АКАДЕМИЈА ТЕХНИЧКО-ВАСПИТАЧКИХ СТРУКОВНИХ СТУДИЈА ОДСЕК НИШ

SERVERSKE TEHNOLOGIJE

Vežba 5 (radi se dve nedelje)

PHP i CRUD operacije

20.11.2024.

Teorijske napomene

CRUD je akronim za osnovne operacije koje se koriste za rad sa podacima:

- **C** - Create (Kreiranje podataka)
- **R** - Read (Čitanje podataka)
- **U** - Update (Ažuriranje podataka)
- **D** - Delete (Brisanje podataka)

Svaka operacija ima odgovarajuću ulogu u radu sa bazama podataka.

1. Create (Kreiranje podataka)

Operacija Create omogućava unos novih podataka u bazu. Ovo je prvi korak u CRUD ciklusu i ključan je za populaciju baze podataka.

- SQL naredba: INSERT INTO
  Ova naredba omogućava umetanje novih redova u tabelu.
- Primer realne primene: Kreiranje korisničkog naloga tokom registracije na web sajtu.
- Napomena: Pri unosu podataka, važno je validirati i sanitizovati ulazne vrednosti kako bi se sprečilo unošenje nevažećih ili zlonamernih podataka.

2. Read (Čitanje podataka)

Operacija Read omogućava dohvat postojećih podataka iz baze.

- SQL naredba: SELECT
  Ova naredba se koristi za čitanje podataka iz tabela na osnovu zadatih kriterijuma.
- Primer realne primene: Prikaz proizvoda iz internet prodavnice.
- Ključne komponente:
  - Filtriranje rezultata (WHERE klauzula).
  - Sortiranje podataka (ORDER BY klauzula).
  - Grupisanje (GROUP BY i HAVING).
- Napomena: Kada radite sa velikim skupovima podataka, možete koristiti paginaciju (deljenje podataka na stranice) kako biste povećali performanse aplikacije.

3. Update (Ažuriranje podataka)

Operacija Update omogućava modifikaciju postojećih podataka. Ovo je neophodno kada korisnik želi da promeni podatke, kao što su kontakt informacije ili lozinka.

- SQL naredba: UPDATE
  Ova naredba modifikuje redove koji ispunjavaju uslov definisan u WHERE klauzuli.
- Primer realne primene: Promena adrese korisnika u korisničkom profilu.
- Napomena: Obavezno koristite WHERE klauzulu kako biste izbegli neželjeno ažuriranje svih redova u tabeli.

4. Delete (Brisanje podataka)

Operacija Delete omogućava uklanjanje podataka iz baze.

- SQL naredba: DELETE
  Ova naredba briše redove koji ispunjavaju uslov definisan u WHERE klauzuli.
- Primer realne primene: Brisanje neaktivnog korisničkog naloga.
- Napomena: Ako se red briše, obavezno proverite da li postoje podaci koji se oslanjaju na njega (strana ograničenja - foreign keys) kako biste izbegli nekonzistentnost u bazi.

Takođe, svaka CRUD operacija se implementira kroz PHP kod koji komunicira sa SQL serverom putem konekcije. U prethodnoj vežbi smo imali prilike da vidimo jednostavnije primere ovog pristupa, a sada ćemo napomenuti još neke bitne detalje.

PHP koristi specifične ekstenzije za komunikaciju sa bazama podataka, koje omogućavaju efikasno i sigurno rukovanje podacima. Najčešće korišćene ekstenzije uključuju:

MySQLi (MySQL Improved):

Ova ekstenzija nudi proceduralni ili objektno orijentisani pristup za rad sa MySQL bazama podataka. MySQLi podržava pripremljene upite, što poboljšava sigurnost i performanse. Prednost MySQLi-a je jednostavnost za rad sa specifičnim funkcijama MySQL-a, ali nije kompatibilan sa drugim tipovima baza.

PDO (PHP Data Objects):

PDO je univerzalnija opcija koja omogućava rad sa više različitih tipova baza podataka (MySQL, PostgreSQL, SQLite, Oracle, itd.). Uz napredne funkcionalnosti poput transakcija i pripremljenih upita, PDO je pogodniji za aplikacije koje zahtevaju fleksibilnost u izboru baze podataka. Dakle, ako postoji potreba za migracijom sa MySQL na neku drugu bazu, PDO omogućava jednostavne promene u kodu.

Proces komunikacije:

1. **Konekcija sa bazom podataka:** PHP uspostavlja vezu sa SQL serverom koristeći kredencijale kao što su host, korisničko ime, lozinka i ime baze.
2. **Priprema SQL upita:** SQL naredba se definiše i priprema za izvršavanje, pri čemu se koriste pripremljeni upiti radi sigurnosti i validacije unosa.
3. **Izvršavanje SQL upita:** PHP šalje SQL naredbu serveru i izvršava je uz odgovarajuće funkcije ili metode.
4. **Obrada rezultata:** Podaci koji se vrate iz baze obrađuju se kao asocijativni niz, objekti ili drugi formati, zavisno od upita.
5. **Zatvaranje konekcije:** Konekcija sa serverom se zatvara nakon završetka rada kako bi se oslobodili resursi.

Obnavljanje SQL upita

1. SELECT upiti

- Koriste se za čitanje podataka iz baze.
- Sintaksa: SELECT kolona1, kolona2 FROM tabela WHERE uslov;

1.1. WHERE klauzula

- Služi za filtriranje podataka.
- Primer: SELECT * FROM korisnici WHERE grad = 'Beograd';

1.2. LIKE klauzula

- Koristi se za pretragu podataka prema obrascu.
- Primer: SELECT * FROM korisnici WHERE ime LIKE 'M%'; -- Sva imena koja počinju na "M".

1.3. ORDER BY klauzula

- Služi za sortiranje rezultata.
- Primer: SELECT * FROM korisnici ORDER BY datum_registracije DESC;

1.4. GROUP BY klauzola

- Koristi se za grupisanje redova u tabeli na osnovu jedne ili više kolona.
- Najčešće se koristi u kombinaciji sa agregatnim funkcijama poput COUNT, SUM, AVG, MAX ili MIN da bi se dobili sažeti podaci po grupama.
- Primer: SELECT hotel_name, COUNT(*) AS broj_recenzija
  FROM reviews
  GROUP BY hotel_name;
- Napomena: Klauzula GROUP BY mora sadržavati sve kolone koje nisu obuhvaćene agregatnim funkcijama.

1.5. HAVING klauzula

- Koristi se za filtriranje rezultata posle grupisanja podataka (za razliku od WHERE, koja filtrira redove pre grupisanja).
- Omogućava da se postave uslovi na rezultate agregatnih funkcija.
- Primer: SELECT grad, COUNT(*) AS broj_korisnika
  FROM korisnici
  GROUP BY grad
  HAVING broj_korisnika > 10;

2. Agregatne funkcije (COUNT, SUM, AVG, MAX ili MIN)

- Omogućavaju izvršavanje operacija na grupama podataka i često se koriste za analizu i sumarizaciju.
- Najčešće se koriste zajedno sa klauzolama kao što su SELECT, GROUP BY i HAVING, kako bi se generisali rezultati.
- Primer: SELECT SUM(price) AS ukupna_cena FROM bookings;

3. INSERT INTO

- Dodavanje novih podataka u tabelu.
- Sintaksa: INSERT INTO tabela (kolona1, kolona2) VALUES (vrednost1, vrednost2);
- Primer: INSERT INTO korisnici (ime, prezime, email)
  VALUES ('Marko', 'Marković', 'marko@example.com')

4. UPDATE

- Ažuriranje postojećih podataka.
- Sintaksa: UPDATE tabela SET kolona1 = vrednost1 WHERE uslov;
- Primer: UPDATE korisnici SET email = 'novi_email@example.com' WHERE id = 1;

5. DELETE

- Brisanje podataka iz tabele.
- Sintaksa: DELETE FROM tabela WHERE uslov;
- Primer: DELETE FROM korisnici WHERE id = 5;

Integracija sa web formom

U ovoj sekciji ćemo detaljno analizirati svaki korak u komunikaciji web forme i baze podataka i navesti primer koji će biti uvod za samostalni praktični zadatak.

Korak 1: Konekcija sa bazom podataka

Primer konekcije sa MySQLi

```
$conn = mysqli_connect("localhost", "username", "password", "database");
if (!$conn) {
    die("Konekcija neuspešna: " . mysqli_connect_error());
}
```

Primer konekcije sa PDO

```
try {
    $pdo = new PDO("mysql:host=localhost;dbname=database", "username", "password");
    $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
} catch (PDOException $e) {
    echo "Konekcija neuspešna: " . $e->getMessage();
}
```

PDO pruža bolju zaštitu od SQL injekcija pomoću pripremljenih upita, iako MySQLi takođe podržava pripremljene upite. Više o tome u odeljku za bezbednost unosa.

PDO koristi izuzetke za obrada grešaka, što omogućava lakše praćenje i rešavanje problema. MySQLi koristi tradicionalnu metodu sa funkcijama za rukovanje greškama.

Svakako, ako radite sa samo MySQL bazom, MySQLi je bolji izbor, dok je PDO pogodniji za aplikacije koje treba raditi sa različitim bazama zbog svoje fleksibilnosti i funkcionalnosti.

Korak 2: Priprema SQL upita

Primer SQL upita sa MySQLi

```
$userId = 1;  // Primer ID

$stmt = $conn->prepare("SELECT * FROM users WHERE id = ?");

$stmt->bind_param("i", $userId);  // Bindovanje parametara
```

U ovom primeru koristimo mysqli_prepare() za pripremu upita koji selektuje sve korisnike iz tabele users sa specifičnim id. Kroz bind_param() vezujemo parametrizovani upit (?) sa stvarnim vrednostima. U ovom slučaju, ? je povezano sa promenljivom $userId, tipa integer.

Primer SQL upita sa PDO

```
$userId = 1;  // Primer ID

$stmt = $pdo->prepare("SELECT * FROM users WHERE id = :id");

$stmt->bindParam(':id', $userId, PDO::PARAM_INT);  // Bindovanje parametra
```

Ovde koristimo PDO::prepare() za pripremu upita koji selektuje sve korisnike sa određenim id-om iz tabele users. Kroz bindParam() vezujemo :id (označeni parametar) sa stvarnom vrednošću ($userId). Ovo je standardna i bezbednija praksa u PDO-u jer omogućava lakšu fleksibilnost prilikom rada sa različitim tipovima podataka.

Korak 3: Izvršavanje SQL upita

Primer izvršenja SQL upita sa MySQLi

```
$stmt->execute();
```

Metoda execute() se koristi nakon pripreme upita da bi se poslala SQL komanda serveru. Ako se koristi parametrijizovani upit, svi bindovani parametri će biti poslati sa njim.

Primer izvršenja SQL upita sa PDO

```
$stmt->execute();
```

PDO koristi istu metodu execute() za izvršenje upita nakon pripreme. Ako je upit pripremljen sa bindovanim parametrima, PDO će poslati odgovarajuće podatke serveru u pravilnom formatu, čime se povećava sigurnost.

Korak 4: Obrada rezultata

Primer obrade rezultata sa MySQLi

```
$result = $stmt->get_result();           // Dobijanje rezultata

while ($row = $result->fetch_assoc()) {  // Hvatanje podataka u asocijativni niz

    echo "Korisničko ime: " . $row['username'] . "<br>";

}
```

Nakon izvršenja upita sa MySQLi, koristimo get_result() da dobijemo rezultate. Zatim prolazimo kroz svaki deobijeni red sa fetch_assoc() funkcijom da dobijemo podatke kao asocijativni niz.

Primer obrade rezultata sa PDO

```
$rows = $stmt->fetchAll(PDO::FETCH_ASSOC);  // Hvatanje rezultata u asocijativne nizove

foreach ($rows as $row) {

    echo "Korisničko ime: " . $row['username'] . "<br>";

}
```

Kod PDO pristupa, nakon izvršenja upita koristimo fetchAll(PDO::FETCH_ASSOC) za dobijanje svih rezultata kroz asocijativne nizove. Ova funkcija vrati niz redova, gde svaki red predstavlja jedan rezultat iz SQL upita.

Korak 5: Zatvaranje konekcije

Primer zatvaranja konekcije sa MySQLi

```
mysqli_close($conn);
```

Nakon što završite sa operacijama oko baze podataka, zatvaranje konekcije sa mysqli_close() će osloboditi resurse i pomoći u upravljanju memorijom.

Primer zatvaranja konekcije sa PDO

```
$pdo = null;
```

PDO automatski zatvara konekciju kada objekat postane neaktivan. Možete postaviti $pdo na null za eksplicitno zatvaranje ako želite.

Bezbednost podataka prilikom rada sa bazama podataka

Za zaštitu podataka od napada, kao što su SQL injekcije, ključno je koristiti **parametrizovane upite**. Kao što smo videli u primeru iznad, ovaj pristup omogućava da korisnički podaci ne budu direktno uključeni u SQL kod, čime se sprečava njihovo izvršavanje kao deo upita.

Rizičan slučaj: `$sql = "SELECT * FROM korisnici WHERE prezime = '" . $_GET['prezime'] . "'";`

Zaštićeni slučaj:
```
$stmt = $pdo->prepare("SELECT * FROM korisnici WHERE prezime = ?");
$stmt->execute([$_GET['prezime']]);
```

Na ovaj način se podaci šalju serveru odvojeno od SQL komandi, čime se sprečava izvršavanje neovlašćenih SQL upita. Za dve nedelje sledi vežba koja će posebno biti posvećena bezbednosti.

Kada obradite rezultate iz baze, važno je i sanitizovati podatke kako biste sprečili izlaganje osetljivih informacija. Sanitizacija korisničkog unosa je proces čišćenja podataka kako bi se sprečili sigurnosni problem i napadi. Cilj je osigurati da podaci uneseni od strane korisnika budu sigurni za dalju obradu ili prikaz. Osnovni primeri sanitizacije su dati u nastavku:

- **Sanitizacija email adrese**: Koristi se filter_input() za čišćenje e-mail unosa:
  `$email = filter_input(INPUT_POST, 'email', FILTER_SANITIZE_EMAIL);`
  Ovaj metod uklanja neželjene karaktere iz e-mail adrese i čini je sigurnom za upotrebu.

- **Sanitizacija za prikaz u HTML-u**: Koristi se htmlspecialchars() da bi se sprečili XSS napadi prilikom prikaza korisničkog unosa:
  `echo htmlspecialchars($unos, ENT_QUOTES, 'UTF-8');`
  Ovaj metod sprečava izvršavanje zlonamernih skripti u pretraživaču.

Nakon što smo obradili sigurnosne aspekte u vezi sa unosima i zaštitom podataka, važno je razumeti kako se podaci obrađuju i komuniciraju između korisnika i servera putem HTML formi. Ipak ono što unesemo u HTML formu "ide dalje" u bazu podataka ili se iz baze izvlači da bi se prikazalo u nekoj komponenti forme na našoj stranici, kao na primer što možemo popuniti padajuću listu podacima iz baze. Dakle, postoje dve osnovne HTTP metode koje se koriste za slanje podataka sa formi na server: **GET** i **POST**.

GET metoda

Karakteristike:

- Podaci se šalju kao deo URL-a.
- Vidljivi su korisnicima u pretraživaču.
- Maksimalna veličina podataka zavisi od pretraživača (obično oko 2000 karaktera).
- Koristi se za zahteve gde nije potrebna zaštita podataka.

Primer upotrebe:

- Pretraga na web sajtovima.
- Filtriranje proizvoda u online prodavnici.

Prednosti:

- Jednostavno debugovanje jer su podaci vidljivi u URL-u.
- URL može biti sačuvan i ponovo korišćen.

Ograničenja:

- Nije bezbedno za osetljive informacije (kao što su lozinke ili brojevi kreditnih kartica).
- Ograničen prostor za slanje podataka.

Kod primene GET metode u formi:

```
<form action="search.php" method="GET">
    <label for="query">Pretraga:</label>
    <input type="text" id="query" name="query">
    <button type="submit">Pretraži</button>
</form>
```

Ovaj obrazac šalje podatke o pretrazi putem URL-a. Na serveru možete obraditi parametar query putem $_GET['query'].

```php
<?php
if (isset($_GET['query'])) {
    // Preuzimanje podataka iz GET zahteva
    $query = $_GET['query'];

    // Sanitizacija unosa (preporučljivo)
    $query = htmlspecialchars($query, ENT_QUOTES, 'UTF-8');

    // Korišćenje unosa za pretragu, na primer u bazi podataka
    echo "Pretraga za: " . $query;
}
?>
```

POST metoda

Karakteristike:

- Podaci se šalju u telu HTTP zahteva.
- Nisu vidljivi u URL-u.
- Nema ograničenja u veličini podataka (osim ograničenja servera).
- Koristi se za slanje osetljivih ili velikih podataka.

Primer upotrebe:

- Prijava korisnika na sistem (korisničko ime i lozinka).
- Slanje podataka sa kontakt forme.

Prednosti:

- Pruža viši nivo sigurnosti za osetljive podatke.
- Može se koristiti za slanje velikih fajlova ili podataka.

Ograničenja:

- Podaci nisu direktno vidljivi za debugovanje.
- URL nije sačuvan za kasniju upotrebu.

Kod primene POST metode u formi:

```
<form action="submit.php" method="POST">
    <label for="name">Ime:</label>
    <input type="text" id="name" name="name">
    <button type="submit">Pošalji</button>
</form>
```

U ovom primeru, podaci se šalju kao telo HTTP zahteva, a na serveru ih možete obraditi putem $_POST['name'].

```php
<?php
if (isset($_POST['name'])) {
    // Preuzimanje podataka iz POST zahteva
    $name = $_POST['name'];

    // Sanitizacija unosa (preporučljivo)
    $name = htmlspecialchars($name, ENT_QUOTES, 'UTF-8');

    // Korišćenje unosa (npr. za unos u bazu podataka ili za prikaz)
    echo "Zdravo, " . $name;
}
?>
```

U GET primeru, kada korisnik pošalje obrazac, URL će biti search.php?query=neki_tekst. PHP će obraditi podatke iz URL-a putem $_GET['query'].

U POST primeru, kada korisnik pošalje obrazac, podaci neće biti vidljivi u URL-u, već će biti poslati putem HTTP tela i obradjeni putem $_POST['name'] u PHP.

Praktični primer

U ovom primeru ćemo dizajnirati jednostavnu web stranicu sa formom za naručivanje burgera, koristićemo PHP za komunikaciju sa bazom podataka i stilizovani HTML za prikazivanje forme. Sledeći koraci pokrivaju operacije od postavljanja baze podataka do kreiranja i implementacije interakcija forme za korisnika.

1. Postavljanje baze podataka

Prvo ćemo kreirati bazu podataka koja sadrži tabelu za burgere i njihove cene. Otvorite XAMPP, uđite na phpMyAdmin i u SQL editoru dodajte sledeći kod:

```
CREATE DATABASE restoran;

USE restoran;

CREATE TABLE burgers (
    id INT AUTO_INCREMENT PRIMARY KEY,
    naziv VARCHAR(100),
    cena DECIMAL(10, 2)
);

INSERT INTO burgers (naziv, cena) VALUES ('Cheeseburger', 5.99);

INSERT INTO burgers (naziv, cena) VALUES ('Hamburger', 4.99);

INSERT INTO burgers (naziv, cena) VALUES ('Veggie Burger', 6.49);
```

Napomena: cene burgera mogu biti u evrima ili dinarima, imate apsolutnu slobodu ovde.

2. Kreiranje PHP koda za povezivanje sa bazom podataka

Povezaćemo se sa MySQL bazom podataka koristeći PDO. Sledi kod za fajl **db.php** koji možete ubaciti preko Visual Studio Code editora. Moja preporuka je da sav kod tu unosite i testirate preko localhosta iz XAMPP-a posle.

```
<?php
// Povezivanje sa MySQL bazom
$host = 'localhost';
$dbname = 'restoran';
$username = 'root';
$password = '';
// Kreiranje PDO objekta
try {
    $pdo = new PDO("mysql:host=$host;dbname=$dbname", $username, $password);
    // Postavljanje PDO opcije za izuzetke
    $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
} catch (PDOException $e) {
    die("Greška pri povezivanju sa bazom: " . $e->getMessage());
}
?>
```

3. HTML forma za odabir burgera i dodataka

Sada možemo kreirati HTML formu koja će omogućiti korisnicima da izaberu burger, količinu, metod dostave i dodatke. Sledi kod za **index.php**, kao i jedna od mogućih ideja za stil, ali svakako imate slobodu i sve boje, margine, fontove i ostale dekoracije da odradite sami. Ubacite i favicon.

```
<?php
// Uključivanje konekcije sa bazom
include 'db.php';

// Uzimanje svih burgera iz baze
$query = $pdo->query("SELECT * FROM burgers");
$burgers = $query->fetchAll(PDO::FETCH_ASSOC);
?>

<!DOCTYPE html>
<html lang="sr">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Burger House</title>
</head>
<body>
    <h1>Naruči svoj burger</h1>
    <!-- Forma za naručivanje burgera -->
    <form action="process.php" method="POST">
        <label for="burger">Izaberi burger:</label>
        <select name="burger" id="burger">
            <?php foreach ($burgers as $burger): ?>
                <option value="<?= $burger['id'] ?>"><?= $burger['naziv'] ?> - $<?=
                            $burger['cena'] ?></option>
            <?php endforeach; ?>
        </select><br><br>

        <label for="quantity">Količina:</label>
        <input type="number" id="quantity" name="quantity" min="1" value="1"><br><br>

        <label for="delivery">Metod dostave:</label><br>
        <input type="radio" id="pickup" name="delivery" value="pickup">
        <label for="pickup">Preuzimanje</label><br>
```

```
        <input type="radio" id="delivery" name="delivery" value="delivery">
        <label for="delivery">Dostava</label><br><br>

        <label for="addons">Dodatak:</label><br>
        <input type="checkbox" id="cheese" name="addons[]" value="cheese">
        <label for="cheese">Topljeni sir</label><br>
        <input type="checkbox" id="bacon" name="addons[]" value="bacon">
        <label for="bacon">Slanina</label><br>
        <input type="checkbox" id="pickles" name="addons[]" value="pickles">
        <label for="pickles">Kiseli krastavci</label><br><br>

        <button type="submit">Izračunaj cenu</button>
    </form>
    <!-- Div gde će se prikazati rezultat -->
    <div id="result" class="result" style="display: none;"></div>
    <script>
        // Dodavanje event listener-a za dugme
        document.getElementById('calculateBtn').addEventListener('click', function () {
            // Prikupljanje podataka iz forme
            const formData = new FormData(document.getElementById('burgerForm'));

            // Slanje AJAX zahteva
            fetch('process.php', {
                method: 'POST',
                body: formData
            })
            .then(response => response.text())
            .then(data => {
                // Prikazivanje rezultata
                const resultDiv = document.getElementById('result');
                resultDiv.innerHTML = data;
                resultDiv.style.display = 'block';
            })
            .catch(error => console.error('Greška:', error));
        });
    </script>
</body>
</html>
```

Predlog za stil:

```
/* Opšti stilovi */
body {
    font-family: 'Arial', sans-serif;
    line-height: 1.6;
    background-color: #f4f4f9;
    margin: 0;
    padding: 0;
    color: #333;
}

/* Kontejner */
.container {
    max-width: 800px;
    margin: 20px auto;
    padding: 20px;
    background-color: #fff;
    border-radius: 8px;
}

/* Naslov */
h1 {
    text-align: center;
    color: #444;
    margin-bottom: 20px;
}

/* Forma */
form {
    display: flex;
    flex-direction: column;
}

label {
    margin-bottom: 8px;
    font-weight: bold;
    color: #555;
}

input, select, button {
    padding: 10px;
    font-size: 1rem;
    margin-bottom: 15px;
    border: 1px solid #ccc;
    border-radius: 4px;
    width: 100%;
}

input[type="number"] {
    width: 80px;
}

input[type="radio"],
input[type="checkbox"] {
    margin-right: 10px;
}

button {
    background-color: #007BFF;
    color: #fff;
    border: none;
    cursor: pointer;
    transition: background-color 0.3s ease;
}

button:hover {
    background-color: #0056b3;
}

/* Fokus i interakcija */
input:focus, select:focus, button:focus {
    outline: none;
    border-color: #007BFF;
}
```

```
/* Rezultat */
.result {
    margin-top: 20px;
    padding: 15px;
    border-radius: 8px;
    background-color: #e7f4e4;
    border: 1px solid #b8e2b4;
    color: #2d572c;
    font-weight: bold;
}


/* Checkbox i radio grupe */
.checkbox-group,
.radio-group {
    display: flex;
    flex-direction: column;
    margin-bottom: 15px;
}
.checkbox-group label,
.radio-group label {
    display: flex;
    align-items: center;
    margin-bottom: 5px;
}

/* Responsivnost */
@media (max-width: 600px) {
    .container {
        padding: 10px;
    }

    button {
        font-size: 0.9rem;
        padding: 8px;
    }
}
```

4. PHP kod za obradu podataka i izračunavanje cene

Kada korisnik pošalje formu, poziva se skripta **process.php** koja obrađuje podatke i vraća HTML sadržaj kao odgovor na AJAX zahtev. Primer takvog koda je dat u nastavku:

```
<?php
// Uključivanje konekcije sa bazom
include 'db.php';

// Preuzimanje podataka iz forme
$burgerId = $_POST['burger'];
$quantity = $_POST['quantity'];
$deliveryMethod = $_POST['delivery'];
$addons = isset($_POST['addons']) ? $_POST['addons'] : [];

// Dohvatanje cene burgera iz baze
$stmt = $pdo->prepare("SELECT * FROM burgers WHERE id = ?");
$stmt->execute([$burgerId]);
$burger = $stmt->fetch(PDO::FETCH_ASSOC);
```

```
// Izračunavanje cene
$totalPrice = $burger['cena'] * $quantity;

// Dodavanje cena dodataka
$addonPrice = 0;
foreach ($addons as $addon) {
    switch ($addon) {
        case 'cheese':
            $addonPrice += 1.00; // Cena za sir
            break;
        case 'bacon':
            $addonPrice += 1.50; // Cena za slaninu
            break;
        case 'pickles':
            $addonPrice += 0.50; // Cena za kisele krastavce
            break;
    }
}

$totalPrice += $addonPrice;

// Ako je dostava, dodajemo cenu dostave
if ($deliveryMethod === 'delivery') {
    $totalPrice += 3.00; // Cena za dostavu
}

// Kreiranje odgovora u HTML formatu
echo "<strong>Vaša narudžbina:</strong><br>";
echo "Burger: " . htmlspecialchars($burger['naziv']) . "<br>";
echo "Količina: " . htmlspecialchars($quantity) . "<br>";
echo "Cena po burgeru: $" . number_format($burger['cena'], 2) . "<br>";
echo "Dodaci: " . (!empty($addons) ? implode(', ', array_map('htmlspecialchars', $addons))
     : 'Nema dodataka') . "<br>";
echo "Metod dostave: " . ($deliveryMethod === 'pickup' ? 'Preuzimanje' : 'Dostava') . "<br>";
echo "<strong>Ukupna cena: $" . number_format($totalPrice, 2) . "</strong>";
?>
```

5. Zaključak

Kako radi?

1. Korisnik popunjava formu na stranici i klikne na dugme **"Izračunaj cenu"**.
2. JavaScript šalje podatke forme ka serveru putem AJAX-a.
3. PHP skripta (process.php) obrađuje podatke, izračunava cenu i vraća HTML odgovor.
4. Rezultat se prikazuje u na istoj stranici, bez osvežavanja.

Ovaj pristup koristi **AJAX** za dinamičko ažuriranje rezultata na stranici, čime se poboljšava korisničko iskustvo. Podaci iz forme se obrađuju server-side, a rezultat se prikazuje direktno na istoj stranici kao komponenta.

Zadatak za samostalni rad

U okviru zadatka za samostalni rad koristite prethodno opisani praktični primer kao osnovu. Neophodno je proširiti bazu podataka tabelom za porudžbine. Svaka porudžbina će imati svoj id, ime kupca, id burgera koji je kupljen, metod dostave, dodatke, količinu, kao i datum porudžbine (koristićemo današnji datum). Za sada ne radimo login za korisnike, dovoljno je samo na web formi napraviti tekstualno polje gde će kupac da unese svoje ime. Obavezno uključite validaciju i ovog polja, ne dozvolite prazne unose. Možete dodati i broj telefona ili adresu kao još jedno polje. Poruždbine treba povezati sa burgerima preko stranog ključa. Sledi predlog SQL koda:

```
CREATE TABLE orders (
    id INT AUTO_INCREMENT PRIMARY KEY, -- Jedinstveni identifikator porudžbine
    customer_name VARCHAR(255) NOT NULL, -- Ime kupca
    burger_id INT NOT NULL, -- ID izabrane vrste burgera
    delivery_method ENUM('pickup', 'delivery') NOT NULL
    cheese BOOLEAN DEFAULT FALSE, -- Dodatak: topljeni sir (0 ili 1)
    bacon BOOLEAN DEFAULT FALSE, -- Dodatak: slanina (0 ili 1)
    pickles BOOLEAN DEFAULT FALSE, -- Dodatak: kiseli krastavci (0 ili 1)
    quantity INT NOT NULL -- Količina naručenih burgera
    order_date DATETIME NOT NULL,
    phone_number VARCHAR(15), -- Broj telefona kupca
    address TEXT, -- Adresa kupca
    FOREIGN KEY (burger_id) REFERENCES burgers(id) -- Povezivanje sa tabelom `burgers`
)
```

PHP kod za ovo dodavanje (sa detaljnom validacijom podataka) bi mogao ovako slično da izgleda:

```
<?php
// Uključivanje konekcije sa bazom
include 'db.php';
// Provera da li je forma poslata metodom POST
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
```

```
// Prikupljanje i validacija unosa sa forme
$customer_name = trim($_POST['customer_name'] ?? '');
$burger_id = intval($_POST['burger'] ?? 0);
$delivery_method = $_POST['delivery'] ?? '';
$quantity = intval($_POST['quantity'] ?? 0);
$addons = $_POST['addons'] ?? [];
$phone_number = trim($_POST['phone_number'] ?? '');
$address = trim($_POST['address'] ?? '');

// Detaljna validacija
if (empty($customer_name)) {
    die('Greška: Ime kupca je obavezno.');
}
if ($burger_id <= 0) {
    die('Greška: Morate izabrati validan burger.');
}
if (!in_array($delivery_method, ['pickup', 'delivery'])) {
    die('Greška: Morate izabrati validan metod dostave.');
}
if ($quantity <= 0) {
    die('Greška: Količina mora biti veća od 0.');
}
if ($delivery_method === 'delivery' && empty($address)) {
    die('Greška: Adresa je obavezna za dostavu.');
}

// Ubacivanje dodataka (default vrednost je FALSE)
$cheese = in_array('cheese', $addons) ? 1 : 0;
$bacon = in_array('bacon', $addons) ? 1 : 0;
$pickles = in_array('pickles', $addons) ? 1 : 0;

// Priprema SQL upita za unos podataka u tabelu
$sql = "INSERT INTO orders (customer_name, burger_id, delivery_method, cheese, bacon,
                            pickles, quantity, phone_number, address)
        VALUES (:customer_name, :burger_id, :delivery_method, :cheese, :bacon, :pickles,
                :quantity, :phone_number, :address)";
$stmt = $pdo->prepare($sql);
```

```
    // Povezivanje vrednosti sa parametrima
    $stmt->bindParam(':customer_name', $customer_name);
    $stmt->bindParam(':burger_id', $burger_id);
    $stmt->bindParam(':delivery_method', $delivery_method);
    $stmt->bindParam(':cheese', $cheese, PDO::PARAM_BOOL);
    $stmt->bindParam(':bacon', $bacon, PDO::PARAM_BOOL);
    $stmt->bindParam(':pickles', $pickles, PDO::PARAM_BOOL);
    $stmt->bindParam(':quantity', $quantity, PDO::PARAM_INT);
    $stmt->bindParam(':phone_number', $phone_number);
    $stmt->bindParam(':address', $address);

    // Izvršavanje SQL upita
    try {
        $stmt->execute();
        echo "Porudžbina je uspešno zabeležena!";
    } catch (PDOException $e) {
        echo "Greška pri unosu podataka: " . $e->getMessage();
    }
} else {
    die('Greška: Forma nije poslata na validan način.');
}
?>
```

Korisni resursi

1. Osnovno vežbanje za hvatavanje i ubacivanje podataka preko PHP-a
   https://tutorial.techaltum.com/data-fetching-from-html-in-php.html
2. Detaljni prikaz MySQLi vs PDO
   https://websitebeaver.com/php-pdo-vs-mysqli
3. Odličan vodič kroz savreemenu i efikasnu validaciju podataka
   https://mailtrap.io/blog/php-form-validation/
4. Korišćenje PHP-a sa funkcijama agregacije
   https://www.geeksforgeeks.org/php-my-sql-avg-aggregate-function/
   https://www.geeksforgeeks.org/php-mysql-minmax-aggregate-operations/
   https://www.geeksforgeeks.org/php-mysql-sum-operation/
5. Update i Delete upiti preko PHP-a (biće nam potrebni za narednu vežbu)
   https://fullstackworld.co.in/courses/full-stack-essentials/lessons/topic-4-php-with-update-delete-2/

   Dodatak: Nacrt mog PHP projekta koji sam radila tokom svojih master studija (2017)
   https://drive.google.com/drive/folders/0ByvDz3t3J6kienhGNHdzRXRndWc?resourcekey=0-HL72Esm-4dYiVjcA7VbNfA&usp=drive_link