

Web programiranje

ITAcademy

HTTP protokol

- Kada govorimo o sajtu/web aplikaciji, osnovni protokol koji koristimo je: **HTTP**
- HTTP protokol podrazumeva **razmenu poruka između klijenta i servera**
- Razmena poruka podrazumeva **HTTP ZAHTEV i HTTP ODGOVOR**
- **Klijent šalje zahtev**, a **server šalje odgovor**
- Kada klijent pošalje zahtev, i server mu odgovori, **komunikacija se prekida** i oni više nisu “svesni” uzajamnog postojanja.

HTTP protokol / poruka

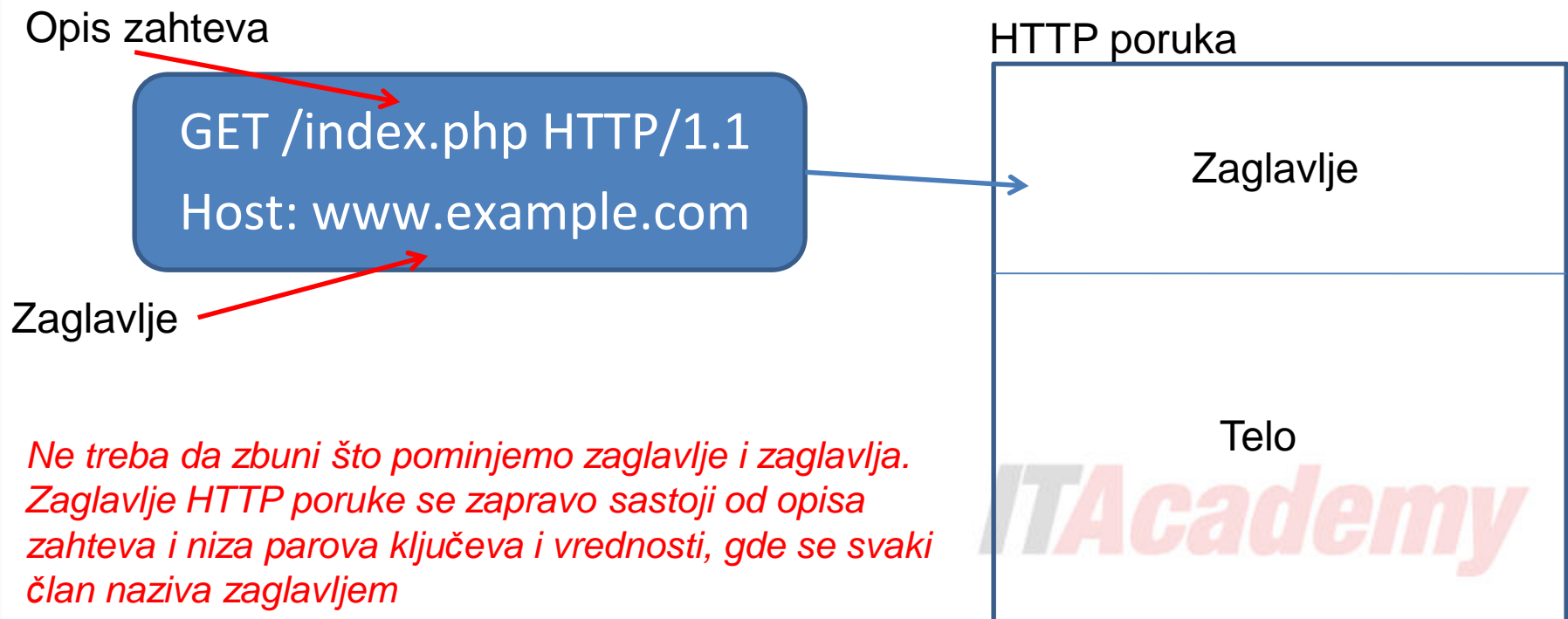
- Svaka http poruka sadrži dva dela:
 - **Zaglavlje**
 - Sastoji se od opisa zahteva/odgovora i liste parova ključeva i vrednosti (header-a)
 - **Telo**
 - **Može sadržati bilo šta**
- Struktura http poruke:

HTTP poruka



HTTP zahtev

- Poruke koje klijent i server šalju jedan drugome su iste strukture, ali nešto drugačijeg sadržaja.
- Http zahtev može izgledati ovako:



HTTP opis zahteva

- Za web aplikaciju, opis zahteva igra veoma značajnu ulogu, jer sadrži informaciju o tome koji dokument se traži u zahtevu, koji http metod je upotrebljen prilikom slanja zahteva i eventualne parametre

GET /index.php HTTP/1.1

HTTP metod

Ciljni dokument

Php je u stanju da identifikuje ove podatke i oni nam stoje na raspolaganju u okviru php aplikacije. Neki su dostupni u nizu: \$_SERVER (u ključevima sa prefiksom HTTP)

Parametrizovani HTTP zahtev

- Često, zahtevi poslati GET metodom, sadrže i parametre. Ovi parametri nalaze se u nastavku naziva dokumenta, identifikovani **znakom pitanja - ?**.
- Svaki parametar je predstavljen parom **ključ/vrednost** odvojenim oznakom jednakosti
- Parametri su uzajamno odvojeni oznakom **&**

GET /index.php?a=2&b=3 HTTP/1.1



Ovo znači da je zahtev poslat sa dva parametra: a i b

Ovako zahtev izgleda u browseru ( 127.0.0.1/index.php?a=1&b=2)

Čemu ovo služi?

HTTP parametri su jedini način da korisnik pošalje neki podatak na server (u aplikaciju)

Upotreba HTTP GET parametara

- PHP je u stanju da prepozna i parsira HTTP GET parametre
- Kada ovakvi parametri stignu do php-a on ih tretira na poseban način, tako što ih smešta u **superglobalnu promenljivu** - asocijativni niz **\$_GET**, a takođe u asocijativni niz **\$_REQUEST**.
- Dakle, get parametri su smešteni na minimum dve lokacije. U nizovima **\$_GET** i **\$_REQUEST**.
- To znači da će tok parametara iz prethodnog primera biti sledeći: _____

Parametri su uneti u address bar browsera



Generisan je http zahtev i poslat na server

GET /index.php?a=2&b=3 HTTP/1.1

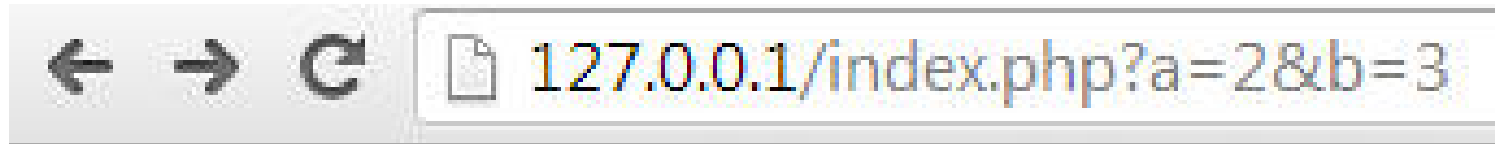
Parametri su smešteni u superglobalne promenljive **\$_GET** i **\$_REQUEST**

\$_GET['a'] = 2;
\$_GET['b'] = 3;

\$_REQUEST['a'] = 2;
\$_REQUEST['b'] = 3;

Primer, parametrizovani kalkulator

- Kreirajmo program koji će na osnovu korisničkog unosa izvršiti sabiranje dva broja, tako da, ako je adresa u address baru browsera: `index.php?a=2&b=3` strana ima sledeći izgled:

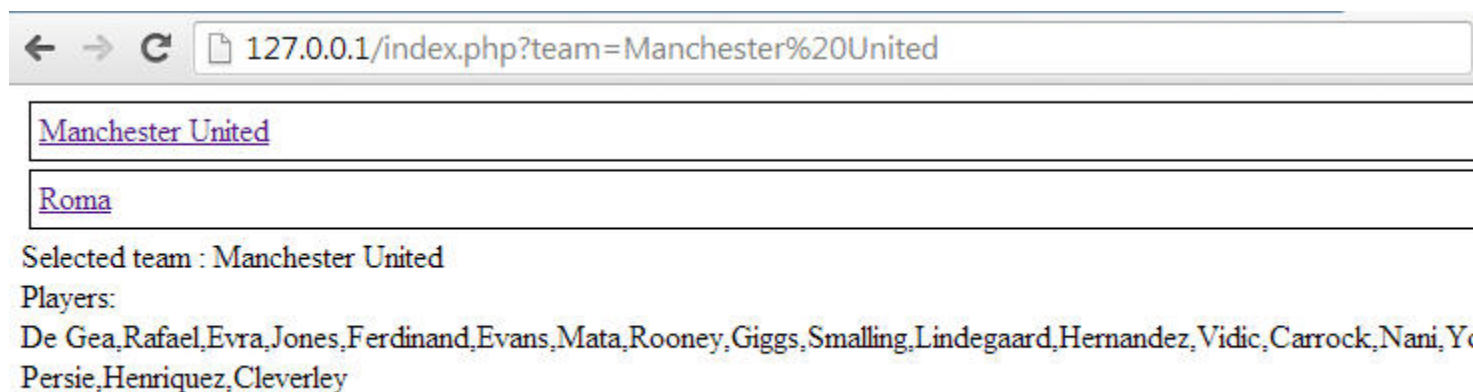


Rezultat je 5

Primer, timovi i igrači

- Pokušajmo da kreiramo program koji će na osnovu korisničkog unosa iz niza timova prikazivati njihove fudbalere
- Da bi program funkcionisao, korisnik mora odabrati jedan od timova iz niza timova predstavljenih kroz div elemente

Kod za niz timova i igrača: <http://pastie.org/pastes/8834735/text>



← → ↻ 127.0.0.1/index.php?team=Manchester%20United

☒ Manchester United

☐ Roma

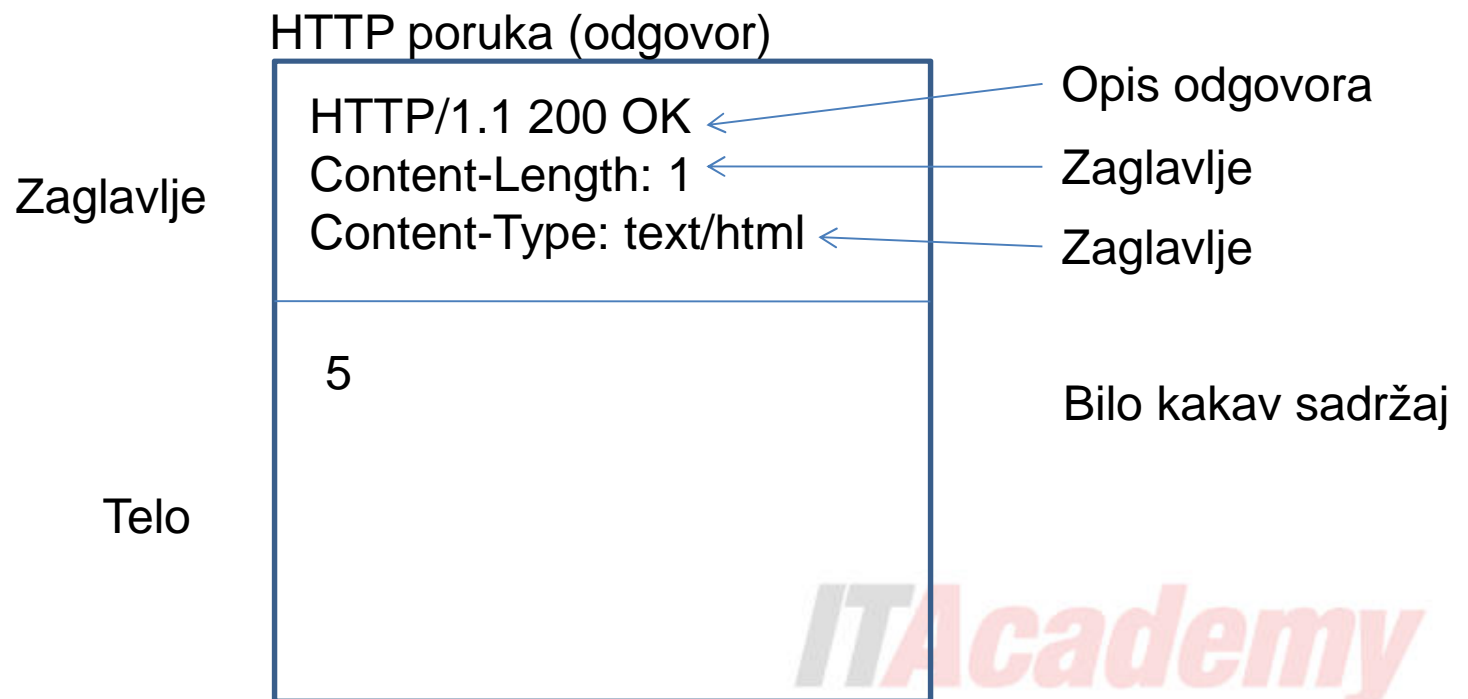
Selected team : Manchester United

Players:

De Gea, Rafael, Evra, Jones, Ferdinand, Evans, Mata, Rooney, Giggs, Smalling, Lindegaard, Hernandez, Vidic, Carrock, Nani, Yo Persie, Henriquez, Cleverley

HTTP odgovor

- Prilikom izvršavanja dva prethodna primera, server (php) je generisao HTTP odgovor. Za primer kalkulatora, ovaj odgovor je imao sledeću strukturu



POST HTTP metod

- Do sada, kroz primere je obrađen samo GET HTTP metod
- Ovaj metod podrazumeva slanje parametara kroz zaglavlje
- POST metod podrazumeva slanje parametara kroz telo zahteva. Pri čemu se parametri formatiraju isto kao i kada se šalju pomoću GET metode

HTTP zahtev / GET HTTP metod

GET /index.php?a=2&b=3 HTTP/1.1

...

Telo je kod get metode
prazno

HTTP zahtev / POST HTTP metod

POST /index.php HTTP/1.1

...

a=2&b=3

ITAcademy

POST HTTP metod

- Kao i u slučaju GET metode, PHP je u stanju da dobavi i parametre poslate POST metodom.
- U ovom slučaju, pored superglobalne promenljive `$_REQUEST`, parametre možemo pronaći i u nizu `$_POST`

HTTP zahtev / POST HTTP metod

POST /index.php HTTP/1.1

...

a=2&b=3

`$_REQUEST['a'] = 2;`

`$_REQUEST['b'] = 3;`

`$_POST['a'] = 2;`

`$_POST['b'] = 3;`

- Da li ovo znači da `$_REQUEST` možemo koristiti i u slučaju post i u slučaju get metode?
- DA – sve dok ne dođe do preklapanja parametara, kad specijalizovani pristup ima prednost nad generalnim.

POST HTTP metod

- Ako nam je potrebno neobrađeno telo HTTP POST zahteva. Možemo ga preuzeti direktnim čitanjem ulaznog toka (php://input):


echo file_get_contents("php://input");

HTTP zahtev / POST HTTP metod

POST /index.php HTTP/1.1

...

a=2&b=3



Kreiranje POST zahteva pomoću forme

- GET metod smo inicirali jednostavnim otvaranjem strane u browseru. A GET parametre realizovali kroz url string.
- Postavlja se pitanje, kako poslati POST parametre, obzirom da nemamo mogućnost direktnog pristupa telu zahteva iz browsera.
- HTML forma ima mogućnost da generiše POST zahtev, da sakupi sve informacije iz kontrola koje sadrži i predstavi ih kao parametre.
- Parametre kasnije možemo preuzeti u php-u na način opisan na prethodnim slajdovima

```
<form action="http://127.0.0.1/form_processor.php" method="post">
  <input type="text" value="Hello" name="tb_first_name" />
  <input type="submit" value="Submit form" />
</form>
```

Peter Submit form

Metod forme mora biti post da bi forma generisala POST zahtev. U suprotnom, forma će generisati GET zahtev.

```
<?php
echo "Hello " . $_POST['tb_first_name'];
```

Kreiranje POST zahteva pomoću forme

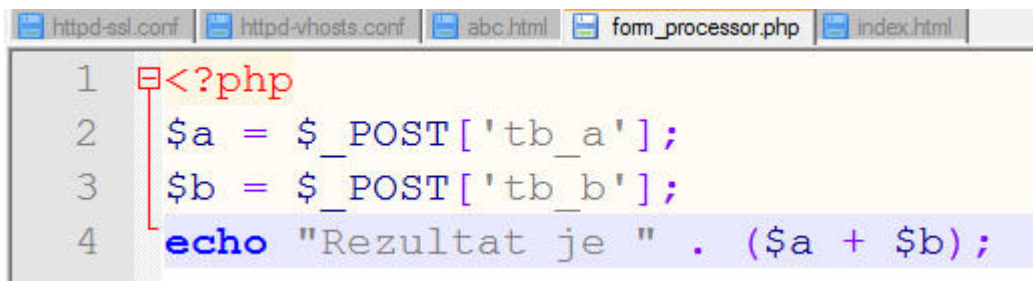
- Pokušajmo da modifikujemo program iz prethodnog primera (sabiranje brojeva), tako što ćemo uvesti mogućnost unosa vrednosti u kontrole forme.
- Forma se može nalaziti na bilo kojoj strani (može i običan html dokument).

```
<form action="form_processor.php" method="post">  
Unesite operand a:<br />  
<input type="text" name="tb_a" /><br />  
Unesite operand b:<br />  
<input type="text" name="tb_b" /><br /><br />  
<input type="submit" value="Saber" />  
</form>
```

- Kroz **action** atribut forme određujemo kojoj će strani forma proslediti podatke. Ovo mora biti serverska skripta, inače podaci forme neće moći da budu obrađeni.

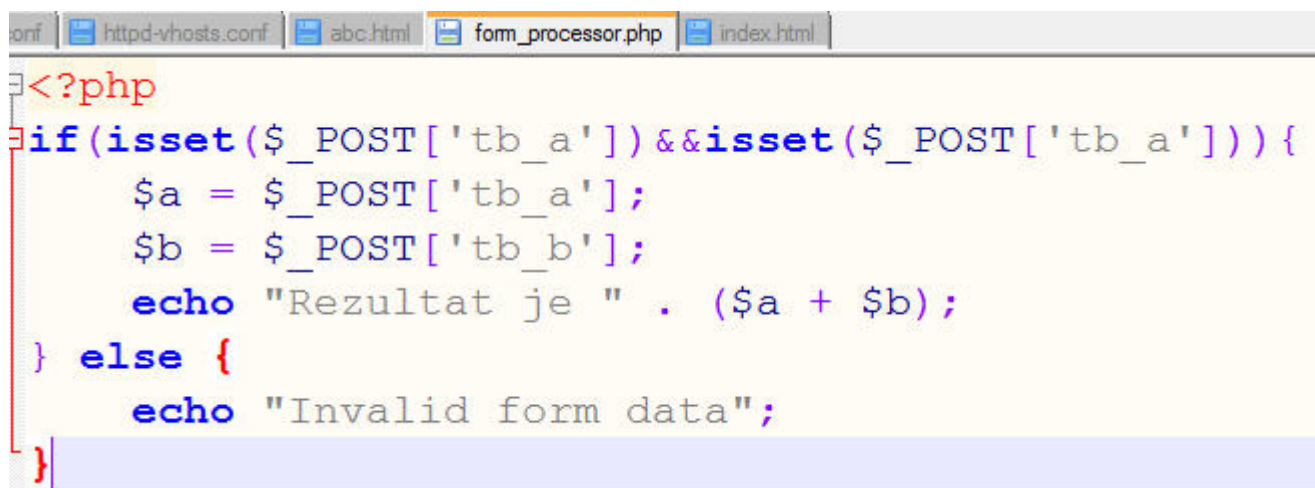
Preuzimanje i obrada parametara forme

- Na serveru, unutar serverske skripte, dočekujemo parametre. Ovo je kritičan trenutak po bezbednost aplikacije, jer je korisnik taj koji definiše podatke, a **KORISNIKU NIKADA NE VERUJEMO**.



```
1 <?php
2 $a = $_POST['tb_a'];
3 $b = $_POST['tb_b'];
4 echo "Rezultat je " . ($a + $b);
```

- Zbog toga uvek treba proveriti parametare forme pre upotrebe. Od provere postojanja, pa do detaljnih provera tipova, vrednosti i slično.



```
<?php
if(isset($_POST['tb_a']) && isset($_POST['tb_b'])) {
    $a = $_POST['tb_a'];
    $b = $_POST['tb_b'];
    echo "Rezultat je " . ($a + $b);
} else {
    echo "Invalid form data";
}
```


Zadatak - kalkulator:

- Potrebno je kreirati program koji od korisnika preuzima dva parametra (operande) kroz text kontrole i jedan parametar (operator) kroz select kontrolu.
- Kada korisnik unese podatke i potvrdi ih tasterom submit, prikazuje se rezultat odabrane operacije

Rešenje

```
<?php
if(isset($_POST['taster'])){
    $operand1 = $_POST['tb_op1'];
    $operand2 = $_POST['tb_op2'];
    switch($_POST['tb_operator']){
        case "+": echo $operand1+$operand2; break;
        case "-": echo $operand1-$operand2; break;
        case "*": echo $operand1*$operand2; break;
        case "/": echo $operand1/$operand2; break;
    }
}
?>
<form action="" method="post">
<input size="2" type="text" name="tb_op1" />
<select name="tb_operator">
<option>+</option>
<option>-</option>
<option>*</option>
<option>/</option>
</select>
<input size="2" type="text" name="tb_op2" /><br />
<input type="submit" name="taster" value="Click" />
</form>
```

Primer: kontakt forma

- Čest primer za upoznavanje sa formama je kontakt forma.
- U nastavku će biti kreirana kontakt forma koja će od korisnika sakupljati sledeće podatke: ime, email, pol, poruku, podatak da li je korisnik stariji od 13 godina.

← → ↻ localhost/contact_form.html

Your name:

Your email:

Your gender:
 Male: ☐ Female: ☒

Message:

I am older than 13 ☐

<http://pastie.org/pastes/8850020/text>

```

<form action="process_contact.php" method="post">
  Your name:<br/>
  <input type="text" name="tb_name" placeholder="Enter your name" required /><br/>
  Your email:<br/>
  <input type="text" name="tb_email" placeholder="Enter your email" required /><br/>
  Your gender:<br/>
  Male: <input type="radio" name="rb_gender" />
  Female: <input type="radio" checked name="rb_gender" /><br/>
  Message:<br/>
  <textarea name="ta_message" placeholder="Enter message" required></textarea><br/>
  I am older than 13 <input type="checkbox" name="cb_age" /><br/>
  <input type="submit" name="btn_send" value="Send" />
  <input type="reset" value="Reset fields" />
</form>

```

Primer: obrada kontakt forme

- U php fajlu za procesiranje forme uraditi što više bezbednosnih provera

<http://pastie.org/pastes/8850180/text>

```
contact_form.html process_contact.php
<?php
if(isset($_POST['btn_send'])){
    $name = isset($_POST['tb_name'])?$_POST['tb_name']: "";
    $email = $_POST['tb_email'];
    $gender = $_POST['rb_gender'];
    $message = $_POST['ta_message'];
    $above_13 = isset($_POST['cb_age'])?true:false;
    if(!$above_13){
        die("Not older than 13 years");
    }
    if(trim($name)==""){
        die("User name field is empty");
    }
    echo "Form data is valid...do something";
} else {
    die("Invalid form data");
}
```

Primer: Tipizacija kontakt forme

- Učinimo formu modularnom, smestivši je u klasu:

<http://pastie.org/pastes/8850342/text>

```
<?php
class ContactForm
{
    public function process(){
        if(isset($_POST['btn_send'])){
            $name = isset($_POST['tb_name'])?$_POST['tb_name']:"";
            $email = $_POST['tb_email'];
            $gender = $_POST['rb_gender'];
            $message = $_POST['ta_message'];
            $above_13 = isset($_POST['cb_age'])?true:false;
            if(!$above_13){
                echo("Not older than 13 years");
                return;
            }
            if(trim($name)==""){
                echo("User name field is empty");
                return;
            }
            echo "Form data is valid...do something";
        } else {
            echo("Invalid form data");
            return;
        }
    }

    public function render(){
        echo '<form action="process_contact.php" method="post">
        Your name:<br/>
        <input type="text" name="tb_name" placeholder="Enter your name" required /><br/>
        Your email:<br/>
        <input type="text" name="tb_email" placeholder="Enter your email" required /><br/>
        Your gender:<br/>
        Male: <input type="radio" name="rb_gender" />
        Female: <input type="radio" checked name="rb_gender" /><br/>
        Message:<br/>
        <textarea name="ta_message" placeholder="Enter message" required></textarea><br/>
        I am older than 13 <input type="checkbox" name="cb_age" /><br/>
        <input type="submit" name="btn_send" value="Send" />
        <input type="reset" value="Reset fields" />
        </form>';
    }
}

$f = new ContactForm;
$f->process();
$f->render();
```

Upload fajla na server pomoću forme

- Pomoću forme je moguće izvršiti upload fajla na server
- U tom slučaju forma mora ispuniti minimalno dva pravila
 - Mora koristiti HTTP POST metod
 - (u atributima forme se postavlja: method="post")
 - Mora biti aktivirano binarno (multipart) slanje podataka
 - (u atributima forme se postavlja: method="post")

```
1 <form action="process_file_upload.php" method="post" enctype="multipart/form-data">
2 <input type="file" name="fup_image" /><br />
3 <input type="submit" name="btn_send" value="Send" />
4 </form>
```

Upload fajla na server pomoću forme

- Kao rezultat fajla uploadovanog pomoću forme, na serveru se popunjava superglobalna promenljiva `$_FILES`. Ona sadrži podatke o jednom ili više fajlova. U sledećoj formi:

`$_FILES`

```
Array
(
    [fup_image] => Array
        (
            [name] => uploaded_file.txt
            [type] => text/plain
            [tmp_name] => C:\wamp\tmp\php721F.tmp
            [error] => 0
            [size] => 4
        )
)
```

Na osnovu ovih podataka možemo saznati sve što nas zanima a vezano je za fajl. Pa lako možemo izvršiti validaciju fajla na serveru

Upload fajla – validacija na serveru

- Na osnovu dobijenih parametara fajla, možemo dobiti sve podatke koji su nam potrebni za validaciju i smeštanje fajla na odgovarajuću lokaciju
- Na primer:

```
<?php
if($_FILES['fup_image']['size']<=0){
    die("Invalid file size");
} else if($_FILES['fup_image']['type']!="image/jpeg"&&$_FILES['fup_image']['type']!="image/png"){
    die("Invalid file format");
} else {
    echo "File is ok";
}
```

- Po istom principu moguće je izvršiti i precizniju validaciju

Upload fajla – kopiranje na konačnu destinaciju

- Kada fajl stigne do servera, on se smešta u tmp direktorijum
- Putanja do uploadovanog fajla se nalazi u ključu **tmp_name**.
- Ovo je ključ koji biva generisan u trenutku kada fajl stigne na server.
- Vrednost ovog ključa je putanja do **PRIVREMENOG** fajla.
- Privremeni fajl biva uništen onog trenutka kada prestane izvršavanje skripte (strane)

[tmp_name] => C:\wamp\tmp\php721F.tmp

- Da bi preuzeli privremeni fajl, koristimo funkciju:
move_uploaded_file
- Funkcija `move_uploaded_file` prihvata dva parametra. Prvi je putanja do privremenog fajla, a drugi, putanja do ciljnog fajla (onaj u koji će privremeni fajl biti kopiran)

```
move_uploaded_file($_FILES['fup_image']['tmp_name'],$_FILES['fup_image']['name']);
```

Primer – registraciona forma

- U nastavku će biti kreirana forma za registraciju korisnika. Forma će sadržati polja za ime korisnika, email korisnika, šifru i avatar.
- Kreiranje forme:

<http://pastie.org/8852462>

```
<form action="reg_form_process.php" method="post" enctype="multipart/form-data">
  Your name:<br/>
  <input type="text" name="tb_name" placeholder="Enter your name" required /><br/>
  Your email:<br/>
  <input type="text" name="tb_email" placeholder="Enter your email" required /><br/>
  Your password:<br/>
  <input type="password" name="tb_password" placeholder="Enter password" required /><br/>
  Your avatar:<br/>
  <input type="file" name="fup_avatar" required/><br />
  <input type="submit" name="btn_register" value="Register" />
</form>
```

Primer – registraciona forma - procesiranje

- U MySql-u kreirati bazu podataka: users_g# (Umesto # uneti broj grupe)

- DDL

<http://pastie.org/8852353>

```
create database users_g1;
use users_g1;
create table users (
    id int primary key auto_increment,
    name varchar(50),
    email varchar(50),
    password varchar(50),
    avatar varchar(256)
);
```

Primer – registraciona forma - procesiranje

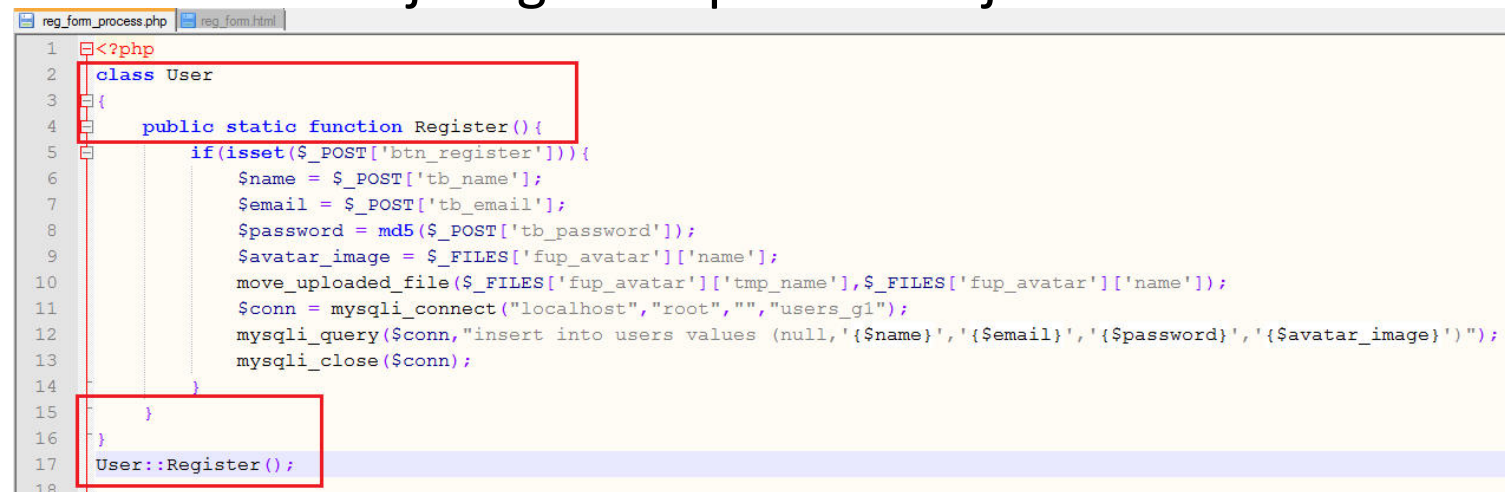
- U fajlu za procesiranje korisnika, treba samo spojiti već naučene elemente:

<http://pastie.org/8852855>

```
<?php
//Verujemo formi i proveravamo samo da li postoji taster
//U produkciji nikada ne radimo na taj nacin, vec vrsimo detaljne provere
if(isset($_POST['btn_register'])){
    $name = $_POST['tb_name'];
    $email = $_POST['tb_email'];
    $password = md5($_POST['tb_password']); //sifra se nikada ne cuva u bazi u izvornom obliku
    $avatar_image = $_FILES['fup_avatar']['name'];
    move_uploaded_file($_FILES['fup_avatar']['tmp_name'],$_FILES['fup_avatar']['name']);
    $conn = mysqli_connect("localhost","root","","users_g1");
    mysqli_query($conn,"insert into users values (null,'{$name}','{$email}','{$password}','{$avatar_image}')");
    mysqli_close($conn);
}
```

<http://pastie.org/8852951>

- Prebacivanje logike za procesiranje forme u klasu



```
reg_form_process.php | reg_form.html
1 <?php
2 class User
3 {
4     public static function Register(){
5         if(isset($_POST['btn_register'])){
6             $name = $_POST['tb_name'];
7             $email = $_POST['tb_email'];
8             $password = md5($_POST['tb_password']);
9             $avatar_image = $_FILES['fup_avatar']['name'];
10            move_uploaded_file($_FILES['fup_avatar']['tmp_name'],$_FILES['fup_avatar']['name']);
11            $conn = mysqli_connect("localhost","root","","users_g1");
12            mysqli_query($conn,"insert into users values (null,'{$name}','{$email}','{$password}','{$avatar_image}')");
13            mysqli_close($conn);
14        }
15    }
16 }
17 User::Register();
18
```

Ostali http metodi

- Pored GET i POST metoda, HTTP takođe ima i druge metode.
 - **DELETE**
 - **PUT**
 - **HEAD**
 - **OPTIONS**
 - **TRACE**
- Ovi metodi su manje zastupljeni, i ne naročito zanimljivi za nas u ovom trenutku

http://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol