**Demo Ispit za Backend Developer Tečaj**

Dobrodošli na demo ispit za Backend Developer tečaj. Ispit traje 90 minuta i sastoji se od tri dijela: Osnove PHP-a, Napredni PHP, i Osnove MySQL-a. Svaki dio sadrži ABC pitalice, pitanja za samostalno odgovaranje, te programski zadatak. Sretno!

**1. Osnove PHP-a**

**Što je PHP? (1 bod)**

A) Programski jezik specifičan za platformu

**B)** Server-side skriptni jezik

C) Klijentski jezik

PHP (Hypertext Preprocessor) je skriptni jezik otvorenog koda koji se koristi za razvoj dinamičkih web stranica i aplikacija. PHP se često koristi u kombinaciji s HTML-om za generiranje sadržaja koji se prikazuje na web stranici, a također se koristi za rad s bazama podataka, sesijama, autentifikacijom korisnika i drugim web funkcijama.

**Koji od sljedećih načina može poslužiti za ubacivanje PHP koda unutar HTML-a? (1 bod)**

A) `<?php ?>`

B) `<script type="php">`

**C)** Oba A i B

**Kako se zove funkcija u PHP-u za spajanje na bazu podataka MySQL? (1 bod)**

A) `connect_mysql()`

B) `mysql_connect()`

**C)** `mysqli_connect()`

**Objasni razliku između echo i print u PHP-u. (2 boda)**

`echo` je jezična konstrukcija a ne funkcija, može ispisati 1 ili više argumenata odvojenih zarezom, nema povratnu vrijednost, `print` je funkcija (iako se koristi kao jezična konstrukcija), vraća uvijek 1, prihvata samo 1 argument, npr. `$result = (print "krumpir"); var_dump($result);` će ispisati `true`. Oba mogu interpolirati varijable u dvostrukim navodnicima ili odvojiti zarezima (za razliku od `print`).

**Kako se definiraju varijable u PHP-u?**

Varijable u PHP-u započinju s znakom dolara (`$`) i ne zahtijevaju prethodnu deklaraciju tipa. PHP je dinamički tipiziran jezik, što znači da tip varijable može biti promijenjen tijekom izvođenja programa.

**Što je to "globalna" i "lokalna" varijabla u PHP-u?**

Globalna varijabla: Definira se izvan funkcija i metoda, dostupna je u cijelom programu, uključujući i unutar funkcija, ali uz korištenje ključne riječi global ili korištenjem `$GLOBALS` superglobalne varijable.

```
$globalVar = 10;

function test() {
    global $globalVar;
    echo $globalVar;
}
test();  // Ispisuje 10
```

Lokalna varijabla definira se unutar funkcije i dostupna je samo unutar te funkcije.

**Objasni kako funkcionira funkcija `isset()`. (2 boda)**

Funkcija se koristi za provjeru da li je varijabla definira i nije `null`. `isset($var)` vraća `true` ako je varijabla definirana i njena vrijednost nije `null`. false vraća ako je definirana ili je njena vrijednost `null`. `empty()` je sličan ali provjerava da li je varijabla prazna (`null`, prazan string, `0`, `false`).

**Napišite PHP skriptu koja će izračunati sumu prvih 100 prirodnih brojeva i ispisati rezultat. (3 boda)**

```
<?php
$suma = 0; // inicijalizacija prije petlje
for ($i = 1; $i <= 100; $i++) {
    $suma += $i;
}
echo "Suma prvih 100 prirodnih brojeva: " . $suma;
?>
```

**Što predstavlja $_GET i $_POST u PHP-u?**

`$_GET` je superglobalna varijabla koja se koristi za pristup podacima koji su poslani putem URL-a, tj. putem `GET` metode. Parametri se prenose u URL-u i mogu se vidjeti u adresnoj traci.

```
// URL: index.php?ime=Ivan&dob=25
$ime = $_GET['ime'];  // Ispisuje "Ivan"
$dob = $_GET['dob'];  // Ispisuje 25
```

`$_POST` je superglobalna varijabla koja se koristi za pristup podacima poslanim putem HTTP POST metode. Ovi podaci se ne prikazuju u URL-u, što ih čini pogodnim za slanje osjetljivih informacija, kao što su lozinke.

```
// Form data with method POST
$ime = $_POST['ime'];
```

`$_GET` se koristi kada želite prenijeti podatke putem URL-a (npr. za pretrage, filtriranje), dok je `$_POST` bolji za slanje osjetljivih podataka (npr. prijava, registracija) jer podaci nisu vidljivi u URL-u.

**Koja je razlika između == i === u PHP-u?**

`==` (dvostruka jednako): Provodi usporedbu vrijednosti, ignorirajući tip varijable. Ako su vrijednosti iste, smatraju se jednakima. `===` (triple equal): Provodi usporedbu vrijednosti i tipa. Varijable moraju imati istu vrijednost i tip.

**2. Napredni PHP**

**Što predstavlja OOP termin "nasljeđivanje"? (1 bod)**

A) Skrivanje unutarnjih detalja

**B)** Ponovna upotreba koda

C) Metoda koja ne vraća vrijednost

OOP termin "nasljeđivanje" (eng. inheritance) odnosi se na sposobnost jedne klase da naslijedi svojstva (atribute) i metode druge klase. To omogućuje ponovnu upotrebu koda i olakšava proširivanje postojećih klasa bez potrebe za njihovim ponovnim definiranjem.

**Koji je operator korišten za stvaranje objekta u PHP-u? (1 bod)**

A) `->`

**B)** `new`

C) `::`

O operator `new` se koristi za stvaranje objekta iz klase. Operator `->` se koristi za pristupanje metodama i svojstvima objekta, ali nije operator za stvaranje objekta. Operator `::` se koristi za pristupanje statičkim metodama i svojstvima klase, ali također nije za stvaranje objekta.

**Koji PHP konstrukt omogućava programerima hvatanje izuzetaka? (1 bod)**

A) `do-catch`

B) `exception`

**C)** `try-catch`

try-catch konstrukcija omogućava programerima da "uhvate" izuzetke i obrade ih na odgovarajući način. U try bloku stavlja se kod za koji postoji mogućnost da baci izuzetak, dok se u catch bloku navodi kako se izuzetak treba obraditi ako do njega dođe. `do-catch` ne postoji, `exception` je PHP klasa koja predstavlja izuzetak.

**Objasni koncept Namespaces u PHP-u. (2 boda)**

Namespaces je koncept koji omogućava jednostavniju organizaciju aplikacije. Omogućava upravljanje imenima konstanti, klasa i funkcija za izbjegavanje kolizija (ista imena u različitim dijelovima aplikacija). Olakšava autoloading (automatsko učitavanje) klasa i organizaciju u logičke cjeline.

**Kako se može implementirati autoloading klasa u PHP-u? (2 boda)**

Prvi način je pomoću `spl_autoload_register` funkcije koja omogućava registraciju više autoload funkcija. Svaki put kada PHP naiđe na novu klasu koja nije prethodno učitana, ova funkcija će biti pozvana.

```
// Autoload funkcija
function myAutoloader($class) {
    include 'classes/' . $class . '.class.php';
}

// Registracija autoload funkcije
spl_autoload_register('myAutoloader');

// Korištenje klase (PHP automatski učitava odgovarajuću datoteku)
$obj = new MyClass();
```

Drugi način je pomoću PSR-4 standarda za autoloading. Za to moramo imati konfiguriran composer i podešenu `composer.json` u root direktoriju projekta:

```
{
  "autoload": {
    "psr-4": {
      "MyNamespace\\": "src/"
    }
  }
}
```

Imamo namespace i direktorij u kojem se nalaze klase koje pripadaju tom prostoru imena.
Pokrenemo Composer kako bi generirali autoload datoteku:

```
bash
composer dump-autoload
```

U PHP kodu, sada možete jednostavno koristiti klase:

```
// Učitavanje autoload datoteke
require 'vendor/autoload.php';
// Korištenje klase
$obj = new MyNamespace\MyClass();
```

**Napiši PHP klasu Auto koja ima svojstva boja i model i metodu `ispis()` koja ispisuje te vrijednosti. (3 boda)**

```php
<?php
class Auto {
    private $boja;
    private $model;

    public function __construct($boja, $model) {
        $this->boja = $boja; // uzmi iz parametara
        $this->model = $model;
    }

    public function ispis() {
        echo "Boja je: " . $this->boja . "<br>";
        echo "Model auta je: " . $this->model . "<br>";
    }
}

// Kreiranje objekta klase Auto
$mojAuto = new Auto("Plava", "Tesla model 3");

$mojAuto->ispis();
```

**Razlika između korištenja apstraktnih klasa i interface-a**

Glavne razlike između korištenja apstraktnih klasa i interfejsa u PHP-u su:

1. Apstraktne klase mogu sadržavati apstraktne i konkretne metode, dok interface-i mogu imati samo apstraktne metode.

2. Apstraktne klase ne podržavaju višestruko nasljeđivanje, dok interfejsi to podržavaju.

3. Apstraktne klase mogu imati svojstva (varijable) koja mogu biti `final`, `non-final`, `static` i `non-static`, dok interfejsi mogu imati samo `static` i `final` svojstva.

4. Apstraktne klase mogu imati `static` metode, `main` metodu i konstruktore, dok interfejsi ne mogu.

5. Apstraktna klasa može pružiti implementaciju interfejsa, dok interfejs ne može pružiti implementaciju apstraktne klase.

6. Apstraktne klase se deklariraju ključnom riječi `abstract class`, dok se interfejsi deklariraju ključnom riječi `interface`.

7. Apstraktne klase postižu djelimičnu apstrakciju (0-100%), dok interfejsi postižu potpunu apstrakciju (100%).

Ukratko, apstraktne klase su fleksibilnije jer mogu imati konkretne metode i svojstva, ali ne podržavaju višestruko nasljeđivanje. S druge strane, interfejsi su rigidniji, ali podržavaju višestruko implementiranje i postižu potpunu apstrakciju.

### 3. Osnove MySQL

**Što je primarni ključ u MySQL bazi podataka? (1 bod)**

**A)** Ključ koji identificira svaki zapis jedinstveno

B) Ključ koji može imati duplicirane vrijednosti

C) Ključ koji se koristi za optimizaciju

Primarni ključ je jedinstveni identifikator za svaki redak u tablici. Svaka tablica može imati samo jedan primarni ključ. Vrijednosti u primarnom ključu moraju biti **jedinstvene i ne mogu biti** `NULL`. Primarni ključ omogućava brzo pretraživanje i povezivanje podataka unutar tablice.

```
CREATE TABLE Students (
    student_id INT NOT NULL,
    name VARCHAR(100),
    PRIMARY KEY (student_id)
);
```

Strani ključ je stupac (ili kombinacija stupaca) u jednoj tablici koji se odnosi na primarni ključ druge tablice. Strani ključ omogućava uspostavljanje odnosa između tablica. Vrijednosti u stranom ključu moraju odgovarati vrijednostima primarnog ključa u povezanoj tablici ili biti NULL (ako je dopušteno). Strani ključ osigurava referencijalni integritet, što znači da se ne mogu unijeti podaci u stranoj tablici koji nemaju odgovarajući redak u tablici kojoj pripada primarni ključ.

```
CREATE TABLE Enrollments (
    enrollment_id INT NOT NULL,
    student_id INT,
    course_id INT,
    PRIMARY KEY (enrollment_id),
    FOREIGN KEY (student_id) REFERENCES Students(student_id)
);
```

**Koji SQL naredba se koristi za brisanje zapisa iz tablice? (1 bod)**

A) `REMOVE`

**B)** `DELETE`

C) `DROP`

```
DELETE FROM ime_tablice WHERE uslov;
```

`REMOVE` ne postoji kao SQL naredba. `DROP` se koristi za brisanje cijele tablice ili baze podataka, a ne pojedinih zapisa.

**Koji podatak definira SQL naredba CREATE TABLE? (1 bod)**

A) Kreira novu bazu podataka

**B)** Kreira novu tablicu

C) Kreira novi stupac u tablici

```
CREATE TABLE ime_tablice (
    id INT PRIMARY KEY,
    naziv VARCHAR(100),
    datum DATETIME
);
```

**Objasni razliku između INNER JOIN i LEFT JOIN. (2 boda)**

`INNER JOIN` vraća samo redove gdje postoji podudaranje (match) u obje tablice iz baze podataka. `LEFT JOIN` vraća sve redove iz lijeve tablice i odgovarajuće redove iz desne tablice (koji imaju podudaranje). Uključuje redove iz lijeve tablice i kada ne postoji odgovarajući red u desnoj tablici, kada kolone iz desne tablice imaju NULL vrijednost. Dakle razlika je u načinu kako se spajaju tablice i koje redove uključuju u rezultat.

**Kako se koristi SQL naredba UPDATE? (2 boda)**

Koristi se za izmjenu postojećih podataka u tablici:

```
UPDATE naziv_tablice
SET naziv_stupca1 = nova_vrijednost1,
    naziv_stupca2 = nova_vrijednost2,
    ...
WHERE uvjet;
```

Ako nema `WHERE` dijela, ažuriraju se svi redovi u tablici `naziv_tablice`.

**Napiši SQL naredbu koja stvara tablicu Korisnici s stupcima ID (primarni ključ, auto-increment), Ime, i Email. (3 boda)**

```
CREATE TABLE Korisnici (
    ID INT AUTO_INCREMENT PRIMARY KEY,
    Ime VARCHAR(30) NOT NULL,
    Email VARCHAR(30) NOT NULL UNIQUE
);
```