Napredno PHP Programiranje

Iznimke

Uvod u iznimke

- **Što su iznimke?**
  - Specijalni objekti u programiranju koji upravljaju neočekivanim greškama.
  - Omogućuju kontrolirano upravljanje greškama umjesto zaustavljanja programa.

- **Zašto koristiti iznimke u PHP-u?**
  - Čišći i čitljiviji kod.
  - Lakše otkrivanje i rješavanje problema.
  - Mogućnost reagiranja na specifične greške na specifične načine.

Sintaksa iznimki

- **Osnovna sintaksa:**

```php
try {
    throw new Exception("Došlo je do greške");
} catch (Exception $e) {
    echo $e->getMessage();
} finally {
    echo "Ovo se uvijek izvodi.";
}
```

- **Ključne riječi:**
  - ***try***: blok koda koji se testira za greške dok se izvodi
  - ***throw***: baca iznimku kada se dogodi problem
  - ***catch***: omogućuje hvatanje i obradu iznimke
  - ***finally***: blok koda koji se izvodi nakon try/catch blokova, opcionalan

Vrste iznimki

- **Standardne iznimke:**
  - ***Exception***: bazna klasa za sve iznimke.
  - ***ErrorException***: koristi se za obradu PHP grešaka kao iznimki.
- **Specijalizirane iznimke:**
  - ***InvalidArgumentException***: baca se kada se funkciji predaju neispravni argumenti.
  - ***LengthException***: baca se kada se funkcija suoči s problemom dužine, npr. prekratki string.
- **Kreiranje prilagođenih iznimki:**

```
class MyCustomException extends Exception {
    // Prilagođene funkcije ili poruke
}
```

Primjeri kodiranja s iznimkama

```php
try {
    $file = fopen("ne_postoji.txt", "r");
    if (!$file) {
        throw new Exception("Datoteka ne postoji.");
    }
} catch (Exception $e) {
    echo "Greška: " . $e->getMessage();
}
```

Primjeri kodiranja s iznimkama

```php
function validateAge($age) {
    if ($age < 0) {
        throw new InvalidArgumentException("Starost ne može biti negativna.");
    }
    echo "Valjana starost";
}

try {
    validateAge(-1);
} catch (InvalidArgumentException $e) {
    echo "Greška: " . $e->getMessage();
}
```

Exception Driven Development (EDD)

- Programerski pristup koji se temelji na pretpostavci da će kod tijekom izvođenja neizbježno naići na greške i iznimke.
- Cilj ovog pristupa je strukturirati i napisati kod na način koji anticipira i elegantno upravlja tim greškama, umjesto da se fokusira isključivo na "sretni put" (engl. happy path) gdje se sve izvodi kako je planirano.
- Potiče programere da misle unaprijed o mogućim problemima i načinima njihove obrade, što rezultira robusnijim i pouzdanijim aplikacijama.

Ključne značajke

1. **Anticipacija grešaka**: Programeri aktivno predviđaju moguće iznimke i greške koje mogu nastati tijekom izvođenja programa.
2. **Proaktivno upravljanje iznimkama**: Kod se piše s jasno definiranim blokovima ***try***, ***catch*** i ***finally*** za obradu iznimaka, što omogućava programima da nastave s radom čak i kad dođe do grešaka.
3. **Testiranje robustnosti**: Uključuje intenzivno testiranje kako bi se osiguralo da aplikacija može elegantno upravljati neočekivanim i izvanrednim situacijama.

Prednosti EDD

- Poboljšava stabilnost i pouzdanost aplikacija.
- Olakšava identifikaciju i rješavanje grešaka.
- Povećava pouzdanost softvera kroz detaljno razmatranje potencijalnih točaka neuspjeha.

Primjer

```php
function processFile($filename) {
    try {
        $file = fopen($filename, "r");
        if (!$file) {
            throw new Exception("Datoteka ne može biti otvorena.");
        }
        // Obradi podatke iz datoteke
        while (!feof($file)) {
            $line = fgets($file);
            // Pretpostavimo da postoji neka specifična logika obrade ovdje
        }
        fclose($file);
    } catch (FileNotFoundException $e) {
        // Specifična obrada za slučaj kada datoteka nije pronađena
        echo "Greška: " . $e->getMessage();
    } catch (Exception $e) {
        // Opća obrada za sve ostale iznimke
        echo "Došlo je do greške: " . $e->getMessage();
    } finally {
        echo "Proces obrade je završen.";
    }
}
```

Hvala na pažnji!