PIEMONTE
FONDI STRUTTURALI E DI INVESTIMENTO EUROPEI 2014/2020
fondo sociale europeo
ITS TECNOLOGIE DELL'INFORMAZIONE E DELLA COMUNICAZIONE

Database in PHP

Corso Backend System Integrator

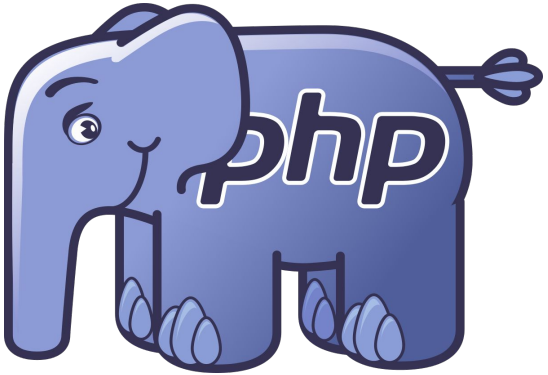
Modulo **Programmazione PHP**

Docente: Dott. Enrico Zimuel

in collaborazione con:
MINISTERO DELL'ISTRUZIONE, DELL'UNIVERSITÀ E DELLA RICERCA
UNIONE EUROPEA
REGIONE PIEMONTE
per una crescita intelligente, sostenibile ed inclusiva
www.regione.piemonte.it/europa2020
INIZIATIVA CO-FINANZIATA CON FSE

Programma

- Database in PHP
- PHP Data Objects (PDO)
  - Collegamento a un DB relazionale
  - Esecuzione di query SQL
  - Gestione degli errori



PDO

- PHP Data Objects (PDO)
- Estensione del PHP per l'accesso ai database tramite un'unica API
- Supporta diversi db: MySQL, SQLite, PostgreSQL, Oracle, SQL Server, DB2, etc

Connessione al DB

```php
$dsn = 'mysql:dbname=testdb;host=127.0.0.1' ;
$user = 'dbuser';
$password = 'dbpass';
try {
    $pdo = new PDO($dsn, $user, $password);
} catch (PDOException $e) {
    printf("Connection failed: %s \n", $e->getMessage());
    exit(1);
}
```

Prepare

- Per poter eseguire una query è necessario "prepararla" (prepare) con dei parametri
- Si utilizza la funzione PDO::prepare()

```
PDO::prepare(string $query, array $options = []): PDOStatement|false
```

- Il risultato è un oggetto PDOStatement o il valore false in caso di errore

Esempio: database

```
CREATE TABLE fruit(
    id INTEGER PRIMARY KEY,
    name TEXT,
    colour TEXT,
    calories INTEGER
);
INSERT INTO fruit VALUES
    (1, 'apple', 'red', 200),
    (2, 'pear', 'green', 250),
    (3, 'orange', 'orange', 300),
    (4, 'lemon', 'yellow', 150);
```

Esempio: PDOStatement

```php
$sql = 'SELECT name, colour, calories
        FROM fruit
        WHERE calories < :calories';
$sth = $pdo->prepare($sql);

var_dump($sth);
// object(PDOStatement)#2 (1) {
//   ["queryString"]=>
//   string(88) "SELECT name, colour, calories..."
// }
```

Eseguire una query

- Una volta creata una query con PDOStatement (prepared statement) è possibile eseguirla con la funzione execute()
- Per poter leggere il risultato dell'esecuzione è necessario effettuare un'operazione di **fetch**:
  - fetchAll() per recuperare tutti i risultati (tutte le righe);
  - fetch() per recuperare un risultato (una riga) alla volta;
  - fetchColumn() per recuperare l'n-esima colonna della riga attuale;
  - fetchObject() recupera un risultato per volta tramite un oggetto, equivalente a fetch(PDO::FETCH_OBJ) o fetch(PDO::FETCH_CLASS)

Esempio: fetchAll con array associativi

```
// SELECT name, colour, calories FROM fruit WHERE calories < :calories
$sth->execute([ ':calories' => 350 ]);
$result = $sth->fetchAll(PDO::FETCH_ASSOC);
var_dump($result);
/*
array(2) {
 [0]=> array(3) {
   ["name"]     => string(5) "apple"
   ["colour"]   => string(3) "red"
   ["calories"] => string(3) "200"
 }
 [1]=> array(3) {
   ["name"]     => string(5) "lemon"
   ["colour"]   => string(6) "yellow"
   ["calories"] => string(3) "150"
}}
*/
```

Esempio: fetchAll con array

```php
// SELECT name, colour, calories FROM fruit WHERE calories < :calories
$sth->execute([ ':calories' => 350 ]);
$result = $sth->fetchAll(PDO::FETCH_NUM);
var_dump($result);
/*
array(2) {
 [0]=> array(3) {
   [0] => string(5) "apple"
   [1] => string(3) "red"
   [2] => string(3) "200"
 }
 [1]=> array(3) {
   [0] => string(5) "lemon"
   [1] => string(6) "yellow"
   [2] => string(3) "150"
}}
*/
```

Esempio: fetchAll con oggetti

```php
// SELECT name, colour, calories FROM fruit WHERE calories < :calories
$sth->execute([ ':calories' => 350 ]);
$result = $sth->fetchAll(PDO::FETCH_OBJ);
var_dump($result);
/*
array(2) {
 [0]=> object(stdClass)#1 (3) {
   ["name"]     => string(5) "apple"
   ["colour"]   => string(3) "red"
   ["calories"] => string(3) "200"
 }
 [1]=> object(stdClass)#2 (3) {
   ["name"]     => string(5) "lemon"
   ["colour"]   => string(6) "yellow"
   ["calories"] => string(3) "150"
}}
*/
```

Esempio: fetchAll con una classe predefinita

```php
class Fruit
{
 public string $name;
 public string $colour;
 public int $calories;
}
```

```php
$sth->execute([ ':calories' => 350 ]);
$result = $sth->fetchAll(PDO::FETCH_CLASS,
Fruit::class);
var_dump($result);
/*
array(2) {
 [0]=> object(Fruit)#1 (3) {
   ["name"]     => string(5) "apple"
   ["colour"]   => string(3) "red"
   ["calories"] => string(3) "200"
 }
 [1]=> object(Fruit)#2 (3) {
   ["name"]     => string(5) "lemon"
   ["colour"]   => string(6) "yellow"
   ["calories"] => string(3) "150"
}}
*/
```

Binding dei parametri

- E' possibile specificare il tipo di ogni parametro nella query SQL
- Questa operazione è chiamata **binding**
- Si utilizzano i metodi PDOStatement::bindValue e PDOStatement::bindParam
- L'operazione di binding dei parametri può essere utile per mitigare attacchi di tipo SQL Injection

Esempio: bind value

```
$sql = 'SELECT name, colour, calories
        FROM fruit
        WHERE calories < :calories';

$sth = $pdo->prepare($sql);
$sth->bindValue(':calories', 150, PDO::PARAM_INT);
$sth->execute();
$result = $sth->fetchAll();
```

Esempio: bind param

```
$sql = 'SELECT name, colour, calories
        FROM fruit
        WHERE calories < :calories';

$calories = 150;
$sth->bindParam(':calories', $calories);
$sth->execute();
$result = $sth->fetchAll();
```

PDO e tipi di dati

- PDO converte i risultati in stringhe, perchè?
- PDO è un driver generico per tutti i database. Alcuni (es. Oracle) potrebbero restituire dati non supportati dal PHP (es. INT con 38 digit)
- E' possibile disabilitare questa opzione impostando a **false** i valori di **PDO::ATTR_STRINGIFY_FETCHES** e **PDO::ATTR_EMULATE_PREPARES** con la funzione PDO::setAttribute()

Esempio

```php
$pdo->setAttribute(PDO::ATTR_STRINGIFY_FETCHES,  false);
$pdo->setAttribute(PDO::ATTR_EMULATE_PREPARES,  false);

$sth = $pdo->prepare($sql);
$sth->execute([
    ':calories' => 350
]);
$result = $sth->fetchAll(PDO::FETCH_ASSOC);
var_dump($result);
```

Esempio: calories è INT

```
/*
array(2) {
 [0]=> array(3) {
   ["name"]     => string(5) "apple"
   ["colour"]   => string(3) "red"
   ["calories"] => int(200)
 }
 [1]=> array(3) {
   ["name"]     => string(5) "lemon"
   ["colour"]   => string(6) "yellow"
   ["calories"] => int(150)
}}
*/
```

ESERCIZIO

- Creare in **MySQL** la tabella fruit vista in precedenza con i campi nome (VARCHAR), colore (VARCHAR) e calorie (INT)
- Inserire **100 record** selezionando a caso tra i nomi (mela, pera, banana, arancia, melone) i colori (rosso, verde, giallo, arancione, marrone) e le calorie (numeri tra 50 e 500)
- Eseguire una query tramite **PDO** per estrarre tutti i frutti di colore rosso con calorie tra 100 e 400

PDO: gestione degli errori

- **PDO** genera l'eccezione PDOException in caso di errori
- Non tutte le classi di PDO generano eccezioni, ad esempio **PDOStatement** restituisce **false** in caso di errore
- E' possibile abilitare la generazione delle eccezioni utilizzando la modalità **PDO:ATTR_ERRMODE**

Esempio

```php
$sql = 'SELECT name, colour, calories
        FROM fruit
        WHERE calor < :calories';
try {
    $pdo = new PDO(/* ... */);
    $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    $sth = $pdo->prepare($sql);
    $sth->execute([ ':calories' => 350 ]);
    $result = $sth->fetchAll(PDO::FETCH_ASSOC);
    var_dump($result);
} catch (PDOException $e) {
    printf("Error: %s\n", $e->getMessage());
    exit(1);
}
// Error: SQLSTATE[42S22]: Column not found:
// 1054 Unknown column 'calor' in 'where clause'
```

Debug dump params

- **PDOStatement** ha un metodo per visualizzare la query SQL inviata al database (debugDumpParams)

```php
try {
    $sth = $pdo->prepare($sql);
    $sth->execute([ ':calories' => 350 ]);
    $result = $sth->fetchAll(PDO::FETCH_ASSOC);
    var_dump($result);
} catch (PDOException $e) {
    $sth->debugDumpParams();
}
```

```
/*
Sent SQL: [60] SELECT name,
colour, calories
FROM fruit
WHERE calor < '350'
*/
```

Esercizio (da consegnare)

- Modificare l’esercizio della Login, consegnato l’ultima volta, prelevando i dati di autenticazione (email e password) da un database **MySQL**
- Nella consegna dell’esercizio, oltre al codice PHP, è necessario inserire anche un file **.sql** con l’**export del database**

Grazie dell’attenzione!

Per informazioni:
enrico.zimuel@its-ictpiemonte.it

POR Piemonte
FSE 2014-2020