Međimursko veleučilište u Čakovcu

Uvod u PHP programiranje

Nastavni materijal za kolegij PHP programiranje

Dr.sc. Sanja Brekalo

2018.

Autor:

Dr. sc. Sanja Brekalo

Recenzenti:

Dr.sc. Dean Valdec

Dr.sc. Mihael Kukec

Lektor:

Maja Bregović

Nakladnik:

Međimursko veleučilište u Čakovcu

Za nakladnika:

doc.dr.sc. Nevenka Breslauer, prof.v.š.

ISBN 978-953-8095-09-2

SADRŽAJ

1. UVOD U PHP 5

1.1. Arhitektura klijent – poslužitelj i tehnologije izvršavanja 7

1.1.1. Klijentske web tehnologije 8

1.1.2. Poslužiteljske web tehnologije 8

1.1.3. HTML/ CSS 10

1.2. Izvršavanje skripte u PHP-u 11

1.3. Razvojno okruženje 13

1.3.1. Najčešći problemi instalacije 17

1.4. Alati za izradu PHP datoteka 18

2. OSNOVE PHP-a 19

2.1. PHP oznake 19

2.1.1. Alternativne PHP oznake 19

2.2. Komentari 20

2.3. Ispisivanje u preglednik 21

3. VARIJABLE I KONSTANTE 23

3.1. Varijable 23

3.2. Konstante 24

3.3. Doseg (engl. *Scope*) varijable 25

3.3.1. Lokalne varijable 25

3.3.2. Globalne varijable 25

3.3.3. Statične varijable 25

3.3.4. Superglobal varijable 26

3.4. PHP podržani tipovi podataka 29

3.5. Konverzija tipova podataka 30

3.5.1. Eksplicitna konverzija tipova podataka (eng. Type Casting) 30

3.5.2. Automatska konverzija tipova podataka (eng. Type Juggling) 30

3.6. Funkcije vezane uz tipove podataka 32

3.6.1. Dobavljanje tipa podataka 32

3.6.2. Mijenjanje tipa podataka 32

3.6.3. Identificiranje tipa podataka 32

4. OPERATORI 34

4.1. Vrste operatora 34

4.1.1. Operatori dodjeljivanja 34

4.1.2. Aritmetički operatori 34

4.1.3. Operatori na razini bitova (engl. Bitwise operators) 34

4.1.4. Operatori usporedbe 35

4.1.5. Operator kontrole pogrešaka 35
4.1.6. Operatori inkrementacije i dekrementacije 35
4.1.7. Logički operatori 35
4.1.8. String operateri 36
4.1.9. Array operateri 36
4.1.10. Operateri tipa podataka 36
4.2. Pregled prioriteta pri izvršavanju operatera 37
4.3. Escape sekvence 39

5. KONTROLNE STRUKTURE 40

6. FUNKCIJE 48

6.1. Prosljeđivanje vrijednosti funkcijama 50
6.2. Zadane i opcionalne vrijednosti parametara 50
6.3. Zadavanje tipova podataka (eng. *Type Hints*) 51
6.4. Vraćanje vrijednosti iz funkcije 53

7. POLJA 54

7.1. Funkcije za rad s poljima 56
7.2. Funkcije za dodavanje i brisanje elemenata iz polja 57
7.3. Funkcije za traženje u polju 59
7.4. Pokazivači 61
7.5. Veličina polja 62
7.6. Mijenjanje i sortiranje polja 63
7.7. Spajanje i razdvajanje polja 65

8. STRING 66

8.1. Regularni izrazi 68
8.1.1. Funkcije koje koriste regularne izraze 70
8.2. String funkcije 73
8.2.1. Određivanje dužine stringa 73
8.2.2. Uspoređivanje 2 stringa 73
8.2.3. Promjena verzala i kurenata (eng. string case) 74
8.2.4. Konvertiranje u i iz HTML-a 74
8.2.5. Alternative regularnim izrazima 76
8.2.6. Brisanje i dodavanje 78

9. DATOTEKE I DIREKTORIJI 80

9.1. Funkcije za rad s putanjom 80
9.2. Funkcije za proceduralan rad s datotekama 81
9.2.1. Funkcije za određivanje veličine 81
9.2.2. Funkcije za određivanje vremena pristupanja i promjene datoteka 81
9.2.3. Rad s datotekama- proceduralni način 82

9.3. Funkcije za proceduralan rad s direktorijima 85
9.3.1. Čitanje sadržaja direktorija 85
9.4. SPL (*Standard PHP Library*) – objektno orijentirani način rada s datotekama, putanjama i direktorijima 86

10. OBJEKTNO ORJENTIRANI PHP 87

10.1. Svojstva objekata/klasa 88
10.2. Metode objekata/klasa 91
10.2.1. Zadavanje tipova varijabli (eng. Type hinting) 92
10.2.2. Konstruktorske i destruktorske metode 93
10.2.3. Overloading metode 95
10.3. Rad s klasama i objektima 97
10.3.1. instanceof 97
10.3.2. Automatsko uključivanje klasa u dokument (eng. autoloading) 97
10.3.3. Kloniranje objekata 98
10.3.4. Interfaces (sučelja) 98
10.3.5. Apstraktne (eng. abstract) klase i metode 100
10.3.6. Pomoćne funkcije za rad s objektima i klasama 100

11. RAD S KORISNIČKIM UNOSOM - FORME 102

11.1. Izrada formi u HTML-u 102
11.2. Dobavljanje korisničkog unosa 105
11.3. Objektno orijentirani pristup izradi i validaciji forme 108
11.3.1. PHP-Bootstrap-Form 108
11.3.2. Validacija podataka u PHP-Bootstrap-Form 119
11.4. Bootstrap confirmation 123

12. MySQL 125

12.1. Podržani tipovi podataka 125
12.2. Ograničenja tipova podataka (eng. *Constraint*), dodatni atributi 127
12.3. Stroj baze podataka (eng. *storage engine*) 129
12.4. Osnovne SQL izjave 130

13. PhpMyAdmin 132

13.1. Izrada baze podataka 133
13.2. Dodavanje novog korisnika i određivanje privilegija 133
13.3. Izrada tablica unutar baze 136

14. MySQL i PHP 137

14.1. MySQLi 137
14.1.1. MySQLi kreiranje konekcije 137
14.1.2. Odabir baze podataka 138

14.1.3. Izvođenje SQL izjava 138
14.1.4. Izvođenje pripremljenih izjava (mysqli_stmt klasa) 139
14.1.5. Korištenje vraćenih podataka (mysqli_result klasa) 140
14.1.6. Rad s pogreškama vezanim uz rad s bazom podataka 143
14.2. Izdvajanje pristupnih podatka za MySQL 144
15. ZAŠTITA WEB STRANICA 146
15.1. Zaštita lozinki 146
15.2. Korištenje HTTPS 148
15.3. Alati za sigurnost web stranice 149
15.4. Dozvole za datoteke (Apache Web Server/Linux-Unix OS) 149
15.5. .htaccess 150
15.5.1. Korisničke stranice s pogreškama 150
15.5.2. Zabrana prikaza indeksa direktorija 151
15.5.3. Zabrana i dopuštanje određenih IP adresa 151
15.5.4. Alternativne indeks stranice 151
15.5.5. Preusmjeravanje 151
15.5.6. Zaštita lozinkom 152
15.5.7. Mod_rewrite 152
15.6. Podešavanje Robots.txt 155
16. SESIJE 156
16.1. Korištenje zaglavlja (eng. header) za preusmjeravanje 158
17. PAGINACIJA 159
18. UPLOADS 161

1. UVOD U PHP

PHP (*Hypertext Preprocessor*) je jedan od najpopularnijih jezika koji se koristi za razvoj web aplikacija. *PHP: Hypertext Preprocessor* je tzv. rekurzivna skraćenica, kod koje se sama skraćenica pojavljuje u značenju. Jednostavan je za učenje zbog sintakse koja je bazirana na C, Java i Perl programskom jeziku s nekoliko jednostavnih specifičnosti. Omogućuje programeru da brzo razvije programe koristeći tehnike proceduralnog i objektno-orijentiranog programiranja. U PHP-u moguće je korištenje mnogih postojećih biblioteka koje su uključene u osnovnu instalaciju ili se mogu instalirati unutar PHP okruženja. Lakoća dodavanja biblioteka u okruženje je jedna od glavnih prednosti i ono što PHP čini popularnim.

PHP je skriptni jezik koji se izvršava na poslužitelju (eng. *server*), a glavna mu je namjena dinamičko stvaranje web stranica. Većina jezika koji se koriste u web programiranju (PHP, JavaScript, ASP i drugi) podskup su skriptnih jezika za koje je karakteristično da se program ne čuva u izvršnoj (eng. *executable*) datoteci (koja je prevedena samo jednom), već se prevođenje naredbi obavlja pri svakom izvršavanju programa. Datoteke i dijelovi kôda napisani u takvim jezicima nazivaju se skriptama.

PHP je nastao 1994. godine kao osobni projekt Rasmus Lerdorfa, a kasnije se u njegov razvoj uključio veliki broj programera koji su doprinijeli razvoju jezika. Originalno je skraćenica PHP označavala *Personal Home Page*, a kasnije je značenje promijenjeno i danas znači *PHP: Hypertext Preprocessor* (PHP: hipertekstualni pretprocesor) što opisuje glavnu funkciju jezika PHP - da na temelju PHP naredbi generira HTML, jezik kojim se opisuje hipertekst.

PHP je u početku tek bila pomoć Rasmusu Lerdorfu za izradu vlastite web stranice tako što je s nekoliko CGI programa pisanih u programskom jeziku C zamijenio programe pisane u programskom jeziku Perl koje je do tada koristio. Inačica 1 i 2 nisu bile previše popularne, ali treća inačica, koja je izašla 1998. godine privukla je velik broj poklonika tada novog programskog jezika. 2000. godine, izlazi i 4. Inačica a 2004. godine izašla još uvijek aktualna 5. inačica. Inačica 6. nikad nije bila objavljena dok je trenutna verzija PHP 7.

Osim kao podrška web aplikacijama danas je PHP moguće koristiti i kao konzolnu aplikaciju, ali je jednako moguće pisati i aplikacije u PHP-u s punim grafičkim korisničkim sučeljem, upotrebljavati OpenGL biblioteke za trodimenzionalnu vizualizaciju i slično. Danas je PHP vrlo moderan programski jezik koji je najčešće u upotrebi na webu.

PHP je jezik otvorenog kôda. Svaka verzija jezika je kreirana koristeći unos od samih programera. To omogućava da jezik, vremenom, napreduje i kreće se u pravcu u kojem ga pokreću korisnici. Programski jezik otvorenog kôda razvija zajednica zainteresiranih korisnika. Zajednica prihvaća unose od programera za preporučene nadgradnje i ispravke. Nekoliko članova zajednice radi zajednički na predstavljanju prijedloga i na ispravljanju grešaka u jeziku. Otvoreni kôd omogućava svim zainteresiranim osobama izravno sudjelovanje u unapređenju programa. Prednosti ovakvih projekata su otvorenost prema svim sudionicima i otvoreni poziv drugima na aktivno sudjelovanje. Jezici otvorenog kôda su besplatni. Jezike koji nisu otvorenog kôda (kao što je Microsoft C#) kreira i ažurira kompanija ili glavna organizacija. Jezici koji nisu otvorenog kôda obično nisu besplatni.

Na početnoj stranici *www.php.net* nalaze se informacije o svakom najnovijem izdanju jezika, kao i informacije o budućim izdanjima, budućim planovima za određena izdanja i planiranim datumima za izdavanje. Na navedenoj stranici moguće je pronaći i druge slične informacije o PHP-u, uključujući poveznice, informacije i savijete za korištenje PHP-a. Na njoj se korisnici mogu uključiti u budući razvoj jezika, testiranje beta verzija te izvještavati o greškama u programu. Najvažniji dio PHP web stranice su stranice dokumentacije. Preko stranica dokumentacije moguće je pronaći opise funkcija jezika te pronaći primjere kôdova koji dodatno objašnjavaju korištenje.

Stranica za preuzimanje omogućuje lak pristup najnovijim verzijama jezika. Često se novije verzije PHP-a za lokalni razvoj preuzimaju u paketima kao što je WAMP, LAMP, MAMP ili XAMPP. Ovi paketi omogućavaju laku instalaciju više proizvoda istovremeno. Bez upotrebe navedenih paketa potrebno je napraviti više posebnih instalacija, te je veća mogućnost pojave greške ako se instaliraju nekompatibilne verzije, a također mogu se javiti problemi u konfiguraciji odvojeno instaliranih paketa. Navedene kombinacije uključuju Apache Web Server, MySQL i PHP za specifičan operativni sistem (Windows, Linux i Mac). Ovi paketi su otvorenog kôda.

PHP 7 je najnovija nadogradnja PHP programskog jezika. Uvela je novi procesor PHP-a (Zend Engine 3), mnogo novih mogućnosti i mnogo promjena. U PHP-u 7 došlo je do velikih ubrzanja u brzini i izrađen je s minimalnom kompatibilnošću s ranijim verzijama.

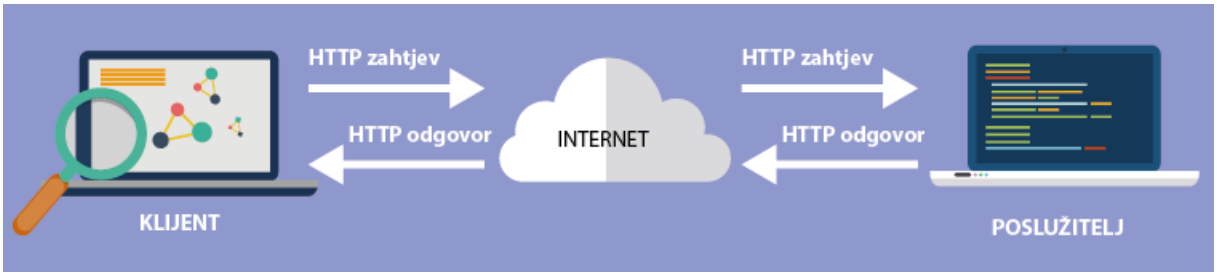
Verzija PHP-a osim sedmice koja je trenutačno aktualna je verzija PHP 5.6 za koju se jedino izrađuju sigurnosna unaprjeđenja. Sve verzije PHP-a prije 5.6 ne dobivaju nadogradnje.

Prednost u novoj verziji PHP-a je olakšana nadogradnja koja je uključena u sam jezik iako PHP 7 ima lošu kompatibilnost s prethodnim verzijama. U posljednjim verzijama PHP-a 5.x brojna svojstva su proglašena zastarjelima i od verzije 7.0 su izbačena.

1.1. ARHITEKTURA KLIJENT – POSLUŽITELJ I TEHNOLOGIJE IZVRŠAVANJA

U komunikaciji putem Interneta sudjeluju dvije strane: klijent i poslužitelj (*server*). Klijent (korisnikov web preglednik) šalje zahtjev, npr. adresu neke stranice, a poslužitelj isporučuje odgovor, koji sadrži HTML kôd tražene stranice (i zajedno s njim binarne datoteke, npr. slike, video...), što će klijent interpretirati i prikazati korisniku kao web stranicu.

Komunikacija između poslužitelja i klijenta odvija se putem HTTP (ili HTTPS) protokola, koji se sastoji od HTTP zahtjeva i odgovora. Komunikacija preko Interneta je dvosmjerna – i klijent može unutar HTTP zahtjeva poslati neke podatke poslužitelju. Slika 1. Prikazuje komunikaciju preko interneta klijenta i poslužitelja.



Slika 1. Komunikacija klijenta i poslužitelja

Pri izradi web stranica mogu se izrađivati statične HTML stranice, stranice s klijentskim skriptama i stranice s poslužiteljskim skriptama.

Sa statičkim HTML stranicama, web stranica će uvijek biti ista, a mogućnosti joj se svode na prikazivanje teksta i slika s mogućnošću povezivanja (preko linkova) s drugim stranicama.

Klijentske skripte se zajedno s HTML kôdom (unutar HTML datoteka ili unutar zasebnih datoteka) nalaze na poslužitelju. Kad stigne zahtjev, zajedno s HTML kôdom se šalju klijentu, a na klijentu se izvršavaju kad budu pozvane (nakon što se stranica učita u preglednik, kad korisnik klikne na dugme i slično). Pod klijentskim skriptama najčešće podrazumijevamo skripte pisane u JavaScript jeziku koji je podržan u najvećem dijelu web preglednika. Uz njega postoje još JScript i VBScript.

Poslužiteljske skripte, kao i klijentske skripte, nalaze se na poslužitelju i mogu biti ubačene u HTML kôd, ili se mogu nalaziti u zasebnim datotekama. Kad stigne zahtjev od klijenta, skripta se izvršava na poslužitelju, a kao rezultat izvršavanja dobiva se HTML kôd koji se šalje klijentu. Najveća prednost poslužiteljskih skripti je njihova mogućnost povezivanja s bazama podataka. Korištenjem poslužiteljskih skripti moguće je izrađivati web aplikacije s dinamičkim sadržajem koji se čita iz baze podataka i koji korisnici mogu mijenjati preko web aplikacije. Tablica 1. Prikazuje usporedbu klijentskih i poslužiteljskih skripti.

Tablica 1. Usporedba poslužiteljskih i klijentskih skripti

Klijentske skripte	Poslužiteljske skripte
Izvršava se na klijentu	Izvršava se na poslužitelju
Može doći do problema zbog nekompatibilnosti preglednika	Izvršavanje skripte je neovisno o pregledniku

Klijentske skripte	Poslužiteljske skripte
Nemogućnost izravnog povezivanja s bazom podataka	Povezivanje s bazama podataka
Mogućnost pokretanja skripti različitim događajima (pokreti mišom, učitavanje prozora i sl.)	Pokreću se samo klikom na link, gumb ili predavanjem forme
Reagiraju trenutačno, stranica se ne osvježava	Potreban je komunikacija sa serverom i osvježavanje cijele stranice
Poslužitelj se rasterećuje korištenjem resursa klijenta	Koriste se samo resursi poslužitelja

Tipične dinamičke web stranice uobičajeno su kombinacija klijentskih i poslužiteljskih skripti pri čemu se uzimaju prednosti obje tehnologije. Klijentsko i poslužiteljsko skriptiranje nisu alternativa jedno drugome, odnosno, većinu stvari koje se može postići jednim načinom programiranja ne može se postići drugim te ih je najbolje koristiti u kombinaciji.

AJAX (*Asynchronous JavaScript and XML*) je tehnologija koja kombinira klijentsko i poslužiteljsko skriptiranje. Na određenu korisnikovu akciju klijentska skripta šalje pozadinski zahtjev skripti na poslužitelju koja obavlja dohvaćanje podataka iz baze podataka i šalje ih korisnikovom pregledniku, bez osvježavanja cijele stranice. Upotrebom AJAX tehnologije web aplikacije postaju brže za uporabu i imaju bolje korisničko sučelje od aplikacija koje se temelje samo na poslužiteljskim tehnologijama.

1.1.1. Klijentske web tehnologije

JavaScript

U posljednjim godinama JavaScript je postao jedan od glavnih skriptnih jezika pri izradi web stranica i aplikacija. Postoje mnogi razlozi za to uključujući njegov rad s objektima i mogućnost brze izrade aplikacija. Poznavanje JavaScripta esencijalno je znanje za svakog web programera. Kako bi se ispravili nedostaci ugrađeni u sam jezik, programeri su kreirali mnogo biblioteka koje pokušavaju riješiti različite probleme. Neke od najšire korištenih biblioteka su Lodash, Ramda i TypeScript. Osim navedenih u upotrebi je još mnogo programskih okvira i biblioteka. Najčešće korišteni su: Vanilla JavaScript, jQuery (najpoznatija biblioteka čije su mogućnosti danas implementirane u novije verzije JavaScripta), i18n (pojednostavljuje lokalizaciju i čini aplikaciju višejezičnom), Moment (pojednostavljuje rad s datumom i vremenom), ThreeJS (omogućuje izradu, prikazivanje i animaciju 3D grafika), Angular, React, Ember, Backbone i dr.

1.1.2. Poslužiteljske web tehnologije

Node.js (JavaScript)

Node.js je među novijim u listi poslužiteljskih web tehnologija i najbrže rastući jezik. U upotrebi je od 2009. godine i koristi se kako bi se pokretao JavaScript na serverskoj strani. Prednost je u tome da nije potrebno učiti novi jezik za razvoj web aplikacija. Koristi se za izradu web aplikacija koje se izvršavaju u stvarnom vremenu (web razgovori (eng. *chat*), igre i sl.).

Pošto je noviji od većine drugih jezika u njega su uključena određena obilježja koja nedostaju u ostalim jezicima. Express je popularni web programski okvir koji može pomoći u razvoju Node.js web aplikacija. Poznate stranice koje koriste Node.js su: PayPal, Uber, LinkedIn, Netflix.

PHP

PHP je najčešće korišten serverski programski jezik. Preko 80% web stranica koristi PHP. Koristi se u Wordpress platformi koja pokreće 25% svjetskih stranica uključujući najpopularnije blogove i stranice s vijestima. PHP je narastao u toj mjeri da se danas koristi kao jezik generalne namjene na webu umjesto samo skriptnog jezika. Još uvijek dobiva česte nadogradnje. PHP je dobio velika unaprjeđenja službenim izlaskom PHP verzije 7 koja daje značajna poboljšanja kao poboljšani engine, uvelike povećana brzina, bolji rad s greškama i dr. Ukoliko se želi izrađivati stranice koje su bazirane na sadržaju, PHP je provjerena metoda.

Postoji i nekoliko *frameworka* (razvojnih platformi) za PHP (Laravel, Code Igniter, Symfony, Zend, Yii 2, CakePHP, Fuel PHP, FATFree, Aura i dr.) čiji je cilj olakšati programiranje, testiranje, uvesti bolje značajke sigurnosti, MVC strukturu i dr.

Poznate stranice koje koriste PHP su: Facebook, Wikipedia, Wordpress i dr.

Java

Java je popularan jezik koji se koristi na mnogim opsežnim web stranicama. Kod izrade manjih aplikacija može zahtijevati više vremena pri izradi pa je takve aplikacije bolje izrađivati s jednostavnijim tehnologijama. Koristi se kod mnogih web aplikacija banaka i osiguravajućih kuća kako bi one mogle komunicirati s ostalim sistemima i programskim rješenjima.

Spring je jedan od popularnijih programskih okvira (engl. *framework*) za Java web aplikacije i uobičajeno se smatra esencijalnim alatom pri razvoju web aplikacija. Poznate stranice koje koriste Javu su: Google, Amazon, eBay i dr.

Ruby

Ruby je popularan za manje aplikacije jer je pogodan za brzi web razvoj. Ruby je objektno orijentirani programski jezik generalne namjene. Rails je najpoznatiji programski okvir koji koristi Ruby. Postao je popularan od 2010. godine. Programski okvir je relativno opširan te postoje jednostavniji programski okviri kao Sinatra koji su dobri za početnike. Poznate web stranice izrađene na Ruby: Twitter (u ranijim danima), GitHub, Groupon, Airbnb.

Python

Pythnon je jezik koji se interpretira, generalne je namjene. On je jedan od najboljih jezika za početnike. To je jezik koji ima sintaksu koja se jednostavno čita, koja omogućuje manji fokus na sintaksu i više na učenje programa. Flask je popularan mikro programski okvir za izradu web aplikacija. Django je Python ekvivalent za Ruby on Rails. Poznate stranice koje koriste Python su: Youtube, Instagram, Dropbox, Quora.

Golang

Golang (ili Go) je jedan od novijih jezika u web razvoju i postaje vrlo popularan. To je jezik koji se prevodi te ima minimalistički pristup i fokus na razvojno iskustvo. Jezik je razvijen od strane Google-a i brzo je postao popularan.

ASP.NET

ASP.NET je programski okvir baziran na C#, jezik koji se izvršava na poslužiteljskoj strani dizajniran od strane Microsofta 2002. godine kako bi se kreirale dinamičke web stranice. .NET se razvijao od tada i još uvijek se koristi.

C#

C# je jezik baziran na C programskom jeziku i dizajniran od strane Microsofta. Ovaj jezik se može koristiti za izradu poslužiteljskih programa ali se češće koristi za izradu Windows programa.

Osim navedenih popularnih jezika postaje i jezici koji se više ili manje koriste na webu kao ColdFusion, Elixir, Clojure, Rust, Haskell, Scala, Erlang, F#, Kotlin, Dart, Crystal, Swift, Pharo i drugi.

1.1.3. HTML/ CSS

HTML i CSS su dva odvojena jezika ali pošto zavise jedan o drugom često se uparuju zajedno. HTML opisuje strukturu stranice dok CSS opisuje kako će stranica izgledati. HTML (skraćeno za *Hypertext Markup Language*) je skriptni jezik koji opsuje tipove sadržaja koje dokument zahtjeva i povezuje podatke na jedno mjesto. CSS (skraćeno od *Cascading Style Sheets*) je jezik koji opisuje stilove dokumenta povezane uz boju, tipografiju, položaje i dr.

HTML5 i CSS3 su posljednja izdanja navedenih jezika koja su uvela velike novine u izradu web sadržaja. Nova poboljšanja i izdanja još uvijek izlaze pod navedenim imenima a njihova implementacija može biti spora od strane web preglednika.

1.2. IZVRŠAVANJE SKRIPTE U PHP-U

Prilikom pristupanja stranicama weba preglednik šalje zahtjev poslužitelju koji mu isporučuje datoteku u HTML obliku. Preglednik tu datoteku grafički oblikuje i prikazuje korisniku. U samom početku datoteke su bile fizički smještene na sam poslužitelj i adresa koju je preglednik tražio od poslužitelja ujedno je bila i njen stvarni naziv na lokalnom disku. Takav oblik smještaja, gdje su sve datoteke na poslužitelju nazivamo dvoslojna arhitektura (eng. *Two-tier architecture*).

Danas je puno češća troslojna arhitektura. U njoj datoteke i podaci nisu smješteni izravno na poslužitelju već i u odvojenoj bazi podataka. Baza podataka može biti smještena na istom ili na odvojenom poslužitelju, a u slučaju velikog broja upita moguće je da postoji velik broj poslužitelja koji se zajedno koriste kao jedna baza podataka. Kada se govori o programskom jeziku PHP on je u pravilu dio troslojne arhitekture budući da se u većini slučajeva dio podataka pohranjuje u bazu podataka. Troslojnu arhitekturu čine, kao što joj i ime govori, tri osnovna sloja. Prvi sloj je klijentski koji uključuje preglednik i samu mrežu Internet. Srednji sloj je poslužitelj weba na kojem se izvršavaju skriptni jezici ili izvršne datoteke, dok je posljednji, treći sloj, onaj u kojem se nalazi sustav za upravljanje bazom podataka i sama baza podataka.

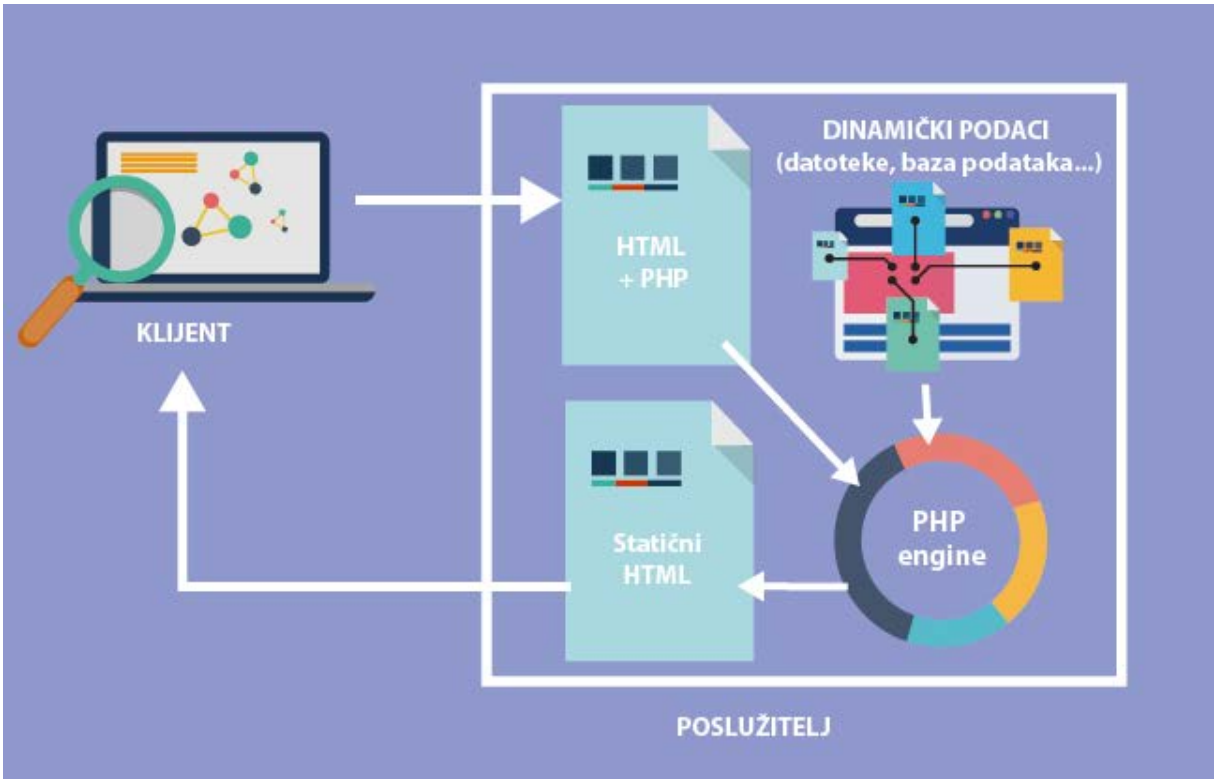
Temelj drugog sloja je poslužitelj weba. Danas se najčešće koristi poslužitelj weba Apache, ali je vrlo čest i Internet Information Server koji je komercijalni poslužitelj. Osim ova dva, koja čine najveći dio svih poslužitelja, na raspolaganju stoji i vrlo velik broj drugih koji su u pravilu specijalizirani za neko područje.

Apache web poslužitelj se najčešće koristi za postavljanje web stranica i rukovanje zahtjeva PHP web stranica. Kao i drugi web serveri, Apache može prihvatiti i vratiti zahtjeve za druge vrste datoteka, uključujući HTML, JavaScript, PERL, slike i RSS dokumente. Poslužitelj određuje koji se procesi trebaju izvršiti iz HTTP zahtjeva tj. iz ekstenzije traženih datoteka. Apache je, kao i PHP, proizvod otvorenog kôda. Svim promjenama na Apache serveru rukovodi Apache Software Foundation koji održava web stranicu apache.org.

Sloj baze podataka se koristi za pohranu i dohvat podataka. Ovaj sloj omogućava istovremeni pristup podacima s nekoliko odvojenih poslužitelja, osigurava sigurnost, tajnost i integritet podataka, a također uz odgovarajuću softversku i hardversku podršku moguće je jednostavno raditi sigurnosnu kopiju (eng. *backup*) svih podataka. Zadaću upravljanja podacima u bazi podataka preuzima sustav za upravljanje bazom podataka (eng. *database management system – DBMS*). Sustav za upravljanje bazom podataka brine o smještaju i dohvatu podataka. Iako je sam po sebi vrlo složen on u pravilu znatno pojednostavnjuje sve operacije. Danas gotovo svi moderni sustavi za upravljanje bazom podatka za dohvat, analizu i obradu koriste jezik SQL (eng. *Structured Query Language*). Na raspolaganju postoje brojni sustavi za upravljanje bazama podataka. Neki od najpoznatijih su: Oracle, DB2, MS SQL, MySQL, MariaDB i PostgreSQL. Od navedenih prva tri su komercijalna, dok su MySQL, MariaDB i PostgreSQL besplatni programi otvorenog kôda.

Prilikom pristupa serveru kada se zatraži određena PHP stranica poslužitelj automatski izvršava PHP datoteku i vraća klijentu statičnu HTML stranicu. Klijent će u web pregledniku primiti samo generiranu HTML stranicu bez PHP kôda tako da je on sakriven od klijenta. Kako

bi server znao da je potrebno izvršiti PHP kôd, datoteke pisane u PHP-u trebaju imati ekstenziju .php. Na slici 2. prikazano je izvršavanje dinamičke skripte.



Slika 2. Izvršavanje poslužiteljske skripte

Poslužitelj će najprije traženu PHP stranicu poslati u PHP procesor, koji ju prevodi (liniju po liniju). Dok se kôd prevodi, procesor će zatražiti izvršavanje eventualnih SQL izjava povezivanjem s bazom podataka. DBMS će vratiti rezultate izvršenja SQL zahtjeva u PHP procesor. PHP procesor će upotrijebiti te rezultate za formatiranje ispisa koji će biti proslijeđen poslužitelju. Poslužitelj će podatke koje je vratio PHP procesor zajedno s HTML (i/ili JavaScript, CSS) vratiti u pretraživač na računalu korisnika. Pretraživač tada prevodi HTML, JavaScript i CSS te prikazuje rezultate tražene stranice.

1.3. RAZVOJNO OKRUŽENJE

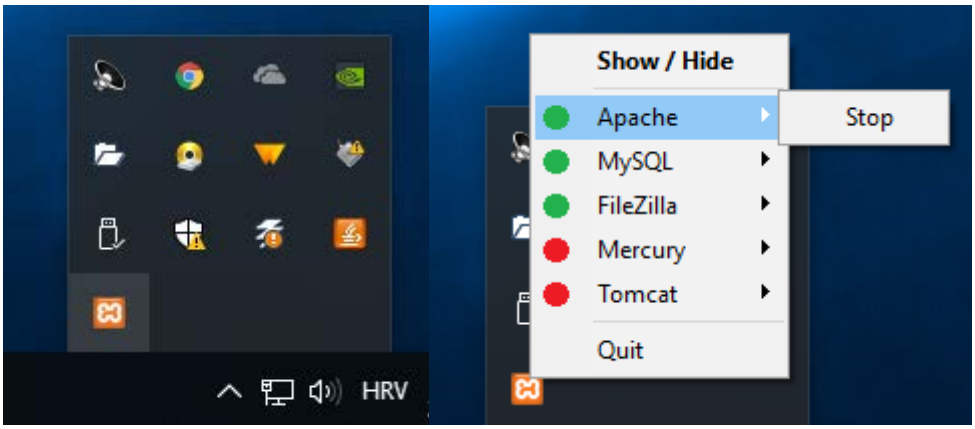
PHP, Apache i MySQL zajedno omogućuju izradu dinamičkih web stranica. Kako bi se stranice mogle izrađivati i testirati na lokalnom računalu potrebno je razvojno okruženje u kojem su dostupne navedene tehnologije. Postoje mnogi razvojni paketi koji omogućuju kombinaciju ovih proizvoda, zajedno sa drugim alatima, kao što je PhpMyAdmin (web sustav za podešavanje baze podataka). Najpopularniji proizvodi su EasyPHP, XAMPP i WampServer.

EasyPHP se može preuzeti sa stranice http://www.easyphp.org/easyphp-devserver.php pri čemu treba obratiti pažnju da je riječ o razvojnom okruženju a ne o paketu koji se postavlja na stvarni poslužitelj (*webserver* verzija). Osnovna instalacija paketa je dovoljna za izvršavanje PHP skripti. Nakon instalacije, datoteke će biti locirane u direktoriju EasyPHP.

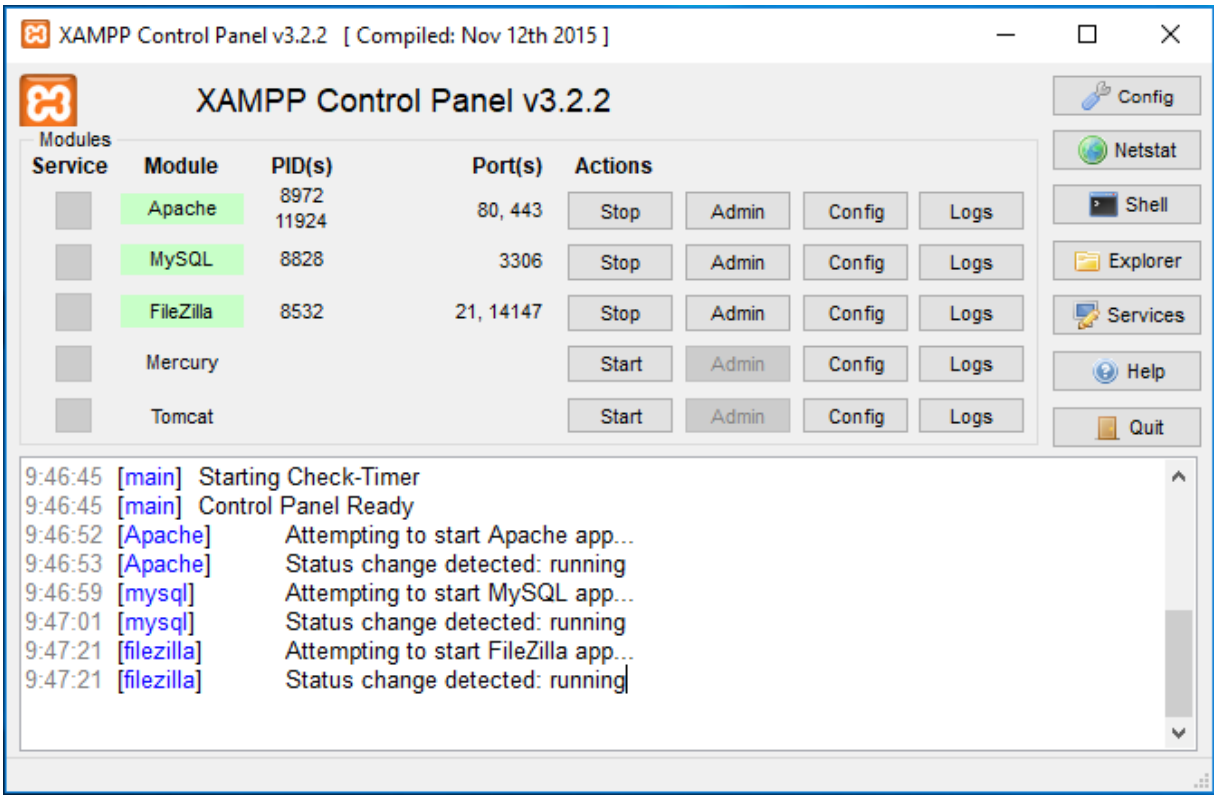
XAMPP je sličan EasyPHP-u te uključuje verzije za Windows, Linux i OS X sustave. Također uključuje puno dodataka među kojima su neki od najpopularnijih sustava za upravljanje sadržajem (engl. *CMS- Content Management System*) – WordPress, Drupal i Joomla. Može se preuzeti sa stranice https://www.apachefriends.org/download.html.

WampServer je također jedan od mogućih alata kojeg je moguće koristiti kako bi se lokalno postavio server za testiranje i pisanje PHP skripti. Moguće ga je preuzeti na http://www.wampserver.com/en/#download-wrapper.

Jednom kad su instalirani sa svim navedenim alatima moguće je upravljati preko ikona programske trake gdje je moguće pronaći ikonu za odabrani proizvod. Slike 3. i 4. prikazuju ikonu i sučelje za rad s XAMP.



Slika 3. Ikona XAMPP programskog rješenja (lijevo) i desni klik na ikonu koji otvara izbornik za upravljanje servisima (desno)

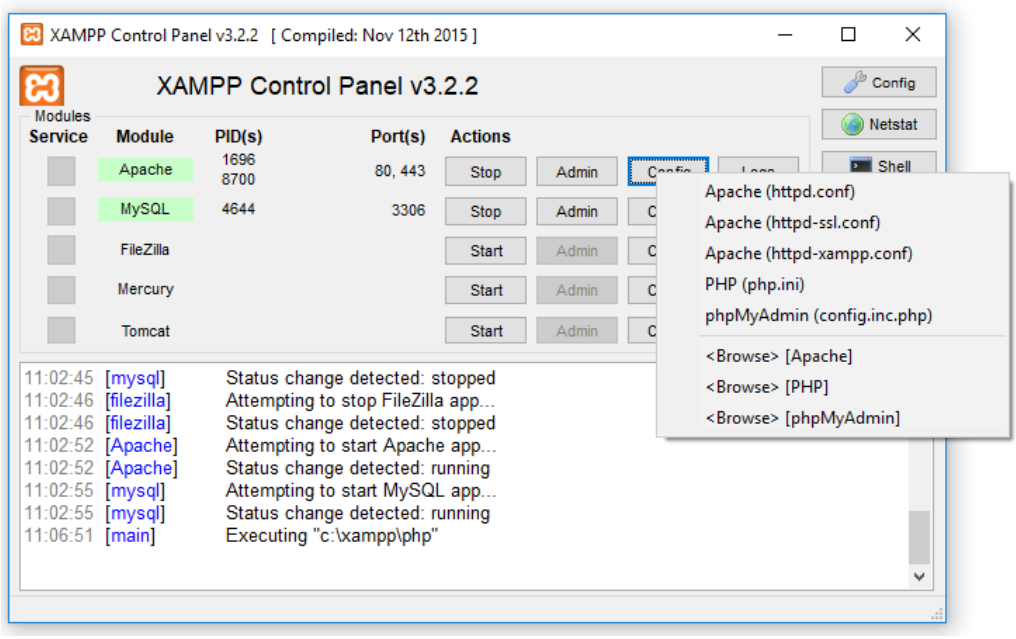


Slika 4. Programsko sučelje XAMPP alata koje se otvara duplim klikom na ikonu u programskoj traci

Svaka od instalacija tipično se postavlja na C disk Windows operativnog sustava (ukoliko se prilikom instalacije ne zada drugačije). Unutar instalacijskog direktorija moguće je pronaći direktorij koji je namijenjen za postavljanje web stranica. Tipično takvi direktoriji imaju naziv *htdocs*, *www* ili *public*. Naziv ovisi o konfiguraciji i moguće ga je promijeniti u konfiguracijskoj datoteci Apache servera (httpd.conf). Slika 5. Prikazuje kako je moguće pristupiti konfiguracijskim datotekama Apachea i PHP-a pritiskom na gumb *Config*.

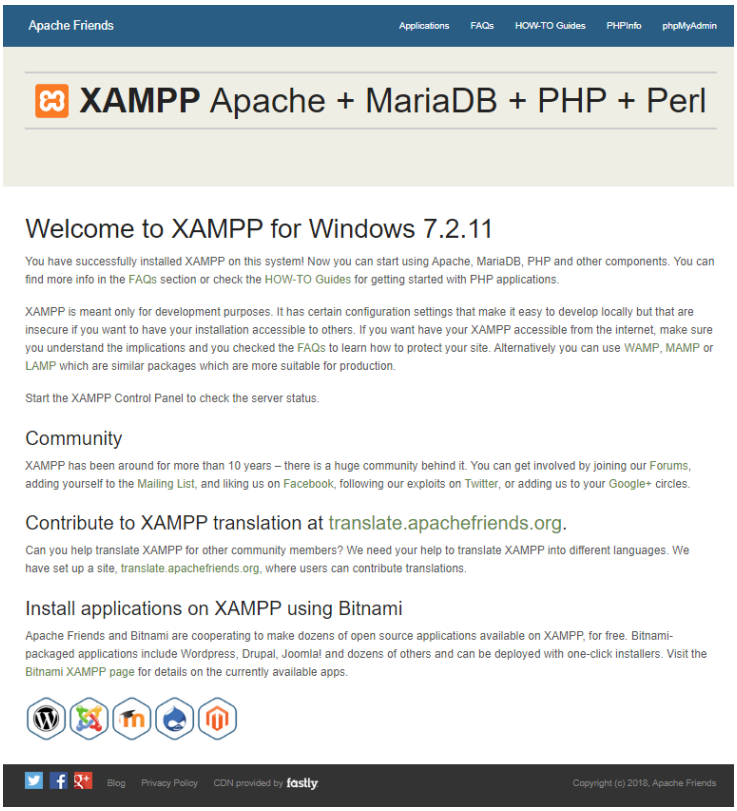
U konfiguracijskoj datoteci Apache servera (httpd.conf) moguće je pronaći i prema potrebi promijeniti liniju *DocumentRoot* koja označava korijenski (eng. *root*) direktorij. Svaka promjena u httpd.conf zahtjeva ponovno pokretanje Apache servera kako bi se promjene učitale. Početne putanje ovisno o instalaciji paketa su:

```
XAMPP: "C:\xampp\htdocs"
EasyPHP: „C:\Program Files (x86)\EasyPHP-DevServer-
14.1VC11\data\localweb\“
WampServer: "C:\wamp\www"
```



Slika 5. Pristup konfiguracijskim datotekama iz XAMPP sučelja

Unutar *root* direktorija postavljaju se sve datoteke koje se žele pregledavati i testirati. Kako bi u pregledniku otvorili pregled web sjedišta potrebno je u pregledniku upisati *localhost* ili *localhost:80* (ili drugi broj porta ukoliko je on izmijenjen). Tipično se tada otvara web stranica samog servera koja je postavljena unutar *root* direktorija. Primjer za XAMPP prikazan je na slici 6.



Slika 6. Otvaranje početne stranice na XAMPP serveru upisivanjem localhost u web pregledniku

Ukoliko se želi nadodati još neki direktorij kao *root* direktorij tj. kao alias (primjerice kada se radi na velikim projektima) moguće je na Apache serveru konfigurirati alias. Alias označava usmjeravanje prema direktoriju koji se može nalaziti izvan osnovne instalacije serverskog okruženja. Usmjeravanje daje dojam kao da se dokumenti nalaze pod web direktorijem servera. Primjerice, XAMPP web direktorij se nalazi na osnovnoj lokaciji "C:\xampp\htdocs" ali se želi održavati stranice postavljene u drugi direktorij "projekt" lociran na "D:\projekt". Kako bi se omogućio navedeni način rada potrebno je kreirati alias koji upućuje poslužitelj da traži datoteke na jednoj lokaciji ali ih prikazuje kao da je direktorij relativan serverskom korijenskom (*root*) direktoriju.

Konfiguracija je prikazana na sljedećim primjerima:

XAMPP

1. Mijenja se httpd-xampp.conf
2. Potrebno je pronaći <IfModule alias_module>
3. Unutar navedenog ubacuje se:

```
Alias /nazivAliasa "C:/putanja"
<Directory "C:/putanja">
Options Indexes FollowSymLinks MultiViews ExecCGI
AllowOverride All
Require all granted
</Directory>
```

Gdje je naziv aliasa i putanju potrebno prilagoditi zahtjevima postavki.

EasyPHP

U EasyPHP-u alternativni se direktoriji mogu kreirati na ekranu *Administration*. Otvara se ikona na programskoj traci (u donjem desnom uglu ekrana), potrebno je napraviti desni klik na ikonicu EasyPHP i odabrati *Administration*. Na sredini ekrana potrebno je odabrati *Local Files*, a na desnoj strani ekrana opciju *Add an Alias* te pratiti upute. Direktorij mora postojati prije dodavanja alternativnog direktorija.

WampServer

Potrebno je pokrenuti WAMP pritiskom na ikonu na programskoj traci. Odabire se "*Apache*" i "*Alias directories*". Nakon toga potrebno je odabrati "*Add an alias*". U *Command prompt* prozoru potrebno je napisati naziv aliasa, enter i putanju do direktorija (primjerice "c:/putanja") i enter kako bi se izašlo iz *Command prompta*. Potrebno je još dodatno podesiti httpd.conf datoteku tako da se traženjem *DocumentRoot*, nešto ispod nje, pronađe linija koja izgleda kao

```
<Directory />
    AllowOverride none
    Require all denied
</Directory>
```

I nakon nje potrebno je dodati linije:

```
<Directory "c:/putanja">
    AllowOverride none
```

```
        Require all granted
    </Directory>
```

Nakon izmjena datoteka se sprema i ponovno pokreće WAMP te je stranici moguće pristupiti sa localhost/nazivaliasa/ upisano u polje preglednika (obavezno s završnom kosom crtom).

1.3.1. Najčešći problemi instalacije

Nedostaje C# biblioteka

PHP 7 (kao i ranije verzije PHP-a) zahtjeva Microsoft Visual Studio C# biblioteku. Ukoliko je na računalu instaliran Windows 8 ili noviji operativni sustav ili novija verzija Microsoft Visual Studio ova biblioteka je već instalirana. Ukoliko se pojavi greška koja ukazuje na nepostojanje C# biblioteke ili na pogrešnu verziju, potrebno ju je instalirati. Odgovor se može pronaći na Microsoft uputama za ispravljanje greške. Odgovor bi trebalo uključivati link za preuzimanje nedostajućih datoteka i upute za instalaciju.

Konflikti ulaza (engl. port)

Ukoliko se na računalu nalazi servis koji koristi port 80, standardni port za HTML promet primiti će se poruka o grešci od Apache servera kada će se pokušati pokrenuti.

Ukoliko se želi može se osloboditi port tako da se isključe drugi servisi koji ga koriste. Nakon programiranja moguće je ponovno pokrenuti servis (ili ponovno pokrenuti računalo). Kako bi ostvarili navedeno mogu se pratiti sljedeće upute:

1. Otvoriti *Microsoft Windows 7/8/10 Task Manager* (Ctrl+Alt+Delete)
2. Selektirati karticu *Services*
3. Potražiti servise *SQL Server Reporter, Web Deployment Agent, BranchCache, Sync Share Service, WAS (IIS Administrator)* ili *W3SVC* (nazivi se mogu razlikovati ovisno o verziji Windows sustava). Ukoliko se pronađe pokrenut servis, potrebno ga je isključiti i ponovno pokrenuti Apache. Ukoliko se Apache i dalje ne pokreće ispravno potrebno je ponovno uključiti ranije isključeni servis te pokušati isključiti drugi.

Ukoliko je potrebno da drugi servisi budu pokrenuti ili račun nema privilegije administratora za isključivanje servisa na portu 80, može se promijeniti lokacija standardnog porta za Apache. Navedeno je moguće ostvariti tako da se promijeni konfiguracijska datoteka za Apache (httpd.conf). Datoteku je moguće otvoriti u Notepadu (ili drugom standardnom editoru teksta). Najprije je potrebno spremiti kopiju datoteke kako bi se eventualno moglo oporaviti pogreške koje se dese. Potrebno je pronaći liniju gdje je zapisano *Listen 80* (XAMPP server) te promijeniti 80 na 8080 ili na 81 *(Napomena: Kada se koristi Notepad ili bilo koji drugi editor za tekst potrebno je obratiti pažnju kako je datoteka spremljena. Datoteka mora imati .conf ekstenziju te ne smije biti spremljena u nekom drugom obliku kao npr. httpd.conf.txt).* Na taj način Apache server će pratiti port koji se ne koristi u većini slučajeva. Nakon izmjene potrebno je ponovno pokrenuti Apache.

1.4. ALATI ZA IZRADU PHP DATOTEKA

Pri pisanju programskog kôda moguće je izabrati tekstualni editor ili IDE (engl. *Integrated Development Environment*). IDE je paket koji omogućuje pisanje, izvršavanje i testiranje kôda na istom mjestu dok je tekstualni editor samo alat koji omogućuje upisivanje kôda. Tipično je IDE orijentiran samo na jedan jezik dok je u tekstualnim editorima moguće raditi s više programskih jezika.

Neka od popularnih IDE rješenja su:

Komercijalna:

PHPStorm (https://www.jetbrains.com/phpstorm/)

ZendStudio (http://www.zend.com/en/products/studio)

Besplatna:

Netbeans (https://netbeans.org/)

Aptana Studio (http://www.aptana.com/)

Eclipse (https://www.eclipse.org/)

PHP dokumenti se mogu izrađivati u bilo kojem tekstualnom editoru. Nije potrebno kupovati editor jer na raspolaganju stoje brojna besplatna rješenja. Kod izbora editora najbolje je odabrati alat koji će bojama istaknuti (*engl. syntax highlighting*) kôd te obilježiti pogreške. Od velike pomoći u takvim alatima mogu biti i prijedlozi naredbi tj. inteligentno dovršavanje kôda (*engl. intelliSense*).

Poznatiji komercijalni editori kôda za PHP su: Sublime Text, Coda, SlickEdit, Rapid, Ultraedit i dr. Neki od besplatnih su: Visual Studio Code, Atom, Notepad++, Brackets, jEdit, Programmer's Notepad, Komodo Edit, RJ TextEd, Codeanywhere i dr.

Bilo koji od navedenih editora ili IDE rješenja mogu olakšati izradu PHP kôda. Postoje i brojni drugi koji su dobri editori i dobra rješenja te se također mogu koristiti.

2. OSNOVE PHP-a

Kao što je ranije napomenuto PHP je skriptni kôd koji se ne prevodi prilikom postavljanja na server. Za PHP datoteke nužno je da imaju .php ekstenziju kako bi se proslijedile PHP procesoru prilikom izvršavanja. Unutar PHP datoteke moguće je postavljati samo PHP kôd ili on može biti postavljen u kombinaciji s HTML, CSS i/ili JavaScript kôdom. Bitno je da se PHP obilježi oznakama kako bi se odvojio od ostatka sadržaja u dokumentu. Postoje različite oznake koje se mogu koristiti.

2.1. PHP OZNAKE

PHP oznake obilježavaju dio kôda koji je pisan u PHP-u i preko oznaka se odvajaju od ostatka sadržaja koji može biti uključen u PHP datoteku.

Osnovna sintaksa

```
<?php
     echo "<p>Dinamički tekst</p>";
?>
```

Kratke oznake

Kratke oznake (engl. *Short-Tags*) mogu se dozvoliti u *php.ini* datoteci postavljanjem svojstva *short_open_tag* na *On*. Kratke oznake mogu biti problematične kad se koriste s XML-om i XHTML-om i zbog toga se ne preporuča njihova upotreba.

```
<?
     echo "<p>Dinamički tekst</p>";
?>
<?=
     echo "<p>Dinamički tekst 2</p>";
?>
```

2.1.1. Alternativne PHP oznake

PHP ranijih verzija podržavao je alternativne PHP oznake koje su uklonjene od verzije 7. pa ih se ne preporuča koristiti.

Script

```
<script language="php">
     print "<p>Dinamički tekst</p>";
</script>
```

ASP stil

ASP stil (php.ini→asp_tags = Off)

```
<%
 print "<p>Dinamički tekst</p>";
%>
```

2.2. KOMENTARI

Linijski komentari (C++ stil)

```
<?php
  //Naslov: PHP programiranje
  //Podnaslov: UVOD
  echo "Ovo je PHP program";
?>
```

Linijski komentari (Unix shell-style (Perl style))

```
<?php
  #Naslov: PHP programiranje
  #Podnaslov: UVOD
  echo "Ovo je PHP program";
?>
```

Više-linijski komentari (C++ stil)

```
<?php
  /* Naslov: PHP programiranje
  Podnaslov: UVOD */
  echo "Ovo je PHP program";
?>
```

2.3. ISPISIVANJE U PREGLEDNIK

Za ispisivanje podataka u preglednik na raspolaganju su funkcije i jezične tvorevine ugrađene u osnovu jezika. One su opisane u nastavku.

int print (string $arg)

Print nije prava funkcija (ona je jezična tvorevina) pa se može koristiti sa i bez zagrada. Glavna razlika u usporedbi s *echo* je da prima samo jedan argument. Vraća 1 ukoliko je izjava uspješno ispisana.

Sa zagradama:

```
<?php
     print ("Ovo je PHP program");
?>
```

Bez zagrada:

```
<?php
     print "Ovo je PHP program";
?>
```

void echo (string $arg1 [, string $...])

Echo nije prava funkcija pa se može koristiti sa i bez zagrada. Ukoliko se funkciji *echo* prosljeđuje više od jednog parametra ona se ne može koristiti sa zagradama. *Echo* ne vraća nikakav podatak.

```
<?php
   $doba1= "Jesen";
   $doba2= "Zima";
   echo $doba1, " i ", $doba2, " su godišnja doba.";
?>
```

Kad se ispisuju vrijednosti varijabli moguće ih je zapisivati i pod dvostrukim navodnicima ali njihove vrijednosti se ne ispisuju kada se koriste jednostruki navodnici, već se ispisuje naziv varijable.

```
echo $doba1; //ispisuje Jesen
echo "$doba1"; //ispisuje Jesen
echo '$doba1'; //ispisuje $doba1
```

int printf (string $format [, mixed $args [, mixed $...]])

Koristi se prilikom ispisa statičnog teksta i dinamičke informacije. Pomoću *printf* moguće je ispisati formatirani ispis. Funkcija vraća dužinu ispisanog stringa.

```
printf("Tjedan ima: %d dana.", 7);
printf("Tjedan ima: %d dana i %d sati.", 7, 168);
```

Type specifiers- češće korišteni

%b -integer; prezentiran kao binaran broj

%c -integer; prezentiran kao slovni znak koji odgovara toj ASCII vrijednosti

%d -integer; prezentiran kao decimalni broj s predznakom

%f -floating-point number; prezentiran kao floating-point number

%o -integer; prezentiran kao oktalni broj

%s - string; prezentiran kao string

%u -integer; prezentiran kao decimalni broj bez predznaka

%x -integer; prezentiran kao heksadecimalan broj- s malim slovima

%X -integer; prezentiran kao heksadecimalan broj- s velikim slovima

string sprintf (string `$format` [, mixed `$args` [, mixed `$...`]])

Razlika sprintf i printf je da se izlaz postavlja u string i ne predaje pregledniku.

3. VARIJABLE I KONSTANTE

3.1. VARIJABLE

Varijabla je imenovana memorijska lokacija koja sadrži podatke kojima je moguće manipulirati provođenjem programa.

Varijable su u PHP-u reprezentirane uz pomoć dolar znaka ($) kojeg prati naziv varijable. Naziv varijable je osjetljiv na verzale i kurente. Valjani naziv varijable započinje sa slovom ili _, nakon čega slijede slova, brojevi, _ i ostali ASCII znakovi od 127 do 255. Varijable ne trebaju biti eksplicitno deklarirane.

Varijablama se mogu dodijeliti vrijednosti:

- preko vrijednosti

```
$broj=25;
$godina=25;
//obje varijable posjeduju svoju kopiju vrijednosti
```

- preko reference (od PHP 4) (varijable koje referenciraju iste podatke)

```
$broj=25;
$godina= & $broj;
// obje varijable pokazuju na istu vrijednost u memoriji
$godina= 30; // obje varijable se mijenjaju
```

Varijabilne varijable (engl. Variable variables)

Varijabilna varijabla uzima vrijednost varijable kao naziv varijable.

```
$a="naziv";
$$a="vrijednost";
echo $naziv; //ispisuje vrijednost
echo "${$a}"; //ispisuje vrijednost
```

3.2. KONSTANTE

Konstanta je vrijednost koja se ne može mijenjati tijekom provođenja programa.

U kôdu se konstante definiraju preko funkcije *define*:

```
define("PI", 3.141592);
```

Konstante se pozivaju navođenjem njenog naziva:

```
$var=2*PI;
```

Pri pozivanju konstante ne koristi se $ znak ispred konstante. Tipično se zadaju velikim slovima (verzalima). Konstante su globalne i mogu se referencirati bilo gdje u skripti.

Postoji cijeli niz konstanti koje su definirane u jezgri samog jezika. Popis predefiniranih konstanti može se pronaći na http://php.net/manual/en/reserved.constants.php.

Konstante koje se deklariraju sa *define* mogle su sadržavati samo skalarne vrijednosti. Od verzije PHP-a 7 mogu se zadavati i *array* konstante.

Primjer 1.

```
define('KONSTANTE', [
  'PI' => '3.141592',
  'e' => '2.71828'
]);
echo KONSTANTE['PI'];
```

3.3. DOSEG (ENGL. *SCOPE*) VARIJABLE

Doseg varijable je određen kontekstom u kojem je deklarirana i načinom deklariranja varijable. Varijable prema dosegu možemo podijeliti na lokalne, globalne, statične i superglobal varijable.

3.3.1. Lokalne varijable

Lokalne varijable su varijable deklarirane u funkciji. Izlaskom iz funkcije lokalne varijable se uništavaju.

3.3.2. Globalne varijable

Globalne varijable su dostupne u svim dijelovima programa. One se deklariraju na osnovnoj razini programskog kôda. U PHP- u unutar funkcija nije moguće direktno pristupati globalnim varijablama već je potrebno koristiti ključnu riječ GLOBAL. Bez ključne riječi GLOBAL unutar funkcija se referencira lokalna varijabla.

Primjer 1.

```
$var=10;
function inkrementiraj() {
   GLOBAL $var;
   $var++;
}
inkrementiraj();
```

Drugi način pristupanja varijablama iz globalnog dosega je korištenjem specijalnog superglobal $GLOBALS polja. Prethodni primjer se može zapisati korištenjem $GLOBALS polja kao:

Primjer 2.

```
$var=10;
function inkrementiraj() {
   $GLOBALS ['var'];
   $var++;
}
inkrementiraj();
```

3.3.3. Statične varijable

Statična varijabla postoji samo unutar funkcija i ne gubi svoju vrijednost kada izvršavanje programa izađe iz lokalnog dosega funkcije. Vrijednost varijabla ostati će sačuvana i varijabla se neće uništavati izlaskom iz funkcije. Deklaracija varijable izvršava se samo prvim pozivanjem funkcije.

Primjer 1.

```
function inkrementiraj() {
   STATIC $broj= 0;
   $broj++;
   echo $broj;
   echo "<br />";
}
inkrementiraj();
inkrementiraj();
inkrementiraj();

Ispis iz programa:
1
2
3
```

3.3.4. Superglobal varijable

Superglobal varijable su dostupne bilo gdje u skripti. Kako bi se koristile u *php.ini* konfiguracijskoj datoteci *track_vars* mora biti omogućen. Imenovane su velikim slovima i započinju sa _.

$_SERVER

$_SERVER je asocijativno polje s unaprijed zadanim ključevima polja (eng. *key*). Ova se superglobal varijabla koristi kako bi se dohvatili podaci o okruženju u kojem se izvršavaju PHP skripte.

Ukoliko se žele ispisati svi podaci zapisani u neku superglobal varijablu može se koristiti sintaksa prikazana u sljedećem primjeru:

```
foreach ($_SERVER as $varijabla => $vrijednost) {
     echo "$varijabla => $vrijednost <br />";
}
```

Dobiveni rezultati izvršavanja navedene skripte ovisiti će o okruženju u kojem se skripta izvršava. Unosi u polje superglobal varijable postavljaju se od strane servera, neke vrijednosti mogu biti nedostupne ili mogu dati vrijednosti koje drugi serveri ne predaju.

Popis vrijednosti koje se mogu dobiti moguće je pogledati na:

http://php.net/manual/en/reserved.variables.server.php.

$_GET

$_GET superglobal varijabla daje asocijativno polje koje sadrži vrijednosti proslijeđene preko GET metode.

Primjer 1.

```
URL→ http://www.mev.hr/index.html?prva=pozdrav&druga=svima
   $_GET['prva']="pozdrav";
   $_GET['druga']="svima";
```

$_POST

$_POST superglobal varijabla sadrži vrijednosti proslijeđene preko POST metode (prilikom predaje HTML formulara). Primjer 1. prikazuje formular s postavljenom POST metodom dok Primjer 2. prikazuje ispis vrijednosti unesenih u formular, nakon predaje formulara

Primjer 1.

```
<form action="obrada_formulara.php" method="post">
   <p>Email adresa:<br />
      <input type="text" name="email" size="20"
      maxlength="50" value="" />
   </p>
   <p>Password:<br />
      <input type="password" name="lozinka" size="20"
      maxlength="15" value="" />
   </p>
   <p>
      <input type="submit" name="predaj" value="predaj!" />
   </p>
</form>
```

Primjer 2.

```
echo $_POST['email'];
echo $_POST['lozinka'];
echo $_POST['predaj'];
```

$GLOBALS

Dohvaća asocijativno polje u kojem se nalaze sve varijable deklarirane u globalnom dosegu skripte. Nazivi varijabli bez dolar znaka su ključevi u asocijativno polje.

$_COOKIE

Asocijativno polje varijabli proslijeđenih trenutačnoj skripti preko HTTP Cookies.

$_FILES

Dohvaća informacije o datotekama postavljenim na server u trenutačnoj skripti korištenjem HTTP POST metode.

$_ENV

Dohvaća informacije o serveru.

$_SESSION

Dohvaća informacije vezane uz varijable u sesiji.

$_REQUEST

Dohvaća asocijativno polje koje sadrži sve varijable definirane u $_POST, $_GET i $_COOKIE.

3.4. PHP PODRŽANI TIPOVI PODATAKA

U PHP-u su podržani sljedeći tipovi podataka:

Boolean

True/False.

Bilo koji znak koji nije nula / 0.

Integer

Maksimalni podržani integer je 2^{31} s predznakom.

Float

Realan broj čija preciznost ovisi o sustavu na kojem se skripte izvršavaju.

String

Niz znakova gdje je znak isto što i bajt (PHP nema ugrađenu podršku za Unicode) Određuju se pomoću jednostrukih ili dvostrukih navodnika.

Array (polja)

Polje u PHP u je poredana mapa (eng. *ordered map*) koja povezuje vrijednosti i ključeve. Prema vrsti ključa postoje polja koja se zadaju kao:

- Numerički indeks

```
$polje[1]="PHP";
```

- Asocijativni indeks

```
$polje["jezik"]="PHP";
```

Object

Tip podataka koji sprema podatke i informacije o njihovom procesiranju. U PHP-u objekti se eksplicitno deklariraju.

NULL

Poseban tip podataka koji može imati samo vrijednost NULL. Kad se varijabla kreira bez vrijednosti automatski joj se dodjeljuje vrijednost NULL. Varijable se također mogu isprazniti postavljanjem vrijednosti na NULL.

Resource

Specijalni tip *Resource* u biti nije stvaran tip podataka. Varijabla koja je tipa *resource* referencira funkcije i resurse izvan PHP skripte. Tipičan primjer korištenja resursa je povezivanje s bazom podataka.

3.5. KONVERZIJA TIPOVA PODATAKA

Konverzija tipova podataka omogućuje pretvorbu varijable iz jednog tipa u drugi tip varijable. Konverzija tipova podataka može biti eksplicitna ili automatska.

3.5.1. Eksplicitna konverzija tipova podataka (eng. Type Casting)

Eksplicitna konverzija eksplicitno navodi u koji tip podataka želimo izvršiti konverziju preko operatera konverzije.

Operatori

- (array)
- (bool) ili (boolean)
- (int) ili (integer)
- (int64) → od PHP 6
- (object)
- (real) ili (double) ili (float)
- (string)

Primjer 1.

```
$rezultat = (double) 13; // $rezultat = 13.0
$rezultat= (int) 14.8; // $rezultat = 14
$recenica= „Ovo je rečenica";

echo (int) $recenica; // vraća 0
$rezultat= 1114;
$polje= (array) $rezultat;
echo $polje[0]; /* izlaz 1114 (OPREZ! kad imamo već
postojeće polje ono se briše i ostaje vrijednost samo na
prvom mjestu polja) */
```

3.5.2. Automatska konverzija tipova podataka (*eng. Type Juggling*)

PHP ne zahtijeva ni ne podržava eksplicitnu definiciju tipa podataka prilikom deklariranja varijabli. Tip varijable se određuje iz konteksta u kojem se varijabla koristi. Ukoliko se varijabli dodijeli vrijednost tipa *string* ona će postati varijabla istog tipa, ukoliko joj se dodijeli *integer* vrijednost i varijabla se mijenja u taj tip podataka.

Slično se događa i prilikom obrade podataka gdje se mogu kombinirati različiti tipovi podataka koji se procjenjuju ovisno o kontekstu kako je prikazano u sljedećim primjerima.

Primjer 1.

```
<?php
  $cijena= 10; // integer
  $pdv= "2.50"; // string
  $cijena += $pdv; // $cijena = 12.5 float
?>
```

Primjer 2.

```
<?php
  $cijena= 10; // integer
  $pdv= "2.50 kuna"; // string
  $cijena += $pdv;
  /* $cijena = 12.5 float →vrijedi samo kad je početak
  stringa broj! */
?>
```

Primjer 3.

```
<?php
  $cijena= "1.21e3";
  // string →1210 kad ima e u sebi tumači se kao float
  $pdv= 0.25; // float
  $cijena += ($cijena * $pdv); // $cijena = 1512.5
?>
```

3.6. FUNKCIJE VEZANE UZ TIPOVE PODATAKA

3.6.1. Dobavljanje tipa podataka

string gettype(mixed var)

Funkcija može vratiti: "boolean", "integer", "double" (umjesto "float " vraća se "double"), "string", "array", "object", "resource", "resource (closed)" od verzije PHP 7.2.0, "NULL" i "unknown type".

Primjer:

```
<?php
   $cijena= "1.21e3"; // string
   $pdv= 0.23; // float
   $cijena += ($cijena * $pdv); // $cijena = 1488.3
   echo gettype($cijena); //ispisuje double
?>
```

3.6.2. Mijenjanje tipa podataka

boolean settype(mixed var, string type)

Moguće vrijednosti za type su: "boolean" ili "bool", ,"integer" ili "int", "float" ili "double", "string", "array", "object" i "null". Funkcija vraća TRUE ukoliko se ispravno izvrši ili FALSE.

Primjer:

```
<?php
   $cijena= "1.21e3"; // string
   $pdv= 0.23; // float
   $cijena += ($cijena * $pdv); // $cijena = 1488.3
   settype($cijena, 'integer');
   echo $cijena; //ispisuje 1488
?>
```

3.6.3. Identificiranje tipa podataka

Prototip funkcije je boolean is_name (mixed var).

Funkcije vraćaju TRUE ili FALSE ovisno o tome da li je uvjet zadovoljen. Slijede neke od is_ funkcija:

```
is_array();
is_bool();
is_float();
is_int ();
is_null();
is_numeric();
```

```
is_object();
is_resource();
is_scalar();
is_string();
```

Primjer 1.

```
$cijena = 10;
printf("Varijabla \$cijena je tipa array: %d <br />",
is_array($cijena));
printf("Varijabla \$cijena je tipa integer: %d <br />",
is_integer($cijena));
printf("Varijabla \$cijena je tipa numeric: %d <br />",
is_numeric($cijena));

Ispisuje:
Varijabla $cijena je tipa array: 0
Varijabla $cijena je tipa integer: 1
Varijabla $cijena je tipa numeric: 1
```

4. OPERATORI

Operator uzima jednu ili više vrijednosti i vraća drugu vrijednost. Pomoću operatera izrađuju se izrazi. Operatori se mogu grupirati ovisno o broju vrijednosti koje uzimaju. Unarni operatori uzimaju samo jednu vrijednost, binarni dvije vrijednosti, te ternarni operator *? :*, koji uzima tri vrijednosti.

4.1. VRSTE OPERATORA

4.1.1. Operatori dodjeljivanja

Operator dodijele vrijednosti je znak jednakosti (=) koji dodjeljuje vrijednost izraza s desne strane operandu s lijeve strane. Postoje i kombinirani operatori koji uključuju osnovne aritmetičke i operacije na razini bitova (primjerice +=).

4.1.2. Aritmetički operatori

Omogućuju standardne aritmetičke operacije.

Primjer	Naziv	Rezultat
+$a	Identitet	Konverzija *$a* u int ili float
-$a	Negacija	Suprotno od *$a*.
$a + $b	Zbrajanje	Suma *$a* i *$b*.
$a - $b	Oduzimanje	Razlika *$a* i *$b*.
$a * $b	Množenje	Umnožak *$a* i *$b*.
$a / $b	Dijeljenje	Kvocijent *$a* i *$b*.
$a % $b	Modulo	Modulo- ostatak nakon dijeljenja *$a* i *$b*.
$a ** $b	Eksponent	Eksponent izračunan kao *$a* na *$b*. Uveden je od verzije PHP 5.6.

4.1.3. Operatori na razini bitova (*engl. Bitwise operators*)

Omogućuju operacije na razini bitova za vrijednosti spremljene kao integer.

Primjer	Naziv	Rezultat
$a & $b	And	Postavljaju se bitovi koji su postavljeni u obje varijable *$a* i *$b*
$a | $b	Or (inclusive or)	Postavljaju se bitovi koji su postavljeni u *$a* ili u *$b*
$a ^ $b	Xor (exclusive or)	Postavljaju se samo bitovi koji su postavljeni u *$a* ili *$b* ali ne u obje varijable
~ $a	Not	Bitovi koji su postavljeni u *$a* se ne postavljaju i obrnuto.
$a << $b	Shift left	Pomiče bitove *$a $b* koraka u lijevo
$a >> $b	Shift right	Pomiče bitove *$a $b* koraka u desno

4.1.4. Operatori usporedbe

Primjer	Naziv	Rezultat
$a == $b	Jednako	**TRUE** ako je *$a* jednak *$b* nakon automatske konverzije
$a === $b	Identično	**TRUE** ako je *$a* jednak *$b* i istog su tipa podataka
$a != $b	Različito	**TRUE** ako je *$a* različit od *$b* nakon automatske konverzije
$a <> $b	Različito	**TRUE** ako je *$a* različit od *$b* nakon automatske konverzije
$a !== $b	Nije identično	**TRUE** ako je *$a* različit od *$b* ili su različitog tipa podataka
$a < $b	Manje od	**TRUE** ako je *$a* manji od *$b*.
$a > $b	Veće od	**TRUE** ako je *$a* veći od *$b*.
$a <= $b	Manje od ili jednako	**TRUE** ako je *$a* manji ili jednak *$b*.
$a >= $b	Veće od ili jednako	**TRUE** ako je *$a* veći ili jednak *$b*.
$a <=> $b	*Eng. Spaceship*	Vraća se 0 ukoliko su jednaki, 1 ako je *$a* veći od *$b* i -1 ako je *$a* manji od *$b*. Dostupno je od verzije PHP 7.
$a ?? $b	*Eng. Null coalescing operator*	Ukoliko je *$a* NULL vraća se *$b* a u ostalim slučajevima *$a*. Provjera da li je varijabla postavljena. Dostupno je od verzije PHP 7

4.1.5. Operator kontrole pogrešaka

PHP podržava jedan operator kontrole pogrešaka i to je znak @. Kada se postavi ispred izraza u PHP-u bilo koja poruka o pogrešci će se ignorirati.

4.1.6. Operatori inkrementacije i dekrementacije

Primjer	Naziv	Rezultat
++$a	Pre-increment	Povećava se *$a* za 1 i vraća se vrijednost od *$a*.
$a++	Post-increment	Vraća se *$a* i nakon toga povećava za 1
--$a	Pre-decrement	Umanjuje se *$a* za 1 i vraća se vrijednost od *$a*.
$a--	Post-decrement	Vraća se *$a* i nakon toga umanji za 1

4.1.7. Logički operatori

Primjer	Naziv	Rezultat
$a and $b	And	**TRUE** ako su *$a* i *$b* **TRUE**.
$a or $b	Or	**TRUE** ako su *$a* ili *$b* **TRUE**.
$a xor $b	Xor	**TRUE** ako su *$a* ili *$b* **TRUE**, ali ne oboje
! $a	Not	**TRUE** ako *$a* nije **TRUE**.
$a && $b	And	**TRUE** ako su *$a* i *$b* **TRUE**.
$a || $b	Or	**TRUE** ako su *$a* ili *$b* **TRUE**.

Razlog postojanja dva and i or je zbog različitog prioriteta u rješavanju izraza.

4.1.8. String operateri

Postoje 2 string operatera: točka ('.') koja radi spajanje argumenata i ('.=') koja dodjeljuje vrijednost argumenta desne strane u argument na lijevoj strani.

4.1.9. Array operateri

Primjer	Naziv	Rezultat
$a + $b	Union	Unija *$a* i *$b*. Pri uniji polju $a se dodaju vrijednosti polja $b. Ukoliko postoje isti ključevi u oba polja uzet će se elementi iz polja $a
$a == $b	Equality	**TRUE** ako polje *$a* i *$b* imaju iste parove key/value
$a === $b	Identity	**TRUE** ako polje *$a* i *$b* imaju iste parove key/value u istom redoslijedu i istog tipa podataka
$a != $b	Inequality	**TRUE** ako je *$a* različito od *$b*.
$a <> $b	Inequality	**TRUE** ako je *$a* različito od *$b*.
$a !== $b	Non-identity	**TRUE** ako *$a* nije identično *$b*.

4.1.10. Operateri tipa podataka

Koristi se *instanceof* kako bi se odredilo da li je PHP varijabla instancirana iz objekta određene klase.

4.2. PREGLED PRIORITETA PRI IZVRŠAVANJU OPERATERA

U tablici su prikazani operatori poredani prema prednosti pri izvršavanju. Ukoliko dva operatera imaju isti prioritet u izvršavanju njihova asocijativnost određuje kako se operatori grupiraju.

Primjer 1:

```
'-' lijeva asocijativnost
1 - 2 - 3 se grupira kao (1 - 2) – 3
Rezultat: -4
```

Primjer 2:

```
'=' desna asocijativnost
$a = $b = $c se grupira kao $a = ($b = $c)
```

Primjer 3:

```
'<' i '>' su istog prioriteta i nisu asocijativni pa se ne
mogu koristiti jedan uz drugog
1 < 2 > 1 je ilegalno
1 <= 1 == 1 je legalno jer == operator ima manji prioritet
```

Korištenje zagrada često može povećati čitljivost izraza iako ih nije potrebno koristiti. Eksplicitno grupiranje je bolje od oslanjanja na prioritet i asocijativnost.

Asocijativnost	Operatori	Svrha
nema	*clone new*	Instanciranje i kloniranje objekta
lijevo	*[*	Polja zatvaranje indeksa
desno	**	Eksponent
desno	++ -- ~ *(int) (float) (string) (array) (object) (bool) @*	Inkrementiranje, dekrementiranje, bitwise NOT, tipovi podataka, supresija pogrešaka
nema	*instanceof*	Provjera da li je objekt instanca neke klase
desno	*!*	Logički NOT
lijevo	** / %*	Množenje, dijeljenje i modulo
lijevo	+ - .	Zbrajanje, oduzimanje, dodavanje (*eng. concatenation*)
lijevo	<< >>	Shift left, shift right (bitwise)
nema	< <= > >=	Manje od, manje ili jednako, veće od, veće ili jednako
nema	== *!*= === *!*== <> <=>	Jednako, različito, identično, nije identično, različito, usporedba
lijevo	&	Bitwise AND i reference
lijevo	^	bitwise XOR

Asocijativnost	Operatori	Svrha
lijevo	|	bitwise OR
lijevo	*&&*	logički AND
lijevo	||	Logički OR
desno	*??*	Usporedba
lijevo	*? :*	Ternarni operator
desno	= += -= *= **= /= .= %= &= |= ^= <<= >>=	Operatori dodjeljivanja
lijevo	*and*	logički AND
lijevo	*xor*	logički XOR
lijevo	*or*	logički OR

4.3. ESCAPE SEKVENCE

U PHP-u kosa crta unatrag ('\') ima nekoliko primjena. Ukoliko ju prati znak koji nije alfanumerički oduzima se bilo kakvo značenje koje bi taj znak mogao imati.

Primjer 1:

* → označava se * i oduzima bilo koje drugo značenje koje bi znak mogao imati

\\ → ukoliko se želi odabrati kosa crta

\$ →Znak dolara

\" →Dupli navodnici

Druga upotreba je zapisivanje znakova koji se ne ispisuju u vidljivom smislu. Vide se u izvornom kôdu, datotekama, ali ne i u prozoru preglednika.

Primjer 2:

\n →Line Feed, New line, LF (Linux, Unix)

\r →Carriage return, CR (Mac, Win (obje \r\n))

(PHP_EOL (od PHP 5.0.2) se može koristiti kao konstanta koja postavlja ispravan prelazak u sljedeći redak ovisno o sustavu).

\t →Tab

Escape sekvence se često koriste u regularnim izrazima.

5. KONTROLNE STRUKTURE

PHP podržava većinu standardnih uvjetnih izjava i petlji kao i ostali programski jezici koji su bazirani na C programskom jeziku. Slijedi pregled istih.

if, else if, else

```
if (izraz) {
     //kôd koji će se izvršiti ukoliko je uvjet zadovoljen
} else if (izraz) {
     //kôd koji će se izvršiti ukoliko je uvjet zadovoljen
} else {
     //kôd koji će se izvršiti ukoliko je uvjet zadovoljen
}
```

Alternativna sintaksa postoji za pojedine kontrolne strukture kao što if, while, for, foreach, and switch. Kod alternativne sintakse mijenja se vitičasta zagrada otvaranja u dvotočku a zagrada zatvaranja u endif;, endwhile;, endfor;, endforeach;, ili endswitch;.

Primjer 1.

```
<?php if ($a == 5): ?>
  A je jednak 5
<?php endif; ?>
```

Primjer 2.

```
if ($a == 5):
    echo "a je jednak 5";
    echo "...";
elseif ($a == 6):
    echo "a je jednak 6";
    echo "!!!";
else:
    echo "a nije ni 5 ni 6";
endif;
```

switch

Primjer 1.

```
switch ($kategorija) {
case "vijesti":
  echo "<p>Događanja u svijetu</p>";
  break;
case "vrijeme":
  echo "<p>Tjedna prognoza</p>";
  break;
case "sport":
  echo "<p>Vijesti iz sporta</p>";
  break;
default:
  echo "<p>Dobrodošli na moju stranicu</p>";
}
```

Primjer 2. (alternativna sintaksa)

```
switch ($i):
     case 0:
        echo "i je jednak 0";
        break;
     case 1:
        echo "i je jednak  1";
        break;
     case 2:
        echo "i je jednak  2";
        break;
     default:
        echo "i nije jednak 0, 1 i 2";
endswitch;
```

Od verzije PHP 7. nije više moguće koristiti više *default* izjava.

while

```
while (izraz) {
     //kôd koji će se izvršiti tako dugo dok je uvjet
     //zadovoljen
}
```

Primjer 1.

```
<?php
   $broj= 1;
   while ($broj< 5) {
      printf("%d na kvadrat= %d <br />", $broj, pow($broj, 2));
   $broj++;
   }
?>

Ispis:
1 na kvadrat= 1
2 na kvadrat= 4
3 na kvadrat= 9
4 na kvadrat= 16
```

Primjer 2. (alternativna sintaksa)

```
$i = 1;
while ($i <= 10):
    echo $i;
    $i++;
endwhile;
```

do while

```
do {
     //kôd koji će se izvršiti tako dugo dok je uvjet
     //zadovoljen
} while (izraz);
```

Primjer 1.

```
<?php
  $broj= 1;
  do {
     printf("%d na kvadrat= %d <br />", $broj, pow($broj, 2));
     $broj++;
  } while ($broj< 5);
?>

Ispis:
1 na kvadrat= 1
2 na kvadrat= 4
3 na kvadrat= 9
4 na kvadrat= 16
```

Za *do while* ne postoji alternativna sintaksa.

for

```
for ($kilometri= 1; $kilometri<= 5; $kilometri++) {
     printf("%d km= %f milja<br />", $kilometri,
       $kilometri*0.62140);
}

Ispis:
1 km = 0.6214 milja
2 km = 1.2428 milja
3 km = 1.8642 milja
4 km = 2.4856 milja
5 km = 3.107 milja
```

Primjer 2. (alternativna sintaksa)

```
for ($i=0; $i<10; $i++):
    echo $i;
endfor;
```

foreach

Koristi se za rad s elementima u polju.

Primjer 1.

```
<?php
  $linkovi=
     array("www.apress.com","www.php.net","www.apache.org");
  echo "<b>Online Resursi</b>:<br />";
foreach($linkovi as $link) {
          echo "<a href=\"http://$link\">$link</a><br />";
}
?>

Ispis:
Online Resursi:
www.apress.com
www.php.net
www.apache.org
```

Primjer 2. (asocijativna polja key=>value parovi)

```
<?php
//definicija asocijativnog polja
$linkovi= array(
     "The Apache Web Server" => "www.apache.org",
     "Apress" => "www.apress.com",
     "The PHP Scripting Language" => "www.php.net");
//petlja za ispis
foreach($linkovi as $naslov=> $link) {
     echo "<a href=\"http://$link\">$naslov</a><br />";
}
?>

Ispis:
The Apache Web Server
Apress
The PHP Scripting Language
```

Primjer 3. (alternativna sintaksa)

```
$polje=array(1,2,3,4);
foreach($polje as $a):
  echo $a;
endforeach;
```

break

Break završava izvođenje trenutačne *for, foreach, while, do-while* ili *switch* strukture. Može primiti opcionalan brojčani parametar koji označava iz koliko ugniježđenih struktura je potrebno izaći. Osnovna vrijednost je 1.

Primjer 1.

```
$i=0;
while (++$i) {
   switch ($i) {
      case 5:
           echo "Broj 5<br />\n";
           break 1;  // Izlazi se samo iz switch.
      case 10:
           echo "Broj 10;<br />\n";
           break 2;  // Izlazi se iz switch i while
      default:
           break;
      }
}
```

continue

Continue se koristi u petljama kako bi se preskočilo izvršavanje kôda u trenutačnoj petlji i nastavilo od sljedeće iteracije petlje. Moguće je dodjeljivati brojčane parametre.

Primjer 1.

```
$i = 0;
while ($i++ < 5) {
     echo "Vanjska petlja<br />\n";
     while (1) {
          echo "Srednja petlja<br />\n";
          while (1) {
               echo "Unutarnja petlja<br />\n";
               continue 3;
          }
          echo "Ovo se nikad ne ispiše.<br />\n";
     }
     echo " Ovo se nikad ne ispiše.<br />\n";
}
```

goto

Primjer 1.

```
for ($broj= 0; $broj< 10; $broj++) {
     $slucajniBroj= rand(1,50);
     if ($slucajniBroj < 10) goto manji;
     else echo "Broj veći od 10: $slucajniBroj<br />";
}
manji:
echo "Broj manji od 10: $slucajniBroj<br />";
```

include, require

Include i *require* služe za uvoženje eksternog kôda. Razlika između njih je da kod *require* ukoliko se uvoženje datoteke ne dogodi, skripta se dalje neće izvoditi, dok *include* ne završava izvođenje skripte.

Način definiranja (vrijedi i za *require*):

include('/path/to/filename')

include "/path/to/filename"

include_once i *require_once* najprije provjeravaju da li je datoteka već uključena kako ne bi došlo do konflikata s imenima varijabli i funkcija.

6. FUNKCIJE

U PHP-u mogu se koristiti funkcije koje su ugrađene u jezik, funkcije koje se uvoze iz biblioteka ili eksternih datoteka pomoću include() i require(). Moguće je kreirati i vlastite funkcije.

Primjer 1.

```
<?php
   $vrijednost= pow(5,3); //vraća 125
   echo $vrijednost;
?>
```

Funkcije i metode koje postoje u jeziku moguće je pogledati na:

http://php.net/manual/en/indexes.functions.php.

Za svaku funkciju ili metodu koja je ugrađena u jezik moguće je pregledati dokumentaciju tako da se navede u pregledniku u polje za url:

```
http://php.net/nazivfunkcije
http://php.net/objekt.metoda

Primjerice:
http://php.net/pow
http://php.net/directory.close
```

Izrada vlastitih funkcija

```
function nazivFunkcije(argumenti)
{
     //tijelo funkcije
}
```

Unutar funkcije može se postavljati bilo koji valjani PHP kôd, druge funkcije i definicije klasa. Nazivi funkcija započinju sa slovom ili _, nakon čega može slijediti kombinacija slova, brojeva i _. Nazivi funkcija osjetljivi su na mala i velika slova.

Funkcije se ne trebaju definirati prije nego što se pozivaju osim u slučajevima kada se definiraju u uvjetnim izjavama ili drugim funkcijama.

Primjer 1.

```
prva(); //ovu funkciju je moguće pozivati
if (true) {
     function druga()
     {
     echo "Funkcija se može pozivati tek nakon izvršenja if
     izjave\n";
     }
}
```

```
druga(); //može se pozivati tek nakon izvršenja if
function prva()
{
     echo "Funkcija koja se može pozivati prije
     deklaracije";
}
```

Primjer 2.

```
function prva()
{
  function druga()
  {
    echo "Funkcija ne postoji dok se ne pozove prva()";
  }
}
prva();
//Funkcija druga() može se pozivati tek nakon pozivanja
//funkcije prva()
druga ();
```

Sve funkcije i klase u PHP-u imaju globalni doseg. Mogu se pozivati izvan funkcije iako su deklarirane u funkciji. Nije moguće redefinirati ili uništiti prethodno deklariranu funkciju.

PHP podržava rekurzivno pozivanje funkcija:

Primjer 3.

```
$a=0;
function rekurzija ($a)
{
    if ($a < 20) {
        echo "$a";
        rekurzija(++$a);
    }
}
```

6.1. PROSLJEĐIVANJE VRIJEDNOSTI FUNKCIJAMA

Vrijednosti se funkciji mogu proslijediti kao statične vrijednosti, kao vrijednosti iz varijabli ili po referenci.

Primjer 1.

```
function cijenasPDV($cijena, $pdv)
{
  $ukupno= $cijena+ ($cijena* $pdv);
  echo "Ukupna cijena: $ukupno";
}

//upotreba statičnih vrijednosti
cijenasPDV(15.00, .25);

//prosljeđivanje po vrijednosti
$cijena1=15.00;
$porez=.25;
cijenasPDV($cijena1, $porez);

//prosljeđivanje po referenci
$cijena1=15.00;
$porez=.25;
cijenasPDV(&$cijena1, $porez);
```

6.2. ZADANE I OPCIONALNE VRIJEDNOSTI PARAMETARA

Zadane vrijednosti (*engl. default*) moraju biti na kraju liste parametara. Moraju biti konstantne vrijednosti i na njihovo mjesto se ne može staviti varijabla ili funkcija (kod definiranja).

Primjer 1.

```
function cijenasPDV($cijena, $pdv=.25)
{
  $ukupno= $cijena+ ($cijena* $pdv);
  echo "Ukupna cijena: $ukupno";
}
cijenasPDV(15.00);
```

Umjesto postavljanja zadane vrijednosti parametri na kraju liste parametara mogu se postaviti kao opcionalni ukoliko im se postavi vrijednost na prazan string "".

Primjer 2.

```
function izracunaj($cijena1, $cijena2="", $cijena3="")
{ echo $cijena1 + $cijena2 + $cijena3;
}
izracunaj (10, "", 3);
```

U primjeru je prikazano i kako je moguće preskočiti parametre prilikom prosljeđivanja argumenata.

Od PHP 5.6 i kasnijim verzijama parametri u listi mogu imati '…' token koji opisuje da funkcija prima varijabilan broj argumenata. Oni će se proslijediti funkciji kao polje.

Primjer 3.

```
function suma(...$brojevi) {
    $zbroj = 0;
    foreach ($brojevi as $n) {
        $zbroj += $n;
    }
    return $zbroj;
}

echo suma(1, 2, 3, 4);
```

6.3. ZADAVANJE TIPOVA PODATAKA (ENG. *TYPE HINTS*)

Od PHP 7 dodana je mogućnost zadavanja skalarnih tipova podataka pri deklaraciji parametara funkcije. Zadavati je moguće skalarne tipove podataka (*bool, int, float, string*), *iterable* i *object*. Od verzije 5. bilo je omogućeno zadavanje *class/interface* (parametar mora biti *instanceof* date klase ili naziva interface-a) i *array*. Korištenje aliasa za osnovne tipove podataka u biti očekuje objekte instancirane iz klase zadanog interface-a (npr. ako je zadan tip *boolean* očekuje se instanca objekta *boolean* umjesto podatka tipa *bool*).

Primjer 1.

```
function typeHints(bool $a, float $b, int $c, string $c)
{
}
typeHints (true, 4.35, 123, "foo");
```

Ukoliko se funkciji proslijede vrijednosti koje nisu u zadanom tipu podataka PHP će pokušati promijeniti tip podatka ukoliko je moguće.

Primjer 2.

```
function sendHttpStatus(int $statusCode, string $message)
{
     header('HTTP/1.0 ' .$statusCode. ' ' .$message);
}
sendHttpStatus(404, "File Not Found");
sendHttpStatus("403", "OK");
```

Oba pozivanja funkcije će ispravno izvršiti kôd jer je konverzija iz jednog tipa u drugi bila moguća.

U slučaju da se proslijedi podatak kojeg nije moguće konvertirati izbacuju se iznimke krivog tipa podataka (*TypeError*).

Osim osnovnog ponašanja podržano je i strogo definiranje tipova podataka. Strogo definiranje zadaje da će bilo koja pogreška u tipu podatka rezultirati iznimkom *TypeError*. Zadavanje strogog načina rada moguće je samo u osnovnoj skripti. To znači da se u bibliotekama ne može zadavati da su one strogo definirane. Zadavanje se vrši na sljedeći način:

```
declare(strict_types=1);
```

Zadavanje korištenja strogih tipova podataka mora biti prva izjava u skripti. Ništa ne smije biti zapisano između početne PHP oznake i *declare* funkcije. Strogi tipovi podataka odnose se samo na skalare.

6.4. VRAĆANJE VRIJEDNOSTI IZ FUNKCIJE

Za vraćanje vrijednosti iz funkcije koristi se return();

Primjer 1.

```
function cijenasPDV($cijena, $pdv)
{
  $ukupno= $cijena+ ($cijena* $pdv);
  return $ukupno;
}
```

Funkcija može vratiti i više vrijednosti tako da vrati *array* ili objekt.

Primjer 2.

```
function vratiKorisnik() {
     $korisnik[]="Ivica";
     $korisnik[]="Ivić";
     $korisnik[]="iivic@mev.hr";
     return $korisnik;
}
list ($ime, $prezime, $email)= vratiKorisnik();
echo "Ime korisnika: $ime, prezime: $prezime, e-mail:
$email" ;
```

Od PHP 7.0 moguće je ograničiti tip podataka koji se vraća iz funkcije za korisnički definirane funkcije. Sintaksa je:

```
function nazivFunkcije ($arg1, $arg2, ...)  : TYPE { ... }
```

TYPE je reprezentacija tipa vrijednosti koja se vraća iz funkcije. To može biti naziv klase ili tip varijable kao *array*, *string*, *bool*, *int* i *float*. Kada je TYPE *str*, *boolean*, *integer*, *real*, *double*, *resource*, *object* i *scalar* u biti je i naziv klase, pa se vraća objekt i za skalare.

Primjer 3.

```
function vracanjeTipa ($var): string {
     return $var;
 }
```

Funkcija navedena u primjeru mora vratiti string i ukoliko ne vrati navedeni tip podataka doći će do ispisa pogreške "*Fatal error: Uncaught TypeError*".

7. POLJA

U PHP-u je moguće deklarirati polja koja spremaju višestruke vrijednosti u jednu varijablu. PHP ne zahtjeva eksplicitnu deklaraciju polja, tipa podatka koji sadrži ili određivanje veličine polja.

Svaki element u polju ima ključ (eng. *key*) i vrijednost (eng. *value*).

Polja se u PHP-u mogu izraditi korištenjem sintakse s ključnom riječi *array*:

```
array(
    key  => value,
    key2 => value2,
    key3 => value3,
    ...
)
```

ili od verzije PHP 5.4 korištenjem skraćene sintakse:

```
[
   key  => value,
   key2 => value2,
   key3 => value3,
   ...
]
```

PHP polja mogu imati numerički (relalan broj) ili asocijativni ključ (string), te je moguće izrađivati polja koja koriste kombinaciju stringova i brojeva. Prilikom dodavanja elemenata u polje ukoliko nije drugačije zadano, uvijek se dodaje inkrementiranjem najvećeg integera koji je korišten u polju, a ukoliko nije bilo korištenih integera *key* se postavlja na nulu.

Primjer 1. numerički *key*

```
$sjedisteZupanije= array("Zagreb", "Čakovec", "Zagreb")
$sjedisteZupanije= array(
     0=> "Zagreb", 19=> "Čakovec", 20=> "Zagreb"
     );
//referenciranje:
$sjedisteZupanije[0];
```

Primjer 2. asocijativan *key*

```
$sjedisteZupanije= array(
   "Zagrebačka"=> "Zagreb",
   "Međimurska"=> "Čakovec",
   "Grad Zagreb"=> "Zagreb"
);
//referenciranje:
$sjedisteZupanije["Međimurska"];
```

Prilikom referenciranja i deklariranja asocijativnog polja ključeve je potrebno postaviti unutar navodnika osim ukoliko se radi s konstantama ili varijablama. Ključevi su izraz kojeg je potrebno riješiti prije referenciranja ili deklariranja.

Moguće je izrađivati i multidimenzionalna polja.

Primjer 3.

```
$sjedisteZupanije= array(
    "Zagrebačka"=> array(
         "grad"=>"Zagreb",
         "stanovnika"=> "309.696"),
    "Međimurska"=> array(
         "grad"=>" Čakovec ",
         "stanovnika"=> " 118.426")
     );

//referenciranje:
$sjedisteZupanije["Međimurska"] ["grad"];
```

Dodavanje u polje moguće je korištenjem ključa upisanog u uglate (ili vitičaste) zagrade, ili korištenjem samo uglatih zagrada bez navođenja ključa.

Korištenjem sljedeće sintakse elementi se slijedno dodaju u polje:

```
$sjedisteZupanije[]= "Zagreb";
$sjedisteZupanije[]= "Krapina";
….
$sjedisteZupanije[]= "Čakovec";
```

Sljedeća sintaksa briše sadržaj polja:

```
$sjedisteZupanije=array("Zagreb", "Krapina");
$sjedisteZupanije= "Čakovec";
```

za razliku od ove, koja dodaje na kraj polja:

```
$sjedisteZupanije=array("Zagreb", "Krapina");
$sjedisteZupanije[]= "Čakovec";
```

U PHP-u postoje funkcije vezane uz rad s poljima koje se mogu pregledati na http://php.net/manual/en/ref.array.php.

7.1. FUNKCIJE ZA RAD S POLJIMA

bool is_array (mixed $var)

Testiranje da li je varijabla polje.

mixed print_r (mixed $expression [, bool $return = FALSE])

Ispisivanje polja za testiranje (eng. *print human-redable*).

Opcionalni parametar *return* vraća podatke pozivatelju, ne postavlja ih na standardan izlaz.

array range (mixed $start , mixed $end [, number $step = 1])

Kreira polje koje sadrži zadani raspon elemenata. *Start* i *end* određuju odakle započinje i završava raspon dok opcionalni parametar *step* određuje razliku između slijednih podataka u rasponu.

Primjer 1.

```
$kocka= range(0, 6);
//isto kao i:
$kocka= array(0,1,2,3,4,5,6);
//step
$parni= range(2, 20, 2);
$parni= array(2,4,6,8,10,12,14,16,18,20);
//slova
$slova= range ("A", "F");
$slova= array("A", "B","C","D","E","F");
```

mixed array_rand (array $array [, int $num = 1])

Slučajan odabir *num* elemenata (*num* je opcionalan parametar).

bool shuffle (array &$array)

Slučajno preslagivanje polja.

number array_sum (array $array)

Zbraja sve članove polja. Ukoliko je moguća pretvorba članova koji nisu brojevi ona će se obaviti prije zbrajanja a ukoliko pretvorba u brojčane tipove podataka nije moguća njihove vrijednosti se zanemaruju.

7.2. FUNKCIJE ZA DODAVANJE I BRISANJE ELEMENATA IZ POLJA

int array_unshift (array &$array [, mixed $...])

Dodavanje vrijednosti na početak polja.

Primjer 1.

```
$zupanije= array("Sisačko-moslavačka", "Karlovačka");
array_unshift(
     $zupanije, "Zagrebačka", "Krapinsko-zagorska"
);
/* Kreirano polje je:
$zupanije= array(
   "Zagrebačka", "Krapinsko-zagorska", "Sisačko-moslavačka",
   "Karlovačka"
   );*/
```

int array_push (array &$array [, mixed $...])

Dodavanje vrijednosti na kraj polja.

Primjer 1.

```
$zupanije= array("Sisačko-moslavačka", "Karlovačka");
array_push($zupanije, "Zagrebačka", "Krapinsko-zagorska");
/* Kreirano polje je:
$zupanije= array("Sisačko-moslavačka", "Karlovačka",
"Zagrebačka", "Krapinsko-zagorska"
);*/
```

mixed array_shift (array &$array)

Uklanjanje vrijednosti sa početka polja (FIFO). Kod numeričkih indeksa, svi indeksi se mijenjanju tako da počinju od nule, a kod asocijativnih indeksa, *key* prvog elementa polja se briše.

Primjer 1.

```
$zupanije= array("Zagrebačka", "Krapinsko-zagorska",
"Sisačko-moslavačka", "Karlovačka");
$zupanija= array_shift($zupanije);

/*$zupanije= array("Krapinsko-zagorska",
"Sisačko-moslavačka", "Karlovačka");
$zupanija="Zagrebačka";*/
```

mixed array_pop (array &$array)

Uklanjanje vrijednosti sa kraja polja (LIFO).

Primjer 1.

```
$zupanije= array("Zagrebačka", "Krapinsko-zagorska", "Sisačko-
moslavačka", "Karlovačka");
$zupanija= array_pop($zupanije);
/*$zupanije= array("Zagrebačka", "Krapinsko-zagorska",
"Sisačko-moslavačka");
$zupanija="Karlovačka";*/
```

7.3. FUNKCIJE ZA TRAŽENJE U POLJU

bool in_array (mixed $needle , array $haystack [, bool $strict = FALSE])

Funkcija služi za traženje vrijednosti u polju (*value*).

Primjer 1.

```
$zupanije= array(
   "Zagrebačka", "Krapinsko-zagorska",
   "Sisačko-moslavačka", "Karlovačka");
$zupanija= "Zagrebačka";
if (in_array($zupanija, $zupanije)) echo "$zupanija se
nalazi u polju";
```

bool array_key_exists (mixed $key , array $array)

Traženje asocijativnih indeksa (*key*).

Primjer 1.

```
$zupanije= array(
   "Zagrebačka"=>"Zagreb",
   "Krapinsko-zagorska"=>"Krapina",
   "Sisačko-moslavačka"=>"Sisak", "Karlovačka"=>"Karlovac");
if (array_key_exists("Zagrebačka", $zupanije)) echo "Glavni
grad Zagrebačke županije je $zupanije['Zagrebačka']";
```

mixed array_search (mixed $needle , array $haystack [, bool $strict = FALSE])

Traženje vrijednosti u asocijativnom polju (*value*) i vraćanje asocijativnog indeksa (*key*). Funkcija vraća *key* ili FALSE ukoliko ne nađe vrijednost.

Primjer 1.

```
$zupanije= array(
   "Zagrebačka"=>"Zagreb",
   "Krapinsko-zagorska"=>"Krapina",
   "Sisačko-moslavačka"=>"Sisak", "Karlovačka"=>"Karlovac");
$zupanija=array_search("Karlovac", $zupanije);
if ($zupanija) echo "Glavni grad $zupanija županije je
$zupanije[$zupanija]";
```

array array_keys (array $array [, mixed $search_value [, bool $strict = FALSE]])

Vraćanje traženih ili svih indeksa u polju

Primjer 1.

```
$zupanije= array(
   "Zagrebačka"=>"Zagreb",
   "Krapinsko-zagorska"=>"Krapina",
   "Sisačko-moslavačka"=>"Sisak", "Karlovačka"=>"Karlovac");
$zupanijePopis=array_keys($zupanije);
```

```
print_r($zupanijePopis);
/*Array ( [0] => Zagrebačka [1] => Krapinsko-zagorska [2]
=> Sisačko-moslavačka [3] => Karlovačka )*/
```

array array_values (array $array)

Vraćanje svih vrijednosti u polju u obliku indeksiranog polja.

Primjer 1.

```
$zupanije= array(
   "Zagrebačka"=>"Zagreb",
   "Krapinsko-zagorska"=>"Krapina",
   "Sisačko-moslavačka"=>"Sisak", "Karlovačka"=>"Karlovac");
$zupanijeGradovi=array_values($zupanije);
print_r($zupanijeGradovi);
/*Array ( [0] => Zagreb [1] => Krapina [2] => Sisak [3] =>
Karlovac )*/
```

array array_unique (array $array [, int $sort_flags = SORT_STRING])

Vraća polje koje nema vrijednosti koje se ponavljaju.

7.4. POKAZIVAČI

U PHP-u polja koriste interne pokazivače koji pokazuju na specifični indeks u polju. Pokazivačem se može upravljati pomoću sljedećih funkcija:

mixed key (array $array)

Vraća *key* na kojem se trenutno nalazi pokazivač (ne pomiče pokazivač).

mixed current (array $array)

Vraća vrijednost na kojem se trenutno nalazi pokazivač (ne pomiče pokazivač).

array each (array &$array)

Vraća *key* i *value* na kojem je pokazivač i pomiče pokazivač (vraća polje).

mixed next (array &$array)

Pomicanje pokazivača u desno.

mixed prev (array &$array)

Pomicanje pokazivača u lijevo.

mixed reset (array &$array)

Vračanje na početak

mixed end (array &$array)

Pomicanje na kraj.

7.5. VELIČINA POLJA

int count (mixed $array_or_countable [, int $mode = COUNT_NORMAL])

Broji sve članove u polju. Ukoliko se opcionalan parametar stavi na COUNT_RECURSIVE moguće je rekurzivno brojati članove u polju što je korisno kod brojanja elemenata u multidimenzionalnom polju. Umjesto *count* moguće je koristiti njezin alias *sizeOf().*

array array_count_values (array $array)

U vraćenom polju *key* su vrijednosti a *value* je broj ponavljanja pojedinih vrijednosti.

Primjer 1.

```
$gradovi= array (
     "Zagreb", "Osijek", "Sisak", "Zagreb", "Sisak");
$ponavljanje= array_count_values ($gradovi);
print_r($ponavljanje");
//Array ( [Zagreb] => 2 [Osijek] => 1 [Sisak] => 2 )
```

7.6. MIJENJANJE I SORTIRANJE POLJA

array array_reverse (array $array [, bool $preserve_keys = FALSE])

Vraća preokrenuto polje (zadnji član je prvi…) (održavaju se asocijativni parovi u polju).

Ukoliko se boolean *preserve_keys* postavi na 1 zadržavaju se parovi i u numeričkom polju.

array array_flip (array $array)

Vraća polje s zamijenjenim *key* i *value.*

bool sort (array &$array [, int $sort_flags = SORT_REGULAR])

Sortira polje uzlazno po vrijednostima. Ne zadržava asocijativne vrijednosti, *key* pretvara u brojčane indekse.

bool rsort (array &$array [, int $sort_flags = SORT_REGULAR])

Sortira polje silazno po vrijednostima. Ne zadržava asocijativne vrijednosti, *key* pretvara u brojčane indekse.

bool asort (array &$array [, int $sort_flags = SORT_REGULAR])

Sortira polje uzlazno po vrijednostima. Zadržava asocijativne vrijednosti. Zadržava *key/value* parove u asocijativnom i numeričkom polju.

bool arsort (array &$array [, int $sort_flags = SORT_REGULAR])

Sortira polje silazno po vrijednostima. Zadržava asocijativne vrijednosti.

bool ksort (array &$array [, int $sort_flags = SORT_REGULAR])

Sortira polje uzlazno po vrijednosti ključa (*key*). Zadržava asocijativne vrijednosti.

bool krsort (array &$array [, int $sort_flags = SORT_REGULAR])

Sortira polje silazno po vrijednosti ključa (*key*). Zadržava asocijativne vrijednosti.

sort_flags parametar:

sort($polje, SORT_NUMERIC)

Sortira polje numerički uzlazno.

sort($polje, SORT_REGULAR)

Sortira polje po ASCII vrijednostima uzlazno (velika slova su prije malih).

sort($polje, SORT_STRING)

Sortira polje po slovnim znakovima uzlazno.

sort($polje, SORT_LOCALE_STRING)

Sortira polje kao stringove ovisno o postavljenim lokalnim vrijednostima jezika uz pomoć setlocale();

sort($polje, SORT_NATURAL)

Sortira polje po slovnim znakovima uzlazno korištenjem "*natural ordering*". Ponaša se kao funkcija *natsort()*. Primjerice, ukoliko su članovi polja recimo slika1, slika10, slika2, slika20; sortiranje će se napraviti kao:

slika1, slika2, slika10, slika20

sort($polje, SORT_FLAG_CASE)

Sortira neovisno o malim i velikim slovima. Može se kombinirati s SORT_STRING ili SORT_NATURAL. Slična je funkcija *natcasesort()* koja sortira korištenjem "*natural ordering*" i sortiranje ne ovisi o malim ili velikim slovima ukoliko se koristi *sort*($polje, SORT_NATURAL | SORT_FLAG_CASE)

Primjer 1. (sortiranje sa hrvatskim znakovima)

```
setlocale(LC_ALL, 'hrv_HRV');
$gradovi= array (
  "Zagreb", "Čakovec", "Osijek", "Sisak", "Zagreb",
  "Sisak");
sort($gradovi, SORT_LOCALE_STRING);
```

Postavljanje lokalnih postavki jezika na hrvatski jezik:

```
setlocale(LC_ALL, 'hr_HR','hrv_HRV', 'hr', 'hrv', 'croatian',
'hrvatski', 'Croatian', 'Hrvatski', 'hr_HR.utf8');
```

7.7. SPAJANJE I RAZDVAJANJE POLJA

array array_merge (array $array1 [, array $...])

Spajanje više polja. Spajanje se vrši tako da se elementi dodaju na kraj prvog polja. Ukoliko se string *key* poklapa u oba polja, vrijednost *key* iz prvog polja se mijenja s drugim poljem.

array array_merge_recursive (array $array1 [, array $...])

Tamo gdje se string *key* poklapa nastaju multidimenzionalna polja.

array array_combine (array $keys , array $values)

Od polja 1 se uzimaju *keys* a od polja 2 *values*.

array array_slice (array $array , int $offset [, int $length = NULL [, bool $preserve_keys = FALSE]])

Vraća novo polje s odabranim elementima.

offset; (odmak)

- pozitivne vrijednosti odmaka označavaju uzimanje vrijednosti od početka polja u odmaku koji je zadan
- negativne vrijednosti označavaju uzimanje vrijednosti od kraja polja u odmaku koji je zadan.

length; (broj znakova)

- ako se ne zada *length* selekcija ide od *offset* do zadnjeg elementa
- pozitivan *length* ide prema kraju polja
- negativan *length* ide do *count($polje) – length*

array array_splice (array &$input , int $offset [, int $length = count($input) [, mixed $replacement = array()]])

Uklanja elemente iz polja i uklonjene vraća u obliku polja. Ukoliko se zada *array replacement* na uklonjena mjesta postavljaju se te vrijednosti.

array array_intersect (array $array1 , array $array2 [, array $...])

Vraća vrijednosti koje se nalaze u svim poljima s očuvanim redoslijedom iz prvog polja.

array array_intersect_assoc (array $array1 , array $array2 [, array $...])

Kraća vrijednosti s očuvanim *key* iz prvog polja koje se nalaze u svim poljima. U usporedbi uključuje i *array key*. Vraća samo parove *key/value* koji se pojavljuju u svim poljima.

array array_diff (array $array1 , array $array2 [, array $...])

Suprotno od *intersect*. Vraća samo one vrijednosti koje su samo u prvom polju a nema ih u ostalima.

array array_diff_assoc (array $array1 , array $array2 [, array $...])

Vraća parove *key/value* iz prvog polja koji se ne pojavljuju u ostalim poljima.

8. STRING

String je niz znakova, gdje je znak isto što i bajt. To znači da PHP podržava samo znakovni set od 256 znakova i nema ugrađenu podršku za Unicode. Stringovi se mogu kreirati korištenjem jednostrukih i dvostrukih navodnika, korištenjem *heredoc* i *nowdoc* sintakse.

Ukoliko se string varijable deklariraju pomoću jednostrukih navodnika sadržaj upisan unutar stringa se ne prevodi prilikom ispisa. Jedino što se prevodi je \' i \\ ukoliko se žele ispisati ' ili \. Sve ostale escape sekvence i varijable se ne prevode.

Primjer 1.

```
print 'Ovaj $string će se ispisivati točno kako je \n
deklariran, osim jednostrukih \' navodnika';

 //ispis:
/*Ovaj $string će se ispisivati točno kako je \n deklariran,
osim jednostrukih ' navodnika*/
```

Kod dvostrukih navodnika prevode se escape sekvence i varijable. Pri tome postoji jednostavna i složena sintaksa. Složena sintaksa prepoznaje se po vitičastim zagradama koje okružuju izraz. Koristi se kako bi se ogradili složeni izrazi.

Primjer 2.

```
$zagrade='vitičaste zagrade';
$polje['kljuc']= "asocijativnog polja";
echo "Ove { $zagrade} se neće dobro prevesti jer postoji
razmak, pa se ispisuju i $zagrade";
echo "Ovo je ispravno korištenje {$zagrade}";
echo "Kod ispisa vrijednosti iz {$polje['kljuc']} koje
sadrži jednostruke navodnike obavezno je korištenje
{$zagrade}. Korištenje $zagrade nije potrebno kod numeričkih
indeksa. Kod multidimenzionalnih polja je također obavezno";
```

Primjer 2. *Heredoc* sintaksa

```
$varijabla=<<<PRIMJER
Ovdje se upisuje tekst
PRIMJER;
//zadnja linija ne smije biti uvučena, smije sadržavati
//samo ; i nikakve bjeline oko sebe
```

Primjer 3. *Nowdoc* sintaksa

```
$varijabla=<<<'PRIMJER'
Ovdje se upisuje tekst
PRIMJER;
//zadnja linija ne smije biti uvučena, smije sadržavati
//samo ; i nikakve bjeline oko sebe
```

String deklariran kao *heredoc* se ponaša kao dvostruki navodnici, što znači da će se varijable prevoditi kao i escape sekvence. *Nowdoc* se ponaša kao jednostruki navodnici, što znači da nema prevođenja unutar stringa.

Znakovima unutar stringa moguće je pristupati preko odmaka korištenjem uglatih ili vitičastih zagrada. U tom slučaju string se ponaša kao polje znakova veličine bajta pa se mogu javiti problemi ukoliko se radi o znakovima koji sadrže više od jednog bajta (npr. ukoliko se koriste hrvatski znakovi). Od PHP 7.1.0 moguće je zadavati i negativne indekse kako bi se odmak definirao od kraja stringa.

Stringovi se spajaju korištenjem točke.

8.1. REGULARNI IZRAZI

Regularni izrazi su posebni tekstualni stringovi za opisivanje uzorka koji se pretražuje. PHP je podržavao POSIX (eng. *Portable Operating System Interface for Unix*) funkcije koje postaju zastarjele od verzije PHP 5.3 a od verzije PHP 7 su izbačene.

U PHP-u je moguće koristiti PCRE (eng. *Perl Complatible Regular Experssions*).

Regularne izraze je teško čitati i kriptični su. Kako bi se mogli čitati potrebno je naučiti značenje pojedinih znakova. Regularni izrazi su izuzetno moćan način provjeravanja upisa korisnika pri validaciji formi.

Delimiteri

Kad se koriste PCRE funkcije izrazi trebaju biti ograđeni delimiterom (razdjeljivač, graničnik). Delimiteri ne smiju biti alfanumerički znakovi, bjeline ili kosa crta unatrag (\).

Često korišteni delimiteri su */, # ili ~* . Ukoliko se u izrazu traži odabrani delimiter prethodi mu escape sekvenca \.

Raspon znakova- klase znakova

Raspon znakova koji se pretražuje obilježava se uglatim zagradama i predstavlja ASCII raspon znakova.

Primjerice:

[0-9] odgovara bilo kojoj decimalnoj brojki od 0 do 9

[a-z] odgovara znakovima od a do z

[A-Z] odgovara znakovima od A do Z

[A-Za-z] odgovara znakovima od A do z.

Određivanje broja ponavljanja (eng. Quantifiers)

U izrazu koji se zapisuje moguće je određivati broj ponavljanja određenog uzorka. Za to se koriste posebni znakovi sa sljedećim značenjem:

Oznaka	Opis
?	0 ili 1 quantifier, izraz se ponavlja 0 ili 1 puta
*****	0 ili više quantifier
+	1 ili više quantifier
{min, max}	min, max quantifier- za sekvence, označavaju minimalan i maksimalan broj ponavljanja

Primjeri korištenja:

p+ odgovara bilo kojem stringu koji ima bar 1 p

p* odgovara bilo kojem stringu koji ima 0 ili više p-a

p? odgovara bilo kojem stringu koji ima 0 ili 1 p

p{2} odgovara bilo kojem stringu koji ima sekvencu od 2p-a

p{2,3} odgovara bilo kojem stringu koji ima sekvencu od 2 ili 3p-a

p{2,} odgovara bilo kojem stringu koji ima sekvencu od najmanje 2p-a.

Posebni znakovi (eng. Meta-characters)

Oznaka	Opis
\	escape character (treba nam za rezervirane znakove ', +, [,), ^, $, |, \, /, *)
^	početak izraza
$	kraj izraza
.	bilo koji znak osim newline ukoliko nije postavljen u klasu. U klasi ima značenje točke kao slovnog znaka.
[character class definition]	Određivanje klase znakova. Uz pomoć znaka ^ moguće je izraditi negiranje klase, dok – označava raspon. Moguće je koristiti i \ kao generalnu escape sekvencu.
(subpattern)	podizraz pretrage
|	alternativna pretraga (ili)

Primjeri korištenja:

p$ odgovara bilo kojem stringu koji ima p na kraju stringa

^p odgovara bilo kojem stringu koji ima p na početku stringa

[^a-zA-Z] odgovara bilo kojem stringu koji nema znakove od a-z i A-Z

p.p odgovara bilo kojem stringu koji ima p zatim bilo koji slovni znak, a zatim opet p

^.{2}$ odgovara bilo kojem stringu koji ima točno 2 slovna znaka

(.*) bilo koji string između i .

p(hp)* bilo koji string koji ima p kojeg prati 0 ili više instanci sekvence hp.

Escape sekvence za specificiranje posebnih generičkih tipova podataka

Oznaka	Opis
\d	Isto kao i klasa [0-9], samo brojevi
\D	Isto kao i klasa [^0-9], sve osim brojeva (eng. nondigit)
\s	Bjeline (eng. whitespace (space, tab, newline, carriage return))
\S	Sve osim bjelina (eng. nonwhitespace)
\w	Riječ (eng. Word) - bilo koji string koji sadrži samo underscore i alphanumeric znakove [a-zA-Z0-9_].
\W	Sve osim riječi (eng. nonword character), [^a-zA-Z0-9_]
\b	obuhvaća samo granice riječi /da\b/ →moda, roda, hoda… NE→danas, nadati
\B	sve osim granica riječi

Dodatne postavke za izraze (eng. Modifiers)

Modifier	Opis
i	(eng. case-insensitive) podudaranje se vrši i na kurente i verzale
m	Kod traženja podudaranja osnovno je ponašanje da se string gleda kao jedna linija iako može sadržavati više linija teksta (newline znakova). Ukoliko se zada pretraživanje jedne linije uvažavaju se prelasci u sljedeći red. Koristi se kod $ i ^ kad se želi uzeti stvarna linija a ne više njih
s	U slučaju postavljanja ove postavke točka (.) odabire i newline znakove u pretragama
x	Ignoriranje bjelina osim ukoliko su u izrazu postavljeni s escape sekvencom ili unutar klase znakova
U	Stani na prvom podudaranju "ungreedy"
u	Uzorak i tekst koji se pretražuje smatraju se kodiranim u utf-8.

U tablici su pobrojane samo neke postavke a ostale se mogu proučiti na http://php.net/manual/en/reference.pcre.pattern.modifiers.php. Dodatne postavke za izraze se stavljaju na kraj regularnog izraza, nakon delimitera.

Primjeri:

/php/i podudaranje se vrši s bilo kojom kombinacijom verzala i kurenta (pr. php, PHP, Php)

/\$\d+/U Vraća prvu cijenu koja je formata $broj

8.1.1. Funkcije koje koriste regularne izraze

Sve funkcije koje započinju sa preg_ su funkcije koje koriste PCRE.

array preg_grep (string $pattern , array $input [, int $flags = 0])

Funkcija *preg_grep* služi za pretraživanje polja i podudaranja vraća u obliku polja. Uzorak koji se pretražuje treba staviti u navodnike jer se očekuje string.

Moguće je kao opcionalni parametar postaviti PREG_GREP_INVERT pri čemu se vraćaju članovi polja koji ne odgovaraju uzorku.

Primjer 1.

```
$hrana = array (
'jabuke', 'kruške', 'jaja', 'marelice', 'banane', 'jagode'
);
//traženje elemenata polja koji započinju slovom j
$hranaJ =preg_grep ("/^j/", $hrana);
print_r ($hranaJ);
```

int preg_match (string $pattern , string $subject [, array &$matches [, int $flags = 0 [, int $offset = 0]]])

Funkcija vraća 0 ili 1 te traži jedno podudaranje u stringu i sprema ga u polje. Ukoliko postoji više izraza u zagradama podudaranja u zagradama se odvojeno spremaju u isto polje.

Primjer 1.

```
$recenica= "Danas je jako lijep dan";
if (preg_match ("/dan/i", $recenica)) print "Riječ dan se
nalazi u stringu";
```

int preg_match_all (string $pattern , string $subject [, array &$matches [, int $flags = PREG_PATTERN_ORDER [, int $offset = 0]]])

Funkcija vraća broj punih podudaranja. Traži sva podudaranja u stringu i može ih spremiti u polje. Ukoliko postoji više izraza u zagradama oni se odvojeno spremaju u isto polje.

Primjer 1.

```
$recenica= "Danas je jako lijep dan";
if (preg_match_all ("/\bdan\S*/i", $recenica, $polje))
print "Riječ dan se nalazi u stringu <br />";
print_r ($polje);
//Ispis polja:
//Array ( [0] => Array ( [0] => Danas [1] => dan ) )
```

mixed preg_replace (mixed $pattern , mixed $replacement , mixed $subject [, int $limit = -1 [, int &$count]])

Funkcija pretražuje *$subject* prema *$pattern* i mijenja ga s *$replacement*. Vraća modificirani rezultat. *$limit* određuje koliko pojavljivanja će se zamijeniti. Varijable *$subject*, *$pattern*, *$replacement* i podatak vraćen iz funkcije mogu biti i polja.

Primjer 1.

```
$primjer= "<b>Danas</b> je <b>utorak</b>, a sutra
<b>srijeda</b>";
$bold = array("/<b>/", "/<\/b>/");
$italic = array("<i>", "</i>");
echo preg_replace ($bold, $italic, $primjer);
//Ispis:
//Danas je utorak, a sutra srijeda
```

array preg_split (string $pattern , string $subject [, int $limit = -1 [, int $flags = 0]])

Funkcija vraća polje koje je izrađeno iz stringa razdijeljenog prema regularnom izrazu.

Limit određuje koliko će se pojavljivanja zamijeniti.

Primjer 1.

```
$tekst = "Međimursko  Veleučilište u Čakovcu";
$polje = preg_split("/\s{1,}/", $tekst);
foreach($polje as $rijec) echo $rijec."<br />";
/*Ispis:
Međimursko
Veleučilište
u
Čakovcu*/
```

8.2. STRING FUNKCIJE

Osim funkcija za regularne izraze postoji više od 100 funkcija za manipuliranje svakim aspektom stringova. Slijedi pregled funkcija koje ostvaruju različite zadatke.

8.2.1. Određivanje dužine stringa

int strlen (string $string)

Primjer 1.

```
$pswd = "skrivenipswd";
if (strlen($pswd) < 10)
echo "Lozinka je prekratka!";
else
echo "Lozinka je u redu!";
```

int mb_strlen (string $str [, string $encoding = mb_internal_encoding()])

Primjer 1.

```
mb_strlen($odlomak, "utf-8")
```

Koristi se kad imamo hrvatske znakove jer strlen u biti broji broj bajtova nekog stringa.

8.2.2. Uspoređivanje 2 stringa

int strcmp (string $str1 , string $str2)

Uspoređivanje koje ovisi o verzalima i kurentima (eng. *case sensitive*).

Funkcija vraća: 0 ukoliko su stringovi jednaki, vrijednost <0 ukoliko je *$str1* manji od *$str2* i >0 ukoliko je *$str1* veći od *$str2*.

Primjer 1.

```
$pswd = "lozinka";
$pswd2 = "lozinka2";
if (strcmp($pswd,$pswd2) != 0)
echo "Niste unijeli ispravnu lozinku!";
else echo "Lozinka odgovara!";
```

int strcasecmp (string $str1 , string $str2)

Funkcija je slična *strcmp* funkciji osim što uspoređivanje ne ovisi o verzalima i kurentima (eng. *case insensitive*).

int strspn (string $subject , string $mask [, int $start [, int $length]])

Funkcija vraća dužinu prvog stringa gdje se računaju samo znakovi koji se nalaze i u drugom stringu. Prvi znak se treba podudarati- ako se ne podudara vraća 0. Kad nađe znak koji nije u setu prestaje traženje.

Primjer 1.

```
$password = "3312345";
if (strspn($password, "1234567890") == strlen($password))
echo "Lozinka ne može sadržavati samo brojeve!";
```

int strcspn (string $subject , string $mask [, int $start [, int $length]])

Suprotno od *strspn* funkcije. Funkcija vraća dužinu prvog stringa gdje se računaju samo znakovi koji se ne nalaze u drugom stringu. Kad nađe prvo podudaranje vraća dužinu do trenutne pozicije.

Primjer 1.

```
$password = "3312345";
if (strcspn($password, "1234567890") == 0)
echo "Lozinka ne može sadržavati samo brojeve!";
```

8.2.3. Promjena verzala i kurenata (eng. *string case*)

string strtolower(string $str)

Svi slovni znakovi se pretvaraju u kurente (mala slova).

string strtoupper(string $str)

Svi slovni znakovi se pretvaraju u verzale (velika slova).

string ucfirst(string $str)

Prvo slovo rečenice postaje verzal.

string ucwords(string $str)

Svaka riječ u rečenici počinje verzalom.

8.2.4. Konvertiranje u i iz HTML-a

string nl2br (string $string [, bool $is_xhtml = TRUE])

Ubacuje HTML prijelaz u sljedeći red (
 ili
) ispred svakog prelaska u sljedeći red (\r\n, \n\r, \n i \r).

string htmlentities (string $string [, int $flags = ENT_COMPAT | ENT_HTML401 [, string $encoding = ini_get("default_charset") [, bool $double_encode = TRUE]]])

Svi znakovi koji imaju HTML znakovni entitet pretvaraju se u te entitete.

Mogu se zadavati parametri kao sljedeći:

ENT_QUOTES: pretvara jednostruke i dvostruke navodnike

ENT_NOQUOTES: ne pretvara navodnike

ENT_COMPAT: pretvara samo dvostruke navodnike

ENT_HTML401: Kôd se gleda kao HTML 4.01

ENT_HTML5: Kôd se gleda kao HTML 5

U kodiranju se može zadati željeni set kodiranja znakova i ovisi o verziji PHP-a. Od verzije PHP 5.6 koristi se osnovni set kodiranja (*default_charset*) koji je postavljen na UTF-8.

Primjer 1.

```
$tekst = "<Međimursko  Veleučilište u Čakovcu®>";
echo htmlentities($tekst,ENT_QUOTES,"UTF-8");

/*HTML KÔD:
&lt;Međimursko  Veleučilište u Čakovcu&reg;&gt;
Ispis u pregledniku:
<Međimursko Veleučilište u Čakovcu®>*/
```

string htmlspecialchars (string $string [, int $flags = ENT_COMPAT | ENT_HTML401 [, string $encoding = ini_get("default_charset") [, bool $double_encode = TRUE]]])

Funkcija je slična *htmlentities* osim što pretvara u entitete samo HTML specijalne znakove kao što su:

& → &

" → "

' → '

< → <

> → >

string strtr (string $str , string $from , string $to)

Prevodi se iz stringa $*from* u string $*to*.

Primjer 1.

```
$adresa = strtr($adresa, "äåö", "aao");
```

string strtr (string $str , array $replace_pairs)

Vraća kopiju stringa sa izvršenim prevođenjem (eng. *translate*) prema posebnom polju za zamjene.

Primjer 1.

```
$tabela = array(
     "<b>" => "<strong>", "</b>" => "</strong>");
$html = "<b>Dobar dan</b>";
echo strtr($html, $tabela);
/*Ispis:
<strong>Dobar dan</strong>;*/
```

string strip_tags (string $str [, string $allowable_tags])

Micanje HTML i PHP oznaka.

8.2.5. Alternative regularnim izrazima

Uobičajeno su brže od regularnih izraza, naročito kad se obrađuje veća količina informacija. Koriste se kod jednostavnih pretraživanja i zamjena.

string strtok (string $str , string $token)

Funkcija se mora uzastopno pozivati. Ona pamti gdje je stala u stringu pa nije potrebno ponovno navoditi string.

Primjer 1.

```
$info = "Međimursko Veleučilište: E-mail:
mev@mev.hr|Čakovec, Hrvatska";
$tokeni = "|,:";
$tokenizirano = strtok($info, $tokeni);

while ($tokenizirano) {
     echo "$tokenizirano<br/>";
     $tokenizirano = strtok($tokeni);
}
/*Ispis:
Međimursko Veleučilište
E-mail
mev@mev.hr
Čakovec
Hrvatska*/
```

array explode (string $delimiter , string $string [, int $limit = PHP_INT_MAX])

Razdvaja string *$string* u polje po separatoru *$delimiter*.

Primjer 1.

```
$gradovi="Čakovec|Zagreb|Varaždin|Koprivnica";
$polje=explode("|", $gradovi);
print_r($polje);
```

string implode (array $pieces)

Spaja polje u string.

string implode (string $glue , array $pieces)

Spaja polje u string i može razdvajati vrijednosti po delimiteru zadanom u *$glue*.

Primjer 1.

```
$gradovi = array(
     "Čakovec", "Zagreb", "Varaždin", "Koprivnica");
echo implode("|", $gradovi);
//Ispis:
//Čakovec|Zagreb|Varaždin|Koprivnica
```

int strpos (string $haystack , mixed $needle [, int $offset = 0])

int strrpos (string $haystack , mixed $needle [, int $offset = 0])

Funkcija *strpos* vraća poziciju početka traženog stringa traženjem od početka stringa dok *strrpos* traži od kraja.

Primjer 1.

```
$odlomak= "Međimursko Veleučilište <br/>
E-mail: mev@mev.hr <br/>
Web: <a href='http://www.mev.hr/'>www.mev.hr</a>";
echo strpos($odlomak, "mev"); //42
echo strrpos($odlomak, "mev"); //98
```

mixed str_replace (mixed $search , mixed $replace , mixed $subject [, int &$count])

mixed str_ireplace (mixed $search , mixed $replace , mixed $subject [, int &$count])

Funkcije služe za zamjenu znakova u traženom stringu ili polju. Razlikuju se u tome da je *str_replace* osjetljiva na verzale i kurente dok *str_ireplace* nije.

Primjer 1.

```
$autor = "mev@mev.hr";
$autor = str_replace("@","(at)",$autor);
echo "Kontaktirajte autora na $autor.";
/*Ispis:
Kontaktirajte autora na mev(at)mev.hr*/
```

string strstr (string $haystack , mixed $needle [, bool $before_needle = FALSE])

Vraća ostatak stringa nakon prvog pojavljivanja predefiniranog stringa.

Primjer 1.

```
$mail = "info@mev.hr";
echo strstr($mail, "@"); //ispisuje @mev.hr
echo ltrim(strstr($mail, "@"),"@"); //ispisuje mev.hr
```

string substr (string $string , int $start [, int $length])

Vraća dio stringa lociranog između predefiniranog startnog *$ofseta* i dužine zadane s *$length*.

Ukoliko je *$start*:

- pozitivan: započinje od početka stringa pomaknutog za vrijednost *start*
- negativan: započinje od *strlength-start*.

Ukoliko je *$length*:

- pozitivan: ide do start+length
- negativan: ide do strlength-length.

int substr_count (string $haystack , string $needle [, int $offset = 0 [, int $length]])

Brojanje koliko puta se neki string pojavljuje.

mixed substr_replace (mixed $string , mixed $replacement , mixed $start [,mixed $length])

Primjer 1.

```
$broj = "1234567899991111";
echo substr_replace($broj,"************",0,12);
//ispis: ************1111
```

8.2.6. Brisanje i dodavanje

string ltrim (string $str [, string $character_mask])

string rtrim (string $str [, string $character_mask])

Ukoliko se ne zada 2 parametar briše bjeline: \t, \n, \r i \0. *ltrim* briše s lijeve strane a *rtrim* s desne.

string trim (string $str [, string $character_mask = " \t\n\r\0\x0B"])

Kombinacija *ltrim* i *rtrim*.

string str_pad (string $input , int $pad_length [, string $pad_string = " " [, int $pad_type = STR_PAD_RIGHT]])

Drugi parametar određuje dužinu konačnog stringa. Ukoliko se ne zada 3 parametar dodaju se bjeline, dok četvrti parametar određuje mjesto dodavanja znakova:

STR_PAD_RIGHT, STR_PAD_LEFT ili STR_PAD_BOTH

Primjer 1.

```
$primjer = "primjer";
echo str_pad ($primjer, 20, "=+", STR_PAD_BOTH);
//ispis: =+=+=+primjer=+=+=+=
```

9. DATOTEKE I DIREKTORIJI

Pri radu s datotekama koristi se *stream*, što je u biti *resource* objekt. Preko *stream* moguće je zapisivati i čitati podatke iz datoteka i direktorija.

9.1. FUNKCIJE ZA RAD S PUTANJOM

string basename (string $path [, string $suffix])

Ukoliko se funkciji proslijedi putanja do datoteke ona će vratiti naziv datoteke s ekstenzijom. Ukoliko je zadan *$suffix* (ekstenzija) tada vraća samo naziv datoteke.

string dirname (string $path [, int $levels = 1])

Ukoliko se funkciji proslijedi putanja do datoteke ona će vratiti samo putanju, koja je jedan ili više nivoa iznad (određeno s *$levels*).

mixed pathinfo (string $path [, int $options = PATHINFO_DIRNAME | PATHINFO_BASENAME | PATHINFO_EXTENSION | PATHINFO_FILENAME])

Funkcija vraća informacije o putanji kao polje ili string, ovisno o *$options*.

Ukoliko vraća polje ono se može sastojati od 4 komponente: *dirname*, *basename*, *extension* i *filename*.

string realpath (string $path)

Identifikacija apsolutne putanje.

Primjer 1.

```
$datoteka= "/home/www/htdocs/vjezba/index.html";
echo  basename ($datoteka) . "<br/>"; //ispisuje index.html
echo basename ($datoteka, ".html") . "<br/>";
//ispisuje index
echo dirname ($datoteka) . "<br/>";
//ispisuje /home/www/htdocs/vjezba
print_r (pathinfo($datoteka));
//ispisuje:
//Array ( [dirname] => /home/www/htdocs/vjezba
[basename] => index.html
[extension] => html
[filename] => index )
```

9.2. FUNKCIJE ZA PROCEDURALAN RAD S DATOTEKAMA

9.2.1. Funkcije za određivanje veličine

int filesize (string $filename)

Određivanje veličine datoteke u bajtovima.

float disk_free_space (string $directory)

Vraća u bajtovima veličinu slobodnog područja diska (ili particije) na kojoj se nalazi specificirani direktorij.

float disk_total_space (string $directory)

Vraća u bajtovima veličinu diska ili particije.

Primjer 1.

```
$datoteka="../5_vjezba/vj5.php";
$velicina= filesize($datoteka);
echo "Veličina datoteke vj5.php" . round ($velicina/1024 ,
2) . " KB";
echo "Veličina slobodnog mjesta na disku/
particiji na kojem se nalazi datoteka "
. round (disk_free_space("/home")/(1024*1024*1024), 2) . "
GB";
echo "Ukupna veličina diska"
. round (disk_total_space("/home")/(1024*1024*1024), 2) . "
GB";
//umjesto home može se pisati i /usr, a kad se lokalno
//testira piše se C:
```

9.2.2. Funkcije za određivanje vremena pristupanja i promjene datoteka

int fileatime (string $filename)

Vraća vrijeme posljednjeg pristupanja datoteci (eng. *access*) kao *Unix timestamp* format (broj sekundi koje su prošle od ponoći *January 1, 1970*). Vrijednost bi se trebala mijenjati kad se datoteka čita pa je u mnogo slučajeva ta funkcionalnost isključena kako bi se uštedjelo na resursima.

int filectime (string $filename)

Obuhvaća promjene (eng. *changed*) na inode podacima (prava, grupe, vlasnici i ostali metapodaci) a ne sadržaja datoteke. Funkcija vraća vrijeme promjene inode podataka.

int filemtime (string $filename)

Funkcija vraća vrijeme promjene (eng. *modified*) u sadržaju datoteke.

Primjer 1.

```
echo "Zadnji pristup datoteci: "
. date( "d.m.Y H:i:s", fileatime($datoteka)) . "<br />";
echo "Zadnja promjena datoteke: „
. date( "d.m.Y H:i:s", filectime($datoteka)) . "<br />";
echo "Zadnja promjena samo sadržaja datoteke: "
. date( "d.m.Y H:i:s", filemtime($datoteka)) . "<br />";
```

bool touch (string $filename [, int $time = time() [, int $atime]])

Funkcija se koristi kako bi se osvježila *access* i *modified* vremena ili postavila na određeno vrijeme (*$time*). *Access* vrijeme se uvijek mijenja bez obzira na broj parametara. Ukoliko datoteka ne postoji ona će se stvoriti.

9.2.3. Rad s datotekama- proceduralni način

resource fopen (string $filename , string $mode [, bool $use_include_path = FALSE [, resource $context]])

Pomoću funkcije otvara se *stream* te se dobiva *resource handle*. Drugi parametar $*mode* specificira način pristupanja streamu koji može biti kako slijedi:

mode	opis
r	samo čitanje - pointer na početku
r+	čitanje i pisanje - pointer na početku
w	pisanje – prije pisanja obrisati sadržaj datoteke (ako ne postoji pokušava se stvoriti) - pointer na početku
w+	čitanje i pisanje – najprije obrisati sadržaj datoteke (ako ne postoji pokušava se stvoriti) - pointer na početku
a	pisanje (append) - pointer na kraj, ako datoteka ne postoji pokušava se stvoriti
a+	ako datoteka ne postoji pokušava se stvoriti
x	kreiraj datoteku i otvori samo za pisanje, ukoliko datoteka već postoji izbaciti će se pogreška
x+	kreiraj datoteku i otvori za pisanje i čitanje, ukoliko datoteka već postoji izbaciti će se pogreška
c	kreiraj datoteku i otvori samo za pisanje, ukoliko datoteka već postoji pointer se postavlja na početak datoteke
c+	kreiraj datoteku i otvori za pisanje i čitanje, ukoliko datoteka već postoji pointer se postavlja na početak datoteke
b	otvori u binarnom modu/ nemoj prevoditi
t	otvori u tekstualnom modu/ prevedi u Windows line Endings (\r\n), vrijedi za Windows OS. Zbog portabilnosti kôda preporuča se ne koristiti.

Treći parametar *$use_include_path* moguće je postaviti na 1 ukoliko se žele koristiti putanje iz php.ini datoteke.

$context omogućuje dijeljenje datoteke i *streama* između više *fopen* zahtjeva.

bool fclose (resource $handle)

Zatvaranje prethodno otvorenih datoteka (vraća TRUE kad je uspješno).

bool feof (resource $handle)

Funkcija vraća TRUE ako naiđe *eof* (*end of file*) a ukoliko pointer nije na kraju vraća FALSE. Funkciji je potrebno proslijediti *file handle*.

int ftell (resource $handle)

Vraća trenutačnu poziciju pointera.

int fseek (resource $handle , int $offset [, int $whence = SEEK_SET]))

Postavlja pointer na poziciju mjerenu u bajtovima od početka datoteke specificirano s odmakom.

bool rewind (resource $handle)

Postavlja pointer na početak datoteke.

int fwrite (resource $handle , string $string [, int $length])

Zapisivanje u datoteku.

string fgets (resource $handle [, int $length])

Čita liniju teksta sve do newline ili EOF oznake. Ukoliko je $*lenght* izostavljen čita do 1,024 znakova.

string fgetss (resource $handle [, int $length [, string $allowable_tags]])

Slično kao i *fgets* osim što briše HTML i PHP oznake.

array file (string $filename [, int $flags = 0 [, resource $context]])

Čita cijelu datoteku u polje (članovi polja određuju se prema *newline* znaku, kopira i *newline* znak). Ne zahtjeva rad s *resource handle*.

string file_get_contents (string $filename [, bool $use_include_path = FALSE [, resource $context [, int $offset = 0 [, int $maxlen]]]])

Čita cijelu datoteku u string.

string fgetc (resource $handle)

Čitanje znak po znak. Kad se naiđe na EOF vraća se *false*.

string fread (resource $handle , int $length)

Čita $*length* znakova (bajtova) ili dok ne dođe do kraja datoteke.

Primjer 1.

```
$datoteka = "datoteke/korisnici.txt";
$hand = fopen($datoteka, "r");
// Čitanje cijele datoteke
$podaci = nl2br( fread($hand, filesize($datoteka)));
echo ($podaci);
fclose($hand);
```

int file_put_contents (string $filename, mixed $data [, int $flags = 0 [, resource $context]])

Skraćeno za *fopen/fwrite/fclose*. Funkcija vraća broj bajtova koji su zapisani u datoteku. Briše datoteku i dodaje sadržaj.

Primjer 1.

```
$velicina= file_put_contents('naziv.txt', $sadrzaj)
```

string file_get_contents (string $filename [, bool $use_include_path = FALSE [, resource $context [, int $offset = 0 [, int $maxlen]]]])

Skraćeno za *fopen/fread/fclose.*

Primjer 1.

```
$sadrzaj= file_get_contents('naziv.txt')
```

9.3. FUNKCIJE ZA PROCEDURALAN RAD S DIREKTORIJIMA

9.3.1. Čitanje sadržaja direktorija

resource opendir (string $path [, resource $context])

Otvaranje *handler* za datotečni sustav.

void closedir ([resource $dir_handle])

Zatvaranje *handler* za datotečni sustav.

string readdir ([resource $dir_handle])

Čitanje sadržaja direktorija u string. Čita se datoteku po datoteku.

void rewinddir ([resource $dir_handle])

Pomicanje pokazivača na početak.

array scandir (string $directory [, int $sorting_order = SCANDIR_SORT_ASCENDING [, resource $context]])

Vraća polje u kojem se nalaze datoteke i direktoriji u traženom direktoriju.

string getcwd (void)

Vraća trenutačni direktorij.

bool mkdir (string $pathname [, int $mode = 0777 [, bool $recursive = FALSE [, resource $context]]])

Pokušava stvoriti direktorij specificiran s putanjom. Omogućava i rekurzivno kreiranje direktorija (više direktorija u putanji koja je specificirana) ukoliko se zada treći parametar.

Primjer 1.

```
mkdir('naziv/test/test2', 0777, true)
```

bool chdir (string $directory)

Omogućuje promjenu direktorija, otvaranje nekog drugog direktorija.

bool rmdir (string $dirname [, resource $context])

Prazan i zatvoren direktorij se može obrisati ukoliko je to dozvoljeno.

bool unlink (string $filename [, resource $context])

Brisanje datoteke.

9.4. SPL (*STANDARD PHP LIBRARY*) – OBJEKTNO ORIJENTIRANI NAČIN RADA S DATOTEKAMA, PUTANJAMA I DIREKTORIJIMA

U SPL- standardnoj PHP biblioteci postoje klase koje omogućuju objektno orijentirani način rada s datotekama. To je omogućeno preko *SplFileInfo, SplFileObject* i *SplTempFileObject* klasa. Navedene klase moguće je koristiti od PHP verzije 5.1.2.

SplFileInfo

SplFileInfo klasa omogućuje dobavljanje podataka o datoteci kao što su putanje, metapodaci, vrsti datoteke, vremenu izmjene podataka, veličini datoteke i dr. Dodatne informacije o mogućim metodama mogu se pogledati na http://php.net/manual/en/class.splfileinfo.php.

Primjer 1.

```
$info = new SplFileInfo('/putanja/do/datoteke.txt');
echo($info->getBasename());
//ispisuje datoteke.txt
```

SplFileObject

Omogućuje objektno orijentirani pristup radu s datotekama i podklasa je *SplFileInfo* pa nasljeđuje i njezine metode. Više informacija o klasi i metodama moguće je pronaći na http://php.net/manual/en/class.splfileobject.php.

Primjer 1.

```
$datoteka="datoteke/korisnici.txt";
$dat = new SplFileObject($datoteka, "r");
echo $dat->fread($dat->getSize());
```

SplTempFileObject

Za privremene datoteke (zapisuju se samo u memoriji). Podklasa je *SplFileObject*. Više informacija o klasi i metodama moguće je pronaći na http://php.net/manual/en/class.spltempfileobject.php.

10. OBJEKTNO ORJENTIRANI PHP

Objektno-orijentirani jezik koristi klase i objekte. Klase su slične nacrtima. Klasa opisuje što objekt može sadržavati, uključujući parametre/varijable i funkcije/metode. Objekt je primjer klase (kao zgrada koja je kreirana iz nacrta). Objektno-orijentirani jezik omogućuje polimorfizam (eng. *polymorphism*), kapsuliranje (eng. *encapsulation*) i nasljeđivanje (eng. *inheritance*).

Objekti su kapsulirani, tako da sadrže sve srodne funkcije/metode i parametre/varijable/svojstva unutar samog objekta.

Polimorfizam omogućava dupliranje naziva metoda/funkcija unutar objektno-orijentiranih objekata. Polimorfizam je moguće postići kombinacijom vrsta varijabli koje se traže u metodi/funkciji i vrste informacije koja je proslijeđena metodi/funkciji. Primjerice, moguće je izraditi nekoliko metoda *add* – jedna koja prihvaća samo cijele brojeve, druga za decimalne brojeve i treća koji prihvaća kombinaciju. Program će odrediti koju metodu/funkciju treba pozvati po onome što je proslijeđeno.

Nasljeđivanje u objektno-orijentiranom programiranju omogućava da objekt „naslijedi“ parametre i funkcije iz drugog objekta. Objekt također može izmijeniti „naslijeđene“ stavke.

Objektno-orijentirani jezici također mogu biti upravljani događajima. Program upravljan događajima neće se izvršavati dok se ne desi događaj.

Osnovna definicija klase

```
class Class_Ime {
// deklaracije konstanti, svojstava i metoda
}
```

Primjer 1.

```
class zaposlenici
{
   //svojstva
   private $ime;
   private $titula;
   protected $placa;
   //metode
   protected function pocetakRV() {
      echo "Član $this->ime upisao se " . date("h:i:s");}
   protected function zavresetakRV() {
      echo " Član $this->ime ispisao se ".date("h:i:s");}
}
```

Kreiranje objekata

Objekti se kreiraju iz klasa kao njihove instance pri čemu se koristi ključna riječ *new*.

```
$zaposlenik= new zaposlenici();
```

10.1. SVOJSTVA OBJEKATA/KLASA

Svojstva klasa (eng. *fields, properties, attributes*) su varijable, konstante ili parametri koji se deklariraju se na početku klasa. Može im se postaviti i zadana vrijednost, primjerice:

```
private $ime= "Ivica";
//konstante
const PI = '3.14159265';
const E = '2.7182818284';
```

Kako bi se pozvala unutar PHP kôda referencira se preko -> i ne koristi se $ znak, primjerice:

```
$zaposlenik->ime
$zaposlenik->titula
$zaposlenik->placa.
```

Kada se svojstva pozivaju unutar klase, recimo u deklariranim metodama, koristi se *$this* varijabla koja je referenca na objekt koji poziva metodu (uobičajeno objekt kojem pripada metoda), primjerice:

```
$this->ime;
```

Primjer korištenja *$this* unutar metode:

```
function postaviIme ($Ime) {
     $this->ime=$Ime;
}
```

Ukoliko se želi dohvatiti vrijednost konstante u klasi koristi se ključna riječ *self::*.

```
class klasa
{
    const KONSTANTA = 'vrijednost';
    function prikaziKonst() {
        echo  self::KONSTANTA . "\n";
    }
}
```

Svako svojstvo deklarirano unutar klase treba obavezno imati zadan doseg (eng. *scope*) koji može biti zadan kao: *public, protected,* i *private*. Prilikom deklariranja svojstva moguće je koristiti i ključnu riječ *static* s kombinacijama zadanih svojstava vidljivosti. Moguće je koristiti i riječ *var* što označava *public* varijablu.

Osnovna vidljivost konstante je *public.* Od verzije PHP 7.1.0 konstantama je također moguće dodjeljivati vidljivost.

public

Public oznaka varijable ili konstante (od verzije PHP 7.1) uobičajeno se ne koristi jer je na navedeni način validacija upisanih podataka onemogućena. Ukoliko je svojstvo postavljeno kao *public*, tom svojstvu se može pristupati iz svih dijelova kôda.

Primjer 1.

```
class zaposlenici {
     public $ime;
}
$zaposlenik= new zaposlenici();
$zaposlenik->ime="Marko";
$ime=$zaposlenik->ime;
echo "Novi zaposlenik: $ime";
```

private

Private svojstva nisu direktno dostupna u kôdu i nisu direktno dostupna podklasama. Dostupna su jedino unutar klase koja ih deklarira. Češće se koriste zbog mogućnosti provjere podataka prije spremanja. Kod takvih svojstava podatke je moguće dohvaćati i mijenjati samo preko funkcija.

Primjer 1.

```
class zaposlenici {
  private $ime;
  public function postaviIme($Ime) {
    $this->ime=$Ime;
  }
  public function vratiIme () {
     return $this->ime;
  }
}
$marko= new zaposlenici();
$marko->postaviIme("Marko");
echo $marko->vratiIme();
```

protected

Svojstva zadana kao *protected* nisu direktno dostupna iz kôda, dostupna su unutar klase, u podklasama i roditeljskim klasama.

static

Svojstva zadana kao *static* pamte prethodne vrijednosti. Kako bi se pristupilo svojstvima koja su zadana kao *static* nije potrebno instancirati objekt. U kôdu takvim svojstvima nije moguće pristupati preko -> već se koristi sintaksa koja uključuje naziv klase *klasa::*.

Primjer 1.

```
class staticnaVar
{
    public static $staticna = 'zadana vrijednost';
    public function staticnaVrijednost() {
        return self::$staticna;
    }
}
print staticnaVar::$staticna . "\n";
$objekt = new staticnaVar();
print $objekt->staticnaVrijednost() . "\n";
```

10.2. METODE OBJEKATA/KLASA

Metode deklarirane u klasi trebaju imati zadan doseg (*scope*). Doseg može biti zadan kao *public*, *private*, *protected*, *abstract*, *final* i *static*.

Public

Primjer 1.

```
class Posjetitelji {
   public function pozdravi() {
      echo "Dobar dan<br />";
   }
   function odzdravi(){
      echo "Doviđenja<br />";
   }
}
$posjetitelj= new Posjetitelji();
$posjetitelj->odzdravi();
```

private

Private metode ne mogu se pozivati direktno iz instanciranog objekta. One su nedostupne u podklasama.

protected

Protected je sličan kao *private* ali su dostupne u podklasama.

abstract

Mogu se koristiti samo u klasama koje su postavljene kao *abstract* koje služe kao baza za druge klase. Deklariraju se u roditeljskoj klasi a implementiraju u *child* klasama.

final

Metode zadane kao *final* ne mogu se mijenjati od strane podklasa. Ključnu riječ *final* moguće je koristiti i kod deklaracije klase što tada označava da se klasa ne može nasljeđivati. *Final* se ne može koristiti kod svojstava objekata.

static

Metoda postavljena kao *static* pamti prethodne vrijednosti. Metodi je moguće pristupiti bez instanciranja objekta. U metodi unutar klase koja je postavljena kao *static* nije moguće koristiti ključnu riječ $*this*. Ukoliko se metode koje nisu zadane kao *static* pozivaju kao statične metode u PHP-u verzije 5 dobiva se upozorenje a u verziji 7 upozorenje da se koristi zastarjela sintaksa. Pozivanje ne statičkih metoda na statičan način moglo bi se izbaciti iz jezika pa se ne preporuča koristiti u kôdu. Statične metode pozivaju se sintaksom koja je prikazana u sljedećim primjerima.

Primjer 1.

```
class staticna {
    public static function staticnaMetoda() {
        // ...
    }
}
staticna::staticnaMetoda();
```

Primjer 2.

```
class objekti {
  private static $objekti= 0;
  function __construct(){
    self::$objekti++;
  }
  static function vratiBrojObjekata(){
    return self::$objekti;
  }
}
/*Instanciranje objekta*/
$objekt1= new objekti();
echo objekti:: vratiBrojObjekata()."<br />";
/* Instanciranje novog objekta*/
$objekt2= new objekti();
echo objekti:: vratiBrojObjekata()."<br />";

//Ispis:
//1
//2
```

10.2.1. Zadavanje tipova varijabli (eng. *Type hinting*)

Zadavanje tipova varijabli pri deklaraciji klasa uvedeno je od verzije PHP 5. Osigurava da objekt koji se prosljeđuje metodi bude član očekivane klase, polje ili se mogu zadavati skalarni tipovi podataka od verzije PHP 7.

Primjer 1.

```
private function hodaj (osobe $osoba)
{
 ...
}
```

10.2.2. Konstruktorske i destruktorske metode

PHP omogućava deklariranje konstruktorskih i destruktorskih metoda unutar klasa. Navedene metode pozivaju se prilikom inicijalizacije ili destrukcije objekta.

Konstruktorske metoda

Konstruktorska metoda koristi se unutar klase i to je metoda koja se poziva prilikom inicijalizacije objekta. Sintaksa je:

```
void __construct ([ mixed $args = "" [, $... ]] )
//napomena:
//ispred riječi construct zapisane su 2 crtice dole
```

Zbog kompatibilnosti s PHP 3 i 4 moglo se umjesto navedene sintakse koristiti kao konstruktorsku metodu funkciju istog naziva kao i sama klasa, i ukoliko nije bila zadana konstruktorska metoda tražila se funkcija istog naziva. Od verzije PHP 7 ovakav način deklariranja konstruktorske funkcije postaje zastario.

Primjer 1.

```
class knjige {
   private $naslov;
   private $isbn;
   private $kopija;
   public function __construct($isbn){
      $this->postaviISBN($isbn);
      $this->dohvatiNaslov();
      $this->dohvatiBRkopija();
   }
   public function postaviISBN($isbn){
      $this->isbn = $isbn;
   }
   public function dohvatiNaslov() {
      $this->naslov= "Naslov knjige";
      print "Naslov: ".$this->naslov."<br />";
   }
   public function dohvatiBRkopija() {
      $this->kopija = "5";
      print "Broj dostupnih kopija: "
           .$this->kopija."<br />";
   }
}
$knjiga= new knjige("159059519X");
//Ispis:
//Naslov: Naslov knjige
//Broj dostupnih kopija: 5
```

Pozivanje roditeljskog (engl. parent) konstruktora

Kada se instancira *child* klasa koja nema konstruktor koristi se konstruktor *parent* klase koji se nasljeđuje kao i ostale metode (ukoliko postoji u roditeljskoj klasi i nije zadan kao *private*). Ukoliko *child* klasa ima konstruktor on će se pozvati bez obzira ima li *parent* klasa konstruktor. U tom slučaju neće se pozvati konstruktor *parent* klase ukoliko to nismo u konstruktorskoj metodi eksplicitno zadali. Ukoliko se želi pozvati konstruktor roditelja potrebno je upotrijebiti izjavu parent::__construct();

Primjer 1.

```
class zaposlenici
{
   protected $ime;
   protected $titula;
   function __construct(){
      echo "<p>Pozvan je konstruktor iz Zaposlenika!</p>";
   }
}
class voditelji extends zaposlenici {
   function __construct(){
      parent::__construct();
      echo "<p>Pozvan je konstruktor iz Voditelja!</p>";
   }
}
$zaposlenik= new voditelji();
//Ispis:
//Pozvan je konstruktor iz Voditelja!
//Pozvan je konstruktor iz Zaposlenika!
```

Ukoliko imamo više klasa koje se nasljeđuju također je moguće pozivati konstruktore iz klasa koje su roditeljske. Primjerice ukoliko imamo sintaksu sličnu sljedećem:

```
class zaposlenici
class voditelji extends zaposlenici
class direktori extends voditelji
```

Konstruktorska funkcija unutar klase direktori mogla bi glasiti:

```
function __construct() {
     zaposlenici::__construct();
     voditelji::__construct();
}
```

Destruktori

Objekti se automatski brišu nakon izvođenja skripte. Kod instanciranja samo korištenjem memorije nije potrebno koristiti destruktore. Destruktori se koriste u slučajevima kad se osim

memorije koriste i drugi zapisi kao npr. zapis u bazu kojeg je kasnije potrebno izbrisati. Sintaksa izrade destruktora je sljedeća:

```
void __destruct ( void )
```

Destruktori se ponašaju slično kao i konstruktori. Ukoliko klasa ne definira destruktor, pozvati će se iz roditeljske klase ukoliko tamo postoji. Destruktor iz roditeljske klase neće se pozvati ukoliko klasa definira vlastiti destruktor ali ga je moguće eksplicitno pozvati s parent::__destruct.

Primjer 1.

```
class knjige{
  private $naslov;
  private $isbn;
  private $kopija;
  function __construct($isbn){
    $this->isbn=$isbn;
    echo "<p>Kreirana je instanca klase Knjiga.</p>";
  }
  function __destruct(){
    echo "<p>Instanca knjige je uništena.</p>";
  }
}
$knjiga= new knjige("1893115852");
//Ispis:
//Kreirana je instanca klase Knjiga.
//Instanca knjige je uništena.
```

10.2.3. Overloading metode

Accessor (getter) metoda

Poziva se kod čitanja podataka iz svojstava koja nisu dostupna. Metoda mora biti postavljena kao *public*. Koristi sljedeću sintaksu:

```
public mixed __get ( string $name )
```

Primjer 1.

```
class osoba {
  public $ime;
  public function __get ($naziv) {
    echo "Ne postoji varijabla " . $naziv ;
  }
}
$ob=new osoba();
```

```
$ob->ime="Ivica";
echo ($ob->ime1);
```

Mutator (setter) metoda

Poziva se kod zapisivanja podataka u svojstva koja nisu dostupna. Mogu se koristiti za ispisivanje poruka o pogreškama ili za izradu novih varijabli. Koristi se sljedeća sintaksa:

```
public void __set ( string $name , mixed $value )
```

Primjer 1.

```
class osoba {
   var $ime;
   function __set ($naziv, $vrijednost) {
      echo "Ne postoji varijabla " .$naziv. " pa joj se ne
      može postaviti vrijednost: ".$vrijednost ;
   }
}
$ob=new osoba();
$ob->ime1="Ivica";
```

Primjer 2.

```
class osoba {
   var $ime;
   function __set ($naziv, $vrijednost) {
      $this->$naziv=$vrijednost;
   }
}
$ob=new osoba();
$ob->prezime=„Ivić";
```

public bool __isset (string $name)

Metoda se poziva s *isset()* ili *empty()* na svojstvima koja nisu zadana.

public void __unset (string $name)

Metoda se poziva s *unset()* na svojstvima koja nisu zadana.

public mixed __call (string $name , array $arguments)

Metoda se poziva pri pozivanju metode koja nije dostupna.

Primjer 1.

```
class koristenjeCall
{
  public function __call($naziv, $argumenti)
  {
    echo "Pozivanje metode '$naziv' "
    implode(', ', $ argumenti). "\n";
  }
}
$obj = new koristenjeCall;
$obj->proba(' u kontekstu objekta);
```

public static mixed __callStatic (string $name , array $arguments)

Metoda se poziva pri pozivanju metode u statičnom kontekstu.

Primjer 1.

```
class koristenjeCallStatic
{
  public static function __callStatic($naziv, $argumenti)
  {
    echo "Pozivanje statičke metode '$naziv' "
    implode(', ', $argumenti). "\n";
  }
}
$obj = new koristenjeCallStatic;
koristenjeCallStatic::proba('u statičnom kontekstu');
```

10.3. RAD S KLASAMA I OBJEKTIMA

10.3.1. instanceof

Provjera da li je objekt instanca neke klase.

Primjer 1.

```
$manager = new Zaposlenik();
...
if ($manager instanceof Zaposlenik) echo "Da";
```

10.3.2. Automatsko uključivanje klasa u dokument (eng. *autoloading*)

Klase se obično drže u zasebnom direktoriju npr. *classes*. Svaka klasa se uobičajeno sprema u zasebnu datoteku. Primjerice, klase bi mogli pohraniti u datotekama kao što su sljedeće:

```
zaposlenici.class.php
```

```
knjige.class.php
```

Kako bi ih uključili u skriptu možemo koristiti:

```
require_once("classes/zaposlenici.class.php")
```

ili iskoristiti mogućnost automatskog ubacivanja preko funkcije *__autolod* koja se poziva ukoliko tražena klasa ne može biti pronađena:

```
function __autoload($class) {
     require_once("classes/$class.class.php");
}
```

Funkcija *__autoload* bi mogla postati zastarjela pa se preporuča koristiti *spl_autoload_register()* funkciju koja omogućava korištenje više *__autoload* funkcija koje primjerice mogu biti uključene u bibliotekama.

```
spl_autoload_register(function ($nazivKlase) {
    require_once $nazivKlase . 'class.php';
});
```

10.3.3. Kloniranje objekata

Svi objekti se tretiraju kao reference na vrijednosti i ukoliko želimo duplicirati neki objekt moramo koristiti *clone*.

```
$zaposlenik1= clone $zaposlenik;
```

Ukoliko želimo izmijeniti način kloniranja u definiciji klase koristimo __clone() metodu. Ona će se pozvati nakon kloniranja objekta ukoliko je potrebno izmijeniti način na koji je izvedena kopija objekta.

10.3.4. *Interfaces* (sučelja)

Interface omogućava kreiranje kôda koji specificira koje metode klasa mora implementirati bez potrebe za implementacijom samih metoda. U njima se deklariraju potrebne metode i konstante. Ne zadaju se varijable. *Interface* postavlja generalni set uputa koje određuju metode koje se trebaju implementirati. Definiraju se slično kao i klase samo umjesto ključne riječi *class* postavlja se *interface*.

Sve metode deklarirane u *interface* moraju biti *public*. Kako bi se implementirao *interface* koristi se ključna riječ *implements*. Sintaksa za izradu sučelja:

```
interface nazivInterface
{
   CONST 1;
   ...
   CONST N;
   function nazivMetode1();
```

```
    ...
    function nazivMetodeN();
}
class NazivKlase implements nazivInterface
{
    function nazivMetode1()
    {
        // implementacija
    }
    function nazivMetodeN()
    {
        // implementacija
    }
}
```

Klasa koja implementira *interface* mora imati implementirane sve funkcije iz *interface*-a ili klasa mora biti deklarirana kao *abstract*.

Primjer 1.

```
interface Istudent
{
    function uciMatematika();
    function poloziIspitMatematika();
}
interface IstudentRAC
{
    function uciPHP();
    function poloziIspitPHP();
}
class Student {
    private $ime;
    private $godina;
    public function setIme($ime) {
        $this->ime=$ime;
    }
    public function getIme() {
        return $this->ime;
    }
    public function setGodina($godina) {
        $this->godina=$godina;
    }
    public function getGodina() {
    return $this->godina;
    }
}
```

```
class SRacunarstva extends Student implements Istudent,
IstudentRAC {
   private $ocjenaPHP;
   function uciMatematika(){
      echo "UCI za ispit matematika na Računarstvu!";
   }
   function poloziIspitMatematika(){
      echo "Prijavi ispit na Računarstvu i položi ga";
   }
   function uciPHP(){
      echo "Napravi stranicu u PHP-u!";
   }
   function poloziIspitPHP(){
      echo "Predaj stranicu i kolokviraj!";
   }
}
```

10.3.5. Apstraktne (eng. *abstract*) klase i metode

Apstraktne klase i metode moguće je definirati od PHP verzije 5. Nisu namijenjene da bi se instancirale već da budu naslijeđene od drugih klasa. Ukoliko neka klasa ima apstraktne metode i sama klasa mora biti apstraktna. Metode deklarirane kao apstraktne određuju opis metode a ne implementaciju.

Kada se nasljeđuje apstraktna klasa potrebno je definirati sve metode koje su označene kao apstraktne. One moraju biti deklarirane s istom ili manje ograničenom vidljivošću (pr. ukoliko je apstraktna metoda zadana kao *protected*, ona može biti u *child* klasi *protected* ili *public* a ne može biti *private*). Implementacija apstraktne metode u *child* klasi mora imati isti broj potrebnih parametara i tipovi podataka se moraju podudarati. Ukoliko implementacija u *child* klasi zahtjeva više parametara oni moraju biti zadani kao opcionalni.

Primjer 1.

```
abstract class Class_Name {
   // umetanje definicija svojstava
   // umetanje definicija metoda
}
```

10.3.6. Pomoćne funkcije za rad s objektima i klasama

bool class_exists (string $class_name [, bool $autoload = TRUE]))

Utvrđivanje da li neka klasa postoji. Opcionalni parametar određuje da li je potrebno pozvati *autoload* funkciju.

string get_class ([object $object])

Vraća ime klase kojoj objekt pripada.

array get_class_methods (mixed $class_name)

Vraća nazive svih metoda koje su deklarirane u klasi.

array get_class_vars (string $class_name)

Vraća asocijativno polje varijabli i njihovih vrijednosti, ukoliko je moguće iz dosega.

array get_declared_classes (void)

Vraća polje deklariranih klasa.

array get_object_vars (object $object)

Funkcija vraća asocijativno polje s definiranim svojstvima i njihovim vrijednostima ovisno o dosegu.

string get_parent_class ([mixed $object])

Funkcija vraća naziv *parent* klase objekta a ako ona ne postoji vraća FALSE.

bool is_a (object $object , string $class_name [, bool $allow_string = FALSE ***])***

Ukoliko je objekt povezan s klasom (instanca) ili pripada klasi koja je *child* od *class_name* vraća TRUE.

bool is_subclass_of (mixed $object , string $class_name [, bool $allow_string = TRUE])

Vraća TRUE ukoliko objekt pripada klasi koja je *child* klase *class_name* ili ju implementira.

bool method_exists (mixed $object , string $method_name)

Provjerava da li je metoda method_name dostupna objektu.

11. RAD S KORISNIČKIM UNOSOM - FORME

11.1. IZRADA FORMI U HTML-U

Forme se zapisuju u HTML kôdu i služe za sakupljanje podataka unesenih od strane korisnika. PHP će kao poslužiteljska komponenta obraditi prikupljene podatke.

Forme se zadaju preko *form* elementa u HTML-u i za PHP je bitno koja metoda je navedena pod *method* atributom. Moguće je koristiti GET ili POST metodu slanja podataka. GET metoda je osnovna metoda za forme i koristi URL za slanje vrijednosti. POST se češće koristi u kombinaciji s PHP-om jer može raditi s većom količinom informacija.

Forma se u HTML-u zadaje kao:

```
<form action="" method="post/get"></form>
```

Atribut *action* određuje koja će se stranica otvoriti nakon predavanja forme. To može biti ista ili neka druga stranica. Ostali atributi koji se mogu koristiti na *form* elementu su:

atribut	značenje
accept-charset	Specificira charset pri predavanju forme (zadano: charset stranice).
autocomplete	Specificira da li preglednik treba automatski dovršiti formu (zadano: on).
enctype	Specificira encoding predanih podataka (zadano: url-encoded).
name	Naziv forme preko koje se forma identificira (DOM : document.forms.name).
novalidate	Specificira da preglednik ne treba provjeravati formu
target	Specificira gdje će se otvoriti stranica nakon predavanja forme (zadano: _self).

Input elementi

Input elementi omogućuju korisnički unos u formu. Osnovna sintaksa *input* elementa je:

```
<input type="" name="" value=""></input>
```

Value određuje unaprijed zadanu vrijednost, *name* je naziv elementa koji će se koristiti u PHP-u za dohvaćanje podataka a *type* određuje tip *input* polja. Moguće je zadavati sljedeće tipove:

Type atribut	Opis:
text	Jednolinijski unos
password	Maskirani jednolinijski unos
submit	Predavanje forme
radio	Radio gumb (ukoliko je *name* isti povezuju se)
checkbox	Potvrdni okvir
button	gumb
color	boja/ovisno o pregledniku prikazuje se *color picker*
date	datum/ovisno o pregledniku prikazuje se *date picker*
datetime	Datum i vrijeme/ovisno o pregledniku prikazuje se *date picker* s odabirom vremenske zone

Type atribut	Opis:
datetime-local	Datum i vrijeme/ovisno o pregledniku prikazuje se *date picker* bez vremenske zone
email	Unošenje valjanih e-mail adresa
month	mjesec/ovisno o pregledniku prikazuje se *date picker* (mjesec/godina)
number	Unošenje brojeva
range	Vrijednost unutar raspona (zadaje se min i max)
search	Polja za pretragu (isto kao *text*)
tel	Za unošenje telefonskih brojeva
time	vrijeme/ovisno o pregledniku prikazuje se *time picker*
url	Za unošenje URL adresa
week	tjedan/ovisno o pregledniku prikazuje se *date picker* (tjedan/godina)

Osim navedenih atributa moguće je zadavati i sljedeće *input* restrikcije:

atribut	Opis
disabled	Polje je onemogućeno
max	Maksimalna vrijednost za *input* polje (npr. kod *number*)
maxlength	Specificiranje maksimalnog broj znakova
min	Specificira minimalnu vrijednost za polje
pattern	Specificira regularni izraz prema kojem se validira polje
readonly	Specificira da je polje samo za čitanje
required	Specificira da je polje obavezno potrebno ispuniti
size	Specificira širinu (u znakovima)
step	Specificira dozvoljene intervale za *input* polje

Mogu se koristiti i sljedeći HTML 5 atributi:

Atribut	Opis
autocomplete	Automatsko umetanje vrijednosti prema prethodnim unosima
autofocus	Element dobiva fokus pri učitavanju stranice
form	Opisuje kojoj formi element pripada; vrijednost mu je id forme
formaction	Mijenjanje *action* atributa *form* elementa preko *submit* ili *image* elementa
formenctype	Samo za *post* forme, mijenja *enctype* atribut *form* elementa, koristi se na *submit* i image *input* elementima
formmethod	Mijenjanje *method* atributa form elementa preko *submit* ili *image* elementa
formnovalidate	Atribut za *form*; podaci iz forme se ne validiraju prilikom predavanja
formtarget	Mijenjanje *target* atributa *form* elementa preko *submit* ili *image* elementa
height and width	Širina i visina *input* image elementa
list	Povezivanje *input* elementa sa *datalist* elementom
multiple	Oznaka da korisnik može unositi više vrijednosti u *input* element
placeholder	Prikaz kako se polje ispunjava (*Hint*)

Padajuća lista <select>

```
<select multiple>
  <optgroup label=„Voće” >
    <option value=„jabuka”>Jabuka</option>
    <option value=„kruska” selected>Kruška</option>
  </optgroup>
  <optgroup label=„Povrće” >
    <option value=„mrkva”>Mrkva</option>
    <option value=„celer” selected>Celer</option>
  </optgroup>
</select>
```

Ostali elementi

Naziv elementa	Opis
<textarea>	Više linijski unos, atributi: rows, cols
<label>	Definira oznaku za <input> element
<fieldset>	Grupira povezane elemente u formi
<legend>	Definira naslov <fieldset> elementa
<button>	Definira gumb
<datalist>	Lista predefiniranih input opcija za unošenje podataka
<keygen>	Koristi se pri predavanju formi; javni(server) i privatni(lokalni) ključ se generira, koristi se za provjeru korisnika
<output>	Definira rezultat kalkulacije

Primjer 1.

```
<input list="preglednici">
  <datalist id="preglednici">
    <option value="Internet Explorer">
    <option value="Firefox">
    <option value="Chrome">
    <option value="Opera">
    <option value="Safari">
  </datalist>
```

11.2. DOBAVLJANJE KORISNIČKOG UNOSA

Primjer 1.

```
<form action="PHP.php" method="post">
  <label for="email">Email address:</label>
  <input type="text" name="email" size="20" maxlength="50"
  id="email"/>
  <label for="pass">Password:</label>
  <input type="password" name="pswd" size="20"
  maxlength="15" id="pass" />
  <input type="submit" name="subscribe" value="subscribe!"
  />
</form>
```

U primjeru 1. prikazana je HTML forma. Nakon ispunjavanja podataka i predavanja forme pritiskom na gumb *submit* forma poziva datoteku PHP.php (to može biti i ista datoteka) i preko POST metode prosljeđuje podatke.

Nakon predavanja forme podaci su dostupni preko $_POST superglobal varijable. $_POST je asocijativno polje gdje su vrijednosti navedene pod *name* atributom u HTML formi ujedno i ključevi u polje, pa je na navedeni način moguće pristupati korisničkom unosu.

Kako bi nakon predaje forme dohvatili podatke koristimo sintaksu:

```
$_POST['email']
```

gdje je email vrijednost *name* atributa prvog *input* polja u formi.

Korisnički unos može biti nesiguran pa bi uvijek trebalo koristiti primjerice funkciju *htmlentities*() prije spremanja u bazu ili prikazivanja na stranici. Ukoliko se podaci prikupljeni s formom prikazuju na stranici korisnikov unos bi drastično mogao promijeniti izgled stranice (npr. na web forumu). Zbog toga se preporuča korištenje:

```
$email= htmlentities($_POST['email']);
```

Ukoliko se iz $_POST dohvaća vrijednost varijable koja nije inicijalizirana dobivamo pogrešku. To se događa u slučajevima kada forma nije predana. Zbog toga bi za formu uvijek trebalo provjeravati da li je predana preko isset() funkcije, naročito kada se forma i PHP skripta nalaze u istoj datoteci, primjerice:

```
if (isset($_POST['email']))
{
  $pswd = htmlentities($_POST['pswd']);
  $email = htmlentities($_POST['email']);
  printf("Pozdrav %s! <br />", $email);
  printf("Vaša lozinka %s je prihvaćena! <br />", $pswd);
}
```

Primjer 2. Prikazuje formu i PHP kôd koji ju validira postavljen u istoj datoteci.

Primjer 2.

```
<!DOCTYPE HTML>
<html>
<head>
   <title>Korisnici</title>
   <meta charset="UTF-8" />
</head>
<body>
<?php
   function provjeraMaila($email) {
      //regularni izraz za provjeru e-maila
      $regEx="/^([_a-z0-9-]+)(\.[_a-z0-9-]+)*@([a-z0-9-
      ]+)(\.[a-z0-9-]+)*(\.[a-z]{2,6})$/i";
      //provjera ispravnosti
      if (preg_match($regEx, $email)) return 1;
      else return 0;
   }
   //Da li je forma predana?
   if (isset($_POST['predano'])) {
      $ime=htmlentities($_POST['ime']);
      $email=htmlentities($_POST['email']);
      if (provjeraMaila($email))
      echo $ime . " Vaša adresa je ispravna.";
      else
      echo $ime . ", adresa " . $email . " je neispravna";
   }
?>

<form action="vj6.php" method="post">
  <p>Unesite ime:<br />
     <input type="text" name="ime" size="20" maxlength="40"
  />
  </p>
  <p>Unesite e-mail:<br />
     <input type="text" name="email" size="20"
     maxlength="40" />
  </p>
  <input type="submit" name="predano" value="Predaj!" />
</form>
</body>
</html>
```

Rad s komponentama forme

Kad radimo s elementima forme kao *select* ili *input type="checkbox"* omogućen je višestruki odabir. Kad je omogućen višestruki odabir pristup odabranim vrijednostima je preko polja spremljenog kao POST unos.

Primjer 1.

```
Odaberite Vaše najdraže jezike: <br />
<select name="jezici[]" multiple="multiple">
  <option value="c++">C++</option>
  <option value="csharp">C#</option>
  <option value="JavaScript">JavaScript</option>
  <option value="ASP.NET">ASP.NET</option>
  <option value="PHP">PHP</option>
</select>

/*ili
 Odaberite Vaše najdraže jezike: <br />
<input type="checkbox" name="jezici[]" value="c++"/>C++<br
/>
<input type="checkbox" name="jezici[]" value="csharp"/>C#<br
/>
<input type="checkbox" name="jezici[]"
value="JavaScript"/>JavaScript<br />
<input type="checkbox" name="jezici[]"
value="ASP.NET"/>ASP.NET<br />
<input type="checkbox" name="jezici[]" value="PHP"/>PHP<br
/> */

<?php
if (isset($_POST['predano'])) {
  echo "Vaši najdraži jezici su: <br/>";
  foreach ($_POST['jezici'] as $jezik) {
    $jezik=htmlentities($jezik);
    echo "$jezik <br/>";
  }
}
?>
```

11.3. OBJEKTNO ORIJENTIRANI PRISTUP IZRADI I VALIDACIJI FORME

11.3.1. PHP-Bootstrap-Form

Objektno orijentirani pristup izradi i radu s formama omogućen je preko eksternih biblioteka. Jedna od biblioteka koja omogućuje brzu izradu i validaciju formi je PHP-Bootstrap-Form. Navedene biblioteku moguće je preuzeti sa stranice http://smarttechdo.com/~avb/pfbc/.

PHP-Bootstrap-Form nastavak je razvoja PHP-form-builder-class. Kreirana je da radi s Bootstrap 3 i 4, podržava responzivni web dizajn, uključuje AJAX predaju forme, validaciju podataka i dodatne elemente kao što su *tinymce, ckeditor* i dr. Posljednja verzija v4.0 podržava PHP 7.

Tipično se stavlja na root servera tako da bi biblioteka bila dostupna svim skriptama koje ju koriste.

Od preuzetih direktorija i datoteka potreban je samo PFBC direktorij. Na lokalnoj instalaciji servera postavlja se u *root* direktorij npr. *www* ili *htdocs*. Radi lakšeg rada s bibliotekom poželjno je promijeniti konfiguracijsku datoteku php.ini tako da *include_path* uključuje *root* direktorij servera primjerice *c:\wamp\www* ili *c:\xampp\htdocs*. U tom slučaju nije potrebno navoditi putanju pri uključivanju biblioteke u dokument.

Pošto PFBC forma koristi sesije za validaciju podataka obavezno je potrebno na početku samog kôda navesti:

```
session_start();
```

Ukoliko se PFBC datoteke nalaze pod *include_path* za njeno ubacivanje u trenutačni dokument dovoljno je napisati:

```
require_once ("PFBC/Form.php");
```

a u suprotnom potrebna je putanja do datoteke *Form.php.*

Primjer 1.

```
<?php
    session_start();
    require_once ("../PFBC/Form.php");
    Form::open ("primjer");
    Form::Textarea ("Poruka:","poruka");
    Form::Button ("Predaj");
    Form::close (false);
?>
```

PHP-Bootstrap-Form podržava 32 elementa: *Button, Captcha, Checkbox, Checksort, CKEditor, Color, Country, Date, DateTimeLocal, DateTime, Email, File, Hidden, HTML, jQueryUIDate, Month, Number, Password, Phone, Radio, Range, Search, Select, Sort, State,*

Textarea, *Textbox*, *Time*, *TinyMCE*, *Url*, *Week* i *YesNo*. Ukoliko preglednik ne podržava HTML5 elemente oni će se prikazati kao tekstualni elementi.

Form::open()

Form::open() je metoda koja služi za instanciranje nove forme. Ukoliko je na istoj stranici potrebno izraditi više formi moguće je dohvatiti objekt kojeg metoda vraća kako bi referencirali željenu formu. Metoda osim standardnih podržava sljedeće atribute:

Atribut	Tip	Opis
action	String	URL koji se otvara nakon predaje forme. Osnovno ponašanje je ista stranica
method	String	Metoda predaje forme. Osnovno ponašanje je POST
ajax	String	Javascript funkcija koja će se pozvati nakon što je forma predana. Osnovno ponašanje je *null* -- onemogućena *ajax* predaja forme
prevent	Array	Jedina vrijednost koja se može predati je *'focus'* – onemogućuje autofokus na prvom elementu forme
view	String	Način izrade forme. Osnovno ponašanje je *'SideBySide'*. Moguće je postaviti sljedeće tipove: *Inline*, *Search*, *SideBySide*, *SideBySide4* i *Vertical*.
errorView	String	Način prikaza pogrešaka. Osnovno: *Standard*
noLabel	Bool	Ako je postavljeno kao true, sve oznake (*labels*) postaju *placeholder*

```
Form::open ("primjer");
```

Izrađuje formu sa sljedećim atributima:

```
<form action="/putanja_do_trenutne_datoteke.php"
id="primjer" method="post" class="form-horizontal">
```

Prvi parametar je *id* forme, drugi parametar su vrijednosti za polja koja se prosljeđuju formi a treći parametar služi za postavljanje atributa.

Primjer 1.

```
session_start();
require_once ("../PFBC/Form.php");
$atributi=array("method"=>'get', "name"=>"mojaForma");
$vrijednosti=array("poruka"=>"ok");
Form::open ("primjer", $vrijednosti, $atributi);
Form::Textarea ("Poruka:","poruka");
Form::Button ("Predaj");
Form::close (false);
```

Kreira formu sa sljedećim *form* elementom:

```
<form action="/pr/forma.php" id="primjer" method="get"
name="mojaForma" class="form-horizontal">
```

Dok je vrijednost postavljena u *textarea* element:

```
<textarea rows="5" name="poruka" id="poruka" class="form-
control">ok</textarea>.
```

Forma u pregledniku izgleda kako je prikazano na sljedećoj slici:

Slika 7. Forma prikazana unutar web preglednika.

Bootstrap

U prethodnom primjeru nedostaje korištenje Bootstrap-a kako bi se forma adekvatno oblikovala. PHP-bootstrap form dizajniran je da koristi Twitter bootstrap. Bootstrap je besplatan sustav za bržu i jednostavniju izradu web stranica. Bootsrap uključuje HTML i CSS predloške za tipografiju, forme, gumbe, tabele, navigaciju, opcionalne JavaScript dodatke i dr. Bootstrap omogućuje jednostavnu izradu responzivnog dizajna. Trenutno aktualne verzije su 3 i 4. Verzija 4 nema IE8-9 podršku i ukoliko je ona potrebna koristi se verzija 3. Verzija 3 više nema nadogradnji. Verzija 4 ima nove komponente i responzivnija je.

Postoje 2 načina da se Bootstrap uključi na stranicu. Može se povezati preko CDN (eng. Content Delivery Network) ili preuzeti s stranice getbootstrap.com.

Prednost kod korištenja CDN je da su mnogi korisnici već preuzeli Bootstrap 4 s MaxCDN kad su posjećivali druge stranice koje ga koriste. Zbog toga će se Bootstrap uključivati iz *cache* memorije kada se posjeti stranica što pridonosi bržem prikazivanju stranice. Većina CDN će pri zahtjevu za datotekama dostaviti datoteke sa servera koji je najbliže korisniku što također pridonosi bržem vremenu učitavanja.

Bootsrtap 4 koristi jQuery i Popper.js za JavaScript komponente (za modale (dijaloški okviri koji se prikazuju na vrhu stranice), skočne prozore i sl.).

Na stranici https://www.w3schools.com/bootstrap4/ postoji mnogo primjera korištenja bootstrap-a i opisa kako se koristi. Ovdje će biti obrađena samo osnova a za naprednije primjere potrebno je posjetiti navedenu stranicu.

Bootstrap 4 koristi HTML5 pa je dokument potrebno graditi kao HTML5 dokument. Bootstrap 4 je dizajniran kao „*mobile-first*“ pa kako bi se osiguralo ispravno prikazivanje potrebno je postaviti *viewport* u *meta* elementu. Sljedeći primjer moguće je preuzeti sa stranice https://www.w3schools.com/bootstrap4/bootstrap_get_started.asp i koristiti kao predložak za izradu stranica.

Primjer 1.

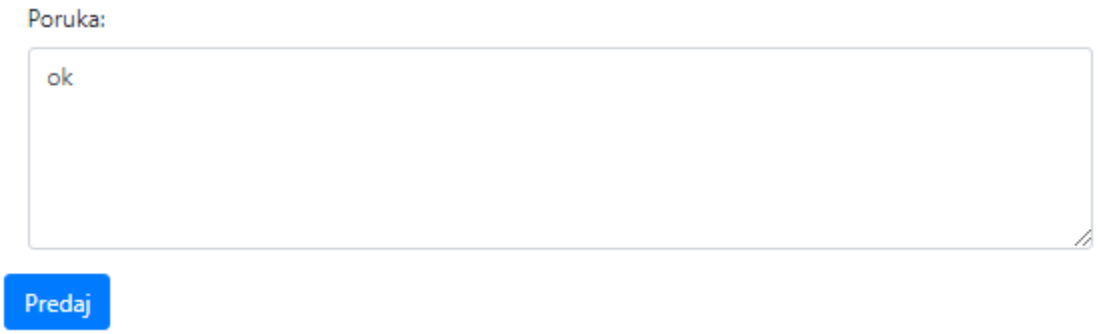
```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Bootstrap 4 Example</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width,
  initial-scale=1">
  <link rel="stylesheet"
  href="https://maxcdn.bootstrapcdn.com/bootstrap/4.1.3/css
  /bootstrap.min.css">
  <script
  src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/j
  query.min.js"></script>
  <script
  src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.1
  4.3/umd/popper.min.js"></script>
  <script
  src="https://maxcdn.bootstrapcdn.com/bootstrap/4.1.3/js/b
  ootstrap.min.js"></script>
</head>
<body>
     <div class="container">
          <h1>My First Bootstrap Page</h1>
          <p>This is some text.</p>
     </div>
</body>
</html>
```

U primjeru je vidljivo postavljanje *meta* elementa *viewport* te uključivanje svih datoteka potrebnih za Bootstrap 4: CSS datoteke i JavaScript datoteke za jQuery, Popper i Bootstrap.

U *body* elementu nalazi se *div* element koji će u sebi sadržavati cijelu stranicu. On ima postavljenu klasu *container* koja omogućava sadržaj fiksne širine. Ukoliko se želi postaviti da se sadržaj prostire cijelom dužinom *viewport* postavlja se vrijednost na *container-fluid*.

Unutar tog *div* elementa nalaziti će se forma koja se izrađuje.

Primjenom navedene konstrukcije stranice forma će izgledati kako je prikazano na sljedećoj slici:



Slika 8. Forma prikazana korištenjem bootstrap

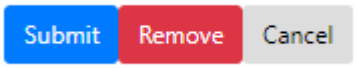
Form::close ()

Form::close() zatvara formu i dovršava ju. Kod zatvaranja forme mogu se koristiti sljedeći parametri:

Parametar	Opis
Form::$SUBMIT	Dodaje submit i cancel gumb. Nakon toga se forma zatvara
Array	Dodaje submit, cancel i ekstra gumbe iz polja
false	Samo zatvara formu bez potrebe dodavanja ekstra gumba

Primjerice:

```
Form::close ();
```




```
Form::close (Form::$SUBMIT);
```

Kako se za zatvaranje forme žele gumbi koji su na hrvatskom jeziku tipično će se koristiti

```
Form::close (false);
```

Gumbi

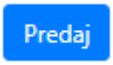
Sintaksa za izradu gumbi je:

```
Form::Button ($value, $type = null, $attributes = null);
```

Prvi parametar je obavezan i određuje *value* atribut *input* elementa, drugi parametar je opcionalan i određuje vrijednost *type* atributa (osnovno je dodavanje *type="submit"* ukoliko nije naveden) a treći opcionalni parametar određuje atribute koji se dodaju na element. Atributi se predaju kao asocijativno polje.

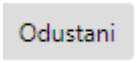
Gumbi se mogu izrađivati na sljedeće načine:

```
Form::Button ("Predaj");
```



Navedena sintaksa izrađuje *submit* gumb. Kad se koristi samo jedan parametar uvijek se izrađuje gumb za predaju forme. Ukoliko se želi izraditi *reset button* pisalo bi se:

```
Form::Button ("Odustani", "reset");
```



Drugi parametar u sintaksi određuje tip gumba (*type* atribut *input* elementa). Treći parametar određuje oblikovanje. Može se zapisati oblikovanje prema Bootstrap-u (https://www.w3schools.com/bootstrap4/bootstrap_buttons.asp):

```
Form::Button ("Odustani", "reset", ["class" => "btn-
danger"]);
```

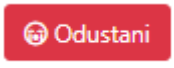
Također se mogu koristiti ikone. Bootstrap 4 podržava sljedeće ikone: Font Awesome 5 i 4, Bootstrap Icons i Google Icons. Kako bi se mogle koristiti potrebno ih je najprije povezati s dokumentom preko link oznake. Za Font Awesome 5 poveznica je:

```
<link rel="stylesheet"
href="https://use.fontawesome.com/releases/v5.5.0/css/all.cs
s" integrity="sha384-
B4dIYHKNBt8Bc12p+WXckhzcICo0wtJAoU8YZTY5qE0Id1GSseTk6S+L3BlX
eVIU" crossorigin="anonymous">
```

Poveznicu je potrebno postaviti u *head* dokumenta. Ostale poveznice mogu se pronaći na stranici: https://www.w3schools.com/icons/default.asp.

Postavljanjem poveznice na ikone omogućuje se korištenje ikona na gumbima. Sve podržane ikone mogu se pogledati na: https://www.w3schools.com/icons/icons_reference.asp.

```
Form::Button ("Odustani", "reset", ["class" => "btn-danger",
"icon"=>"far fa-sad-cry" ]);
```



Treći parametar je polje u kojem su key/value parovi u biti kombinacija atribut/vrijednost (osim u slučaju *icon*). Tako možemo dodati i JavaScript kôd.

```
Form::Button ("Pozdravi", "button",
["onclick"=>"alert('Pozdrav!!')"] );
```

Hidden element

```
Form::Hidden ($nameId, $value = null);
```

U navedenoj sintaksi prvi obavezan parametar određuje *id* i *name* atribut *hidden* elementa. Drugi opcionalan parametar određuje *value* atribut. Primjerice:

```
Form::Hidden ("naziv", "vrijednost polja");
```

izrađuje

```
<input type="hidden" name="naziv" value="vrijednost polja"
id="naziv">
```

Tekstualno polje

```
Form::Textbox ($label, $nameId, $attributes = null);
```

Prva dva parametra su obavezna dok je treći opcionalan. Prvi parametar određuje sadržaj *label* elementa koji se odnosi na *input type=“text“* element. Drugi parametar određuje vrijednost za *for* atribut *label* elementa te *name* i *id* atribut *input* elementa. Treći parametar su dodatni atributi koje se žele postaviti na *input* element.

Primjer 1.

```
Form::Textbox ("Ime: ", "ime", ["placeholder"=>"Unesite vaše
ime"]);
```

Email

```
Form::Email ($label, $nameId, $attributes = null);
```

Password

```
Form::Password ($label, $nameId, $attributes = null);
```

File

```
Form::File ($label, $nameId, $attributes = null);
```

Textarea

```
Form::Textarea ($label, $nameId, $attributes = null);
```

Element za unos brojeva(HTML 5)

```
Form::Number($label, $nameId, $attributes = null);
```

Primjer 1.

```
Form::Number("Broj", "broj", ['min'=>'1', 'value'=>'1']);
```

Phone (HTML 5)

```
Form::Phone ($label, $nameId, $attributes = null);
```

Search (HTML 5)

```
Form::Search ($label, $nameId, $attributes = null);
```

URL (HTML 5)

```
Form::Url ($label, $nameId, $attributes = null);
```

Range (HTML 5)

```
Form::Range ($label, $nameId, $attributes = null);
```

Color (HTML 5)

```
Form::Color ($label, $nameId, $attributes = null);
```

Select

```
Form::Select($label, $nameId, $options, $attributes = null);
```

Select element omogućuje selektiranje iz padajuće liste. Treći obavezan parametar očekuje polje s popisom opcija. Polje koje se prosljeđuje određuje *value* atribut i sadržaj elementa.

Primjer 1.

```
$opcije= array ("1" => "opcija #1", "2" => "opcija #2");
Form::Select("Odaberi: ", "odabir", $opcije,
["multiple"=>"multiple"]);
```

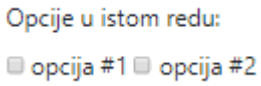
Checkbox

```
Form::Checkbox ($label, $nameId, $options,$attributes =
null);
```

Sintaksa je slična kao u ostalim primjerima jedino se može dodatno postaviti da li se opcije prikazuju u liniji ili jedna ispod druge.

Primjer 1.

```
$opcije= array ("1" => "opcija #1", "2" => "opcija #2");
Form::Checkbox ("Opcije u istom redu: ", "odabir", $opcije,
["inline" => 1]);
```

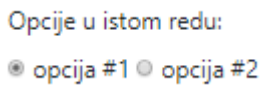


Options

Polje se može podešavati kao u prethodnom primjeru, da li je horizontalno ili vertikalno.

Primjer 1.

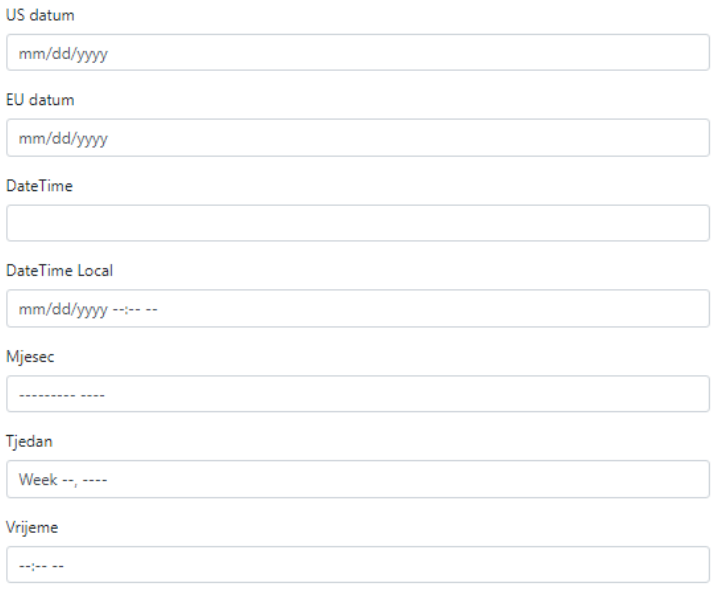
```
$opcije= array ("1" => "opcija #1", "2" => "opcija #2");
Form::Radio ("Opcije u istom redu: ", "odabir", $opcije,
["inline" => 1]);
```



Elementi za datum (HTML 5)

Primjer 1.

```
Form::Date("US datum", "datum", $attributes = null);
Form::Date("EU datum", "datum", array ("pattern" =>
"\d{2}.\d{2}.\d{4}", "placeholder" => "DD.MM.YYYY"));
Form::DateTime("DateTime", "datetime", $attributes = null);
Form::DateTimeLocal("DateTime Local", "DateTimeLocal",
$attributes = null);
Form::Month("Mjesec", "month", $attributes = null);
Form::Week("Tjedan", "week", $attributes = null);
Form::Time("Vrijeme", "time", $attributes = null);
```



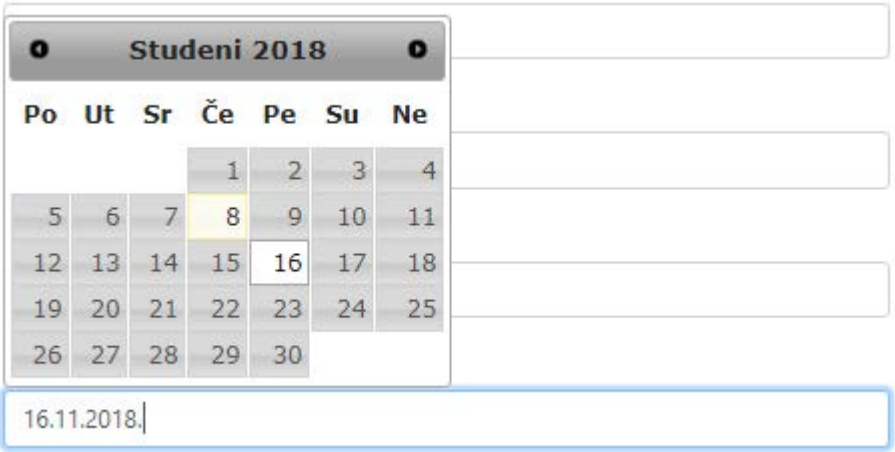
jQuery Datepicker

jQuery datepicker objekt podržava povezivanje s jQueryUI i omogućuje izradu polja za unos s kalendarom. Lokalizacijska datoteka za *datepicker* može se preuzeti sa: https://github.com/jquery/jquery-ui/tree/master/ui/i18n. Ona će omogućiti nazive na željenom jeziku. Potrebno ju je uključiti u datoteku. Lokalizacijsku datoteku je potrebno uključiti u dokument nakon forme kada je *datepicker* objekt inicijaliziran. To je iz razloga jer PHP-Bootstrap-Form povezuje dokument s jQueryUI i tek nakon ubacivanja komponenti, moguće ih je mijenjati s lokalizacijskim datotekama. PHP- Bootstrap-Form omogućuje da se određene postavke iz jQuery biblioteke postavljaju prilikom izrade objekta. Potrebno je navesti jQueryOptions i proslijediti svojstva u obliku polja. jQuery *datepicker* svojstva mogu se pogledati na: https://jqueryui.com/datepicker/.

Primjer 1.

```
<div class="container">
<?php
   session_start();
   require_once ("PFBC/Form.php");
   Form::open ("primjer");
   Form::jQueryUIDate("Date", "jQueryUIDate",
      ["jQueryOptions" => [
         "dateFormat"=> "dd.mm.yy.",
         "firstDay"=>"1"
      ]]);
   Form::Button ("Predaj");
   Form::close (false);
?>
</div>
<script src="datepicker-hr.js"></script>
```

Navedeni primjer daje *datepicker* kako je prikazano na slici:



Odabir zemlje

```
Form::Country ($label, $nameId, $attributes = null);
```

Captcha

Captcha je vrsta autentikacije "izazov-odgovor" koji se koristi u računarstvu da bi odredilo je li korisnik čovjek ili računalo, s ciljem sprječavanja pristupa zlonamjernim računalnim programima.

```
Form::Captcha($label, $attributes = null);
```

U PFBC-bootstrap-form za sada se koristi zastarjela verzija *captcha* koja se ne prikazuje ispravno. To se može ispraviti zamjenom datoteke u PFBC/Resources/recaptchalib.php s datotekom na sljedećoj poveznici:

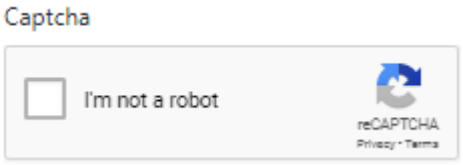
https://github.com/sperti/recaptchav2phplib/blob/master/recaptchalibv2.php

Datoteku je potrebno preimenovati u isti naziv koji se koristi u PFBC, dakle preuzeta datoteka treba se zvati recaptchalib.php.

Potrebno je zamijeniti i ključeve za pristup *recaptcha*. Novi ključevi mogu se dobiti na https://www.google.com/recaptcha/admin. Kako se stranica razvija pod *localhost* pod domenu je potrebno napisati isto.

Ključeve je potrebno unijeti u datoteku *Captcha.php* koja se nalazi pod *PFBC/Elements/Captcha.php. $privateKey* je *Secret key* dobiven od strane Google dok je *Site key $publicKey*.

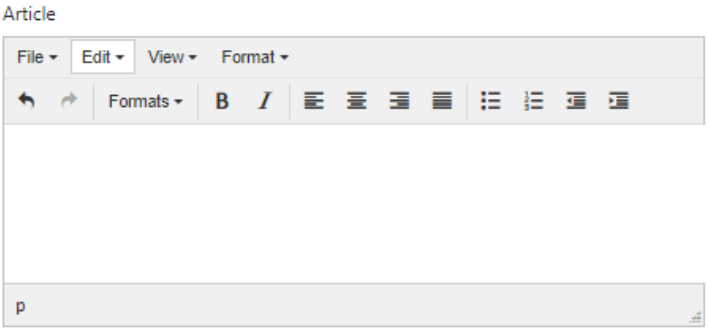
Nakon podešavanja na stranici se prikazuje:



Dodatno je još potrebno ispraviti datoteku za validaciju *Captcha*.

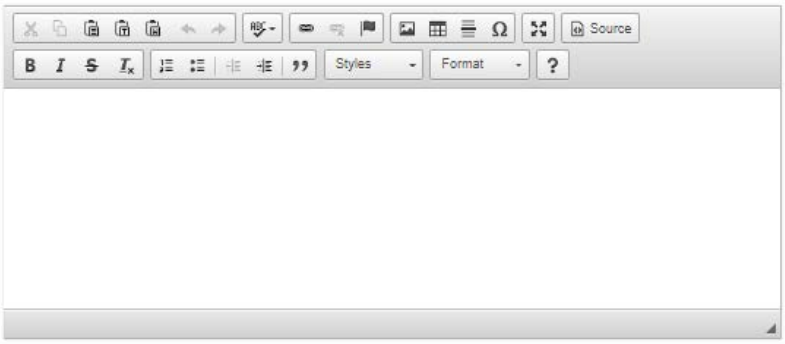
TinyMCE Editor

Form::TinyMCE($label, $nameId, $attributes = null);



CKEditor

Form::CKEditor($label, $nameId, $attributes = null);

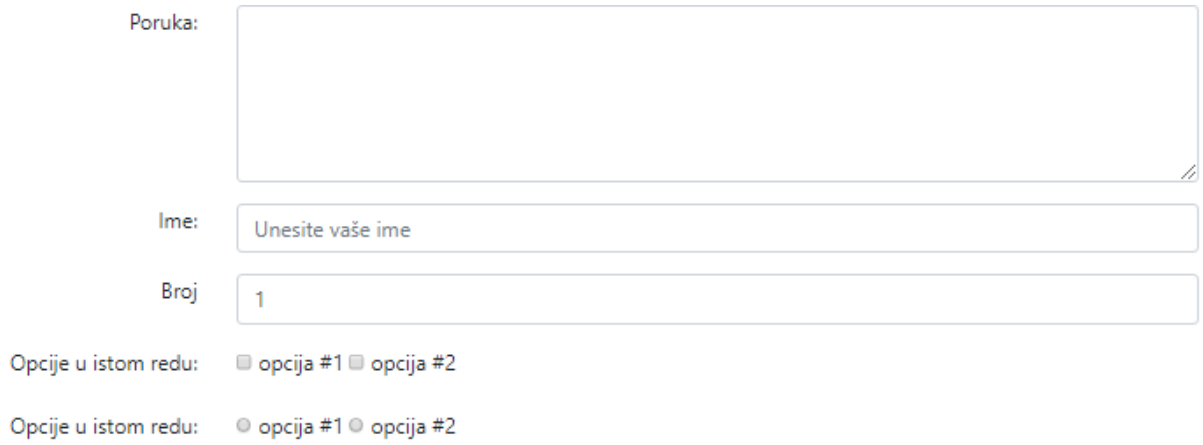


Promjena izgleda forme

Izgled forme (eng. *View*) određuje kako će forma izgledati. Postoje 4 opcije: *SideBySide, SideBySide4 (boostrap4), Vertical* i *Inline*. *SideBySide* je osnovni izgled forme.

Primjer 1.

```
$atributi=array("view" => "SideBySide4");
  Form::open ("primjer", '', $atributi);
```



11.3.2. Validacija podataka u PHP-Bootstrap-Form

Validacija u PHP-Bootstrap-Form ostvaruje se u 2 koraka. Radi se klijentska i serverska validacija. Neki elementi kao *Captcha, Color, Date, Email, jQueryUIDate, Month, Number, Url* i *Week* imaju validaciju koja se postavlja kao osnovno ponašanje elementa. Nije potrebno zadavati dodatna pravila.

PHP-Bootstrap-Form podržava 7 tipova validacije: *AlphaNumeric, Captcha, Date, Email, Numeric, RegExp, Required* i *Url*.

Validacija podataka na serveru

Validacija na serverskoj strani pokreće se pozivanjem statične metode:

```
Form::isValid("naziv");
```

Parametar koji se navodi je naziv forme zadan u

```
Form::open("naziv");
```

isValid() metoda podržava i drugi opcionalni parametar koji sprječava brisanje podataka iz forme ukoliko se podaci trebaju dodatno provjeravati. Ukoliko se postavi na *false*, podaci se ne brišu iako je forma ispravno validirana s *isValid()*.

```
Form::isValid("naziv", false);
```

Obavezni elementi

Obavezni elementi se zadaju pri definiranju forme. Za obavezne elemente moguće je raditi validaciju u HTML5 i na serverskoj strani.

```
Form::Textbox ("Prezime: ", "prezime", ["required"=>1]);
```

Svaki element se može na navedeni način zadati kao obavezan element. U prikazu dobiva crvenu zvjezdicu.

Validacija s regularnim izrazima

PHP-Bootstrap-Form podržava automatsku validaciju prema regularnom izrazu. Sintaksa je:

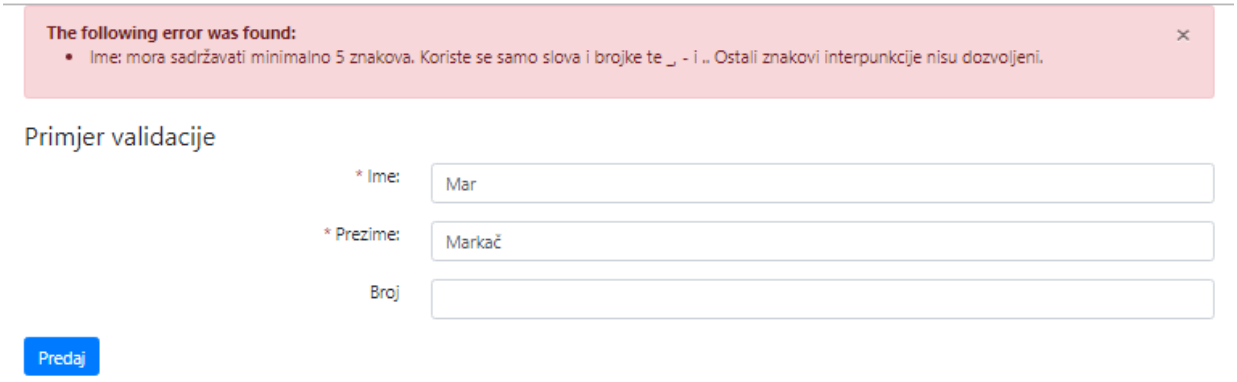
```
Form::Textbox ("Ime: ", "ime", [
"placeholder"=>"Unesite vaše ime",
"required"=>1,
"validation" => new Validation_RegExp(
     "/^[a-zA-Z0-9_\.\-]{5,50}$/",
     "%element% mora sadržavati minimalno 5 znakova.
     Koriste se samo slova i brojke te _, - i ..
     Ostali znakovi interpunkcije nisu dozvoljeni.")
     ]);
```

Koristi se *Validation_RegExp* objekt kojem se prosljeđuje uzorak za validaciju i poruka koja će se ispisati ukoliko je validacija neuspješna. *%element%* se mijenja s prvim parametrom objekta koji se kreira tj. s *label* atributom.

Primjer 1.

```
<?php
  //validacija
    session_start();
    require_once ("../PFBC/Form.php");
    Form::isValid('primjer',false);
?>

<div class="container">
  <?php
  $atributi=array("view" => "SideBySide4");
   Form::open ("primjer", '', $atributi);
   echo "<legend>Primjer validacije</legend>";
   Form::Textbox ("Ime: ", "ime", [
   "placeholder"=>"Unesite vaše ime",
   "required"=>1,
   "validation" => new Validation_RegExp(
   "/^[a-zA-Z0-9_\.\-]{5,50}$/",
   "%element% mora sadržavati minimalno 5 znakova.
   Koriste se samo slova i brojke te _, - i ..
   Ostali znakovi interpunkcije nisu dozvoljeni.")
   ]);
   Form::Textbox ("Prezime: ", "prezime", ["required"=>1]);
   Form::Number("Broj", "broj", ['min'=>'1']);
   Form::Button ("Predaj");
   Form::close (false);
?>
```



Ponekad je poželjno da se već pri unošenju podataka objasni korisniku što se očekuje od njega. To je omogućeno s *longDesc* atributom. Primjerice:

```
Form::Textbox ("Prezime: ", "prezime", ["required"=>1,
"longDesc"=>"Opis unosa"]);
```

Dodatna validacija

PFBC-Bootstrap-Form omogućuje dodatnu validaciju ili rad s podacima forme kao što je recimo zapisivanje u bazu podataka.

Primjer 1.

```
if(Form::isValid("naziv", false)) {
     //ukoliko je forma validirana
     //dodatna validacija
  if(valjaniKor($_POST["Email"], $_POST["Password"])) {
     //imaginarna funkcija za validaciju vratila je
     //true pa je moguće izbrisati podatke u formi
     Form::clearValues("naziv");
     //moguće je izraditi i dodatne korake kao npr.
     //zapisivanje u bazu podataka
     header("Location: nekaStr.php");
  } else {
     //imaginarna funkcija za validaciju vratila je
     //false pa se ispisuje pogreška
     Form::setError("naziv", "Greška: Neispravni Email/
     Password");
     header("Location: ponovi.php");
  }
} else {
     //ukoliko forma nije dobro validirana
     header("Location: ponovi.php");
}
```

Dodatno je moguće brisati pogreške sa:

```
Form::clearErrors('naziv');
```

11.4. BOOTSTRAP CONFIRMATION

Bootstrap confirmation omogućuje izradu potvrdnih polja za akcije koje se žele izvršiti. Tipično se koristi kako bi se provjerilo da li korisnik nije slučajno pritisnuo na neki link ili gumb, naročito ukoliko pokreću akciju koje mijenja podatke primjerice u bazi podataka.

Bootstrap confirmation se može preuzeti sa http://bootstrap-confirmation.js.org/.

Kako bi se koristio potrebno je preuzeti paket ili se povezati preko cdn:

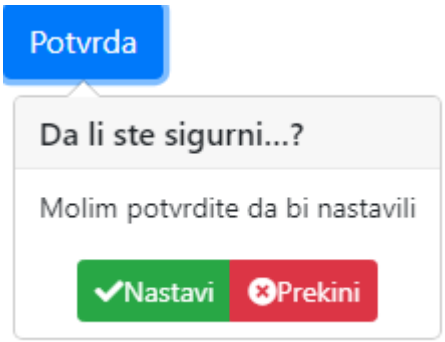
https://cdn.jsdelivr.net/npm/bootstrap-confirmation2@4.0.2/dist/bootstrap-confirmation.min.js.

Koristi jQuery, Popper.js (za Bootstrap 4) i Bootstrap 3 ili 4 pa je i njih potrebno uključiti u datoteku koja će koristiti potvrdne okvire. Primjeri korištenja i moguće opcije nalaze se na stranici dodatka.

Može se dodavati na gumb ili poveznicu te ga je moguće personalizirati.

Primjer 1.

```
<button class="btn btn-large btn-primary"
  data-toggle="potvrda"
  data-btn-ok-label="Nastavi"
  data-btn-ok-class="btn-success"
  data-btn-ok-icon-class="fa fa-check"
  data-btn-cancel-label="Prekini"
  data-btn-cancel-class="btn-danger"
  data-btn-cancel-icon-class="fas fa-times-circle"
  data-title="Da li ste sigurni…?"
  data-content="Molim potvrdite da bi nastavili">
  Potvrda
</button>
```



Primjer izrađuje gumb ovisno o vrijednostima atributa koje se zadaju. *Class* određuje kako se gumb prikazuje što je određeno s uključenim Bootstrap, dok sadržaj elementa *button* određuje tekst koji će se prikazati na njemu. *data-toggle* određuje da se radi o skočnom prozoru potvrde te je potrebno napisati jQuery kôd koji će omogućiti akcije potvrde za dugme. Bez dodatka jQuery gumb neće prikazivati potvrdni okvir. *data-btn-ok-label* određuje tekstualni sadržaj za potvrdno gumb, *data-btn-ok-class* izgled i boju, *data-btn-ok-icon-class* poveznicu na ikonu koja se pokazuje iz Font Awesome 5 kojeg je potrebno povezati kako bi se ikone prikazivale (korišten je

Bootstrap 4.). Isto vrijedi za prekini (*cancel*) gumb. *data-title* određuje naslovni tekst a *data-content* sadržaj okvira.

Kako bi gumb prikazivalo skočni okvir i kako bi se odredila akcija nakon potvrde koristi se jQuery.

Primjer 2.

```
$('[data-toggle=potvrda]').confirmation({
  rootSelector: '[data-toggle=potvrda]',
  onConfirm: function() {
       //akcija koja će se obaviti nakon potvrde
  }
});
```

Bez dodavanja .confirmation funkcije na gumb, ne bi se prikazivao skočni okvir.

Ukoliko bi željeli predati podatke preko potvrdnog okvira recimo s GET metodom mogli bi zapisati:

```
location.href='obrisi.php?id=' + $(this).attr('id');
```

što je JavaScript kôd za promjenu URL-a. U navedenom primjeru otvorila bi se datoteka obrisi.php kojoj bi se proslijedio kao *id* parametar, *id* vrijednost na *button* elementu.

Isti primjer sa predavanjem vrijednosti preko POST metode:

```
$.post(window.location.protocol + "//" + window.location.host
    + "/" + "obrisi.php", {id: $(this).attr('id')});
```

Dodatak omogućuje i zadavanje dodatnih atributa kao primjerice *data-placement* koji se može postaviti na *left, top, bottom* i *right* kako bi se odredilo gdje će se prikazati okvir. *data-singleton* ukoliko se postavi na "*true*" određuje da je moguće imati samo jedan okvir otvoren na stranici i ukoliko se otvori novi okvir bez zatvaranja prošlog okvira, on će se automatski zatvoriti. *data-popout* ukoliko se postavi na "*true*" označava da će se okviri zatvarati ukoliko se klikne bilo gdje na stranici.

Ostale mogućnosti mogu se proučiti na stranici dodatka.

12. MySQL

MySQL je besplatan sustav otvorenog kôda za upravljanje bazom podataka. Uz PostgreSQL MySQL je čest izbor baze za projekte otvorenog kôda, te se distribuira kao sastavni dio serverskih Linux distribucija, no također postoje inačice i za ostale operacijske sustave poput Mac OS-a, Windowse itd. Postoji i mnogo drugih komercijalnih i besplatnih baza podataka. U posljednje vrijeme MariaDB često se isporučuje kao zamjena za MySQL. MariaDB nastala je iz MySQL baze podataka kao potpuno otvoren projekt i iznimno je kompatibilna s MySQL bazom podataka. Razlog njenog nastanka bila je kupnja MySQL od strane Oracle i strah korisnika da će prestati razvoj MySQL. Za daljnji razvoj MySQL zadužen je Oracle a za MariaDB udruženje korisnika.

MySQL baze su relacijskog tipa, relacije se koriste za skladištenje i pretraživanje velikih količina podataka i često čine osnovu svakog informacijskog sustava. Shema baze koja je potrebna prevodi se u tablice koje se koriste za pohranjivanje podataka. Tablice se sastoje od stupaca i redova. Osnovi element koji se pohranjuje u bazi naziva se entitet. U relacijskim bazama podataka odnosi ili relacije između raznih entiteta na odgovarajući način predstavljaju se unutar same baze. Stupci se nazivaju još i poljima ili atributima, a služe za opis podataka o određenom entitetu. Redovi se nazivaju zapisima (eng. *record*) i sadrže sve podatke jednog entiteta.

12.1. PODRŽANI TIPOVI PODATAKA

Tekstualni tipovi podataka

Tip podatka	Najveća veličina	Objašnjenje
CHAR(size)	255 znakova	Size određuje koliko znakova će se spremiti a ne kako će polje biti veliko. String fiksne dužine. Ukoliko je manji broj znakova od zadanog postavljaju se razmaknice s desne strane
VARCHAR(size)	255 znakova	Size određuje koliko znakova će se spremiti. String varijabilne dužine.
TINYTEXT(size)	255 znakova	Size određuje koliko znakova će se spremiti.
TEXT(size)	65.535 znakova	Size određuje koliko znakova će se spremiti.
MEDIUMTEXT(size)	16.777.215 znakova	Size određuje koliko znakova će se spremiti.
LONGTEXT(size)	4GB ili 4.294.967.295 znakova	Size određuje koliko znakova će se spremiti.
BINARY(size)	255 znakova	Size određuje koliko binarnih znakova će se spremiti. String fiksne dužine. Ukoliko je manji broj znakova od zadanog postavljaju se razmaknice s desne strane
VARBINARY(size)	255 znakova	Size određuje koliko binarnih znakova će se spremiti. String varijabilne dužine.

Veliki objekti (eng. LOB - Large Object)

Tip podatka	Najveća veličina	Objašnjenje
TINYBLOB	255 bajtova	Binarni objekt
BLOB(*size*)	65.535 bajtova	Binarni objekt
MEDIUMBLOB	16.777.215 bajtova	Binarni objekt

Numerički tipovi podataka

Tip podataka	Najveća veličina	Objašnjenje
BIT	Vrijednosti s predznakom: -128 do 127. Vrijednosti bez predznaka: 0 do 255.	Mali integer koji je jednak kao i TINYINT(1).
TINYINT(*m*)	Vrijednosti s predznakom: -128 do 127. Vrijednosti bez predznaka: 0 do 255	
SMALLINT(*m*)	Vrijednosti s predznakom: -32768 do 32767. Vrijednosti bez predznaka: 0 do 65535.	
MEDIUMINT(*m*)	Vrijednosti s predznakom: -8388608 do 8388607. Vrijednosti bez predznaka: 0 do 16777215.	
INT(*m*)	Vrijednosti s predznakom: -2147483648 do 2147483647 Vrijednosti bez predznaka: 0 do 4294967295.	
INTEGER(*m*)	Vrijednosti s predznakom: -2147483648 do 2147483647 Vrijednosti bez predznaka: 0 do 4294967295.	Sinonim za INT.
BIGINT(*m*)	Vrijednosti s predznakom: -9223372036854775808 do 9223372036854775807. Vrijednosti bez predznaka: 0 do 18446744073709551615.	
DECIMAL(*m*,*d*)	Broj s decimalnim zarezom ***m*** je 10, ako nije specificiran ***d*** je 0, ako nije specificiran	***m*** je ukupan broj znamenki, ***d*** je broj znamenki nakon decimalnog zareza
DEC(*m*,*d*)	Broj s decimalnim zarezom ***m*** je 10, ako nije specificiran ***d*** je 0, ako nije specificiran	Sinonim za DECIMAL.
NUMERIC(*m*,*d*)	Broj s decimalnim zarezom ***m*** je 10, ako nije specificiran ***d*** je 0, ako nije specificiran	Sinonim za DECIMAL.
FIXED(*m*,*d*)	Broj s decimalnim zarezom ***m*** je 10, ako nije specificiran ***d*** je 0, ako nije specificiran	Sinonim za DECIMAL.

Tip podataka	Najveća veličina	Objašnjenje
FLOAT(*m*,*d*)	Broj s pomičnim zarezom- jednostruka preciznost	***m*** je ukupan broj znamenki, ***d*** je broj znamenki nakon decimalnog zareza
DOUBLE(*m*,*d*)	Broj s pomičnim zarezom- dvostruka preciznost	***m*** je ukupan broj znamenki, ***d*** je broj znamenki nakon decimalnog zareza
DOUBLE PRECISION(*m*,*d*)	Broj s pomičnim zarezom- dvostruka preciznost	Sinonim za DOUBLE.
REAL(*m*,*d*)	Broj s pomičnim zarezom- dvostruka preciznost	Sinonim za DOUBLE.
FLOAT(p)	Broj s pomičnim zarezom	***p*** je preciznost
BOOL	Sinonim za TINYINT(1)	0 je FALSE a ostale vrijednosti su TRUE.
BOOLEAN	Sinonim za TINYINT(1)	0 je FALSE a ostale vrijednosti su TRUE.

Osnovna vrijednost cijelih brojeva uzima se kao vrijednost s predznakom. To se može izmijeniti korištenjem opcije UNSIGNED.

Datum i vrijeme

Tip podataka	Najveća veličina	Objašnjenje
DATE	Od '1000-01-01' do '9999-12-31'.	Prikazuje se kao 'YYYY-MM-DD'.
DATETIME	Od '1000-01-01 00:00:00' do '9999-12-31 23:59:59'.	Prikazuje se kao 'YYYY-MM-DD HH:MM:SS'.
TIMESTAMP(*m*)	Od '1970-01-01 00:00:01' UTC do '2038-01-19 03:14:07' UTC.	Prikazuje se kao 'YYYY-MM-DD HH:MM:SS'. Postavlja se automatski kod INSERT i UPDATE. Podržava i formate: YYYYMMDDHHMISS, YYMMDDHHMISS, YYYYMMDD, or YYMMDD.
TIME	Od '-838:59:59' do '838:59:59'.	Prikazuje se kao 'HH:MM:SS'.
YEAR[(2|4)]	Godina sa 2 ili 4 znamenke	Osnovno ponašanje 4 znamenke

12.2. OGRANIČENJA TIPOVA PODATAKA (ENG. *CONSTRAINT*), DODATNI ATRIBUTI

Ograničenja (eng. *SQL constraints*) vezana uz podatke koji se zapisuju u tabele definiraju pravila za podatke. Oni ograničavaju tipove podataka koja se mogu zapisati u tabelu. Time se dobiva preciznost i pouzdanost zapisanih podataka. Ukoliko postoji povreda pravila između ograničenja i akcije nad podacima, akcija će se obustaviti. Ograničenja mogu biti postavljena na bazi cijele tablice ili pojedinog stupca.

NOT NULL

Onemogućava unošenje NULL u tabelu, čime se osigurava da se osnovne informacije proslјeđuju. Osnovno ponašanje tabele je da prima NULL vrijednosti. NOT NULL će osigurati da polje uvijek ima vrijednost, pa je vrijednost obavezno potrebno proslijediti pri ubacivanju i ažuriranju podatka.

UNIQUE

Osigurava da su sve vrijednosti u stupcu različite. Samo NULL vrijednosti se mogu ponavljati. (Primjerice e-mail vrijednosti za *newsletter*). U tabeli više stupaca može biti postavljeno kao UNIQUE.

PRIMARY KEY

Ne može biti nula, mora imati jedinstvene vrijednosti. Osigurava jedinstvenost retka i identificira svaki zapis u tabeli. Tabela može imati samo jedan primarni ključ koji se sastoji od jednog ili više polja.

FOREIGN KEY

To je ključ koji se koristi kako bi se povezale dvije tablice zajedno. Može biti zadan kao jedno ili više polja a referencira PRIMARY KEY u drugoj tabeli. Ograničenje FOREIGN KEY onemogućuje pogrešne podatke da budu upisani u tabelu, jer to mora biti podatak sadržan u tabeli na koju pokazuje.

CHECK

Ovo ograničenje se koristi kako bi se limitirao raspon podataka koji se ubacuje u stupac. Ukoliko se postavi na stupac, ograničava upise koji se mogu upisati u njega. Ukoliko se postavi na tabelu, pojedini stupci se mogu limitirati prema vrijednostima u drugim stupcima u redcima.

DEFAULT

Određena konstantna vrijednost će biti pridodana kad nije specificirana druga vrijednost.

INDEX

Izrađuje sortirano polje ključeva za određenu tabelu koji ukazuje na određeni redak tabele. Indeksi se koriste kako bi se podaci brzo dohvatili iz baze. Važan je korak u ubrzavanju rada baze. Indeksi se rade samo za stupce koji se često pretražuju.

AUTO_INCREMENT

Najčešće se koristi u stupcu koji se zadaje kao PRIMARY KEY. Time se dobiva jedinstven broj automatski. Samo jedan stupac u tabeli može biti zadan s AUTO_INCREMENT.

BINARY

Koristi se samo kod CHAR i VARCHAR. Koristi se kad želimo da se stupac sortira case-sensitive (ASCII), tj. da se uspoređuje po bajtovima a ne znakovima.

NULL

Oznaka da vrijednost ne postoji u polju. Osnovno ponašanje je da su sva polja postavljena na null.

ZEROFILL

Koristi se za numeričke tipove podataka (dodaje nule ispred brojeva).

12.3. STROJ BAZE PODATAKA (ENG. *STORAGE ENGINE*)

Kada se kreira baza podataka jedna od važnih odluka je odabir stroja baze podataka. Ispravan odabir može biti kritičan što se tiče performansi, naročito kako baza raste. Tipično je u MySQL-u dostupno 10 strojeva (InnoDB, MyISAM, Memory, CSV, Archive, Blackhole, NDB, Merge, Federated i Example) ali se mogu dodati i drugi.

MyISAM

MyISAM je ranije bi osnovni *storage engine*. Nezavisan je o OS-u. Na raspolaganju ima alate za provjeru integriteta tabele i alate za kompresiju. Ne može raditi s transakcijama. Vrlo je čest u izradi web stranica.

Koristi se kod:

- tabela koje imaju puno SELECT i INSERT naredbi
- omogućava istovremene SELECT i INSERT naredbe.

Postoji 3 MyISAM formata:

- ***static***

Static format je najbrži. Često zauzima više prostora (svaki stupac zahtjeva maksimalan prostor, bez obzira da li se taj prostor koristi). Automatski se postavlja ako su svi stupci statični (nema xBLOB, xTEXT ili VARCHAR). Koristi se gdje je moguće.

- ***dinamic***

Zauzima manje prostora ali ima nedostatke u brzini (ako se neka lokacija promijeni, podaci se moraju preseliti na drugo mjesto). *Dinamic* zahtjeva redovitu defragmentaciju (OPTIMAZE TABLE naredba).

- ***compresed***

Koristi se kod tabela koje su samo za čitanje. Zauzima mali memorijski prostor. Za kompresiju koristi se *myisampack utility*.

InnoDB

Stroj koji se najčešće koristi. Koristi transakcijski engine. Koristi se za rad s velikom količinom podataka. Koristi se kod tabela koje imaju puno UPDATE naredbi. Dobar je za više simultanih upisa u bazu. Moguće je automatsko vraćanje stanja nakon pada sustava. Podržava FOREIGN KEY. To je ujedno i osnovni stroj u MySQL bazi podataka.

Uvijek dok je potrebno izrađivati stranice gdje je uključeno plaćanje koristi se transakcijski engine.

MEMORY

Memory ima najbrže vrijeme odaziva. Podaci se spremaju samo u memoriju. Nedostatak je da se podaci gube pri padu sustava. Nije namijenjen za dugotrajno čuvanje podataka. Koristi se kod manjih količina podataka koje se često dohvaćaju.

MERGE (MRG_MYISAM)

Služi za spajanje više *MyISAM* tabela zajedno. Upiti se tada rade na tu tabelu.

FEDERATED

Koristi se za povezivanje na baza na udaljenim serverima. Preko tog stroja kreira se jedna logička baza za više fizičkih servera. Upiti na lokalnom serveru automatski se izvršavaju na udaljenim tabelama. U lokalnim tabelama ne spremaju se podaci.

ARCHIVE

Komprimira podatke koji se rijetko koriste. Ne koristi indekse. Za kompresiju koristi *zlib* biblioteku. Kad zatrebaju podaci se dekomprimiraju.

CSV

Podaci se spremaju odvojeni zarezom, to su u biti tekstualne datoteke. Ne koristi indekse.

EXAMPLE

Služi za izradu vlastitih strojeva.

BLACKHOLE

Podaci koji se tu zapisuju nestaju. Koristi se za testiranja.

12.4. OSNOVNE SQL IZJAVE

SQL je skraćenica za *Structured Query Language* i označava programski jezik namijenjen za upravljanje podacima u relacijskim bazama podataka. SQL obuhvaća zadatke kao što su unos podataka, upiti, ažuriranje i brisanje, kreiranje i mijenjanje te kontrolu pristupa.

SQL je najviše korišten programski jezik za baze podataka. Glavna karakteristika mu je deklarativnost što znači da korisnik određuje što SQL treba napraviti, ali ne i kako doći do rezultata, te nema potrebe poznavati složene aktivnosti koje se događaju kad se unese SQL naredba.

Naredbe u SQL-u se mogu podijeliti u četiri kategorije: DDL(eng. *data definition language*) naredbe, DML(eng. *data manipulation language*) naredbe, DCL (eng. *data control language*) naredbe i TCL(eng. *transaction control language*) naredbe.

DCL naredbe se koriste kod dodjeljivanja dozvola za određene operacije nad bazom podataka. U njih spadaju GRANT koja dodjeljuje pravo, te REVOKE koja oduzima pravo nad određenom operacijom nad bazom podataka.

TCL naredbe se koriste kod upravljanja transakcijama u SQL-u. Transakcija predstavlja skup (obično DML) naredbi koje se izvršavaju u bazi podataka. U TCL naredbe spada COMMIT naredba kojom se spremaju promjene u bazi podataka i ROLLBACK kojom se ukidaju sve promjene od zadnje COMMIT naredbe.

Pri izradi web sadržaja najznačajnije su DML naredbe. DML naredbe se koriste kod kreiranja i brisanja tablica, indeksa i pogleda. Osnovne DML naredbe koje se koriste u SQL-u su CREATE TABLE, CREATE INDEX, CREATE VIEW, ALTER TABLE, DROP TABLE, DROP VIEW i DROP INDEKS. DML naredbe se koriste za dodavanje redaka, izmjenu i brisanje podataka u tablicama, te se najviše koriste u radu s bazama podataka. U DML naredbe spadaju SELECT, INSERT, UPDATE i DELETE.

SELECT

SELECT naredba se koristi za dohvat podataka iz tablica u bazama podataka. Sintaksa joj je sljedeća:

```
SELECT naziv_atributa FROM ime_tablice [WHERE uvjet] [GROUP
BY naziv_atributa][HAVING uvjet] [ORDER BY naziv_atributa
[ASC | DESC]]
```

Sve što je stavljeno u uglate zagrade je opcionalan dio naredbe.

INSERT

INSERT naredba se koristi za dodavanje novih redaka u tablicu. Sintaksa INSERT naredbe izgleda je:

```
INSERT INTO ime_tablice VALUES (vrijednost_1,
vrijednost_2...)
INSERT INTO ime_tablice (atribut_1, atribut_2...) VALUES
(vrijednost_1, vrijednost_2...)
```

UPDATE

UPDATE se koristi za izmjenu podataka u tablici. Sintaksa UPDATE naredbe je sljedeća:

```
UPDATE ime_tablice SET atribut_1=vrijednost_1,
atribut_2=vrijednost_2... WHERE atribut_X=vrijednost_X
```

DELETE

DELETE naredba se koristi za brisanje redaka iz tablice. Sintaksa DELETE naredbe je sljedeća:

```
DELETE FROM ime_tablice WHERE atribut_X=vrijednost_X
```

Ostale naredbe neće biti pokrivene ovdje jer će se većina zadataka obavljati preko *phpMyAdmin* sučelja. Navedene naredbe potrebno je poznavati, jer će se koristiti u PHP kôdu za izvršavanje zadataka nad bazom podataka.

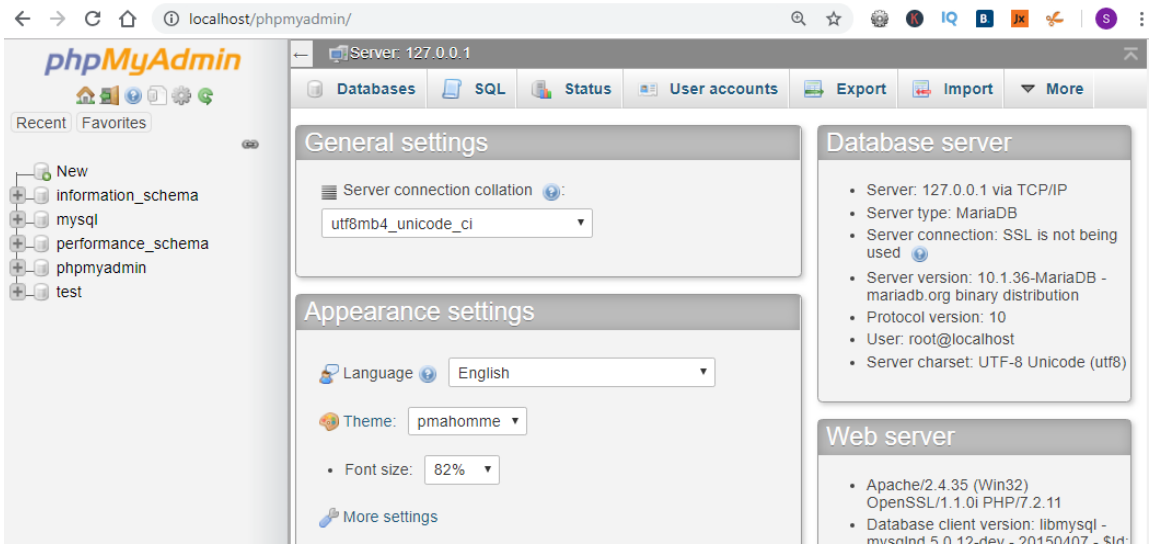
13. PhpMyAdmin

PhpMyAdmin je grafičko korisničko sučelje (eng. *graphical user interface, GUI*) koje se koristi za upravljanje podacima unutar MySQL baze podataka. PhpMyAdmin je besplatan alat pisan u PHP-u, a služi za upravljanje i administraciju MySQL-a preko World Wide Web-a. Uz pomoć njega mogu se izvršavati mnoge MySQL operacije putem korisničkog sučelja izravno u pregledniku.

U phpMyAdmin moguće je upravljati bazama podataka, tablicama, poljima, indeksima, korisnicima, dozvolama pristupa samim bazama, izvršavati upite na bazu i dr.

Na lokalnom poslužitelju pristupa mu se upisivanjem adrese http://localhost/phpmyadmin, (domena je localhost ili 127.0.0.1 i /phpmyadmin).

U osnovnoj instalaciji korisničko ime je *root*, dok lozinke nema. Navedene postavke nisu sigurne pa bi bilo dobro izmijeniti ih tako da *root* korisnik koristi lozinku. *Root* korisnik nije određen samo za phpMyAdmin već je i korisnik koji ima apsolutna prava nad cijelom MySQL bazom. Takav korisnik se može koristiti u lokalnom razvojnom okruženju dok na produkcijskim serverima tipično koristimo korisnike s ograničenim pravima na samo jednu bazu, tablicu ili stupac.



Slika 9. PhpMyAdmin sučelje

Kad otvorimo phpMyAdmin sučelje otvara nam se web stranica kako je prikazano na slici. S lijeve strane su pobrojane baze podataka. Od navedenih sve su potrebne osim test baze pa se ne bi trebale mijenjati. U glavnom izborniku nalaze se stranice preko kojih je moguće upravljati bazama. Ukoliko se želi dodati lozinka za *root* korisnika pod SQL karticom bi zapisali:

```
SET PASSWORD FOR root@localhost = PASSWORD('lozinka');
```

Kako bi phpMyAdmin radio s novom lozinkom potrebno je promijeniti i njegovu konfiguracijsku datoteku. Potrebno ju je otvoriti (config.inc.php -ovisno o okruženju) i izmijeniti sljedeće dvije linije:

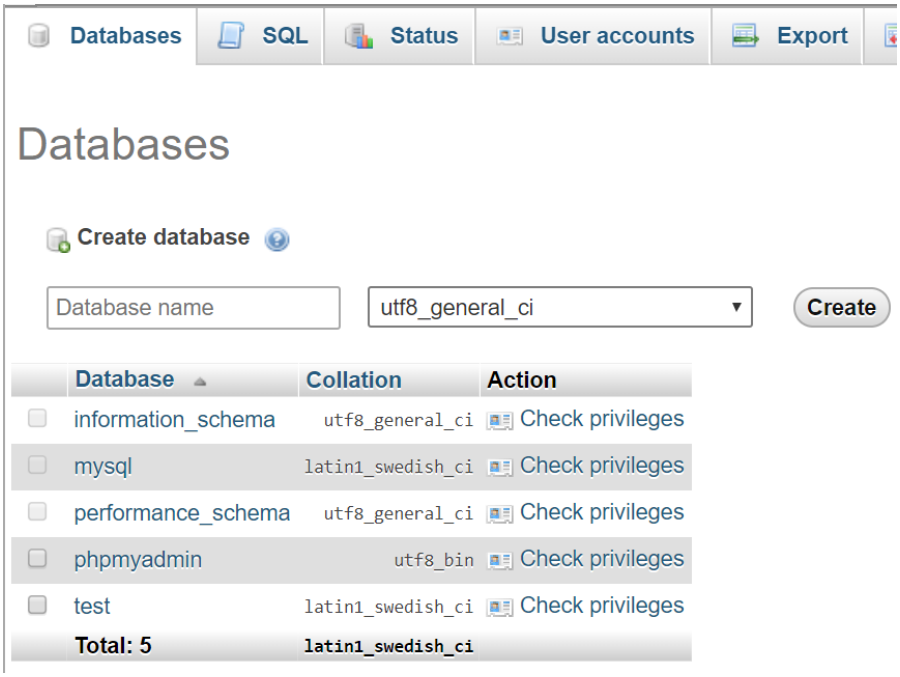
```
$cfg['Servers'][$i]['auth_type'] = 'cookie';
$cfg['Servers'][$i]['AllowNoPassword'] = false;
```

Na taj način root korisnik zahtjeva lozinku.

13.1. IZRADA BAZE PODATAKA

Novu bazu je moguće dodati pod karticom *Databases*. Pod *create database* u polje *database name* upisuje se naziv baze. *Collation* određuje koje će se kodiranje znakova koristiti. Dobra opcija je odabrati kako je zadano *utf8_general_ci* koji će omogućiti ispravno prikazivanje hrvatskih znakova. *Collation* (uspoređivanje) je moguće postaviti i na *utf8_unicode_ci* koji ima opširnija pravila uspoređivanja i bolje sortira i uspoređuje specijalne znakove u različitim jezicima. *utf8_general_ci* je brži jer koristi manje pravila, ali mu je uspoređivanje lošije.

Gumb *create* kreira novu bazu prema unosu.

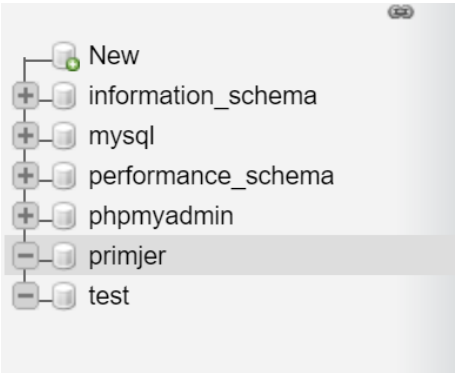


Slika 10. Kartica za dodavanje nove baze

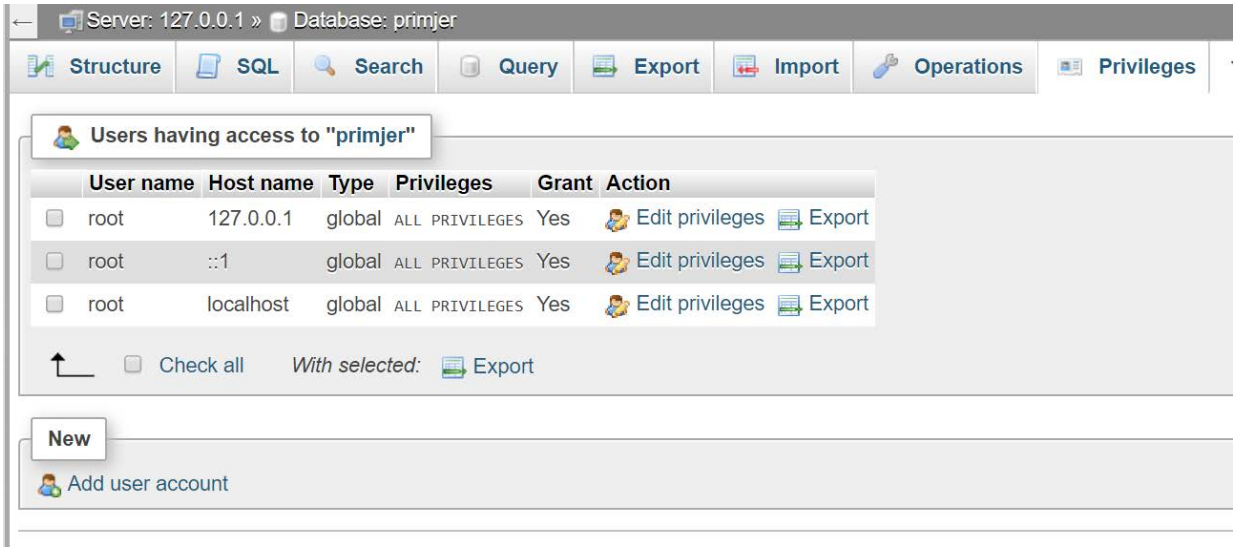
Kad je baza kreirana ona se nalazi pod popisom baza i automatski je selektirana.

13.2. DODAVANJE NOVOG KORISNIKA I ODREĐIVANJE PRIVILEGIJA

Prvi sljedeći korak je izrada korisnika samo za tu bazu. To je moguće pod karticom *Privileges* na selektiranoj bazi.

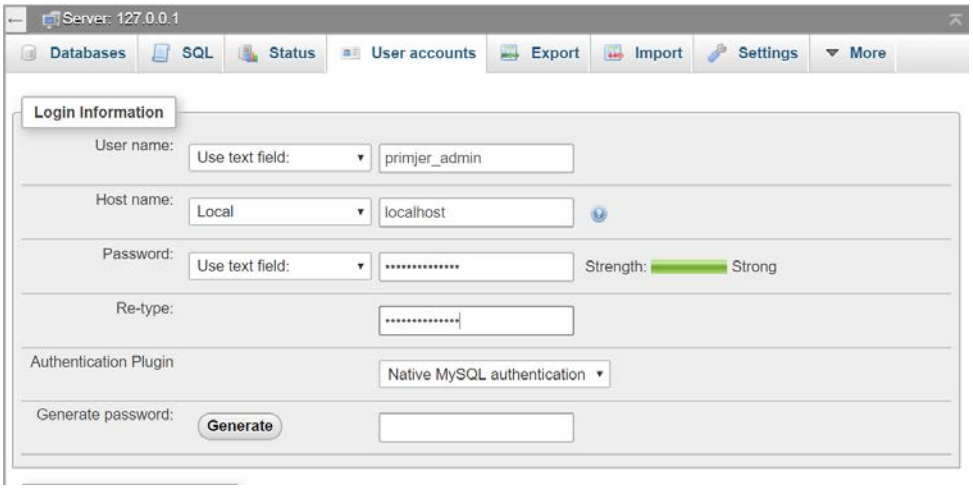


Slika 11. Prikaz nove baze u popisu baza



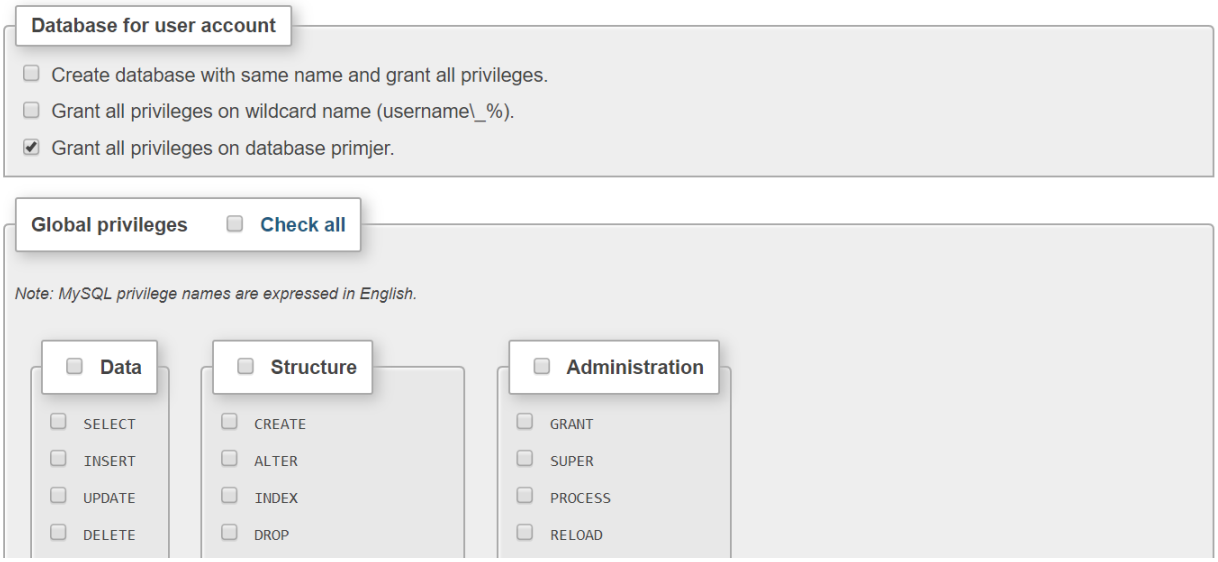
Slika 11. Kartica Privileges selektirane nove baze

Na dnu kartice *Privileges* nalazi se poveznica *Add user account*. Pritiskom na poveznicu dolazi se na *User accounts* gdje je moguće popuniti podatke za novog korisnika. Unosi se korisničko ime (eng. *Username*) i lozinka (eng. *Password*). Potrebno je navesti da se pristupa bazi preko *localhost*.



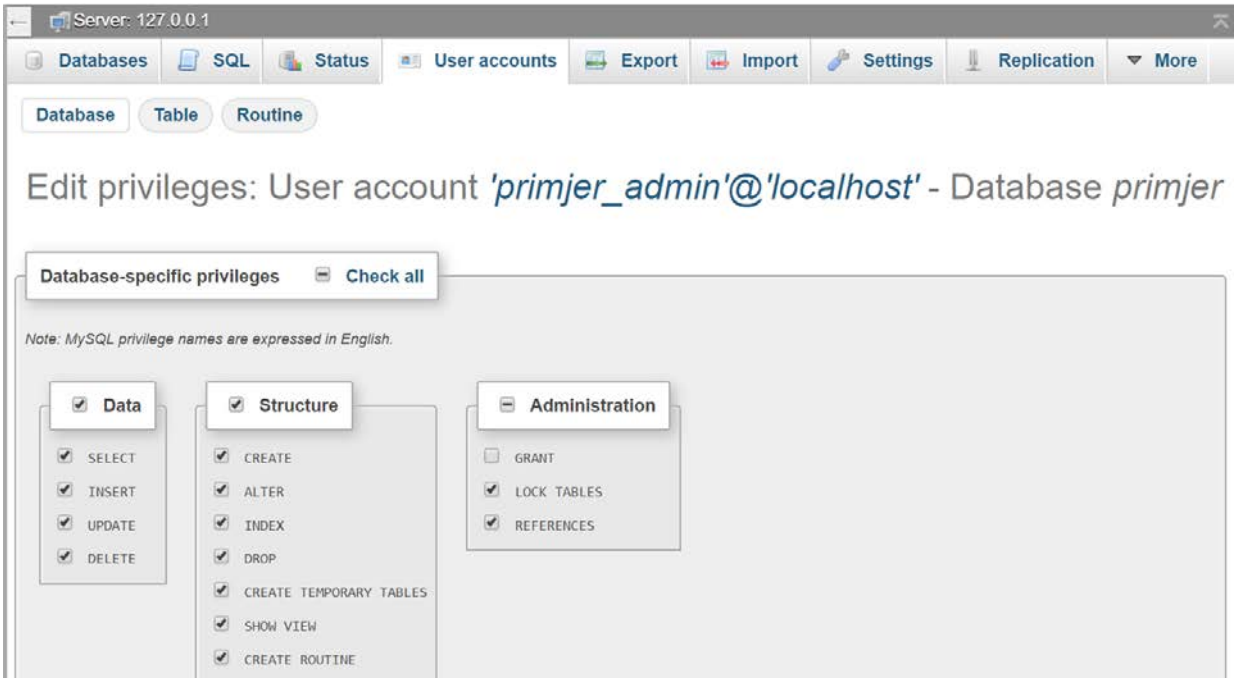
Slika 12. Određivanje korisnika za novu bazu

U sljedećem polju obilježena je treća opcija kojom se korisniku dodjeljuju sva prava nad izrađenom bazom *primjer*. Polje *Global privileges* odnosi se na globalne privilegije nad svim MySQL opcijama. Globalne privilegije neće biti potrebne za korisnika baze.



Slika 13. Određivanje privilegija za novog korisnika

Kad se odabere baza s lijeve strane moguće je pod *Privileges* odabrati stvorenog korisnika i opciju *Edit privileges* i promijeniti privilegije korisnika za tu bazu podataka. Otvara se prozor kako je prikazano na slici.



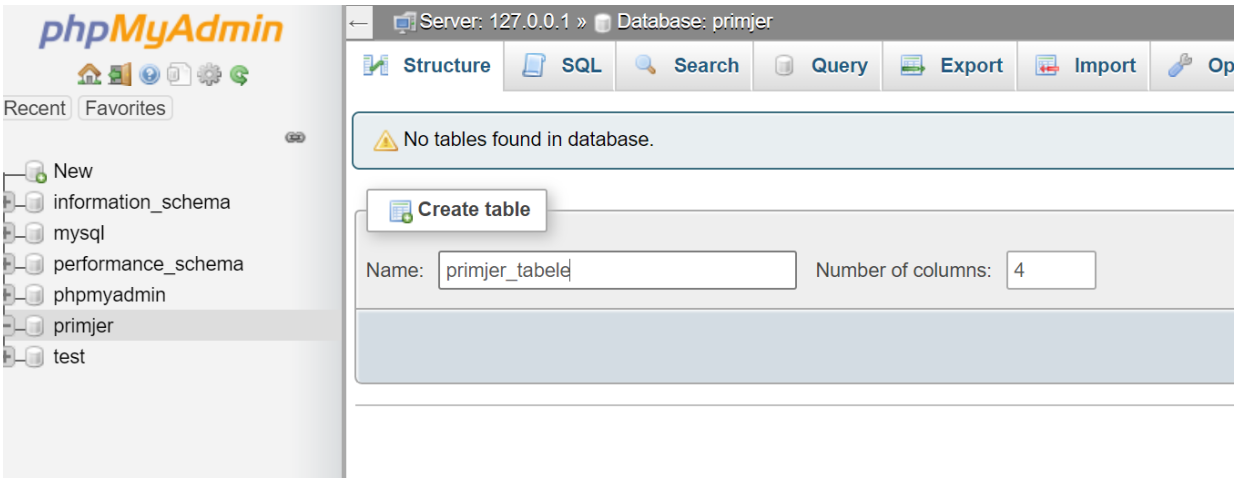
Slika 14. određivanje prava korisnika na novo kreiranoj bazi

Na navedenom izborniku moguće je uređivati prava korisnika na pojedinačnoj bazi. Isto je moguće učiniti kad su kreirane tablice u bazi. Moguće je privilegije dati samo na pojedinačnu tablicu ili stupac u njoj.

Dodjeljivanje privilegija mijenja podatke spremljene u *mysql* bazi. Tamo se mogu i pregledati. Baza *mysql* i tablica *user* određuju popis korisnika s lozinkama i globalnim privilegijama. *mysql.db* određuje prava korisnika na razini pojedinih baza. *mysql.tables_priv* definira prava korisnika na razini pojedinih tabela a *mysql.columns_priv* prava korisnika na razini pojedinih stupaca.

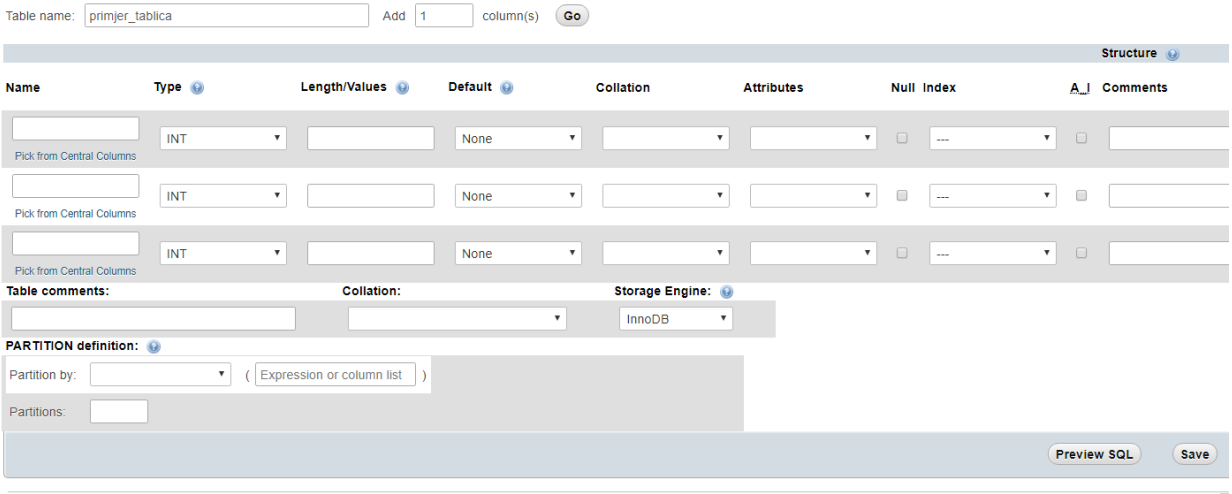
13.3. IZRADA TABLICA UNUTAR BAZE

Odabere se baza i pod karticom *Structure* odabere izrada tabele kojoj se zadaje naziv i broj stupaca.



Slika 15. Izrada tabele u PhpMyAdmin

Kako bi se izradila tablica popunjavaju se polja u formularu koji se otvara.



Slika 16. Izrada stupaca u tabeli baze

14. MySQL i PHP

PHP omogućuje spajanje i rad s bazom podataka. Preduvjet je kreirana baza podataka s tabelama i korisnik koji ima pravo pristupa kreiranoj bazi. Rad s bazom podataka ostvaruje se u 5 koraka:

1. kreiranje konekcije
2. odabir baze
3. izvođenje SQL izjave
4. korištenje vraćenih podataka
5. zatvaranje konekcije.

Ranije se za konekciju s bazom podataka koristio proceduralni način rada koji je u novim verzijama jezika zastario i više se ne koristi od PHP verzije 7. Proceduralni način je koristio metode *mysql_connect(), mysql_select_db(), mysql_query(),* razne metode za rad s podacima i *mysql_close().* Nedostatak navedenog pristupa je sigurnost i podložnost napadima. Ovaj način rada moguć je na mnogim serverima zbog starije verzije PHP-a, ali se ne preporuča. Preporuka je koristiti *MySQLi* ili *PDO_MySQL* ekstenzije pri radu s bazom podataka.

14.1. MYSQLI

MySQLi je objektno orijentirana ekstenzija koja omogućuje rad s bazom podataka. Omogućena je od PHP verzije 5 i MySQL verzije veće od 4.1.3. Prednosti korištenja *mysqli* ekstenzije su: objektno orijentirani pristup, podrška za pripremljene izjave, višestruke izjave, transakcije i dr. Ekstenzija također podržava proceduralni način rada. Ovdje će biti pregled samo objektnog pristupa. Pregled ekstenzije i svih metoda i svojstava nalazi se na:

http://php.net/manual/en/book.mysqli.php.

14.1.1. MySQLi kreiranje konekcije

U *mysqli* konekcija se ostvaruje kreiranjem objekta *mysqli* kojem se kao parametri prosljeđuju naziv ili IP adresa servera (eng. *host*), korisničko ime (eng. *user*), lozinka (eng. *password*) i selekcija baze s kojom se želi raditi (eng. *database*).

Primjer 1.

```
$konekcija= new mysqli
("localhost","korisnik","lozinka","naziv_baze");
//host, user, password, db
```

Svi parametri su opcionalni i moguće je kreirati konekciju bez zadavanja parametara, te ih definirati u sljedećem koraku pomoću metode *connect*.

Primjer 2.

```
$konekcija= new mysqli ();
$konekcija->connect
("localhost","korisnik","lozinka","naziv_baze");
```

Ukoliko se ne zadaju parametri konekcije, konekcija će se pokušati ostvariti sa zapisom iz PHP konfiguracijske (php.ini) datoteke koja se odnosi na sljedeća polja:

```
mysqli.default_host=127.0.0.1
mysqli.default_user=root
mysqli.default_pw=""
mysqli.default_port=3306
mysqli.default_socket=/tmp/mysql.sock
```

Nakon kreiranja konekcije prema bazi podataka zbog hrvatskih znakova uvijek bi trebalo zadati znakovni set. To će se ostvariti korištenjem *set_charset* svojstva:

```
$konekcija->set_charset("utf8");
```

Prilikom kreiranja konekcije odabir baze podataka se može izostaviti pa se baza može odabrati u kasnijem koraku.

14.1.2. Odabir baze podataka

Ukoliko baza podataka s kojom se želi raditi nije definirana u početnom koraku to je moguće napraviti naknadno korištenjem:

```
$konekcija->select_db("naziv_baze");
```

14.1.3. Izvođenje SQL izjava

Query metoda

Izvođenje SQL izjava ostvaruje se preko *query* metode.

Primjer 1.

```
$query="INSERT INTO naziv_tabele (stupac_1, stupac_2) VALUE
('vrijednost1', 'vrijednost2')"
$rezultat=$konekcija->query($query);
```

Sintaksa *query* metode je:

```
mixed mysqli::query ( string $query [, int $resultmode =
MYSQLI_STORE_RESULT ] )
```

Kao opcionalan parametar može se koristiti MYSQLI_USE_RESULT ili MYSQLI_STORE_RESULT. MYSQLI_STORE_RESULT će dohvatiti cijeli rezultat od

MySQL servera dok će MYSQLI_USE_RESULT dohvaćati redak po redak. MYSQLI_USE_RESULT ne prenosi cijeli rezultat iz baze podataka pa se kroz vraćeni set ne može proizvoljno pretraživati.

Metoda vraća FALSE ukoliko izjava nije uspješno izvršena. U slučaju uspješnih SELECT, SHOW, DESCRIBE ili EXPLAIN izjava metoda će vratiti *mysqli_result* objekt. Za ostale uspješne izjave vraća TRUE.

mysqli::$affected_rows

```
int $konekcija->affected_rows;
```

Vraća broj promijenjenih ili dohvaćenih redaka u prethodnoj SELECT, INSERT, UPDATE, REPLACE ili DELETE izjavi.

14.1.4. Izvođenje pripremljenih izjava (*mysqli_stmt* klasa)

MySQL baza podržava izvođenje pripremljenih izjava. Pripremljene izjave omogućuju izvođenje iste izjave s izmijenjenim parametrima s visokom efikasnošću. Često se koriste kao zaštita od SQL napada.

Pripremljena izjava izvodi se u 2 faze: priprema i izvođenje. U fazi pripreme šalje se uzorak izjave koja će se izvoditi a u fazi izvršavanja izjava koje će se izvršiti. Izjava se priprema tako da se na mjesta na koja će doći vrijednosti varijabli označe s ?. Naknadno se šalju vrijednosti koje su potrebne da se izjava izvrši.

Primjer 1.

```
$stmt= $konekcija->prepare('SELECT * FROM naziv_tabele WHERE
ime=?');
$stmt->bind_param('s', $ime);
$stmt->execute();
$rezultat= $stmt->get_result();
```

U primjeru je $*stmt* objekt *mysqli_stmt* klase. Kako bi se dobio rezultat izjava koje čitaju podatke iz baze podataka koristi se *get_result()* metoda *mysqli_stmt* klase koja vraća objekt *mysqli_result* klase.

Metoda *bind_param()* kao prvi argument prima vrijednosti:

- i- ukoliko se prosljeđuje cjelobrojna vrijednost
- d- ukoliko se prosljeđuje decimalna vrijednost
- s- ukoliko se prosljeđuje string
- b – ukoliko se prosljeđuje BLOB.

Pri prosljeđivanju više parametara izjava bi glasila primjerice:

```
$stmt->bind_param('si', $ime, $godine);
```

Ostali argumenti su varijable koje se prosljeđuju. Određivanjem tipova varijabli smanjuje se rizik SQL napada. Ukoliko se prosljeđuju podaci koje je unio korisnik potrebno ih je sanirati i provjeriti kako bi ispravno izvršavali SQL naredbe.

Izvršavanje izjave ostvaruje se preko *execute* metode. Ona će vratiti TRUE ili FALSE ovisno o izvršavanju izjave.

Ukoliko se želi izvoditi više istih izjava s različitim vrijednostima pri promjeni vrijednosti poziva se samo *execute* metoda, za izjave koje su pripremljene i povezani su parametri.

Primjer 2.

```
$stmt= $konekcija->prepare("INSERT INTO korisnici (ime,
prezime, email) VALUES (?, ?, ?)");
$stmt->bind_param("sss", $ime, $prezime, $email);
$ime= "Iva";
$prezime= "Ivić";
$email= "iva@example.com";
$stmt->execute();
$ime= "Marko";
$prezime= "Markač";
$email= "marko@example.com";
$stmt->execute();
```

Ukoliko se u istoj skripti želi izvršavati više izjava prema bazi *prepared statement* se može zatvoriti s *close*(), primjerice:

```
$stmt->close();
```

Kod pripremljenih izjava može se koristiti i *bind_result()* metoda koja će preko *fetch*() metode dobivene vrijednosti postaviti u varijable.

Primjer 3.

```
$stmt= $konekcija->prepare('SELECT prezime, godine FROM
korisnici WHERE ime=?');
$stmt->bind_param('s', $ime);
$stmt->execute();
$stmt->bind_result($prezime, $godine);
while ($stmt->fetch()) {
        echo $prezime . $godine;
}
$stmt->close();
```

14.1.5. Korištenje vraćenih podataka (*mysqli_result* klasa)

Nakon izvršene SQL izjave moguće je raditi s *mysqli_result* objektom u slučaju čitanja podataka iz baze podataka. *Mysqli_result* je rezultat upita u bazu.

Od PHP verzije 5.4 mogu se koristiti iteratori za prolaz kroz rezultate.

Primjer 2.

```
$query="SELECT id, sifra, naziv, opis, kolicina FROM
proizvodi ORDER by id";
foreach ($konekcija->query($query) as $red ) {
  printf("id: %s (%s ) %s: %s <br /> Količina: %s <br /> ",
  $red['id'], $red['sifra'], $red['naziv'], $red['opis'],
  $red['kolicina']);
}
```

array mysqli_result::fetch_assoc (void)

Metoda vraća asocijativno polje koje predstavlja redak iz seta dobivenih rezultata ili NULL ukoliko nema više redaka u setu rezultata. Svako uzastopno pozivanje funkcije vraća jedan redak rezultata. Ključ u polju predstavlja naziv stupca u tabeli i osjetljiv je na mala i velika slova. Vrijednost pod ključem je vrijednost spremljena u stupcu u retku koji se dohvaća. Ukoliko dva ili više stupaca u rezultatu imaju jednak naziv uzima se vrijednost od posljednjeg stupca.

Primjer 1.

```
$query="SELECT id, sifra, naziv, opis, kolicina FROM
proizvodi ORDER by id";
$rezultat=$konekcija->query($query);
while ($red=$rezultat->fetch_assoc()) {
     $id=$red['id'];
     $sifra=$red['sifra'];
     $naziv=$red['naziv'];
     $opis=$red['opis'];
     $kolicina=$red['kolicina'];
     printf("id: %s (%s ) %s: %s <br /> Količina: %s <br />
     ", $id, $sifra, $naziv, $opis, $kolicina);
}
```

mixed mysqli_result::fetch_row (void)

Vraća indeksirano polje gdje je indeks određen redoslijedom stupca u tabeli ako se podaci dohvaćaju s *. Ukoliko se preko upita dohvaćaju pobrojane vrijednosti redoslijed indeksa je određen izrađenim upitom.

Vraća redak iz seta rezultata ili NULL ukoliko ne postoji više redaka u vraćenom setu.

Primjer 1.

```
$query="SELECT id, sifra, naziv, opis, kolicina FROM
proizvodi ORDER by id";
$rezultat=$konekcija->query($query);
while ($red=$rezultat->fetch_row()) {
     printf("id: %s (%s ) %s: %s <br /> Količina: %s <br />
     ", $red[0], $red[1], $red[2], $red[3], $red[4]);
}
```

mixed mysqli_result::fetch_array ([int $resulttype = MYSQLI_BOTH])

Predefinirano dohvaća i indeksirano i asocijativno polje. Vraća NULL kada nema više rezultata. Vraća redak iz seta rezultata ili NULL ukoliko ne postoji više redaka u vraćenom setu.

Kao argument može se proslijediti MYSQLI_ASSOC koji vraća samo asocijativno polje (vraća rezultat kao *fetch_assoc()*). MYSQLI_NUM (vraća rezultat kao *fetch_row()*) vraća indeksirano polje. MYSQLI_BOTH vraća indeksirano i numeričko polje.

Primjer 1.

```
$query="SELECT id, sifra, naziv, opis, kolicina FROM
proizvodi ORDER by id";
$rezultat=$konekcija->query($query);
while ($red=$rezultat->fetch_array()) {
     printf("id: %s (%s ) %s: %s <br /> Količina: %s <br />
     ", $red[0], $red[1], $red[2], $red['opis'],
     $red['kolicina']);
}
```

mixed mysqli_result::fetch_all ([int $resulttype = MYSQLI_NUM])

Metoda dohvaća sve retke iz seta rezultata i može ih vratiti kao numeričko polje (MYSQLI_NUM), asocijativno polje (MYSQLI_ASSOC) ili oboje (MYSQLI_BOTH). Kako metoda vraća sve retke koji su dohvaćeni u jednom koraku može zauzeti više memorije od ostalih metoda. Iz navedenog razloga metoda bi trebala biti korištena samo u situacijama gdje se rezultat prosljeđuje drugom sloju za obradu.

object mysqli_result::fetch_object ([string $class_name = "stdClass" [, array $params]])

Vraća redak rezultata kao objekt gdje su nazivi stupaca svojstva objekta. Ukoliko nema više rezultata vraća NULL. Opcionalni parametar *string $class_name* određuje naziv klase koja se instancira, ukoliko se ne zada vraća se *stdClass* objekt. Drugi opcionalni parametar *array $params* očekuje polje parametara koji se prosljeđuju konstruktorskoj funkciji za *class_name* objekte.

Primjer 1.

```
$query="SELECT id, sifra, naziv, opis, kolicina FROM
proizvodi ORDER by id";
$rezultat=$konekcija->query($query);
while ($red=$rezultat->fetch_object()) {
     printf("id: %s (%s ) %s: %s <br /> Količina: %s <br />
     ", $red->id, $red->sifra, $red->naziv, $red->opis,
     $red->kolicina);
}
```

int $mysqli_result->num_rows;

Vraća broj redaka u rezultatu.

Primjer 1.

```
$query="SELECT id, sifra, naziv, opis, kolicina FROM
proizvodi ORDER by id";
$rezultat=$konekcija->query($query);
$broj=$rezultat->num_rows;
echo ("Pronađen broj redaka: " . $broj);
```

void mysqli_result::free (void)

void mysqli_result::close (void)

void mysqli_result::free_result (void)

Bilo koja od navedenih metoda poziva se ukoliko želimo osloboditi memoriju povezanu s rezultatom, kada rezultat više nije potreban. Koristi se kada se žele izvršavati drugi upiti u istoj skripti.

14.1.6. Rad s pogreškama vezanim uz rad s bazom podataka

Prilikom izrade konekcije ili dohvaćanja podataka može doći do pogrešaka. Pogreške bi se trebale provjeravati ukoliko skripta ovisi o dohvaćenim podacima. *Mysqli* i *mysqli_stmt* klase podržavaju rad s pogreškama. Pogreške se mogu vratiti kao broj pogreške ili opis pogreške.

int mysqli::connect_errno;

Vrača posljednji kôd pogreške u posljednjem pozivu *connect()* metode. Nula označava da nije došlo do pogreške.

string mysqli::connect_error;

Vraća opis posljednje pogreške pri povezivanju s bazom podataka. Vraća se NULL ukoliko nije pronađena pogreška.

Primjer 1.

```
if ($konekcija->connect_errno) {
   die ('Greška sa spajanjem na bazu: ' . $konekcija->
      connect_error);
}
```

int mysqli::errno;

int mysqli_stmt::$errno;

Vrača posljednji kôd pogreške posljednje *mysqli* ili *mysqli_stmt* metode koja je pozvana.

string mysqli::error;

string mysqli_stmt::$error;

Vraća opis posljednje pogreške nastale pri pozivanju *mysqli* ili *mysqli_stmt* metoda.

Primjer 1.

```
if ($konekcija->errno) {
   die ('Greška: ' .  $konekcija->error);
}
```

array mysqli::error_list;

array mysqli_stmt::$error_list;

Vraća asocijativno polje pogrešaka u kojem je zapisan kôd pogreške, opis pogreške i SQLSTATE pogreška nastale pri posljednjem pozivanju *mysqli* ili *mysqli_stmt* metoda.

14.2. IZDVAJANJE PRISTUPNIH PODATKA ZA MYSQL

PHP skripte nisu vidljive preko web preglednika pa su lozinke kojima se pristupa bazi podataka relativno sigurne u PHP-u, osim ukoliko nastaju pogreške servera koje ispisuju sadržaj datoteke, primjerice ukoliko server nije dobro konfiguriran ili dođe do napada koji promijeni serversku konfiguraciju. U tom slučaju datoteke bi se mogle prikazati kao tekst.

Dobra je praksa držati pristupne podatke izvan pojedinačnih datoteka na jednom mjestu kako bi se pristupni podaci lakše izmijenili i kako bi ih bilo lakše zaštititi. Jedan od dobrih načina je izdvajanje konekcije u zasebnu datoteku koja se sprema izvan root direktorija servera.

Primjerice root direktorij na serveru je "/localhost/public_html/". Moguće je kreirati direktorij izvan root direktorija primjerice "/localhost/privatno/". Ta lokacija nije direktno dostupna preko url-a jer se nalazi izvan root direktorija.

Pristupni podaci se ugrađuju u .ini datoteku primjerice config.ini. Sadržaj datoteke bio bi primjerice:

```
[database]
host = localhost
korisnik = root
lozinka = 1234
baza = mojabaza
```

PHP ima dobru podršku za .ini datoteke. Datoteka se ubacuje prilikom izrade konekcije. Primjerice:

```
$konfig = parse_ini_file('../privatno/config.ini');
$konekcija = new mysqli ($konfig['host'],
$konfig['korisnik'], $konfig['lozinka'], $konfig['baza']);
```

Kod kreiranja konekcije na bazu može se koristiti kôd koji je prikazan u sljedećem primjeru.

Primjer 1.

```
function konekcijaDB() {
    static $konekcija;
    if(!isset($konekcija)) {
       $konfig = parse_ini_file('../privatno/config.ini');
```

```
        $konekcija = new mysqli ($konfig['host'],
        $konfig['korisnik'], $konfig['lozinka'],
        $konfig['baza']);
        $konekcija->set_charset('utf8');
    }
    if($konekcija->connect_errno) {
        echo '<div class="alert alert-danger">
        <strong>Nije se moguće spojiti na bazu
        podataka</strong> </div>';
        exit();
        }
    return $konekcija;
}
$konekcija= konekcijaDB();
```

Ukoliko je prethodni primjer spremljen kao dbkonekcija.php kada se želi povezati s bazom koristi se primjerice:

```
require_once('../putanja/dbkonekcija.php');
$query = "SELECT id, ime, prezime FROM korisnici";
$rezultat = $konekcija->query($sql);
```

15. ZAŠTITA WEB STRANICA

Zaštita jedan je od najvažnijih aspekata izrade web stranice. Uključuje korake koji se implementiraju u brojne dijelove same izrade web stranica. Jednom kad je web stranica postavljena na server i u upotrebi također je potrebno raditi na sigurnosti redovitim sigurnosnim kopiranjem stranice i baze podataka, redovitim ažuriranjem platforme, skripti, biblioteka i dr.

Stranicu je potrebno zaštititi od SQL napada (eng. SQL injection) kako je bilo govoreno u prethodnom poglavlju (korištenjem pripremljenih izjava) te je potrebno provjeravati sve unose u formulare i podešavati ih za SQL izjave koje se izvode.

Cross-Site Scripting (XSS) je također jedan od češćih napada. Napad se ostvaruje tako da se na stranicu recimo preko polja za komentare ugrađuje JavaScript kôd koji se izvršava u pregledniku korisnika. Na taj način omogućena je krađa podataka i slanje prema napadaču. Primjerice, ukoliko se komentari prikazuju na stranici bez validacije, tada napadač može predati skriptu koja može ukrasti *cookie* koji se koristi za *login* i time omogućuje napadaču da preuzme kontrolu računa svakog korisnika koji je pregledao komentar. Postoje razni načini kako je moguće obraniti se od ovakvih napada. U PHP-u je obavezna sanacija unosa korisnika korištenjem metoda kao *strip_tags* ili *htmlentities*. Postoje i drugi alati i metode koje mogu osigurati veću razinu zaštite (npr. Content Security Policy (CSP) primjer. https://developer.mozilla.org/en-US/docs/Web/HTTP/CSP).

Validaciju je potrebno raditi na strani klijenta i servera. Preglednik može pronaći jednostavne greške kao recimo obavezna polja ili unos koji nije u željenom formatu. Takva validacija na klijentskoj strani se može zaobići pa je bitno vršiti istu i složeniju validaciju na serveru. Ukoliko se ne radi validacija može doći do ubacivanja zlonamjernog kôda u bazu podataka ili neželjenih rezultata na web stranici.

Pri izradi web stranica potrebno je dati manje podataka o greškama ili iznimkama koje su nastale kako se ne bi odali podaci koji bi omogućili lakše napade na stranicu.

Osim navedenog slijedi pregled dijela zaštite kojeg bi nužno trebalo ugraditi u web stranice. Postoje i druge tehnike koje se koriste pri zaštiti web stranica, a nisu navedene u ovom dijelu.

15.1. ZAŠTITA LOZINKI

Poznato je da je potrebno koristiti kompleksne lozinke, no korisnici to često ne rade. Potrebno je koristiti snažne lozinke prema serveru ili administratorskom dijelu stranice te također prisiliti korisnike stranice da koriste snažne lozinke kako bi osigurali sigurnost vlastitih računa.

Za korisnike računa na nekoj stranici potrebno je zadavati minimalne uvijete pri određivanju lozinki (primjerice; osam znakova, korištenje kombinacije verzala i kurenta). Lozinke se u bazi podataka uvijek trebaju spremati kao kriptirane vrijednosti i to korištenjem jednosmjernih algoritama. Pri korištenju jednosmjernih algoritama uvijek se provjeravaju kriptirane vrijednosti. Radi povećane sigurnosti dobro je koristiti i *salt* izraze, pri čemu se koristi dodatni novi *salt* po lozinci.

Ukoliko napadač uspije doći do lozinki one će biti u kriptiranom obliku. Moguće je pokušati dekriptirati lozinke korištenjem rječnika (primjerice usporedbom spremljenih pojmova s kriptiranim rječnicima) ili *brute force attack* (automatizirani *software* generira velik broj pojmova kako bi došao do željenih podataka), u načelu pogađanjem velikog broja kombinacija dok se ne pronađe podudaranje. Kad se koriste lozinke koje su dodatno zaštićene sa *salt* proces dolaska do originalne lozinke je još dulji jer je potrebno svako pogađanje kombinirati sa *salt* dijelom lozinke.

Za izradu kriptirane lozinke može se koristiti primjerice hash () funkcija:

```
string hash ( string $algo , string $data [, bool
$raw_output = FALSE ] )
```

Prvi argument određuje algoritam koji će se koristiti dok drugi argument određuje podatke koji će se kriptirati. Funkcijom hash_algos() moguće je saznati koji algoritmi su dostupni. Funkcija vraća polje s listom podržanih algoritama kriptiranja.

Algoritmi za kriptiranje kao MD5, SHA1 i SHA256 su dizajnirani da budu brzi i efikasni. Sa modernim tehnikama i računalnom opremom vrlo je lagano izraditi napade. Upravo zbog brzine algoritma preporuka je da se ne koriste za kriptiranje lozinki.

Preporučeni algoritam za kriptiranje lozinki je Blowfish ili bcrypt koji se i koristi kao zadani u *native password hashing API* (http://php.net/manual/en/book.password.php). Metoda je:

```
string password_hash ( string $password , int $algo [, array
$options ] )
```

Trenutno su podržani sljedeći algoritmi (zadaju se kao konstante za parametar $algo):

PASSWORD_DEFAULT - koristi *bcrypt* algoritam (zadano od PHP 5.5.0). Ova će se konstanta mijenjati kroz vrijeme kako se dodaju novi i jači algoritmi u PHP. Iz navedenog razloga dužina rezultata pri korištenju ove konstante se može s vremenom promijeniti pa se preporuča spremati lozinku u polje koje može biti veće od 60 znakova (255 znakova).

PASSWORD_BCRYPT - koristi CRYPT_BLOWFISH algoritam. Time se izrađuje standardno kriptiranje kao i kod *crypt()* metode. Rezultat je uvijek 60 znakova ili FALSE pri neuspjehu. Osnovna vrijednost *cost* je 10 a može varirati od 4 do 31. *Cost* određuje koliko iteracija je potrebno od 2^4 do 2^{31}. Time se postiže potreba za dugim vremenom kriptiranja i velikim hardverskim zahtjevima za kriptiranje što omogućuje zaštitu od *brute force attack. Cost* se podešava prema hardveru servera na kojem se skripta izvršava. Preporuka je da se *salt* ne zadaje kao opcija funkcije jer će se *salt* generirati automatski. Od verzije PHP 7 zadavanje *salt* postaje zastarjeli način rada i može se ukloniti u novijim verzijama PHP.

PASSWORD_ARGON2I - koristi Argon2i algoritam. Algoritam je dostupan samo ukoliko je PHP podešen s podrškom za Argon2i (dostupan od PHP verzije 7.2).

Metoda vraća kriptirani izraz. Izraz uključuje korišteni algoritam, *cost* i *salt* kao dio kriptiranog izraza. Sve informacije koje su potrebne da bi se verificirao uključene su u njemu. To

omogućuje da *password_verify()* funkcija verificira izraz bez potrebe za posebnim spremanjem *salt*, *cost* ili korištenog algoritma.

Za korištenje na serverima preporuka je da se *cost* parametar testira na serveru i prilagodi tako da izvršavanje funkcije bude kraće od 100 milisekundi.

Primjer 1.

```
//Određivanje cost  parametra na serveru
//preuzeto sa:
//http://php.net/manual/en/function.password-hash.php
$timeTarget = 0.05; // 50 milliseconds
$cost = 8;
do {
     $cost++;
     $start = microtime(true);
     password_hash("test", PASSWORD_BCRYPT, ["cost" =>
     $cost]);
     $end = microtime(true);
} while (($end - $start) < $timeTarget);
echo "Appropriate Cost Found: " . $cost;
```

Lozinku koju je korisnik unio potrebno je najprije provjeriti da li odgovara u broju znakova i korištenim znakovima (mala, velika slova, brojke). Kriptirana lozinka se sprema u bazu podataka. Kako bi izradili kriptiranu lozinku iz korisničkog unosa koristili bi sljedeću sintaksu:

```
$kriptiranaLozinka =
password_hash($_POST["lozinka"],PASSWORD_DEFAULT);
```

Provjera kriptirane lozinke s novim unosom korisnika koristila bi sljedeću sintaksu:

```
if(password_verify($_POST["lozinka"],$lozinkaIzBaze))
     echo "Ispravna lozinka";
else
     echo "Neispravna lozinka";
```

15.2. KORIŠTENJE HTTPS

HTTPS je protokol koji omogućava sigurnost na internetu. HTTPS garantira da korisnici komuniciraju sa serverom kojeg očekuju i da nitko ne može presresti ili promijeniti sadržaj koji se pregledava.

Preporuka je da se https koristi na cijelom sjedištu a naročito kod sadržaja koji je privatan kao stranice za prijavu ili plaćanje. U biti trebao bi se koristiti na cijelom web sjedištu. Naročito je bitan kod korištenja sesija jer se u tim slučajevima često postavlja *session cookie* koji sadrži informaciju o sesiji korisnika. Ukoliko se ne zaštiti stranica moguće je ukrasti taj *cookie* i njegove informacije. U tom slučaju napadač može imitirati korisnika i preuzeti njihovu sesiju.

Postavljenje https protokola nije više toliko zahtjevno ili skupo. Certifikat koji je potreban za postavljanje https moguće je dobiti besplatno (pr. stranica Let's Encrypt) te postoje alati, platforme i programski okviri koji automatski podešavaju https.

Google je najavio da će stranice koje imaju https dobiti bolji rang pri indeksiranju pa je postavljanje https također i SEO prednost. Nesiguran http se više neće toliko koristiti.

Postoji i HTTP Strict Transport Security (HSTS), jednostavan *header* koji onemogućuje nesiguran HTTP za cijelu domenu.

15.3. ALATI ZA SIGURNOST WEB STRANICE

Nakon što je stranica izrađena dobro je testirati njenu sigurnost. To se može ostvariti korištenjem alata za web sigurnost koje se uobičajeno naziva alatima za *penetration testing* ili *pen testing.*

Postoji mnogo komercijalnih i besplatnih alata koji pomažu u testiranju. Neki od besplatnih alata su: Netsparker, OpenVAS, SecurityHeaders.io, Xenotix XSS Exploit Framework i dr.

15.4. DOZVOLE ZA DATOTEKE (APACHE WEB SERVER/LINUX-UNIX OS)

Dozvole za datoteke mogu spriječiti neautoriziran pristup ili ograničiti pristup pojedinim direktorijima i datotekama. Pojedine dozvole dozvoljavaju korisnicima da čitaju, zapisuju ili izvršavaju datoteke. Prava se mogu ograničiti za *owner* (vlasnik datoteke), *group* (korisnici unutar iste grupe) ili *world* (javni korisnici).

Prava nad datotekama i direktorijima mogu se postaviti preko FTP klijenta, kao primjerice FileZilla, Command Prompt- Windows, Terminal- Macintosh ili Command Line- Linux.

Prava je moguće podešavati postavljanjem broja ili podešavanjem *rwx* oznaka. Kada se podešava broj on se tipično zadaje s 3 znamenke, gdje prva znamenka određuje prava vlasnika, druga prava grupe a treća prava javnih korisnika. Moguće je koristiti sljedeće znamenke:

0 – bez pristupa

1 – izvršavanje (execute)

2 – zapisivanje (write)

3 – write i execute

4 – čitanje (read)

5 – read i execute

6 – read i write

7 – read, write i execute.

Root direktorij web stranice bi trebao imati dozvolu postavljenu na 755. Ta dozvola daje sva prava za vlasnika i prava čitanja i zapisivanja za grupu i javne korisnike. Većina direktorija na serveru imati će takva prava. Većina datoteka na serveru postavlja se s pravima 644.

Datoteke koje sadrže osjetljive podatke kao primjerice konekcija na bazu podataka (s korisnikom i lozinkom), trebale bi se posebno štititi. Kod takvih datoteka prava se postavljaju na 400, gdje samo vlasnik može čitati datoteku. Konfiguracijske datoteke koje sadrže osjetljive podatke uobičajeno se postavljaju na 600 kako bi se mogle i mijenjati od strane vlasnika.

.htaccess datoteka može imati prava 404 (ili 444). Može se postaviti i na 644 ili 604 ukoliko se pišu skripte koje mijenjaju sadržaj datoteke.

Rwx oznake podešavaju se prema uzorku *–rwx rwx rwx*. Prva oznaka „–„ može se postaviti na d što tada označava prava direktorija. Slijedom se zapisuju prava vlasnika, grupe ili javna prava. *R* je oznaka za čitanje, *w* za zapisivanje a *x* za izvođenje. Ukoliko se neka prava ne žele dodijeliti ostavlja se -.

15.5. .HTACCESS

.htaccess datoteke su važne kada se radi s direktorijima za koje se želi da se ne mogu javno čitati. *.htaccess* datoteke omogućuju definiranje specifičnih pravila za pojedine datoteke, tipove datoteka ili korisnike. *.htaccess* datoteka nije podržana na svakom poslužitelju. Koristi se na Linux i Unix ili bilo kojoj verziji Apache servera ali na njima može biti zabranjena konfiguracijom.

Upotrebljava se za zaštitu datoteka lozinkama, automatsku preusmjeravanje, korisničke stranice o greškama, promjenu ekstenzija datoteka, zabranu ili dozvolu pristupa određenim IP adresama, zabranu ispisivanja popisa datoteka u direktoriju te upotrebu drugih naziva datoteka kao index datoteke i dr.

Direktive koje se zadaju unutar *.htaccess* datoteke odnose se na direktorij u kojem se nalaze te na sve poddirektorije koji se nalaze u njemu. *.htaccess* datoteka omogućuje promjene globalne Apache konfiguracijske datoteke (httpd.conf). Često se koristi na dijeljenim serverima gdje ne postoji mogućnost da se mijenja globalna konfiguracijska datoteka koja bi izmijenila konfiguraciju za sve korisnike servera.

Što se može napraviti unutar *.htaccess* određeno je u globalnoj konfiguracijskoj datoteci. Postoji mnogo mogućnosti pri korištenju navedenih datoteka koja nisu pokrivena u ovom poglavlju a neke se mogu pogledati na http://www.htaccess-guide.com/.

15.5.1. Korisničke stranice s pogreškama

Kreira se *.htaccess* datoteka. Na Windows sustavu ponekad je problem da se datoteka zove ".htaccess" tj. da ima samo ekstenziju bez naziva. Taj problem se može zaobići tako da se datoteka preimenuje na serveru.

Primjer 1.

```
ErrorDocument  404  /putanja/nazivErrDatoteke.html
```

U primjeru izrađena je korisnička stranica za pogrešku 404 koja označava pogrešku prilikom otvaranja stranice koja ne postoji na serveru. Ukoliko nastane navedena pogreška otvorila bi se stranica definirana u putanji. Putanja se uvijek zadaje od root direktorija.

Uobičajene pogreške za koje se mogu izrađivati korisničke stranice su:

401 - Authorization Required

400 - Bad request

403 – Forbidden

500 - Internal Server Error

404 - Wrong page.

15.5.2. Zabrana prikaza indeksa direktorija

Primjer 1.

```
Options –Indexes
```

15.5.3. Zabrana i dopuštanje određenih IP adresa

Primjer 1.

```
deny from 000.000.000.000
#ukoliko se postavi samo jedna ili 2 grupe
#zabranjuje se cijeli raspon
allow from 000.000.000.000
deny from all
#zabrana za sve osim za skripte
```

15.5.4. Alternativne indeks stranice

Primjer 1.

```
DirectoryIndex index.php index3.php index.html index.htm
ime.php
```

15.5.5. Preusmjeravanje

Koristi se kad se promijeni naziv datoteke a korisnicima još uvijek želimo omogućiti da ju nađu ili kada želimo izraditi preusmjeravanje na duži URL.

Primjer 1.

```
#Primjer preusmjeravanja datoteke
Redirect /lokacija/stare/datoteke/naziv.ext
http://www.drugastr.hr/lokacija/datoteke/drugiNaziv.xyz
```

```
#Primjer preusmjeravanja direktorija
Redirect /stariDir http://www.novaStr.hr/noviDir
```

15.5.6. Zaštita lozinkom

Izrađuju se dvije datoteke: *.htpasswd* i *.htaccess*.

.htpasswd se uobičajeno sprema izvan *root* direktorija stranice kako se ne bi mogla dohvatiti s weba. U datoteku se sprema lozinka i korisničko ime u obliku:

```
username:password
```

Dobro je i kriptirati lozinku (npr. http://aspirine.org/htpasswd_en.html ili http://www.htaccesstools.com/htpasswd-generator/).

Primjerice unosom vrijednosti *ime* i *lozinka* u polja na stranici *htaccesstools.com* dobivamo:

```
ime:$apr1$h8UiOhs2$TSOwTq3zoY.8iQnlB4NAZ.
```

Kreira se i .htaccess datoteka. Ona se kreira kako je prikazano u primjeru.

Primjer 1.

```
AuthName "Dio za korisnike"
AuthType Basic
AuthUserFile /Users/Korisnik/Desktop/.htpasswd
#obavezno cijela putanja u sustavu, za server
Require valid-user
```

15.5.7. Mod_rewrite

Mod_rewrite je Apache modul za manipulaciju URL-ova. Često se koristi kako bi se uzeo URL kojeg traži korisnik i šalje mu se sadržaj drugog URL-a. Važno je napomenuti da se mijenjanje URL-a događa na serveru i preglednik još uvijek prikazuje početni URL a podaci se vraćaju sa promijenjenog URL-a.

Prije podešavanja *mod_rewrite* trebalo bi provjeriti da li je omogućen na serveru na kojem će se koristiti. *mod_rewrite* se često koristi za:

- Izradu URL-ova koji su kraći i lakše se upisuju i pamte. Primjerice www.primjer.com/clanci/moj-clanak/ može biti povezan na poveznicu www. primjer.com/prikaziclanak.php?idclanka=moj-clanak.
- Sprječavanje povezivanja drugih stranica na slike u izrađenom web sjedištu (eng. *image leeching/hotlinking*). Ukoliko se druge stranice povezuju na slike u određenom web sjedištu može im se poslati pogreška "*Forbidden*".
- Mnogim se web stranicama može pristupiti preko nekoliko URL-ova. Primjerice www.primjer.com/stranica.html i primjer.com/stranica.html. Preko *mod_rewrite* moguće je zadati da se uvijek prikazuje ispravna stranica.
- Izbjegavanje 404 pogrešaka kada se stranica reorganizira. Tada je stare URL-ove potrebno povezati s novim URL-ovima.

Mod_rewrite direktive se tipično postavljaju u *.htaccess* datoteku koja se nalazi u *root* direktoriju web sjedišta i primjenjuju se na sve direktorije. Dvije najvažnije *mod_rewrite* direktive su:

- *RewriteEngine*: uključuje ili isključuje *mod_rewrite*
- *RewriteRule*: pravila prepisivanja jednog URL-a u drugi.

Primjer 1.

```
RewriteEngine on
RewriteRule ^trazi\.html$ http://www.google.com/ [R=301]
```

U navedenom primjeru ukoliko traži stranica trazi.html ona će biti preusmjerena na Google korištenjem 301 HTTP *redirect*. ^trazi\.html$ je regularan izraz. *Mod_rewrite* podržava PCRE regularne izraze. Označava da se traži od početka URL-a (bez dijela domene), nakon čega slijedi tekst *trazi.html*, nakon čega slijedi kraj URL-a.

Osnovna sintaksa za *RewriteRule* je:

```
RewriteRule Pattern Substitution [Optional Flags]
```

U dokumentu se može nalaziti više *RewriteRule* izjava. *mod_rewrite* prolazi kroz sva pravila jedno po jedno i procesira URL. Ukoliko se promijeni traženi URL u novi URL, novi URL će se dalje pretraživati za uzorke napisane u pravilima te se može ponovno izmijeniti. Ukoliko se ne želi navedena funkcionalnost koristi se [L] zastavica („*last rule*“).

Primjer 2.

```
RewriteEngine on
RewriteRule ^stari-url\.html$ /novi-url.html [R=301,L]
```

Primjer pokazuje kako je moguće napraviti preusmjeravanje primjerice kod promjene organizacije stranice. Ovakav kôd šalje HTTP *header* svakom pregledniku koji pristupa staroj stranici i upućuje kako je stranica premještena na novi URL. Ovakva naredba također upućuje *search engine* da ažurira svoj *index* s novim URL-om.

Primjer 3.

```
RewriteEngine on
RewriteRule ^clanci/([^/]+)/?$
prikazi_clanak.php?clanakID=$1 [L]
```

Primjer prikazuje kako je moguće formatirati jednostavan URL u kompliciraniji koji se koristi. Ovakvo pravilo promijenilo bi URL tipa

http://www.primjer.com/clanci/moj-clanak/

u

http://www.primjer.com/prikazi_clanak.php?clanakID=moj-clanak

U primjeru je kao regularni izraz [^/]+ korišteno podudaranje s bilo kojim znakom osim / koji se ponavlja jedan ili više puta, dok /? označava da se kosa crta ponavlja 1 ili 0 puta. Dakle kosa crta je opcionalna. Dio uzorka je ugrađen u zagradu što daje pod-uzorak. $1 označava pod-uzorak spremljen u zagradama (eng. *backreference*). Ukoliko bi se koristio dodatni pod-uzorak on bi se dohvatio s $2.

Primjer 4.

```
RewriteEngine on
RewriteCond %{HTTP_REFERER} !^$
RewriteCond %{HTTP_REFERER} !^http://(www\.)?primjer\.com/.*$
[NC]
RewriteRule .+\.(gif|jpg|png)$ - [F]
```

U navedenom primjeru onemogućeno je da se druge stranice povezuju na slikovne datoteke na serveru. Bilo koji preglednik koji traži sliku sa servera bit će odbijen s "403 Forbidden" greškom ukoliko se slika traži s URL-a koji započinje s domenom koja je različita od *www.primjer.com* ili *primjer.com*. Na stranici koja povezuje neće se prikazati slika i neće doći do povećanog prometa na serveru.

RewriteCond je još jedna direktiva *mod_rewrite*. Omogućuje da se postavi uvjet koji mora biti zadovoljen kako bi se sljedeći *RewriteRule* procesirao. Linije kôda u primjeru ispituju uvijete zadane pod *RewriteCond;*

```
RewriteCond %{HTTP_REFERER} !^$
```

provjerava da li podatak u HTTP_REFERER nije prazan string

```
RewriteCond %{HTTP_REFERER} !^http://(www\.)?primjer\.com/.*$
[NC]
```

Provjerava da li je HTTP_REFERER različit od http://www.primjer.com/ ili http://primjer.com/

Ukoliko su oba uvijeta zadovoljena izraditi će se izmjena URL-a. Zastavica [NC] označava da je testiranje neosjetljivo na mala i velika slova (eng. *case-insensitive*).

Ukoliko jedan od uvjeta nije istinit *RewriteRule* se neće izvršiti.

Pravilo izmjene URL-a:

```
RewriteRule .+\.(gif|jpg|png)$ - [F]
```

koristi [F] zastavicu kako bi se vratila "403 Forbidden" greška ukoliko URL sadrži .gif ili .jpg ili .png ekstenziju. Znak „–„ označava da nije potrebno prepisivati URL u drugi URL.

Navedeni primjeri samo su manji dio mogućnosti *mode_rewrite*. Potpuna dokumentacija može se pronaći na službenoj stranici:

http://httpd.apache.org/docs/current/mod/mod_rewrite.html.

15.6. PODEŠAVANJE ROBOTS.TXT

Neke datoteke su privatne i ne bi se trebale pronaći preko tražilica. Iz navedenog razloga pojedine direktorije potrebno je zadati da se ne indeksiraju u robots.txt datoteci koja se nalazi na root web direktoriju. Koristi se '*Disallow*' naredba. Da li će se poštovati naredba ovisi isključivo o robotu koji pretražuje stranicu.

Primjer 1.

```
User-agent: *
     Disallow: /stilovi/css/
     Disallow: /includes/
```

16. SESIJE

Sesije omogućuju spremanje podataka u varijable koje se mogu koristiti kroz više stranica. Nisu ovisne o klijentu kao *cookie*, te se informacije ne spremaju na korisničko računalo. Korisnik ne može obrisati podatke kako to može napraviti s *cookie*. Sesije mogu spremiti više podataka nego *cookies*. Podaci se spremaju na serveru te su sigurniji i pouzdaniji nego podaci u *cookies*. Vrijednosti u sesiji ostaju zabilježene kroz stranice i skripte, dok se ne unište. Automatski se uništavaju kad sesija završi. Završetak sesije je kad korisnik ugasi preglednik, čime se završava sesija preglednika ili kad se odjavi.

Glavni nedostatak sesija prema *cookie* je u tome da ne daju mogućnost određivanja koliko dugo će sesija trajati te nije moguće postaviti vrijeme uništavanja. Podaci u sesijama postoje tako dugo dok sesija traje.

Kako bi se radilo sa sesijama potrebno ih je prvo započeti i spremiti vrijednost u varijablu sesije.

bool session_start ([array $options = array()])

session_start() je funkcija koja određuje početak sesije. Koristi se i kako bi se uključili u postojeću sesiju i zato se poziva na svim stranicama koje koriste sesiju. Neće stvarati novu sesiju ukoliko sesija već postoji. Sesija je interno reprezentirana jedinstvenim *session indentifier*-om (koji se automatski generira početkom sesije). To je ID kojeg preglednik koristi kako bi povezao sesiju s višestrukim stranicama, kako bi te stranice mogle pristupiti podacima u sesiji. ID sesije se može prosljeđivati preko GET i POST zahtjeva. SID (*session* ID) može biti zapisan u *cookie* ili se prosljeđuje preko URL-a.

Od verzije PHP 7. dodan je *options* parametar. To je opcionalno asocijativno polje koje mijenja direktive izrade sesije.

Primjer 1.

```
//postavljanje trajanja cookie sesije na 1 dan
//osnovna postavka je 0, koja označava da sesija traje
//dok se ne ugasi preglednik
//vrijeme se zadaje u sekundama
session_start([
    'cookie_lifetime' => 86400,
]);
```

Kod izrade sesija koje su bazirane na *cookie, session_start*() mora biti pozvan prije ispisivanja sadržaja u preglednik.

Rad sa sesijama uključuje spremanje vrijednosti unutar sesijskih varijabli. Sesijske varijable se postavljaju s $_SESSION superglobal varijablom, primjerice:

```
$_SESSION ['korisnik']='admin';
```

Vrijednost sesijske varijable može se dohvatiti ukoliko se na početku stranice vežemo na otvorenu sesiju (na svakoj stranici se koristi *sesssion_start()*). Tada možemo raditi s njima.

Ukoliko imamo stranice koje su zaštićene prijavom korisnika na početku svake stranice preko sesije provjeravamo da li su postavljene sesijske varijable. Ukoliko one nisu postavljene radimo preusmjeravanje korisnika na stranicu za prijavu. Preusmjeravanje je moguće ostvariti preko HTTP zaglavlja (eng. *header*).

Primjer 2.

```
<?php
   session_start();
   if(!isset($_SESSION['korisnik'])){
      header("location: prijava.php");
      exit();
   }
   $ime=$_SESSION['ime'];
?>
<html>
<head>
     <title>Profil od <?php echo $ime;?></title>
</head>
<body>
     <h1>Pozdrav <?php echo $ime;?></h1>
     <h3><a href="odjava.php">Klikni ovdje za
     odjavu</a></h3>
</body>
</html>
```

bool session_destroy (void)

Funkcija koja uništava *session* ID. Ona ne uništava podatke spremljene u sesiji ni *session cookie*. Kako bi se nakon pozivanja *session_destroy()* dalje koristili sesijski podaci dovoljno je pozvati *session_start*(). Podaci se brišu gašenjem preglednika ili postavljanjem u prazno polje:

```
$_SESSION=array();
```

a *session cookie* s funkcijom *setcookie*(). Naziv *cookie*-a kojeg treba uništiti saznajemo preko *session_name*() funkcije.

Primjer 1.

```
session_start();
//poništavanje svih varijabli
$_SESSION=array();
//uništavanje cookie postavljanjem trajanja na 1 sat u
//prošlost
if (isset($_COOKIE[session_name()])) {
    setcookie(session_name(), '', time()-36000, '/');
}
//uništavanje sesije
session_destroy();
```

Sesije mogu koristiti *cookies* ukoliko preglednik omogućava *cookies*. To je predefinirano ponašanje i tada će sesija privremeno spremiti *session* ID u *cookie*.

Ukoliko preglednik ne dozvoljava spremanje *cookie* SID se prosljeđuje preko URL-a stranice. U php.ini treba biti postavljen *session.use_trans_id* na 1. Ukoliko se ne može mijenjati php.ini potrebno je slati SID preko URL-a u kôdu:

```
<a href="stranica.php?<?php echo SID; ?">Naziv linka</a>
```

Ukoliko želimo da prijava traje duže od sesije preglednika potrebno je raditi sa sesijama i *cookie*. Iste podatke spremamo i u sesije i u *cookie*. Ukoliko korisnik ugasi preglednik i ponovno dođe na stranicu sesiju možemo kreirati iz *cookie*.

16.1. KORIŠTENJE ZAGLAVLJA (ENG. HEADER) ZA PREUSMJERAVANJE

Kad server napravi zahtjev prema PHP stranici prije nego što odgovori pregledniku korisnika s HTML-om napraviti će set zaglavlja koji se najprije šalju korisniku. Zaglavlja opisuju što preglednik može sljedeće očekivati.

Zaglavlja je u PHP-u moguće podešavati sa funkcijom header:

```
void header ( string $header [, bool $replace = TRUE [, int
$http_response_code ]] )
```

Primjeri *Header*:

```
header("Content-type: application/pdf");
header("Content-type: application/vnd.ms-excel;
name='excel'");
header("Content-disposition: attachment;
filename=datoteka.xls");
heder("HTTP/1.0 404 Not Found"); exit;
```

Preusmjeravanje (302 rederect)

```
header("Location: stranica.html");
exit;
```

Zaglavlje treba biti poslano prije ispisa bilokakvog sadržaja uključujući bjeline ili HTML oznake, PHP ispis. Česta je pogreška uključivanje kôda s *include* i *require* funkcijama ili prazne linije koje se ispisuju prije pozivanja funkcije.

Pošto su zaglavlja prvo što server šalje ukoliko ih želimo promijeniti to bi trebalo napraviti na samom početku stranice. To se može zaobići preko php.ini datoteke u kojoj se postavlja *output_buffering.* On omogućava da se šalju linije zaglavlja i nakon što je poslan sadržaj jer se koristi izlazni međuspremnik (eng. *output buffer*). Ukoliko nije moguće podešavati php.ini mogu se koristiti funkcije koje omogućuju rad s izlaznim međuspremnikom.

Primjer 1.

```
ob_start();
//sadržaj datoteke
ob_end_flush();
```

17. PAGINACIJA

Kada se gradi CMS sustav i unose se podaci u njega često je mnogo podataka koji su spremljeni u bazi podataka i koje nije preporučljivo ispisivati sve na istoj stranici. Paginacija se koristi kako bi se otvorila i prikazala velika količina sadržaja. Paginacija je proces dijeljenja sadržaja web stranice na zasebne stranice.

Slijedi primjer koji podešava ispis paginacije neke stranice. Primjer je izrađen za ispisivanje korisnika iz baze.

Primjer 1.

```
//provjera da li se nalazi u bazi
//određivanje trenutačne stranice
//ukoliko ništa nemamo zapisano u URL-u,
//stranica se postavlja na 1,
//a ukoliko imamo zapisan broj u $_GET postavljamo ga u
//varijablu stranica
     $stranica=(empty($_GET['stranica'])) ? 1 : (int)
     $_GET['stranica'];

//broj članaka koji će se prikazivati po stranici
     $brojPoStranici=2;

//brojanje koliko je stavki u bazi
     $query="SELECT COUNT(*) FROM korisnici";
     $rezultat=$baza->query($query);
     if ($rezultat) {
          $polje=$rezultat->fetch_row();
          $ukupno_korisnika=$polje[0];
     } else {
          echo "Nije bilo moguće pročitati bazu";
     }

//određivanje koliko ukupno imamo stranica
     $brojStranica=ceil($ukupno_korisnika/$brojPoStranici);

//ukolko korisnik upiše u URL broj stranice koji ne postoji
     if ($stranica<1) $stranica=1;
     else if ($stranica>$brojStranica-1)$stranica=$brojStranica;

//Odeđivanje koji korisnici će se dohvatiti
     $odmak=$brojPoStranici*($stranica-1);

//dohvaćanje korisnika ovisno o stranici na kojoj smo,
//poredamo ih ASC,
```

```
//ukoliko želimo da idu od mlađeg prema starijem stavimo DESC
     $query="SELECT * FROM korisnici ORDER BY id ASC LIMIT
               $brojPoStranici OFFSET $odmak";
     $rezultat=$baza->query($query);
```

Nakon što je određeno koja stranica se prikazuje i nakon što se pročita baza samo za korisnike koje trebamo može se izraditi ispis korisnika. Nakon ispisa slijedi izrada paginacije što je prikazano u sljedećem primjeru.

Primjer 2.

```
//paginaciju budemo ispisivali jedino ukoliko imamo
//više od jedne stranice
   if ($brojStranica>1) {
        echo "<div style='clear: left;'>";
      //ukoliko nismo na prvoj stranici ispisujemo prethodna,
      //kad bi bili na prvoj stranici ne bi ispisali prethodna
        echo '<ul class="pagination pagination-sm">';
        if ($stranica>1) {
        //prethodna je promjenljivi link i ovisi o stranici
        //na kojoj se trenutačno nalazimo --> $stranica-1
             echo "<li>
                  <a href='svi_korisnici.php?stranica=" .
                  ($stranica-1) . "' >
                   &laquo Prethodna </a></li>";
               }
        for ($i=1; $i<=$brojStranica; $i++) {
             if ($i==$stranica) echo "<li class='active'><span>
             $i </span></li>";
             else echo "<li><a
             href='svi_korisnici.php?stranica=$i'> $i
             </a></li>";
        }
        if ($stranica<$brojStranica) {
             echo "<li><a href='svi_korisnici.php?stranica=" .
             ($stranica+1) . "' >
                       Sljedeća&raquo </a></li>";
        }
        echo "</ul>";
        echo "</div>";
   }
```

18. UPLOADS

U PHP-u moguće je dozvoliti prijenos datoteka na server. Kako bi se omogućio prijenos datoteke određene postavke moraju biti zadane u PHP konfiguracijskoj datoteci (php.ini). Postavke koje se tiču prijenosa u php.ini su sljedeće:

- file_uploads

  Treba biti postavljen na *on, true* ili 1 kako bi se omogućio prijenos datoteka.

- upload_tmp_dir

  Ukoliko je postavljen na NULL koristi se *tmp dir* sistema (npr. na Wamp-u je to "c:/wamp/tmp")

- upload_max_filesize

  Maksimalna veličina za datoteku koja se prenosi (tipično je zadano 20MB).

- max_input_time

  Određuje koliko vremena će PHP čekati da dobije podatke datoteke. Ukoliko se postavi na -1 nema vremenskog ograničenja.

- post_max_size

  Maksimalna veličina POST podataka koje će PHP primiti, treba biti postavljen na vrijednost veću od *upload_max_filesize*.

- memory_limit

  Maksimalna količina memorije koju skripta može zauzeti.

- max_execution_time

  Maksimalno vrijeme izvođenja skripte.

Kako bi omogućili prijenos datoteka potreban je *form* element. *Form* element treba imati atribut:

```
enctype="multipart/form-data"
```

Enctype atribut označava da se ne šalje samo tekst nego će biti i drugi podaci u formi. Također je bitno da se za formu postavi metoda POST.

Izrada elementa koji omogućuje prijenos datoteka je preko *input type="file"* elementa:

```
<input type="file" name="datoteka_upload" />
```

Moguće je u HTML-u zadavati postavke prijenosa datoteke primjerice:

```
<input type="hidden" name="MAX_FILE_SIZE" value="1000000"
/>
```

Sakriveno polje deklarira maksimalnu veličinu datoteke, kojoj se vrijednost zadaje u bajtovima (1MB). Mora biti deklarirana prije *input file* polja i ne smije biti veća od *upload_max_filesize* u php.ini. Neki preglednici uzimaju vrijednost polja i neće dozvoliti prijenos koji je veći od zadanog ali nije preporučljivo oslanjati se samo na navedenu opciju.

Sljedeći primjer prikazuje izradu forme u HTML-u.

Primjer 1.

```
<form method="post" action="uploads.php"
enctype="multipart/form-data" >
    <input type="hidden" name="MAX_FILE_SIZE" value="1000000" />
    <input type="file" name="slika" />
   <input type="submit" name="predaj" value="Predaj" />
  </form>
```

Jednom kad je forma predana do prenesenih datoteka moguće je doći preko $_FILES superglobal varijable. Svi ostali podaci i dalje se nalaze u $_POST varijabli. Prilikom prijenosa datoteke najprije se na serveru datoteka postavlja u privremeni direktorij.

$_FILES je dvodimenzionalno polje. Kako bi se došlo do podataka o datoteci koja se prenosi pod prvim ključem navodi se vrijednost *name* atributa zadanog pri definiranju forme („slika“ iz primjera) a pod drugim ključem jedna od sljedećih vrijednosti:

- *name*

  Vraća originalan naziv datoteke, kako se datoteka zvala kad se premještala na server.

- *type*

  Vraća *mime type*, tj. tip datoteke ovisno o njenoj ekstenziji (primjer "image/gif"). Podatak prosljeđuje preglednik.

- *size*

  Veličina datoteke u bajtovima.

- *tmp_name*

  Putanja i privremeni naziv datoteke na serveru. Datoteke se najprije postavljaju u privremeni direktorij gdje dobivaju privremeni naziv odakle ih je potrebno premjestiti u željeni direktorij.

- *Error*

  Određuje oznaku greške koja je nastala prilikom učitavanja datoteke.

Primjer korištenja:

```
$_FILES['slika]['name']
```

gdje je prvi ključ vrijednost *name* atributa a drugi parametar omogućuje dohvaćanje originalnog naziva datoteke.

Prilikom prijenosa datoteka mogu nastati pogreške koje je moguće pratiti preko $_FILES varijable. Pogreške koje mogu nastati pri prijenosu datoteka su sljedeće:

- 0 → UPLOAD_ERR_OK

  Bez greške prilikom prijenosa datoteka.

- 1→ UPLOAD_ERR_INI_SIZE

  Prijenos je bio veći od onog što je deklarirano u php.ini (*upload_max_filesize*).
- 2→ UPLOAD_ERR_FORM_SIZE

  Veće od MAX_FILE_SIZE koji je zadan u formi.
- 3→ UPLOAD_ERR_PARTIAL

  Prijenos nije dovršen.
- 4→ UPLOAD_ERR_NO_FILE

  Datoteka nije poslana.
- 6→ UPLOAD_ERR_NO_TMP_DIR

  Nema privremenog direktorija.
- 7→ UPLOAD_ERR_CANT_WRITE

  Problemi s direktorijima (pravima).
- 8→ UPLOAD_ERR_EXTENSION

  Neka ekstenzija sprječava da se datoteka prenese.

Detaljnije o pogreškama moguće je pročitati na: http://php.net/manual/en/features.file-upload.errors.php.

Kada se radi prijenos datoteka na server treba biti iznimno oprezan i zaštititi web sjedište jer je prijenos datoteka mogućnost koja se može iskoristiti kako bi se ugrozilo web sjedište. Pri tome treba obratiti pažnju na mnogo pojedinosti. Sam prijenos je jednostavan ali zaštita mora biti adekvatno postavljena.

Kako bi se datoteka preselila iz privremenog direktorija u direktorij gdje će biti pohranjena koristi se metoda:

```
bool move_uploaded_file ( string $filename , string
$destination )
```

Funkcija najprije provjerava da li je datoteka zadana pod *$filename* valjana datoteka dobivena s prijenosom (da je prenesena s HTTP POST mehanizmom). Ukoliko je datoteka valjana funkcija će ju preseliti na lokaciju zadanu pod *$destination*. Funkcija vraća TRUE ukoliko je datoteka uspješno premještena.

PHP skripta bi trebala vršiti više promjena i provjera na datoteci koja se prenosi.

Najprije je potrebno provjeriti da li je datoteka predana. To je moguće izjavama koje provjeravaju da li se podaci nalaze u $_FILES superglobal varijabli:

```
empty($_FILES['slika'])
```

i da li su nastale pogreške pri prijenosu:

```
$_FILES['slika']['error'] == 0.
```

Sljedeća provjera je da li je predana datoteka odgovarajućeg *mime* tipa:

```
$dozvoljeni_MIME=array("image/jpeg", "image/gif", "image/png",
"image/bmp");
in_array($_FILES['slika']['type'], $dozvoljeni_MIME)
```

Provjera je bitna jer napadač može predati i druge tipove datoteka primjerice PHP skriptu koju zatim izvršava direktnom poveznicom. Navedena provjera nije dovoljna jer napadač može promijeniti zaglavlje (eng. *request header*) da označava željeni tip datoteke a u biti prosljeđuje drugi tip datoteke. Kako bi se napravila dodatna provjera moguće je koristiti metodu *getimagesize()* iz GD biblioteke ili *FileInfo* ekstenziju.

```
$provjera=getimagesize($_FILES['slika']['tmp_name'])
if(!in_array($provjera['mime'], $dozvoljeni_MIME)) {
    echo "Kriva datoteka!";
    exit;
}
```

Ukoliko se koristi *FileInfo* ekstenzija također je moguće dohvatiti *mime* tip podataka:

```
mime_content_type($_FILES['slika']['tmp_name']).
```

Mnoge slikovne datoteke omogućavaju dodavanje komentara pa je moguće da PHP interpreter ignorira sliku i izvrši php kôd u komentaru. To će ovisiti o konfiguraciji PHP-a koja određuje koje datoteke će se procesirati kao PHP. Često se ne može utjecati na konfiguracijsku datoteku. Iz navedenog razloga slike se spremaju na lokaciji gdje napadač ne može direktno dohvatiti datoteku. To može biti i izvan root direktorija servera i zaštićeno s *.htaccess* datotekom.

Sadržaj *.htaccess* bi bio:

```
order deny,allow
deny from all
allow from 127.0.0.1
```

Dodatno možemo zapisati da ne dozvoljavamo izvršavanje skripti:

```
AddHandler cgi-script .php .pl .jsp .asp .sh .cgi
Options –ExecCGI
```

Primjer različitih opcija pri izradi *.htaccess* datoteke može se pogledati na: https://tomolivercv.wordpress.com/2011/07/24/protect-your-uploads-folder-with-htaccess/.

Korištenje direktorija izvan *root* servera sigurnija je varijanta jer *.htaccess* nije uvijek pouzdan. Ukoliko se datoteke spremaju izvan *root* direktorija da bi se dohvatile potrebna je skripta za dohvaćanje datoteka. Primjer 2. prikazuje skriptu za dohvaćanje slikovnih datoteka koje se nalaze izvan *root* direktorija.

Primjer 2.

```
<?php
// Apsolutna putanja do lokacije slika
$direktorij = '/var/sigurno/slike/';

// Dohvaćanje naziva datoteke preko URL.
// "basename" funkcija se koristi zbog sigurnosti
```

```
// kako bi dohvatili samo naziv datoteke a ne putanju
$naziv = basename($_GET['slika']);

// Izrada putanje datoteke
$putanja = $direktorij . $naziv;

// Provjera da li datoteka postoji
if(!file_exists($putanja) || !is_file($putanja)) {
    header('HTTP/1.0 404 Not Found');
    die('Datoteka ne postoji');
}

// Provjera da li je datoteka slika
$podaciSlike = getimagesize($putanja);
if(!$podaciSlike) {
    header('HTTP/1.0 403 Forbidden');
    die('Datoteka nije slika.');
}

// Podešavanje header
header('Content-type: ' . $podaciSlike ['mime']);
header('Content-length: ' . filesize($putanja));

// Ispis slikovnih podataka
readfile($imgPath);
?>
```

Ukoliko bi se skripta iz Primjer 2. pohranila u datoteku naziva *prikazi_sliku.php* na stranici bi slike prikazivali pomoću sljedećeg kôda:

```
<img src="prikazi_sliku?slika=prva.jpg" alt="Prva slika"><br>
<img src="prikazi_sliku?slika=druga.jpg" alt="Druga
 slika"><br>
```

Dodatno je potrebno izraditi i provjere naziva datoteke. Datoteka bi trebala biti ispravno nazvana (koristi samo dozvoljene slovne znakove). To možemo testirati sa:

```
preg_match("/^[-0-9A-Z_\.]+$/i",$naziv)
```

Ukoliko naziv datoteke ne sadrži samo dozvoljene znakove možemo ih zamijeniti sa – primjerice:

```
//zamjena svih nedozvoljenih znakova
$naziv = preg_replace('/[^\\pL0-9_]+/u', '-', $naziv);
//brisanje crtice (-) ukoliko se pojavljuje na
//početku ili kraju
$naziv = trim($naziv, "-");
//pretvorba stringa u željeni set kodiranja slovnih znakova
$naziv = iconv("utf-8", "ASCII//TRANSLIT", $naziv);
```

```
     $naziv = strtolower($naziv);
     //brisanje preostalih znakova nepodržanih u nazivu
     $naziv = preg_replace('/[^-a-z0-9_]+/', '', $url);
```

Bitno je također da naziv datoteke ima manje od 250 znakova što možemo provjeriti sa:

```
mb_strlen($naziv,"UTF-8") > 225)
```

Možemo provjeriti i da li datoteka ima potrebnu ekstenziju:

```
$dozvoljeno = array('gif','jpg','jpe','jpeg','png');
$posljednjaTocka = strrpos($datoteka_spremanja, ".");
$ekstenzija= substr($datoteka_spremanja, $posljednjaTocka);
if (!in_array($ekstenzija, $dozvoljeno)) {
     echo 'Ekstenzija datoteke nije podržana.';
}
```

Skripta za prijenos mogla bi biti kako slijedi:

Primjer 3.

```
$dozvoljeni_MIME=array("image/jpeg", "image/gif", "image/png",
"image/bmp");
$provjera=getimagesize($_FILES['slika']['tmp_name'])

if(!empty($_FILES['slika'])&&
!in_array($_FILES['slika']['type'], $dozvoljeni_MIME)&&
!in_array($provjera['mime'], $dozvoljeni_MIME)) {
     echo "Niste odabrali ispravan tip datoteke!";
     exit;
} else {
     $greska = $_FILES['slika']['error'];
     $upload_greske = array(
          UPLOAD_ERR_OK=> "Datoteka je uspješno predana",
          UPLOAD_ERR_INI_SIZE  => "Datoteka je prevelika",
          UPLOAD_ERR_FORM_SIZE => "Datoteka je prevelika",
          UPLOAD_ERR_PARTIAL => "Djelomični prijenos.",
          UPLOAD_ERR_NO_FILE => "Niste predali datoteku",
          UPLOAD_ERR_NO_TMP_DIR => "Greška sa serverom",
          UPLOAD_ERR_CANT_WRITE => "Greška sa serverom",
          UPLOAD_ERR_EXTENSION=> "Greška vezana uz ekstenziju
          datoteke.");
     if ($greska>0) {
          echo $upload_greske[$greska];
          exit;
     } else {
          $privremena_datoteka=$_FILES['slika']['tmp_name'];
          $datoteka_spremanja=basename($_FILES['slika']['name']);
          $posljednjaTocka = strrpos($datoteka_spremanja, ".");
```

```
        $ekstenzija= substr($datoteka_spremanja,
        $posljednjaTocka);
        //brisanje točaka ukoliko ih je više u nazivu datoteke
        $datoteka_spremanja= str_replace(".", "",
        substr($datoteka_spremanja, 0, $posljednjaTocka));
        //micanje_razmaknica
        $datoteka_spremanja= str_replace(" ", "",
        $datoteka_spremanja);
        //skrati naslov do 50 znakova
        if (strlen($datoteka_spremanja)>50)
        $datoteka_spremanja= substr($datoteka_spremanja, 0,
        50);
        //vraćanje ekstenzije
        $datoteka_spremanja.=$ekstenzija;
        //naziv direktorija na serveru, putanja do njega,
        //gdje će se slike spremati
        $upload_dir="slike";

        //PROVJERA DA LI DATOTEKA S ISTIM NAZIVOM VEĆ NE
        //POSTOJI
        //ako datoteka postoji u folderu slike novu ćemo
        spremiti s dodatkom broja
        $i=0;
        while
        (file_exists($upload_dir."/".$datoteka_spremanja)) {
             list ($naziv, $ekstenzija)=explode(".",
             $datoteka_spremanja);
             //pomoću rtrim brišemo posljednji broj koji je bio
             //dodan na kraj naziva i dodajemo novi broj i
             //ekstenziju
             $datoteka_spremanja=rtrim($naziv, strval($i-1)) .
             $i . "." . $ekstenzija;
             $i++;
        }
        $slika=$upload_dir. "/" . $datoteka_spremanja;
        if (move_uploaded_file($privremena_datoteka, $slika)) {
             //upisivanje u bazu
             echo 'Uspješan prijenos!';
        } else {
             echo 'Slika nije prebačena u direktorij na
             serveru!';
        }
}}
```