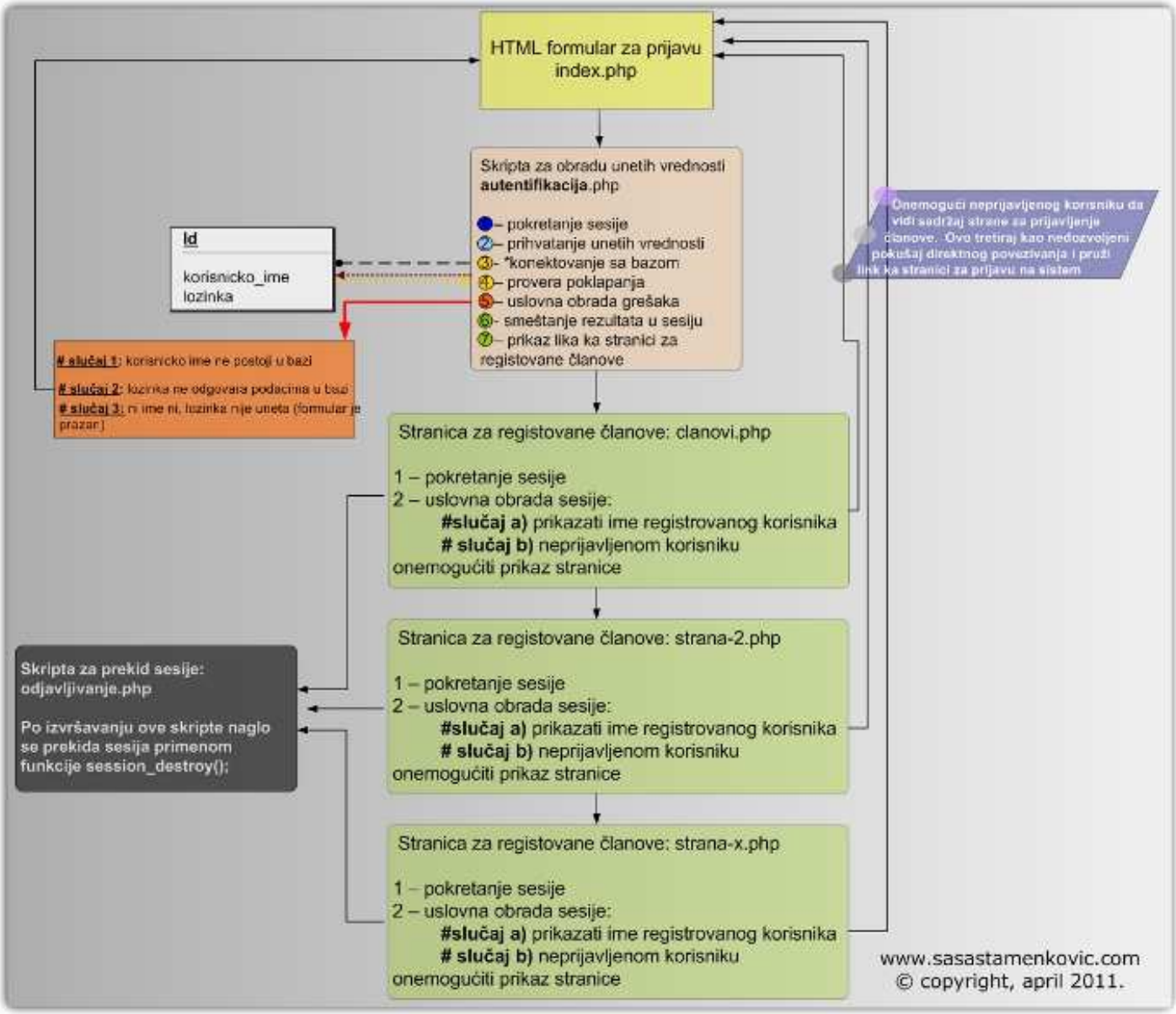
U nastavku ćemo dakle, raditi na razvoju aplikacije za autentifikaciju registrovanih korisnika. Naš cilj će biti usmeren isključivo ka projektovanju aplikacione logike koja će pomoću promenljive sesije vršiti proveru korisnika, odnosno njegovog statusa (prijavljen i ne). Budući da smo se u proteklom delu kursa, prilikom interperacije rada sa MySQL-om upoznali sa osnovama formiranja PHP skripti pomoću kojih smo tada vršili unos podataka u bazu, kako bismo plastičnije ilustrovali princip rada u primeni koncepta troslojne arhitekture, sada se time nećemo baviti, ali i dalje ostajemo u domenu troslojnog modela.

Dakle, naš primarni cilj u toku ovog dela kursa je da utvrdimo šta je sve potrebno za formiranje aplikacije koja bi mogla da prijavi "registrovanog korisnika" upoređivanjem

podataka koje je uneo u formular u odnosu na realne podatke u bazi. Potom ćemo raditi sa seijom uz pomoć koje ćemo obeležiti korisnika primenom identifikatora sesije. Na kraju ćemo naučiti kako da obavestimo korinsika o njegovom statusu (prijavljen ili ne), uz adekvatnu obradu grešaka.

U cilju boljeg razumevanja formiran je dijagram toka sistema aplikacije za autentifikaciju korisnika prikazanog na slici 3. Pri vrhu dijagrama imaćemo jedan HTML formular koji će se sastojati od svega dva polja za unos. Jedno polje će reprezentovati korisničko ime, a drugo lozinku. Korisnik po dolasku na naš sistem može odabrati opciju prijave i to, primenom pomenutog formulara. Ukoliko klikne na dugme submit podaci se prosleđuju POST metodom unutar skripte autentifikacija.php koja je ujedno i centralni deo naše aplikacije, budući da se sastoji od povećeg dela aplikacione logike. Zadatak ove skripte je da pokrene sesiju, prihvati podatke i odmah izvrši obradu, uz uslov prethodnog konektovanja sa bazom podataka. Prihvaćeni podaci u o korisničkom imenu i lozinki se šalju unutar ba, pri čemu je glavni cilj da se uporede ove vrednosti. Ukoliko se vrednosti poklapaju, korisnik je autentifikovan smeštanjem vrednosti o korisničkom imenu unutar promenljive sesije.

Ako se uneti podaci ne poklapaju ili je pritisnuto dugme submit, a da pritom nije uneta nikakva vrednost sa kojom bi moglo da se radi, doći će do greške. Do greške takođe dolazi i kada se korisnik direktno poveže na stranu zaštićenu sesijom za proveru autentifikovanosti. Sa druge strane, za korisnika koji je uneo tačne podatke smatra se da je autentifikovan i jedino takav korinik može videti sadržaj zaštićenih strana. U ovom delu se posebno naglašava da funkcija za pokretanje sesije mora biti postavljena uvek na početku skripte. Čak i jedno mesto razmaka ili red može ugroziti pravilno formiranje sesije i zato se postarajte da izgradnju takvih strana uvek počnete PHP oznakom u XML stilu i odmah nakon otvaranja pozovete funkciju za pokretanje sesije session_start();



Slika 3. Dijagram toka sistema

***Uvećaj prikaz dijagrama ovde**: http://www.sasastamenkovic.com/e-knjiga/kurs-7/dijagram-toka-autentifikacije.php*

Formiranje aplikacione logike

Struktura baze "autentifikacija"

Za rad sa vrednostima biće nam potrebna baza, a u u okviru nje jedna tabela. Za ime baze uzeto je: "autentifikacija", a za ime tabele "korisnici".

1) Biće nam potreban identifikator novootvorenih redova, pri čemu će svaki novi red imati jedinstveni identifikator uvećan za jedan u odnosu na prethodni. Ova kolona neće primati negativne vrednosti i ona, ujedno, ne sme biti prazna, odnosno, ona uvek mora sadržati vrednost.

2) Kolona korisničkog imena je tipa varchar sa maksimalo 25 karaktera.

3) Lozinka je identična po formi, a broj dozvoljenih karaktera određen na 32, u slučaju da budete imali nameru da kasnije ovaj skript unapredite kriptogrfskim metodom zaštite, npr. jednosmernom MD5 heš funkcijom.

4) Konačno, bazu moramo ispuniti nekim početnim vrednostima budući da u ovom delu kursa nismo radili na sistemu za registraciju korisnika, jer su metode upisivanja podataka u bazu već opisane u PHP i MySQL priručniku. Naše korisničko ime će biti "korisnik", a lozinka "lozinka123".

```
CREATE DATABASE `autentifikacija` DEFAULT CHARACTER SET utf8 COLLATE
utf8_unicode_ci;
USE `autentifikacija`;
CREATE TABLE IF NOT EXISTS `korisnici` (
`identifikator` int(8) unsigned NOT NULL AUTO_INCREMENT,
`korisnicko_ime` varchar(25) COLLATE utf8_unicode_ci NOT NULL,
`lozinka` varchar(32) COLLATE utf8_unicode_ci NOT NULL,
PRIMARY KEY (`identifikator`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci
AUTO_INCREMENT=2 ;
INSERT INTO `korisnici` (`identifikator`, `korisnicko_ime`, `lozinka`)
VALUES
(1, 'korisnik', 'lozinka123');
```

Skripta index.php

Do sada smo se više puta susretali sa načinom formiranja HTML formulara, a ovaj se ni po čemu ne razlikuje od prethodnih osim po vrednostima koje u njega unose. Struktura koda je data u nastavku:

```
<html>

<form action='autentifikacija.php' method='POST'>
Korisnicko ime: <input type ='text' name='korisnicko_ime' size="13"><br>
Lozinka: <input type ='password' name='lozinka'><br>
<input type ='submit' value='Uloguj se'>
</form>

</html>
```

Skripta autentifikacija.php

Na osnovu teorijskih razmatranja iznetih na ovom kursu trebalo bi da je jasna uloga sesija pri radu sa PHP-om.

Da bismo i praktično primenili ova znanja, formirali smo maločas pomenutu skriptu. Zbog složenosti operacija koje se ovde odvijaju, biće izvršena nešto detaljnija analiza skripte, tako da sam kod nećete moći da direktno kopirate, ali zato ćete na kraju ovog kursa biti u mogućnosti da u izolovanom kontekstu analizirate **samo strukturu i po potrebi kopirate.**

```
<?php
```

Najpre pokrećemo sesiju funkcijom (session_start();).

```
session_start ();
```

Formirali smo dve promenljive koje će prihvatiti vrednosti poslate iz HTML formulara metodom **POST**

```
$korisnicko_ime = $_POST['korisnicko_ime'];
$lozinka= $_POST['lozinka'];
```

Postavljamo prvi uslov: ukoliko je korisnik pritisnuo dugme submit u HTML formi, mi mi ćemo sada proveriti da li je pritom uneo nešto u formular i to u oba polja, budući da smo koristili operator && koji daje 1 (true) jedino ukoliko je i a i b true odnosno 1. Ovaj princip je već dovoljno opisan u uvodnim delovima kursa prilikom interpetacije operatora.

```
if ($korisnicko_ime&&$lozinka)
{
```

Ukoliko je korisnik uneo vrednosti, a one kopirane u maločas istaknute promeljive, izvršiće se ovaj blok koda. Njegov cilj Vam je već dovoljno dobro poznat. On uspostavlja konekciju sa bazom.

```
$connect = @mysql_connect("localhost", "root", "") or die ("konekcija
neuspesna!") ;
mysql_select_db("autentifikacija") or die ("tabela nije slektovana");
```

Po uspešno obavljenoj konekciji, formiraćemo promenljivu upit čiji je zadatak da ostvari, selektuje kolonu "korisničko_ime unutar tabele "korisnici"

```
$query = mysql_query ("SELECT * FROM korisnici WHERE korisnicko_ime
='$korisnicko_ime'");
```

Važno je sada da definišemo još jednu promenljivu. U našem slučaju reč je promenljivoj $numrows (Vi je možete definisati po želji), a njen primarni zadatak će biti da ispita da li ima redova u tabeli. Primećujete da je u okviru petlje loop upotrebljena negacija pomoću koje definišemo da, ukoliko nije tačno da nema redova u tabeli, onda ih zasigurno ima! Ukoliko ih ima, onda još samo treba da pronaćđemo u okviru kog specifičnog reda se nalazi korisnik koji je uneo specifično korisničko ime i lozinku. U ovom slučaju, uneto korisničko ime odgovara korisničkom imenu u bazi, a isto važi i za lozinku.

```
$numrows = mysql_num_rows($query);

if ($numrows!=0)
{

while ($row = mysql_fetch_assoc($query))
{
```

```
$dbkorisnicko_ime = $row['korisnicko_ime'];


$dblozinka = $row['lozinka'];
}
```

Upotrebom uslova uz pomoć operatora jednakosti i operator && utvrđuje se podudarnost unetih podataka. Upamtite da upotrebom ovog operatora i a i b moraju biti true.

```
if ($korisnicko_ime==$dbkorisnicko_ime&&$lozinka==$dblozinka)


{
```

Ukoliko se podaci podudaraju, korisnik će biti obavešten o tome i biće mu pružen link ka unutrašnjosti sajta isključivo namenjenog autentifikovanju korisnika.

Ključni deo naše skripte se nalazi upravo ovde, budući da ćemo vrednost korisnika (korisnicko_ime) smestiti u sesiju i njome se rukovoditi pri daljem radu. Na ovaj način, sesija o ovom korisniku će biti ubeležena i na korisnikovom Web čitaču, pa ćemo svaki put morati samo da proverimo da li podaci još uvek postoje (ubeženi u Web čitaču), a ako ne postoje, to će ujedno značiti da je sesija istekla. Ukoliko postoje, funkcija session_start() će ih prepoznati.

```
echo "Ulogovani ste! Sada možete pristupiti stranici za <a
href='clanovi.php'>registrovane korisnike.</a>";
$_SESSION ['korisnicko_ime'] = $korisnicko_ime;
}
```

Ukoliko nema podudarnosti, obavestićemo korisnika da je, ili uneo netačnu lozinku, ili nepostojeće korisničko ime.

```
else
echo "Uneta lozinka je netacna!";

}
else die ("Uneto korisnicko ime ne postoji!");
}

else
die ("Morate uneti ime i lozinku!");
?>
```

Skripta clanovi.php

Ključno je da upamtite da možete imati veliki broj sesija i kolačića, a na prethodno opisan način ćete ih i formirati, ali njihova dalja upotreba mora poprimiti sledeću strukturu da bi one zaista služle svrsi.

Ideja ove skripte je da omogući prijavljenim i oneomogući neprijavljenim korisnicima da vide sadržaj strane. To znači da strukturu ove skripte morate da umetate svugde gde želite da onemogućite direktan pristup, a šta se konkretno dešav, videćemo u daljem toku skripte:

```
<?php
session_start();
```

Pošto smo formirali sesiju to će značiti da će ona po izvršavanju proveriti u Web čitaču da li je registovana sesija.

```
if ($_SESSION['korisnicko_ime'])
```

Ukoliko je registrovana sesija, jezički konstrukt će dati vrednost korisničkog imena koje je autentifikovano, a ukoliko ne, jednostvano dati link gde bi to moglo da se učini.

```
echo "Ulogovani ste kao: ".$_SESSION['korisnicko_ime']."<br/>Odjavite se<a

href='logout.php'>ovde</a>";


else

die ("Ovu stranicu mogu videti samo <a href='index.php'> prijavljeni
korisnici</a>");

?>
```

Celovit pregled skripte autentifikacija.php

```
<?php

session_start ();

$korisnicko_ime = $_POST['korisnicko_ime'];
$lozinka= $_POST['lozinka'];

if ($korisnicko_ime&&$lozinka)
{

$connect = @mysql_connect("localhost", "root", "") or die ("konekcija
neuspesna!") ;
mysql_select_db("autentifikacija") or die ("tabela nije slektovana");


$query = mysql_query ("SELECT * FROM korisnici WHERE korisnicko_ime
='$korisnicko_ime'");

$numrows = mysql_num_rows($query);

if ($numrows!=0)
{

while ($row = mysql_fetch_assoc($query))
{
$dbkorisnicko_ime = $row['korisnicko_ime'];
$dblozinka = $row['lozinka'];
}

// provera poklapanja

if ($korisnicko_ime==$dbkorisnicko_ime&&$lozinka==$dblozinka)

{
echo "Ulogovani ste! Sada možete pristupiti stranici za <a
href='clanovi.php'>registrovane korisnike.</a>";
$_SESSION ['korisnicko_ime'] = $korisnicko_ime;
}
else
echo "Uneta lozinka je netacna!";
```

```
}
else die ("Uneto korisnicko ime ne postoji!");
}
else


die ("Morate uneti ime i lozinku!");
?>
```

Cilj ove skripte je, dakle, bio isključivo usmeren ka isticanju značaja primene sesija pri razvoju Web aplikacija. Aplikacija koja je nastala u ovakvoj formi je samo osnova koju

bi trebalo najpre osigurati, što primenom kriptografskih metoda, što nužnim filtriranjem podataka. Zato se postarajte, ukoliko želite da je koristite u praksi da izolujete neke njene delove kao što je segment koda koji se konektuje sa bazom, zatim upit u tabelu korisnici itd.

LITERATURA

[1] – http://www.php.net

[2] – Jason E. Sweat, *PHP|ARCHITECT'S GUIDE TO PHP DESIGN PATTERNS, 2005.*

[3] – David Powers*, PHP Object-Oriented Solutions, 2008*