Pràctica 3 PHP + MariaDB

Guillem Serrat Bahí

Índex

Introducció

Tipus de codis

Durant l'activitat es presenten 2 tipus de codis:

- Els codis base que s'han ofert com a enunciat en la pàgina web d'W3Schools
- Codis propis, modificant l'entorn de MariaDB (noms de la BD i la taula, noms dels camps, etc) i noms de variables de PHP, entre altres aspectes.

Mentre que els codis base d'W3Schools s'han reproduït amb absoluta exactitud, els codis anomenats "propis" s'han desenvolupat canviant aspectes que afecten no només la nomenclatura d'alguns objectes, sinó el comportament d'alguns codis respecte als originals d'W3Schools.

Explicacions dels codis

Per documentar i entendre cada codi realitzat, dins de cada un hi ha comentaris descriptius, però també es fa una explicació amb model d'informe abans de cada codi, on s'explica quin és l'objectiu, les ordres (PHP) i les instruccions (SQL) treballades.

L'explicació amb model d'informe es farà sobre el codi d'exemple d'W3Schools, amb els seus noms de variables i les seves ordres i instruccions, no sobre els codis propis.

En la documentació en format d'informe, únicament s'exposarà les novetats d'aquell codi, sense detallar conceptes ja treballats en exemples anteriors.

Primera connexió (i ordres PHP bàsiques)

Explicació

L'objectiu d'aquest exemple és verificar que podem connectar-nos a una base de dades.

Per poder connectar-nos a un gestor de base de dades, sempre especificarem 4 elements:

- El servidor (sempre la IP de localhost)
- L'usuari
- La contrasenya
- La base de dades a connectar

Això ho fem mitjançant variables:

```
$servername = "127.0.0.1";
$username = "root";
$password = "fjeclot";
$dbname = "myDBPDO";
```

Seguidament, crearem la connexió en sí. La connexió és un objecte de la classe PDO de PHP. Dins d'aquest objecte especificarem els 4 elements necessaris per poder realitzar la connexió

```
$conn = new PDO("mysql:host=$servername;dbname=$dbname",
$username, $password);
```

Per tant, la variable $conn és la connexió, i especifiquem les propietats d'aquest nou objecte amb els valors comentats anteriorment. En aquest cas ens connectem:

- Al localhost
- Amb l'usuari "root"
- Amb la contrasenya "fjeclot"
- A la base de dades "myDBPDO" (prèviament creada)

A partir d'aquí, tot el que tingui a veure amb la connexió, farem servir la variable $conn.

Per poder detectar possibles errors durant l'execució del codi, hem de fer servir la funció setAttribute per rescatar el tipus d'error, que més endavant podrà ser interceptat per un "catch"

```
$conn->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);
```

Com hem comentat abans, fem servir la variable $conn.

Aquesta línia té la funció de llençar un error de tipus PDOException en cas de que hi hagi algun error en l'execució del codi.

Mitjançant un try - catch, podem agafar aquesta PDOException i executar codi en cas d'haver un error, per no deixar la pantalla en blanc.

Mitjaçant el "try", intentem reproduir el següent codi:

```
try {
$conn = new PDO("mysql:host=$servername;dbname=$dbname",
$username, $password);

$conn->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);

echo "Connectat a la base de dades.";
}
```

En cas que hi hagi algun error, PHP llençarà una excepció de tipus PDOException, i el catch actuarà, executant el codi que hi ha en el seu interior:

```
catch(PDOException $e) {
echo "No es pot connectar. Motiu: " . $e->getMessage();
}
```

En el cas de que el catch intercepti alguna PDOException, la variable $e la dona la propia PDOException, la variable $e conté l'error que ha provocat el catch.

Dins del catch, imprimir el missatge d'error, a partir de la funció getMessage() a la variable $e (que conté l'error)

Finalment, tanquem la connexió amb la següent instrucció:

```
$conn = null;
```

Resultat

Connectat a la base de dades.

Codi en PHP

```php
<!DOCTYPE html>
<html lang="ca">
    <head>
        <title>Exemple PDO Guillem Serrat</title>
        <meta http-equiv="Content-type" content="text/html; charset=UTF-8">
    </head>
    <body>
        <?php
            // Definim els paràmetres per realitzar la connexió
            $servername = "127.0.0.1";      // El servidor on ens connectarem
            $username = "root";             // L'usuari de MariaDB
            $password = "fjeclot";          // La contrasenya de l'usuari
            $dbname = "myDBPDO";            // La base de dades que farem servir

            try {
            // Definim un nou objecte de la classe PDO amb els atributs: host al que ens connectarem, la base de dades, el nom d'usuari i contrasenya.
            $conn = new PDO("mysql:host=$servername;dbname=$dbname", $username, $password);

            // A través de la funció setAttribute, agafem el tipus d'error en cas de que n'hi hagi algun
            $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

            echo "Connectat a la base de dades.";

            } catch(PDOException $e) {                              // En cas de que hi hagi un error, PHP llença una PDOException, i la variable $e agafa aquesta excepció
                echo "No es pot connectar. Motiu: " . $e->getMessage();     // Es mostra el missatge
            }

            $conn = null;                                           // Tanca la connexió amb la BBDD
        ?>
        <h1>Codi en PHP</h1>
        <?php
            show_source("exemple.php");
        ?>
    </body>
</html>
```

Codi

```
<!DOCTYPE html>
<html lang="ca">
    <head>
        <title>Exemple PDO Guillem Serrat</title>
                <meta  http-equiv="Content-type"  content="text/html;
charset=UTF-8">
    </head>
    <body>
        <?php
            // Definim els paràmetres per realitzar la connexió
                $servername = "127.0.0.1";        // El servidor on ens
connectarem
            $username = "root";              // L'usuari de MariaDB
                $password = "fjeclot";             // La contrasenya de
l'usuari
              $dbname = "myDBPDO";              // La base de dades que
farem servir

            try {
                  // Definim un nou objecte de la classe PDO amb els
atributs: host al que ens connectarem, la base de dades, el nom
d'usuari i contrasenya.
                $conn = new PDO("mysql:host=$servername;dbname=$dbname",
$username, $password);

                // A través de la funció setAttribute, agafem el tipus
d'error en cas de que n'hi hagi algun
                                    $conn->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);

            echo "Connectat a la base de dades.";

                                        }   catch(PDOException   $e)   {
// En cas de que hi hagi un error, PHP llença una PDOException, i la
variable $e agafa aquesta excepció
                 echo "No es pot connectar. Motiu: " . $e->getMessage();
// Es mostra el missatge
            }

                                                    $conn    =    null;
// Tanca la connexió amb la BBDD
```

```
        ?>
        <h1>Codi en PHP</h1>
        <?php
            show_source("exemple.php");
        ?>
    </body>
</html>
```

Exemple de creació de base de dades (Ex1)

Explicació de l'exemple

L'objectiu d'aquest exemple és crear una base de dades.

Primer de tot, en cas de voler crear una base de dades, a l'hora de crear la connexió NO s'ha d'especificar l'argument "dbname" (ja que no ens estem connectant a cap)

En cas de voler executar instruccions de MySQL (create, insert, update, delete, etc) hem de fer servir el mètode exec() de la classe PDO (recordem que $conn és un objecte de la classe PDO), on l'argument del mètode (el que hi ha dins del parèntesi) és el que s'executarà.

```
$conn->exec(<ordeSQL>);
```

Normalment, enlloc d'escriure directament el codi com l'argument del mètode, s'escriu dins d'una variable i després "s'executa la variable"

```
$sql = "CREATE DATABASE myDBPDO";
$conn->exec($sql);
```

En aquest cas, estem creant una base de dades anomenada "myDBPDO"

Codi W3Schools

Resultat

Database created successfully

Codi en PHP

```php
<!DOCTYPE html>
<html lang="ca">
    <head>
        <title>Exemple Creació Base de Dades Guillem Serrat</title>
        <meta http-equiv="Content-type" content="text/html; charset=UTF-8">
    </head>
    <body>
        <?php
            // Definim els paràmetres per realitzar la connexió
            $servername = "127.0.0.1";
            $username = "root";
            $password = "fjeclot";

            try {
                // Definim un nou objecte de la classe PDO amb els atributs: host al que ens connectarem, el nom d'usuari i contrasenya.
                $conn = new PDO("mysql:host=$servername", $username, $password);

                // A través de la funció setAttribute, agafem el tipus d'error en cas de que n'hi hagi algun
                $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

                // Definim una variable que és l'ordre que s'executara, la qual és crear una BBDD nova amb nom "myDBPDO"
                $sql = "CREATE DATABASE myDBPDO";

                // Dins de la connexió, executem la variable sql (que és l'ordre vista anteriorment)
                $conn->exec($sql);

                // Si es crea correctament
                echo "Database created successfully<br>";
            } catch(PDOException $e) { // En cas de que hi hagi un error, PHP llença una PDOException, i la variable $e agafa aquesta excepció
                // Si no es crea correctament, imprimeix el missatge d'error de PDOException
                echo $sql . "<br>" . $e->getMessage();
            }

            $conn = null; // Tanca la connexió amb la BBDD
            ?>
        <h1>Codi en PHP</h1>
        <?php
            show_source("ex1.php");
        ?>
    </body>
</html>
```

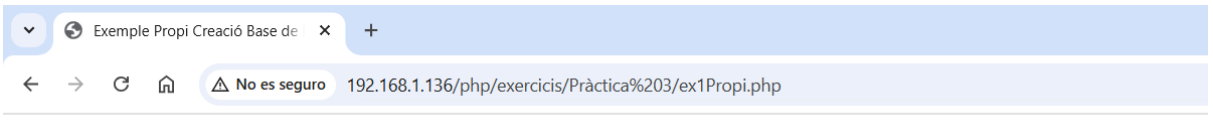
Codi

```
<!DOCTYPE html>
<html lang="ca">
    <head>
        <title>Exemple Creació Base de Dades Guillem Serrat</title>
                    <meta  http-equiv="Content-type"  content="text/html;
charset=UTF-8">
    </head>
    <body>
        <?php
            // Definim els paràmetres per realitzar la connexió
            $servername = "127.0.0.1";
            $username = "root";
            $password = "fjeclot";

            try {
                    // Definim un nou objecte de la classe PDO amb els
atributs: host al que ens connectarem, el nom d'usuari i contrasenya.
                    $conn = new PDO("mysql:host=$servername", $username,
$password);

                // A través de la funció setAttribute, agafem el tipus
d'error en cas de que n'hi hagi algun
                                    $conn->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);

                // Definim una variable que és l'ordre que s'executara,
la qual és crear una BBDD nova amb nom "myDBPDO"
                $sql = "CREATE DATABASE myDBPDO";

                // Dins de la connexió, executem la variable sql (que
és l'ordre vista anteriorment)
                $conn->exec($sql);

                // Si es crea correctament
                echo "Database created successfully<br>";
               } catch(PDOException $e) { // En cas de que hi hagi un
error, PHP llença una PDOException, i la variable $e agafa aquesta
excepció
                    // Si no es crea correctament, imprimeix el missatge
d'error de PDOException
                echo $sql . "<br>" . $e->getMessage();
```

```
            }

            $conn = null; // Tanca la connexió amb la BBDD
            ?>
        <h1>Codi en PHP</h1>
        <?php
            show_source("ex1.php");
        ?>
    </body>
</html>
```

Codi Propi

Resultat



Base de dades dbGserrat creada correctament

Codi en PHP

```php
<!DOCTYPE html>
<html lang="ca">
    <head>
        <title>Exemple Propi Creació Base de Dades Guillem Serrat</title>
        <meta http-equiv="Content-type" content="text/html; charset=UTF-8">
    </head>
    <body>
        <?php
            // Definim els paràmetres per realitzar la connexió
            $servidor = "127.0.0.1";
            $usuari = "root";
            $contrasenya = "fjeclot";

            try {
                // Definim un nou objecte de la classe PDO amb els atributs: host al que ens connectarem, el nom d'usuari i contrasenya.
                $connexio = new PDO("mysql:host=$servidor", $usuari, $contrasenya);

                // A través de la funció setAttribute, agafem el tipus d'error en cas de que n'hi hagi algun
                $connexio->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

                // Definim una variable per indicar el nom de la base de dades a crear
                $nomBDaCrear = "dbGserrat";

                // Definim una variable que és l'ordre que s'executara, la qual és crear una BBDD nova amb nom el valor de la variable $nomBDaCrear
                $instruccioSQL = "CREATE DATABASE $nomBDaCrear";

                // Dins de la connexió, executem la variable sql (que és l'ordre vista anteriorment)
                $connexio->exec($instruccioSQL);

                // Si es crea correctament
                echo "Base de dades $nomBDaCrear creada correctament<br>";
            } catch(PDOException $e) { // En cas de que hi hagi un error, PHP llença una PDOException, i la variable $e agafa aquesta excepció
                // Si no es crea correctament, imprimeix el missatge d'error de PDOException
                echo "La base de dades no s'ha creat correctament" . "<br>" . $e->getMessage();
            }

            $connexio = null; // Tanca la connexió amb la BBDD
            ?>
        <h1>Codi en PHP</h1>
        <?php
            show_source("ex1Propi.php");
        ?>
    </body>
</html>
```

Codi

```
<!DOCTYPE html>
<html lang="ca">
    <head>
                <title>Exemple  Propi  Creació  Base  de  Dades  Guillem
Serrat</title>
                  <meta  http-equiv="Content-type"  content="text/html;
charset=UTF-8">
    </head>
    <body>
        <?php
            // Definim els paràmetres per realitzar la connexió
            $servidor = "127.0.0.1";
            $usuari = "root";
            $contrasenya = "fjeclot";

            try {
                    // Definim un nou objecte de la classe PDO amb els
atributs: host al que ens connectarem, el nom d'usuari i contrasenya.
                    $connexio = new PDO("mysql:host=$servidor", $usuari,
$contrasenya);

                // A través de la funció setAttribute, agafem el tipus
d'error en cas de que n'hi hagi algun
                            $connexio->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);

                // Definim una variable per indicar el nom de la base
de dades a crear
                $nomBDaCrear = "dbGserrat";

                // Definim una variable que és l'ordre que s'executara,
la  qual  és  crear  una  BBDD  nova  amb  nom  el  valor  de  la  variable
$nomBDaCrear
                $instruccioSQL = "CREATE DATABASE $nomBDaCrear";

                // Dins de la connexió, executem la variable sql (que
és l'ordre vista anteriorment)
                $connexio->exec($instruccioSQL);

                // Si es crea correctament
```

```
                            echo "Base de dades $nomBDaCrear creada
correctament<br>";
                } catch(PDOException $e) { // En cas de que hi hagi un
error, PHP llença una PDOException, i la variable $e agafa aquesta
excepció
                    // Si no es crea correctament, imprimeix el missatge
d'error de PDOException
                    echo "La base de dades no s'ha creat correctament" .
"<br>" . $e->getMessage();
                }

                $connexio = null; // Tanca la connexió amb la BBDD
                ?>
            <h1>Codi en PHP</h1>
            <?php
                show_source("ex1Propi.php");
            ?>
        </body>
</html>
```

Exemple de creació d'una taula (Ex2)

Explicació de l'exemple

L'objectiu d'aquest exemple és crear una taula dins de la base de dades creada anteriorment.

A partir d'ara, com la base de dades està creada, haurem d'especificar l'argument "dbname" dins de la connexió, per treballar sobre aquella base de dades.

Per crear una taula, ho farem de la mateixa manera que per crear una base de dades. Establirem la instrucció SQL dins d'una variable, i mitjançant el mètode exec() executarem la instrucció.

En aquest cas, al ser la creació d'una taula, la instrucció és la següent:

```
$sql = "CREATE TABLE MyGuests (
id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
firstname VARCHAR(30) NOT NULL,
lastname VARCHAR(30) NOT NULL,
email VARCHAR(50),
reg_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP
)";
```

On especifiquem el nom de la taula MyGuests, i:

- Una id que s'autoincrementa
- El primer nom
- El segon nom
- L'email
- Data de registre, automàtic que registre la data la qual s'afageix el registre a la taula

Codi W3Schools

Resultat

Table MyGuests created successfully

90 %

Codi en PHP

```php
<!DOCTYPE html>
<html lang="ca">
    <head>
        <title>Exemple Creacio Taula Guillem Serrat</title>
        <meta http-equiv="Content-type" content="text/html; charset=UTF-8">
    </head>
    <body>
            <?php
                // Definim els paràmetres per realitzar la connexió
                $servername = "127.0.0.1";
                $username = "root";
                $password = "fjeclot";
                $dbname = "myDBPDO";

                try {
                    // Definim un nou objecte de la classe PDO amb els atributs: host al que ens connectarem, el nom d'usuari i contrasenya.
                    $conn = new PDO("mysql:host=$servername;dbname=$dbname", $username, $password);

                    // A través de la funció setAttribute, agafem el tipus d'error en cas de que n'hi hagi algun
                    $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

                    // Definim una variable que és l'ordre que s'executara, la qual és crear una taula de nom MyGuests. La BBDD en la cual es treballarà està especificat a la connexió
                    $sql = "CREATE TABLE MyGuests (
                    id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
                    firstname VARCHAR(30) NOT NULL,
                    lastname VARCHAR(30) NOT NULL,
                    email VARCHAR(50),
                    reg_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP
                    )";

                    // Dins de la connexió, executem la variable sql (que és l'ordre vista anteriorment)
                    $conn->exec($sql);

                    echo "Table MyGuests created successfully";

                } catch(PDOException $e) {                          // En cas de que hi hagi un error, PHP llença una PDOException, i la variable $e agafa aquesta excepció
                    echo $sql . "<br>" . $e->getMessage();          // Si no es crea correctament, imprimeix el missatge d'error de PDOException
                }
                $conn = null;                                       // Tanca la connexió amb la BBDD
            ?>
        <h1>Codi en PHP</h1>
        <?php
        show_source("ex2.php");
        ?>
    </body>
</html>
```

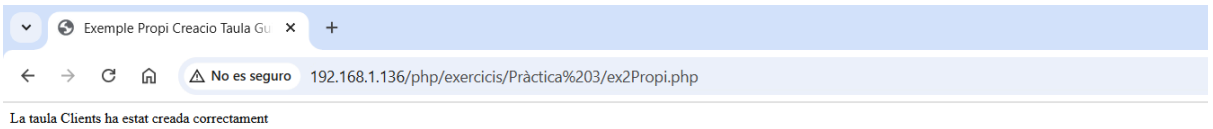
Codi

```
<!DOCTYPE html>
<html lang="ca">
    <head>
        <title>Exemple Creacio Taula Guillem Serrat</title>
                    <meta  http-equiv="Content-type"  content="text/html;
charset=UTF-8">
    </head>
    <body>
            <?php
                // Definim els paràmetres per realitzar la connexió
                $servername = "127.0.0.1";
                $username = "root";
                $password = "fjeclot";
                $dbname = "myDBPDO";

                try {
                     // Definim un nou objecte de la classe PDO amb els
atributs: host al que ens connectarem, el nom d'usuari i contrasenya.
                                                        $conn   =   new
PDO("mysql:host=$servername;dbname=$dbname", $username, $password);

                      // A través de la funció setAttribute, agafem el
tipus d'error en cas de que n'hi hagi algun
                                    $conn->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);

                           // Definim una variable que és l'ordre que
s'executara, la qual és crear una taula de nom MyGuests. La BBDD en la
cual es treballarà està especificat a la connexió
                    $sql = "CREATE TABLE MyGuests (
                    id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
                    firstname VARCHAR(30) NOT NULL,
                    lastname VARCHAR(30) NOT NULL,
                    email VARCHAR(50),
                        reg_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON
UPDATE CURRENT_TIMESTAMP
                    )";

                      // Dins de la connexió, executem la variable sql
(que és l'ordre vista anteriorment)
                    $conn->exec($sql);
```

```
                    echo "Table MyGuests created successfully";

                                        } catch(PDOException $e) {
// En cas de que hi hagi un error, PHP llença una PDOException, i la
variable $e agafa aquesta excepció
                                echo $sql . "<br>" . $e->getMessage();
//  Si no es crea correctament, imprimeix el missatge d'error de
PDOException
                }
                                                        $conn = null;
// Tanca la connexió amb la BBDD
            ?>
        <h1>Codi en PHP</h1>
        <?php
        show_source("ex2.php");
        ?>
    </body>
</html>
```

Codi Propi

Resultat



La taula Clients ha estat creada correctament

Codi en PHP

```php
<!DOCTYPE html>
<html lang="ca">
    <head>
        <title>Exemple Propi Creacio Taula Guillem Serrat</title>
        <meta http-equiv="Content-type" content="text/html; charset=UTF-8">
    </head>
    <body>
            <?php
                // Definim els paràmetres per realitzar la connexió
                $servidor = "127.0.0.1";
                $usuari = "root";
                $contrasenya = "fjeclot";
                $nomDB = "dbGserrat";

                try {
                    // Definim un nou objecte de la classe PDO amb els atributs: host al que ens connexioectarem, el nom d'usuari i contrasenya.
                    $connexio = new PDO("mysql:host=$servidor;dbname=$nomDB", $usuari, $contrasenya);

                    // A través de la funció setAttribute, agafem el tipus d'error en cas de que n'hi hagi algun
                    $connexio->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

                    // Definim una variable amb el nom de la taula a crear
                    $nomTaulaACrear = "Clients";

                    // Definim una variable que és l'ordre que s'executara, la qual és crear una taula de nom Clients. La BBDD en la cual es treballarà està especificat a la connexioexió
                    $instruccioSQL = "CREATE TABLE $nomTaulaACrear (
                    id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
                    nom VARCHAR(30) NOT NULL,
                    cognom VARCHAR(30) NOT NULL,
                    mail VARCHAR(50),
                    dataRegistre TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP
                    )";

                    // Dins de la connexió, executem la variable sql (que és l'ordre vista anteriorment)
                    $connexio->exec($instruccioSQL);

                    echo "La taula $nomTaulaACrear ha estat creada correctament";

                } catch(PDOException $e) {                          // En cas de que hi hagi un error, PHP llença una PDOException, i la variable $e agafa aquesta excepció
                    echo "La taula no s'ha creat correctament" . "<br>" . $e->getMessage();              // Si no es crea correctament, imprimeix el missatge d'error de PDOException
                }
                $connexio = null;                                       // Tanca la connexió amb la BBDD
            ?>
        <h1>Codi en PHP</h1>
        <?php
        show_source("ex2Propi.php");
        ?>
    </body>
</html>
```

Codi

```
<!DOCTYPE html>
<html lang="ca">
    <head>
        <title>Exemple Propi Creacio Taula Guillem Serrat</title>
                    <meta  http-equiv="Content-type"  content="text/html;
charset=UTF-8">
    </head>
    <body>
            <?php
                // Definim els paràmetres per realitzar la connexió
                $servidor = "127.0.0.1";
                $usuari = "root";
                $contrasenya = "fjeclot";
                $nomDB = "dbGserrat";

                try {
                      // Definim un nou objecte de la classe PDO amb els
atributs:  host  al  que  ens  connexioectarem,  el  nom  d'usuari  i
contrasenya.
                                                        $connexio   =   new
PDO("mysql:host=$servidor;dbname=$nomDB", $usuari, $contrasenya);

                       // A través de la funció setAttribute, agafem el
tipus d'error en cas de que n'hi hagi algun
                                $connexio->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);

                       // Definim una variable amb el nom de la taula a
crear
                    $nomTaulaACrear = "Clients";

                            // Definim una variable que és l'ordre que
s'executara, la qual és crear una taula de nom Clients. La BBDD en la
cual es treballarà està especificat a la connexioexió
                    $instruccioSQL = "CREATE TABLE $nomTaulaACrear (
                    id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
                    nom VARCHAR(30) NOT NULL,
                    cognom VARCHAR(30) NOT NULL,
                    mail VARCHAR(50),
                     dataRegistre TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON
UPDATE CURRENT_TIMESTAMP
```

```
                    )";

                        // Dins de la connexió, executem la variable sql
(que és l'ordre vista anteriorment)
                    $connexio->exec($instruccioSQL);

                        echo "La taula $nomTaulaACrear ha estat creada
correctament";

                                            } catch(PDOException $e) {
// En cas de que hi hagi un error, PHP llença una PDOException, i la
variable $e agafa aquesta excepció
                    echo "La taula no s'ha creat correctament" . "<br>"
. $e->getMessage();                    // Si no es crea correctament,
imprimeix el missatge d'error de PDOException
                }
                                                    $connexio = null;
// Tanca la connexió amb la BBDD
            ?>
        <h1>Codi en PHP</h1>
        <?php
        show_source("ex2Propi.php");
        ?>
    </body>
</html>
```

Exemple d'inserció de dades (Ex3)

Explicació de l'exemple

L'objectiu d'aquest exemple és inserir dades dins la taula creada anteriorment

Per inserir dades a una taula, ho farem de la mateixa manera que per crear una taula. Establirem la instrucció SQL dins d'una variable, i mitjançant el mètode exec() executarem la instrucció.

Per inserir dades a una taula, la instrucció és la següent:

```
$sql = "INSERT INTO MyGuests (firstname, lastname, email)
VALUES ('John', 'Doe', 'john@example.com')";
```

Codi W3Schools

Resultat

New record created successfully

Codi en PHP

```php
<!DOCTYPE html>
<html lang="ca">
    <head>
        <title>Exemple Insercio Dades Guillem Serrat</title>
        <meta http-equiv="Content-type" content="text/html; charset=UTF-8">
    </head>
    <body>
        <?php
            // Definim els paràmetres per realitzar la connexió
            $servername = "127.0.0.1";
            $username = "root";
            $password = "fjeclot";
            $dbname = "myDBPDO";

            try {
                // Definim un nou objecte de la classe PDO amb els atributs: host al que ens connectarem, el nom d'usuari i contrasenya.
                $conn = new PDO("mysql:host=$servername;dbname=$dbname", $username, $password);

                // A través de la funció setAttribute, agafem el tipus d'error en cas de que n'hi hagi algun
                $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

                // Definim una variable que és l'ordre que s'executara, la qual és inserir dades a la taula MyGuests. La BBDD en la cual es treballarà està especificat a la connexió
                $sql = "INSERT INTO MyGuests (firstname, lastname, email)
                VALUES ('John', 'Doe', 'john@example.com')";

                // Dins de la connexió, executem la variable sql (que és l'ordre vista anteriorment)
                $conn->exec($sql);
                echo "New record created successfully";

            } catch(PDOException $e) {                      // En cas de que hi hagi un error, PHP llença una PDOException, i la variable $e agafa aquesta excepció
                echo $sql . "<br>" . $e->getMessage();      // Si no es crea correctament, imprimeix el missatge d'error de PDOException
            }
            $conn = null;                                   // Tanca la connexió amb la BBDD
        ?>

        <h1>Codi en PHP</h1>
        <?php
        show_source("ex3.php");
        ?>
    </body>
</html>
```

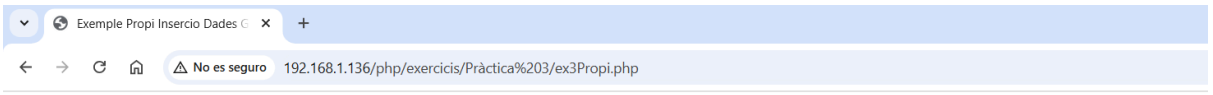
Codi

```
<!DOCTYPE html>
<html lang="ca">
    <head>
        <title>Exemple Insercio Dades Guillem Serrat</title>
                    <meta  http-equiv="Content-type"  content="text/html;
charset=UTF-8">
    </head>
    <body>
        <?php
            // Definim els paràmetres per realitzar la connexió
            $servername = "127.0.0.1";
            $username = "root";
            $password = "fjeclot";
            $dbname = "myDBPDO";

            try {
                    // Definim un nou objecte de la classe PDO amb els
atributs: host al que ens connectarem, el nom d'usuari i contrasenya.
                                                        $conn    =    new
PDO("mysql:host=$servername;dbname=$dbname", $username, $password);

                // A través de la funció setAttribute, agafem el tipus
d'error en cas de que n'hi hagi algun
                                    $conn->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);

                // Definim una variable que és l'ordre que s'executara,
la qual és inserir dades a la taula MyGuests. La BBDD en la cual es
treballarà està especificat a la connexió
                    $sql = "INSERT INTO MyGuests (firstname, lastname,
email)
                VALUES ('John', 'Doe', 'john@example.com')";

                 // Dins de la connexió, executem la variable sql (que
és l'ordre vista anteriorment)
                $conn->exec($sql);
                echo "New record created successfully";

            } catch(PDOException $e) {                          // En cas
de que hi hagi un error, PHP llença una PDOException, i la variable $e
agafa aquesta excepció
```

```
                echo $sql . "<br>" . $e->getMessage();      // Si no es
crea correctament, imprimeix el missatge d'error de PDOException
            }
            $conn = null;                                   // Tanca la
connexió amb la BBDD
        ?>

        <h1>Codi en PHP</h1>
        <?php
        show_source("ex3.php");
        ?>
    </body>
</html>
```

Codi Propi

Resultat



Registre inserit correctament

Codi en PHP

```php
<!DOCTYPE html>
<html lang="ca">
    <head>
        <title>Exemple Propi Insercio Dades Guillem Serrat</title>
        <meta http-equiv="Content-type" content="text/html; charset=UTF-8">
    </head>
    <body>
        <?php
            // Definim els paràmetres per realitzar la connexió
            $servidor = "127.0.0.1";
            $usuari = "root";
            $contrasenya = "fjeclot";
            $nomDB = "dbGserrat";
            $nomTaula = "Clients";

            try {
                // Definim un nou objecte de la classe PDO amb els atributs: host al que ens connectarem, el nom d'usuari i contrasenya.
                $connexio = new PDO("mysql:host=$servidor;dbname=$nomDB", $usuari, $contrasenya);

                // A través de la funció setAttribute, agafem el tipus d'error en cas de que n'hi hagi algun
                $connexio->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

                // Definim una variable que és l'ordre que s'executara, la qual és inserir dades a la taula Clients. La BBDD en la cual es treballarà està especificat a la connexió
                $instruccioSQL = "INSERT INTO $nomTaula (nom, cognom, mail)
                VALUES ('Guillem', 'Serrat', 'guillem@example.com')";

                // Dins de la connexió, executem la variable sql (que és l'ordre vista anteriorment)
                $connexio->exec($instruccioSQL);
                echo "Registre inserit correctament";

            } catch(PDOException $e) {                          // En cas de que hi hagi un error, PHP llença una PDOException, i la variable $e agafa aquesta excepció
                echo "El registre no s'ha creat correctament" . "<br>" . $e->getMessage();      // Si no es crea correctament, imprimeix el missatge d'error de PDOException
            }
            $connexio = null;                                   // Tanca la connexió amb la BBDD
        ?>

        <h1>Codi en PHP</h1>
        <?php
        show_source("ex3Propi.php");
        ?>
    </body>
</html>
```

Codi

```
<!DOCTYPE html>
<html lang="ca">
    <head>
        <title>Exemple Propi Insercio Dades Guillem Serrat</title>
                    <meta  http-equiv="Content-type"  content="text/html;
charset=UTF-8">
    </head>
    <body>
        <?php
            // Definim els paràmetres per realitzar la connexió
            $servidor = "127.0.0.1";
            $usuari = "root";
            $contrasenya = "fjeclot";
            $nomDB = "dbGserrat";
            $nomTaula = "Clients";

            try {
                    // Definim un nou objecte de la classe PDO amb els
atributs: host al que ens connectarem, el nom d'usuari i contrasenya.
                                                    $connexio    =    new
PDO("mysql:host=$servidor;dbname=$nomDB", $usuari, $contrasenya);

                // A través de la funció setAttribute, agafem el tipus
d'error en cas de que n'hi hagi algun
                                $connexio->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);

                // Definim una variable que és l'ordre que s'executara,
la qual és inserir dades a la taula Clients. La BBDD en la cual es
treballarà està especificat a la connexió
                 $instruccioSQL = "INSERT INTO $nomTaula (nom, cognom,
mail)
               VALUES ('Guillem', 'Serrat', 'guillem@example.com')";

                 // Dins de la connexió, executem la variable sql (que
és l'ordre vista anteriorment)
               $connexio->exec($instruccioSQL);
               echo "Registre inserit correctament";
```

```
                                        } catch(PDOException $e) {
// En cas de que hi hagi un error, PHP llença una PDOException, i la
variable $e agafa aquesta excepció
                echo "El registre no s'ha creat correctament" . "<br>"
. $e->getMessage();        // Si no es crea correctament, imprimeix el
missatge d'error de PDOException
            }
                                                $connexio = null;
// Tanca la connexió amb la BBDD
        ?>

        <h1>Codi en PHP</h1>
        <?php
        show_source("ex3Propi.php");
        ?>
    </body>
</html>
```

Exemple d'obtenció del darrer ID (Ex4)

Explicació de l'exemple

L'objectiu d'aquest exemple és obtenir la ID de l'últim registre insertat.

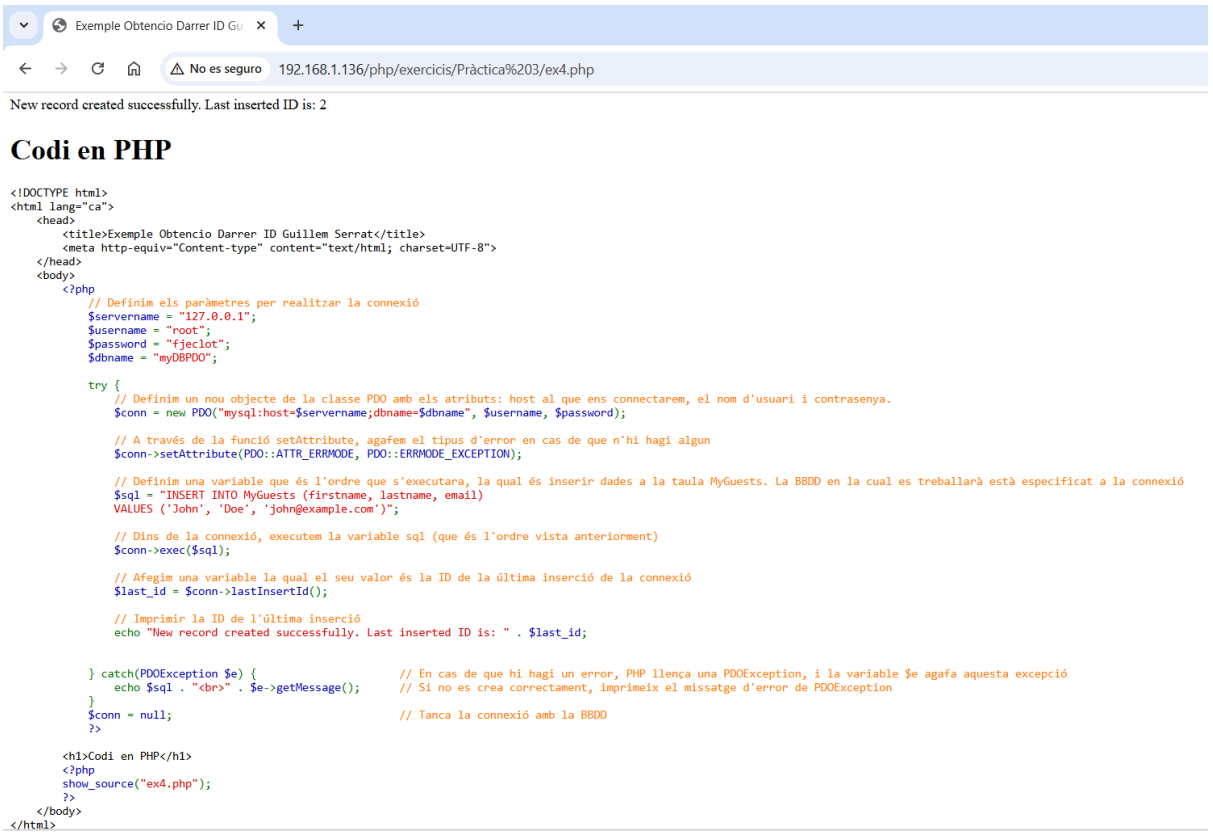
En cas d'inserir o actualizar un registre dins d'una taula, tenim la possibilitat d'obtenir l'ID d'aquell registre inserit o modificat, sempre i quan l'ID sigui autoincremental.

Per obtenir la l'última ID farem servir el mètode lastInsertId de la classe PDO, i la desarem a una variable anomenada last_id

```
$last_id = $conn->lastInsertId();
```

Codi W3Schools

Resultat



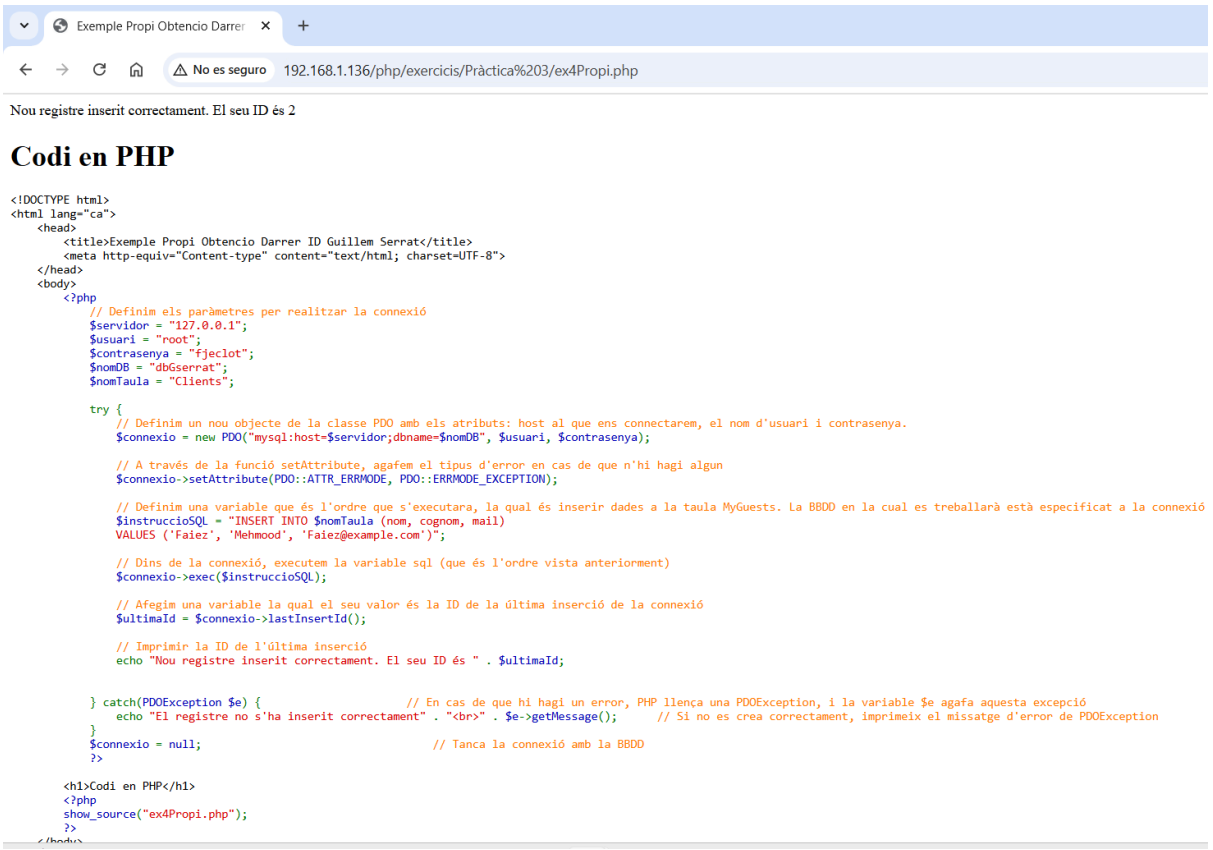
Codi

```
<!DOCTYPE html>
<html lang="ca">
    <head>
        <title>Exemple Obtencio Darrer ID Guillem Serrat</title>
                    <meta  http-equiv="Content-type"  content="text/html;
charset=UTF-8">
    </head>
    <body>
        <?php
            // Definim els paràmetres per realitzar la connexió
            $servername = "127.0.0.1";
            $username = "root";
            $password = "fjeclot";
            $dbname = "myDBPDO";

            try {
                    // Definim un nou objecte de la classe PDO amb els
atributs: host al que ens connectarem, el nom d'usuari i contrasenya.
                                                        $conn    =    new
PDO("mysql:host=$servername;dbname=$dbname", $username, $password);

                // A través de la funció setAttribute, agafem el tipus
d'error en cas de que n'hi hagi algun
                                    $conn->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);

                // Definim una variable que és l'ordre que s'executara,
la qual és inserir dades a la taula MyGuests. La BBDD en la cual es
treballarà està especificat a la connexió
                    $sql = "INSERT INTO MyGuests (firstname, lastname,
email)
                VALUES ('John', 'Doe', 'john@example.com')";

                // Dins de la connexió, executem la variable sql (que
és l'ordre vista anteriorment)
                $conn->exec($sql);

                // Afegim una variable la qual el seu valor és la ID de
la última inserció de la connexió
                $last_id = $conn->lastInsertId();
```

```
                // Imprimir la ID de l'última inserció
                 echo "New record created successfully. Last inserted ID
is: " . $last_id;


             } catch(PDOException $e) {                        // En cas
de que hi hagi un error, PHP llença una PDOException, i la variable $e
agafa aquesta excepció
                 echo $sql . "<br>" . $e->getMessage();       // Si no es
crea correctament, imprimeix el missatge d'error de PDOException
            }
             $conn = null;                                    // Tanca la
connexió amb la BBDD
            ?>

        <h1>Codi en PHP</h1>
        <?php
        show_source("ex4.php");
        ?>
    </body>
</html>
```

Codi Propi

Resultat




```php
<!DOCTYPE html>
<html lang="ca">
    <head>
        <title>Exemple Propi Obtencio Darrer ID Guillem Serrat</title>
        <meta http-equiv="Content-type" content="text/html; charset=UTF-8">
    </head>
    <body>
        <?php
            // Definim els paràmetres per realitzar la connexió
            $servidor = "127.0.0.1";
            $usuari = "root";
            $contrasenya = "fjeclot";
            $nomDB = "dbGserrat";
            $nomTaula = "Clients";

            try {
                // Definim un nou objecte de la classe PDO amb els atributs: host al que ens connectarem, el nom d'usuari i contrasenya.
                $connexio = new PDO("mysql:host=$servidor;dbname=$nomDB", $usuari, $contrasenya);

                // A través de la funció setAttribute, agafem el tipus d'error en cas de que n'hi hagi algun
                $connexio->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

                // Definim una variable que és l'ordre que s'executara, la qual és inserir dades a la taula MyGuests. La BBDD en la cual es treballarà està especificat a la connexió
                $instruccioSQL = "INSERT INTO $nomTaula (nom, cognom, mail)
                VALUES ('Faiez', 'Mehmood', 'Faiez@example.com')";

                // Dins de la connexió, executem la variable sql (que és l'ordre vista anteriorment)
                $connexio->exec($instruccioSQL);

                // Afegim una variable la qual el seu valor és la ID de la última inserció de la connexió
                $ultimaId = $connexio->lastInsertId();

                // Imprimir la ID de l'última inserció
                echo "Nou registre inserit correctament. El seu ID és " . $ultimaId;


            } catch(PDOException $e) {                      // En cas de que hi hagi un error, PHP llença una PDOException, i la variable $e agafa aquesta excepció
                echo "El registre no s'ha inserit correctament" . "<br>" . $e->getMessage();      // Si no es crea correctament, imprimeix el missatge d'error de PDOException
            }
            $connexio = null;                               // Tanca la connexió amb la BBDD
            ?>

        <h1>Codi en PHP</h1>
        <?php
        show_source("ex4Propi.php");
        ?>
    </body>
```

Codi

```
<!DOCTYPE html>
<html lang="ca">
    <head>
        <title>Exemple Propi Obtencio Darrer ID Guillem Serrat</title>
                    <meta  http-equiv="Content-type"  content="text/html;
charset=UTF-8">
    </head>
    <body>
        <?php
            // Definim els paràmetres per realitzar la connexió
            $servidor = "127.0.0.1";
            $usuari = "root";
            $contrasenya = "fjeclot";
            $nomDB = "dbGserrat";
            $nomTaula = "Clients";

            try {
                    // Definim un nou objecte de la classe PDO amb els
atributs: host al que ens connectarem, el nom d'usuari i contrasenya.
                                                    $connexio    =    new
PDO("mysql:host=$servidor;dbname=$nomDB", $usuari, $contrasenya);

                // A través de la funció setAttribute, agafem el tipus
d'error en cas de que n'hi hagi algun
                                $connexio->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);

                // Definim una variable que és l'ordre que s'executara,
la qual és inserir dades a la taula MyGuests. La BBDD en la cual es
treballarà està especificat a la connexió
                 $instruccioSQL = "INSERT INTO $nomTaula (nom, cognom,
mail)
                VALUES ('Faiez', 'Mehmood', 'Faiez@example.com')";

                 // Dins de la connexió, executem la variable sql (que
és l'ordre vista anteriorment)
                $connexio->exec($instruccioSQL);

                // Afegim una variable la qual el seu valor és la ID de
la última inserció de la connexió
                $ultimaId = $connexio->lastInsertId();
```

```
                // Imprimir la ID de l'última inserció
                 echo "Nou registre inserit correctament. El seu ID és "
. $ultimaId;


            } catch(PDOException $e) {                      // En cas
de que hi hagi un error, PHP llença una PDOException, i la variable $e
agafa aquesta excepció
                    echo "El registre no s'ha inserit correctament" .
"<br>"  .  $e->getMessage();          // Si no es crea correctament,
imprimeix el missatge d'error de PDOException
            }
             $connexio = null;                                      //
Tanca la connexió amb la BBDD
            ?>

        <h1>Codi en PHP</h1>
        <?php
        show_source("ex4Propi.php");
        ?>
    </body>
</html>
```

Exemple d'inserció múltiple (Ex5)

Explicació de l'exemple

L'objectiu d'aquest exemple és inserir diversos registres d'una sola vegada a partir d'una transacció.

En cas de voler executar diferents operacions SQL que obligatòriament s'han de realitzar totes correctament (és a dir, o s'executen totes o ninguna), fem servir el que s'anomena una transacció.

```
$conn->beginTransaction();
```

Un cop declarada la transacció, definim totes les operacions SQL a realitzar, i un cop es realitzen totes, es fa un commit per fer definitius els canvis

```
$conn->commit();
```

Per tant el codi és:

```
$conn->beginTransaction();

$conn->exec()
$conn->exec()
......

$conn->commit();
```

Tot el que estigui entre les instruccions beginTransaction i commit es considerarà la transacció.

En cas que alguna instrucció dongui algun error, el catch la recuperarà, però l'important és revertir els possibles canvis que s'hagin realitzat en tota la transacció, i ho aconseguim amb el mètode rollback()

$conn->rollback();

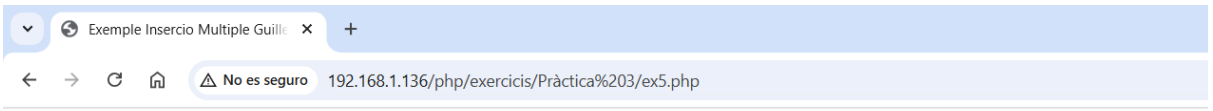
Així, totes les instruccions que s'hagin executat correctament en la transacció abans de que sortís l'error, s'eliminaran.

Necessitem un commit (tot i tenir rollback, que desfà els canvis), ja que els canvis es desen temporalment, però es desen. Per això necessitem un commit per fer-ho definitiu i un rollback per esborrar-los.

Informació sobre transaccions: https://www.php.net/manual/en/pdo.transactions.php

Codi W3Schools

Resultat



New records created successfully

Codi en PHP

```php
<!DOCTYPE html>
<html lang="ca">
    <head>
        <title>Exemple Insercio Multiple Guillem Serrat</title>
        <meta http-equiv="Content-type" content="text/html; charset=UTF-8">
    </head>
    <body>
        <?php
            // Definim els paràmetres per realitzar la connexió
            $servername = "127.0.0.1";
            $username = "root";
            $password = "fjeclot";
            $dbname = "myDBPDO";

            try {
                // Definim un nou objecte de la classe PDO amb els atributs: host al que ens connectarem, el nom d'usuari i contrasenya.
                $conn = new PDO("mysql:host=$servername;dbname=$dbname", $username, $password);

                // A través de la funció setAttribute, agafem el tipus d'error en cas de que n'hi hagi algun
                $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

                // Comença una transacció. Una transacció és un conjunt d'ordres que es fan a la vegada. A partir d'aquí es pot programar un rollback
                $conn->beginTransaction();

                // Dins de la connexió, executem la següent ordre, que es desa temporalment:
                $conn->exec("INSERT INTO MyGuests (firstname, lastname, email)
                VALUES ('John', 'Doe', 'john@example.com')");

                // Dins de la connexió, executem la següent ordre, que es desa temporalment:
                $conn->exec("INSERT INTO MyGuests (firstname, lastname, email)
                VALUES ('Mary', 'Moe', 'mary@example.com')");

                // Dins de la connexió, executem la següent ordre, que es desa temporalment:
                $conn->exec("INSERT INTO MyGuests (firstname, lastname, email)
                VALUES ('Julie', 'Dooley', 'julie@example.com')");

                $conn->commit();                        // Es confirma la transacció, es fan definitius els canvis
                echo "New records created successfully";

            } catch(PDOException $e) {                  // En cas de que hi hagi un error, PHP llença una PDOException, i la variable $e agafa aquesta excepció
                // En cas d'error es realitza un rollback. Un rollback desfà totes les operacions fetes en la transacció, per evitar quedar-se a "mitges"
                $conn->rollback();
                echo "Error: " . $e->getMessage();      // Si no es crea correctament, imprimeix el missatge d'error de PDOException
            }
                $conn = null;                           // Tanca la connexió amb la BBDD
            ?>

        <h1>Codi en PHP</h1>
        <?php
        show_source("ex5.php");
        ?>
    </body>
</html>
```

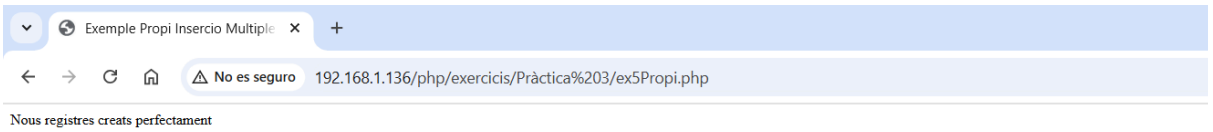
Codi

```
<!DOCTYPE html>
<html lang="ca">
    <head>
        <title>Exemple Insercio Multiple Guillem Serrat</title>
                    <meta  http-equiv="Content-type"  content="text/html;
charset=UTF-8">
    </head>
    <body>
        <?php
            // Definim els paràmetres per realitzar la connexió
            $servername = "127.0.0.1";
            $username = "root";
            $password = "fjeclot";
            $dbname = "myDBPDO";

            try {
                    // Definim un nou objecte de la classe PDO amb els
atributs: host al que ens connectarem, el nom d'usuari i contrasenya.
                                                        $conn    =    new
PDO("mysql:host=$servername;dbname=$dbname", $username, $password);

                // A través de la funció setAttribute, agafem el tipus
d'error en cas de que n'hi hagi algun
                                    $conn->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);

                // Comença una transacció. Una transacció és un conjunt
d'ordres que es fan a la vegada. A partir d'aquí es pot programar un
rollback
                $conn->beginTransaction();

                // Dins de la connexió, executem la següent ordre, que
es desa temporalment:
                $conn->exec("INSERT INTO MyGuests (firstname, lastname,
email)
                VALUES ('John', 'Doe', 'john@example.com')");

                // Dins de la connexió, executem la següent ordre, que
es desa temporalment:
                $conn->exec("INSERT INTO MyGuests (firstname, lastname,
email)
```

```
                VALUES ('Mary', 'Moe', 'mary@example.com')");

                  // Dins de la connexió, executem la següent ordre, que
es desa temporalment:
                $conn->exec("INSERT INTO MyGuests (firstname, lastname,
email)
                VALUES ('Julie', 'Dooley', 'julie@example.com')");

                   $conn->commit();                                  // Es
confirma la transacció, es fan definitius els canvis
                echo "New records created successfully";

             } catch(PDOException $e) {                          // En cas
de que hi hagi un error, PHP llença una PDOException, i la variable $e
agafa aquesta excepció
                  // En cas d'error es realitza un rollback. Un rollback
desfà totes les operacions fetes en la transacció, per evitar quedar-se
a "mitges"
                $conn->rollback();
                 echo "Error: " . $e->getMessage();          // Si no es
crea correctament, imprimeix el missatge d'error de PDOException
            }
                 $conn = null;                                     // Tanca la
connexió amb la BBDD
            ?>

        <h1>Codi en PHP</h1>
        <?php
        show_source("ex5.php");
        ?>
    </body>
</html>
```

Codi Propi

Resultat



Nous registres creats perfectament

Codi en PHP

```php
<!DOCTYPE html>
<html lang="ca">
    <head>
        <title>Exemple Propi Insercio Multiple Guillem Serrat</title>
        <meta http-equiv="Content-type" content="text/html; charset=UTF-8">
    </head>
    <body>
        <?php
            // Definim els paràmetres per realitzar la connexió
            $servidor = "127.0.0.1";
            $usuari = "root";
            $contrasenya = "fjeclot";
            $nomDB = "dbGserrat";
            $nomTaula = "Clients";

            try {
                // Definim un nou objecte de la classe PDO amb els atributs: host al que ens connectarem, el nom d'usuari i contrasenya.
                $connexio = new PDO("mysql:host=$servidor;dbname=$nomDB", $usuari, $contrasenya);

                // A través de la funció setAttribute, agafem el tipus d'error en cas de que n'hi hagi algun
                $connexio->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

                // Comença una transacció. Una transacció és un conjunt d'ordres que es fan a la vegada. A partir d'aquí es pot programar un rollback
                $connexio->beginTransaction();

                // Dins de la connexió, executem la següent ordre, que es desa temporalment:
                $connexio->exec("INSERT INTO $nomTaula (nom, cognom, mail)
                VALUES ('Guillem', 'Serrat', 'guillem@example.com')");

                // Dins de la connexió, executem la següent ordre, que es desa temporalment:
                $connexio->exec("INSERT INTO $nomTaula (nom, cognom, mail)
                VALUES ('Faiez', 'Mehmood', 'faiez@example.com')");

                // Dins de la connexió, executem la següent ordre, que es desa temporalment:
                $connexio->exec("INSERT INTO $nomTaula (nom, cognom, mail)
                VALUES ('Jordi', 'Binefa', 'microsoftTheBest@binefa.com')");

                $connexio->commit();                            // Es confirma la transacció, es fan definitius els canvis
                echo "Nous registres creats perfectament";

            } catch(PDOException $e) {                      // En cas de que hi hagi un error, PHP llença una PDOException, i la variable $e agafa aquesta excepció
                // En cas d'error es realitza un rollback. Un rollback desfà totes les operacions fetes en la transacció, per evitar quedar-se a "mitges"
                $connexio->rollback();
                echo "No s'ha inserit cap registre degut a un error" . "<br>" . $e->getMessage();          // Si no es crea correctament, imprimeix el missatge d'error de PDOException
            }
                $connexio = null;                               // Tanca la connexió amb la BBDD
            ?>

        <h1>Codi en PHP</h1>
        <?php
        show_source("ex5Propi.php");
        ?>
    </body>
```

Codi

```
<!DOCTYPE html>
<html lang="ca">
    <head>
        <title>Exemple Propi Insercio Multiple Guillem Serrat</title>
                    <meta  http-equiv="Content-type"  content="text/html;
charset=UTF-8">
    </head>
    <body>
        <?php
            // Definim els paràmetres per realitzar la connexió
            $servidor = "127.0.0.1";
            $usuari = "root";
            $contrasenya = "fjeclot";
            $nomDB = "dbGserrat";
            $nomTaula = "Clients";

            try {
                    // Definim un nou objecte de la classe PDO amb els
atributs: host al que ens connectarem, el nom d'usuari i contrasenya.
                                                    $connexio    =    new
PDO("mysql:host=$servidor;dbname=$nomDB", $usuari, $contrasenya);

                // A través de la funció setAttribute, agafem el tipus
d'error en cas de que n'hi hagi algun
                                $connexio->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);

                // Comença una transacció. Una transacció és un conjunt
d'ordres que es fan a la vegada. A partir d'aquí es pot programar un
rollback
                $connexio->beginTransaction();

                // Dins de la connexió, executem la següent ordre, que
es desa temporalment:
                    $connexio->exec("INSERT INTO $nomTaula (nom, cognom,
mail)
                VALUES ('Guillem', 'Serrat', 'guillem@example.com')");

                // Dins de la connexió, executem la següent ordre, que
es desa temporalment:
```

```
                $connexio->exec("INSERT INTO $nomTaula (nom, cognom,
mail)
            VALUES ('Faiez', 'Mehmood', 'faiez@example.com')");

                // Dins de la connexió, executem la següent ordre, que
es desa temporalment:
                $connexio->exec("INSERT INTO $nomTaula (nom, cognom,
mail)
                                        VALUES   ('Jordi',   'Binefa',
'microsoftTheBest@binefa.com')");

                $connexio->commit();                            // Es
confirma la transacció, es fan definitius els canvis
            echo "Nous registres creats perfectament";

        } catch(PDOException $e) {                      // En cas
de que hi hagi un error, PHP llença una PDOException, i la variable $e
agafa aquesta excepció
                // En cas d'error es realitza un rollback. Un rollback
desfà totes les operacions fetes en la transacció, per evitar quedar-se
a "mitges"
            $connexio->rollback();
                echo "No s'ha inserit cap registre degut a un error" .
"<br>" . $e->getMessage();              // Si no es crea correctament,
imprimeix el missatge d'error de PDOException
        }
                $connexio = null;                               //
Tanca la connexió amb la BBDD
        ?>

    <h1>Codi en PHP</h1>
    <?php
    show_source("ex5Propi.php");
    ?>
</body>
</html>
```

Exemple de declaracions preparades i paràmetres enllaçats (Ex6)

Explicació de l'exemple

L'objectiu d'aquest exemple és inserir dades diverses vegades a partir de declaracions preparades.

Al igual que en C, Python o altres llenguatges de programació fem servir funcions per no haver de repetir el mateix codi diverses vegades, en PHP existeixen les declaracions preparades.

Aquestes declaracions són instruccions tradicionals d'SQL, però que els paràmetres són variables, i cal cridar a la declaració un cop aquestes variables tinguin el valor que desitjem (exactament igual que una funció a C o Python)

La declaració preparada es desa dins d'una variable, i es crea mitjançant el mètode "prepare"

```
$stmt = $conn->prepare("INSERT INTO MyGuests (firstname, lastname, email)
VALUES (:firstname, :lastname, :email)");
```

Quan definim els valors, enlloc d'especifar un valor fix, especifiquem el que s'anomena **marcadors.** Aquests marcadors (identificats amb ":") se li assignen variables ("$") amb la funció bindParam sobre la declaració preparada ($stmt), no sobre la connexió ($conn)

```
$stmt->bindParam(':firstname', $firstname);
$stmt->bindParam(':lastname', $lastname);
$stmt->bindParam(':email', $email);
```

Aquí declarem que el marcador :firstname agafarà el valor de la variable $firstname, i així amb els 3 marcadors.

Finalment, definim els valors de les tres variables i cridem a la declaració preparada (cridem a la funció amb els paràmetres establerts)

```
$firstname = "John";
$lastname = "Doe";
$email = "john@example.com";
$stmt->execute();
```

Per tant, en aquest cas, executem la instrucció insert amb els 3 paràmetres.

I, en cas de voler realitzar més inserts, únicament cal canviar el valor de les variables i tornar a cridar la declaració preparada

Codi W3Schools

Resultat

New records created successfully

Codi en PHP

```php
<!DOCTYPE html>
<html lang="ca">
    <head>
        <title>Exemple Declaracions Preparades Guillem Serrat</title>
        <meta http-equiv="Content-type" content="text/html; charset=UTF-8">
    </head>
    <body>
        <?php
            // Definim els paràmetres per realitzar la connexió
            $servername = "127.0.0.1";
            $username = "root";
            $password = "fjeclot";
            $dbname = "myDBPDO";

            try {
                // Definim un nou objecte de la classe PDO amb els atributs: host al que ens connectarem, el nom d'usuari i contrasenya.
                $conn = new PDO("mysql:host=$servername;dbname=$dbname", $username, $password);

                // A través de la funció setAttribute, agafem el tipus d'error en cas de que n'hi hagi algun
                $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

                // Definim una preparació en la variable $stmt. Bàsicament passem l'estructura de l'ordre però no els seus paràmetres, els quals els passarem més endavant
                $stmt = $conn->prepare("INSERT INTO MyGuests (firstname, lastname, email) # Estructura de l'ordre
                VALUES (:firstname, :lastname, :email)"); // Els paràmetres definits amb : (els anomenarem marcadors) vol dir que són les "variables"

                $stmt->bindParam(':firstname', $firstname); // Dins de l'estructura, definim que el marcador tindrà com a valor la variable $firtsname
                $stmt->bindParam(':lastname', $lastname);   // Dins de l'estructura, definim que el marcador tindrà com a valor la variable $lastname
                $stmt->bindParam(':email', $email);         // Dins de l'estructura, definim que el marcador tindrà com a valor la variable $email


                $firstname = "John";            // Assignem un valor a la variable
                $lastname = "Doe";              // Assignem un valor a la variable
                $email = "john@example.com";    // Assignem un valor a la variable
                $stmt->execute();               // Agafem la preparació, i ja que les variables tenen un valor, els marcadors també. (Substituim els marcadors per valors reals de les variables)



                $firstname = "Julie";          // Canviem el valor de la mateixa variable que abans
                $lastname = "Dooley";          // Canviem el valor de la mateixa variable que abans
                $email = "julie@example.com";  // Canviem el valor de la mateixa variable que abans
                $stmt->execute();              // Tornem a agafar la preparació, però com els valors de les variables han canviat, els valors dels marcadors també i per tant l'ordre final és diferent

                echo "New records created successfully";

            } catch(PDOException $e) {                  // En cas de que hi hagi un error, PHP llença una PDOException, i la variable $e agafa aquesta excepció
                echo "Error: " . $e->getMessage();      // Si no s'executa correctament, imprimeix el missatge d'error de PDOException
            }
                $conn = null;                           // Tanca la connexió amb la BBDD
            ?>

        <h1>Codi en PHP</h1>
        <?php
        show_source("ex6.php");
        ?>
    </body>
</html>
```

Codi

```
<!DOCTYPE html>
<html lang="ca">
    <head>
        <title>Exemple Declaracions Preparades Guillem Serrat</title>
                    <meta  http-equiv="Content-type"  content="text/html;
charset=UTF-8">
    </head>
    <body>
        <?php
            // Definim els paràmetres per realitzar la connexió
            $servername = "127.0.0.1";
            $username = "root";
            $password = "fjeclot";
            $dbname = "myDBPDO";

            try {
                    // Definim un nou objecte de la classe PDO amb els
atributs: host al que ens connectarem, el nom d'usuari i contrasenya.
                                                        $conn    =    new
PDO("mysql:host=$servername;dbname=$dbname", $username, $password);

                // A través de la funció setAttribute, agafem el tipus
d'error en cas de que n'hi hagi algun
                                    $conn->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);

                    // Definim una preparació en la variable $stmt.
Bàsicament passem l'estructura de l'ordre però no els seus paràmetres,
els quals els passarem més endavant
                        $stmt  =  $conn->prepare("INSERT  INTO  MyGuests
(firstname, lastname, email) # Estructura de l'ordre
                    VALUES  (:firstname,  :lastname,  :email)"); // Els
paràmetres definits amb : (els anomenarem marcadors) vol dir que són
les "variables"

                $stmt->bindParam(':firstname', $firstname); // Dins de
l'estructura, definim que el marcador tindrà com a valor la variable
$firtsname
                $stmt->bindParam(':lastname', $lastname);   // Dins de
l'estructura, definim que el marcador tindrà com a valor la variable
$lastname
```

```
                $stmt->bindParam(':email', $email);         // Dins de
l'estructura, definim que el marcador tindrà com a valor la variable
$email



                $firstname = "John";            // Assignem un valor a
la variable
                $lastname = "Doe";              // Assignem un valor a
la variable
                $email = "john@example.com";    // Assignem un valor a
la variable
                    $stmt->execute();                   // Agafem la
preparació, i ja que les variables tenen un valor, els marcadors també.
(Substituim els marcadors per valors reals de les variables)



                $firstname = "Mary";            // Canviem el valor de
la mateixa variable que abans
                $lastname = "Moe";              // Canviem el valor de
la mateixa variable que abans
                $email = "mary@example.com";    // Canviem el valor de
la mateixa variable que abans
                 $stmt->execute();               // Tornem a agafar la
preparació, però com els valors de les variables han canviat, els
valors dels marcadors també i per tant l'ordre final és diferent



                $firstname = "Julie";           // Canviem el valor de
la mateixa variable que abans
                $lastname = "Dooley";           // Canviem el valor de
la mateixa variable que abans
                $email = "julie@example.com";   // Canviem el valor de
la mateixa variable que abans
                 $stmt->execute();               // Tornem a agafar la
preparació, però com els valors de les variables han canviat, els
valors dels marcadors també i per tant l'ordre final és diferent

               echo "New records created successfully";

            } catch(PDOException $e) {                  // En cas de
que hi hagi un error, PHP llença una PDOException, i la variable $e
agafa aquesta excepció
```

```
                    echo "Error: " . $e->getMessage();        // Si no
s'executa correctament, imprimeix el missatge d'error de PDOException
            }
                  $conn = null;                                 // Tanca la
connexió amb la BBDD
            ?>

        <h1>Codi en PHP</h1>
        <?php
        show_source("ex6.php");
        ?>
    </body>
</html>
```

Codi Propi

Resultat

Tots els registres inserits correctament

Codi en PHP

```
<!DOCTYPE html>
<html lang="ca">
    <head>
        <title>Exemple Propi Declaracions Preparades Guillem Serrat</title>
        <meta http-equiv="Content-type" content="text/html; charset=UTF-8">
    </head>
    <body>
        <?php
            // Definim els paràmetres per realitzar la connexió
            $servidor = "127.0.0.1";
            $usuari = "root";
            $contrasenya= "fjeclot";
            $nomDB = "dbGserrat";
            $nomTaula = "Clients";
            $camps = "nom, cognom, mail";

            try {
                // Definim un nou objecte de la classe PDO amb els atributs: host al que ens connectarem, el nom d'usuari i contrasenya.
                $connexio = new PDO("mysql:host=$servidor;dbname=$nomDB", $usuari, $contrasenya);

                // A través de la funció setAttribute, agafem el tipus d'error en cas de que n'hi hagi algun
                $connexio->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

                // Definim una preparació en la variable $consultaPreparada. Bàsicament passem l'estructura de l'ordre però no els seus paràmetres, els quals els passarem més endavant
                $consultaPreparada = $connexio->prepare("INSERT INTO $nomTaula ($camps) # Estructura de l'ordre
                VALUES (:firstname, :lastname, :email)"); // Els paràmetres definits amb : (els anomenarem marcadors) vol dir que són les "variables"

                $consultaPreparada->bindParam(':firstname', $nom); // Dins de l'estructura, definim que el marcador tindrà com a valor la variable $firtsname
                $consultaPreparada->bindParam(':lastname', $cognom);   // Dins de l'estructura, definim que el marcador tindrà com a valor la variable $cognom
                $consultaPreparada->bindParam(':email', $mail);         // Dins de l'estructura, definim que el marcador tindrà com a valor la variable $mail


                $nom = "Guillem";            // Assignem un valor a la variable
                $cognom = "Serrat";             // Assignem un valor a la variable
                $mail = "guillem@example.com";    // Assignem un valor a la variable
                $consultaPreparada->execute();              // Agafem la preparació, i ja que les variables tenen un valor, els marcadors també. (Substituim els marcadors per valors reals de les variables)


                $nom = "Faiez";           // Canviem el valor de la mateixa variable que abans
                $cognom = "Mehmood";              // Canviem el valor de la mateixa variable que abans
                $mail = "faiez@example.com";    // Canviem el valor de la mateixa variable que abans
                $consultaPreparada->execute();              // Tornem a agafar la preparació, però com els valors de les variables han canviat, els valors dels marcadors també i per tant l'ordre final és diferent


                $nom = "Jordi";          // Canviem el valor de la mateixa variable que abans
                $cognom = "Binefa";           // Canviem el valor de la mateixa variable que abans
                $mail = "microsoftTheBest@binefa.com";   // Canviem el valor de la mateixa variable que abans
                $consultaPreparada->execute();              // Tornem a agafar la preparació, però com els valors de les variables han canviat, els valors dels marcadors també i per tant l'ordre final és diferent

                echo "Tots els registres inserits correctament";

            } catch(PDOException $e) {                // En cas de que hi hagi un error, PHP llença una PDOException, i la variable $e agafa aquesta excepció
                echo "Algun registre no s'ha inserit correctament" . "<br>" . $e->getMessage();      // Si no es crea correctament, imprimeix el missatge d'error de PDOException
            }
                $connexio = null;                         // Tanca la connexió amb la BBDD
            ?>

        <h1>Codi en PHP</h1>
        <?php
        show_source("ex6Propi.php");
        ?>
    </body>
</html>
```

Codi

```
<!DOCTYPE html>
<html lang="ca">
    <head>
                <title>Exemple  Propi  Declaracions  Preparades  Guillem
Serrat</title>
                  <meta  http-equiv="Content-type"  content="text/html;
charset=UTF-8">
    </head>
    <body>
        <?php
            // Definim els paràmetres per realitzar la connexió
            $servidor = "127.0.0.1";
            $usuari = "root";
            $contrasenya= "fjeclot";
            $nomDB = "dbGserrat";
            $nomTaula = "Clients";
            $camps = "nom, cognom, mail";

            try {
                    // Definim un nou objecte de la classe PDO amb els
atributs: host al que ens connectarem, el nom d'usuari i contrasenya.
                                                    $connexio    =    new
PDO("mysql:host=$servidor;dbname=$nomDB", $usuari, $contrasenya);

                // A través de la funció setAttribute, agafem el tipus
d'error en cas de que n'hi hagi algun
                                $connexio->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);

                            // Definim una preparació en la variable
$consultaPreparada. Bàsicament passem l'estructura de l'ordre però no
els seus paràmetres, els quals els passarem més endavant
                  $consultaPreparada = $connexio->prepare("INSERT INTO
$nomTaula ($camps) # Estructura de l'ordre
                    VALUES (:firstname, :lastname, :email)"); // Els
paràmetres definits amb : (els anomenarem marcadors) vol dir que són
les "variables"

                $consultaPreparada->bindParam(':firstname', $nom); //
Dins de l'estructura, definim que el marcador tindrà com a valor la
variable $firtsname
```

```
                    $consultaPreparada->bindParam(':lastname', $cognom);
// Dins de l'estructura, definim que el marcador tindrà com a valor la
variable $cognom
                          $consultaPreparada->bindParam(':email', $mail);
// Dins de l'estructura, definim que el marcador tindrà com a valor la
variable $mail


                    $nom = "Guillem";             // Assignem un valor a la
variable
                   $cognom = "Serrat";               // Assignem un valor a
la variable
                    $mail = "guillem@example.com";     // Assignem un valor
a la variable
                    $consultaPreparada->execute();                // Agafem
la preparació, i ja que les variables tenen un valor, els marcadors
també. (Substituim els marcadors per valors reals de les variables)


                    $nom = "Faiez";               // Canviem el valor de la
mateixa variable que abans
                    $cognom = "Mehmood";                // Canviem el valor
de la mateixa variable que abans
                    $mail = "faiez@example.com";     // Canviem el valor de
la mateixa variable que abans
                    $consultaPreparada->execute();                // Tornem
a agafar la preparació, però com els valors de les variables han
canviat, els valors dels marcadors també i per tant l'ordre final és
diferent


                    $nom = "Jordi";             // Canviem el valor de la
mateixa variable que abans
                   $cognom = "Binefa";           // Canviem el valor de la
mateixa variable que abans
                    $mail = "microsoftTheBest@binefa.com";   // Canviem el
valor de la mateixa variable que abans
                    $consultaPreparada->execute();                // Tornem
a agafar la preparació, però com els valors de les variables han
canviat, els valors dels marcadors també i per tant l'ordre final és
diferent
```

```
                echo "Tots els registres inserits correctament";

            } catch(PDOException $e) {                    // En cas de
que hi hagi un error, PHP llença una PDOException, i la variable $e
agafa aquesta excepció
                    echo "Algun registre no s'ha inserit correctament" .
"<br>"  .  $e->getMessage();          //  Si  no  es  crea  correctament,
imprimeix el missatge d'error de PDOException
            }
                $connexio = null;                          // Tanca la
connexió amb la BBDD
            ?>

        <h1>Codi en PHP</h1>
        <?php
        show_source("ex6Propi.php");
        ?>
    </body>
</html>
```

Exemple de selecció de dades (Ex7)

Explicació de l'exemple

L'objectiu d'aquest exemple és extreure les dades de la base de dades i mostrar-les en format taula HTML

Primerament, necessitem recórrer la taula (que, dit d'una altra forma, és una estructura d'arrays), i ho farem amb una classe anomenada RecursiveIteratorIterator. Primer, crearem una classe filla anomenada TableRows

```
class TableRows extends RecursiveIteratorIterator
```

Seguidament, redefinirem 4 mètodes:

1 Constructor

```
function __construct($it) {
            parent::__construct($it, self::LEAVES_ONLY);
        }
```

El constructor s'invoca 1 sola vegada de manera automàtica quan es crea un objecte de la classe. L'objectiu del constructor és retornar únicament els valors finals de les files i columnes de la BD

2 beginChildren()

```
function beginChildren() :void {
            echo "<tr>";
}
```

El mètode beginChildren s'invoca automàticament cada vegada que comença una iteració. L'objectiu és crear l'etiqueta <tr> per obrir una fila

3 current()

```
function current() :string {
 return "<td style='width:150px;border:1px solid black;'>" . parent::current(). "</td>";
}
```

El mètode current() s'invoca automàticament cada vegada que s'itera (per tant, cada vegada que agafem un registre de la BD). L'objectiu és obtenir el valor de la columna en la que està i introduir-la dins de una cel·la

4 endChildren()

```
function endChildren() :void {
        echo "</tr>" . "\n";
    }
```

El mètode endChildren s'invoca automàticament cada vegada que acaba una iteració. L'objectiu és crear l'etiqueta </tr> per tancar la fila ja començada

Un flux d'iteració segueix les següents passes:

- 1. Inici de l'Iteració (Per a la primera fila): Es crida beginChildren() --> Imprimeix <tr>.
- 2. Iteració de la Primera Columna (id): Es crida current() --> Imprimeix <td ...>1</td>.
- 3. Iteració de la Segona Columna (firstname): Es crida current() --> Imprimeix <td ...>John</td>.
- 4. Iteració de la Tercera Columna (lastname): Es crida current() --> Imprimeix <td ...>Doe</td>.
- 5. Fi de l'Iteració de la Fila: Es crida endChildren() --> Imprimeix </tr>\n.
- 6. Repetició (Per a la següent fila), tornant al punt 1.

Un cop definits els mètodes de la classe, afegim una consultaPreparada per seleccionar totes les dades de la taula mitjançant la següent ordre SQL

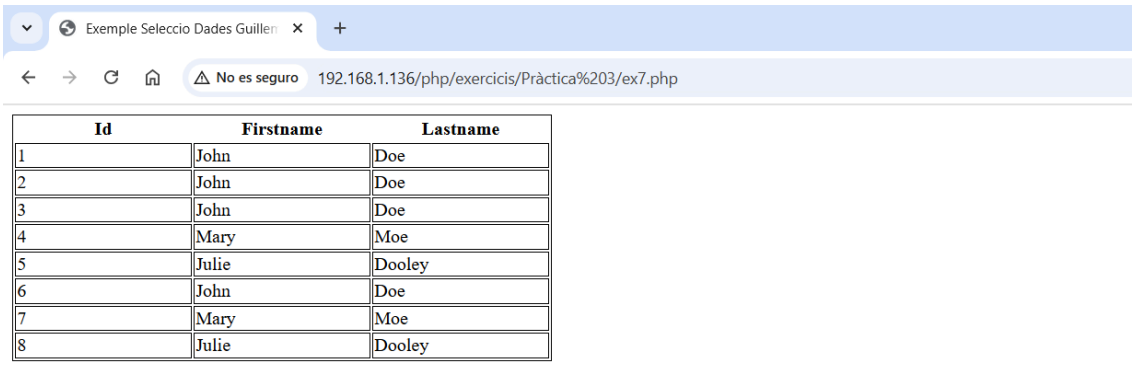
```
$stmt = $conn->prepare("SELECT id, firstname, lastname FROM MyGuests");
```

I per últim, amb la classe creada anteriorment, creem la taula HTML

```
$result = $stmt->setFetchMode(PDO::FETCH_ASSOC);
foreach(new TableRows(new RecursiveArrayIterator($stmt->fetchAll())) as $k=>$v) {
// new RecursiveArrayIterator($stmt->fetchAll()) converteix els registres en iteratius
echo $v; // Imprimeix tota la fila HTML ja processada per les funcions
}
```

Codi W3Schools

Resultat



Id	Firstname	Lastname
1	John	Doe
2	John	Doe
3	John	Doe
4	Mary	Moe
5	Julie	Dooley
6	John	Doe
7	Mary	Moe
8	Julie	Dooley

Codi en PHP

```php
<!DOCTYPE html>
<html lang="ca">
    <head>
        <title>Exemple Seleccio Dades Guillem Serrat</title>
        <meta http-equiv="Content-type" content="text/html; charset=UTF-8">
    </head>
    <body>
        <?php
            echo "<table style='border: solid 1px black;'>";                    // Crea la taula
            echo "<tr><th>Id</th><th>Firstname</th><th>Lastname</th></tr>";     // Crea una fila per les capçeleres de la taula

            // Creem una classe que és filla de RecursiveIteratorIterator, una classe de PHP que permet recórrer estructures d'arrays
            class TableRows extends RecursiveIteratorIterator {
                // Sempre es crida al constructor 1 vegada abans de començar a iterar sobre un array
                function __construct($it) {
                    parent::__construct($it, self::LEAVES_ONLY);
                    // Amb l'operador LEAVES_ONLY ens assegurem que únicament agafa el valor i res més
                }

                // Creem una funció anomenada current. L'objectiu de la funció és crear una cel·la i agafar el valor d'un registre SQL.
                // Aquesta funció la crida automàticament per l'objecte RecursiveIteratorIterator (no cal cridar-la, es fa sol)
                function current() :string { // Afegim :string per evitar el missatge "deprecated". Indiquem que retorna un String
                    return "<td style='width:150px;border:1px solid black;'>" . parent::current(). "</td>";
                }


                // Creem una funció anomenada beginChildren. L'objectiu de la funció és crear una nova fila.
                // Aquesta funció la crida automàticament per l'objecte RecursiveIteratorIterator (no cal cridar-la, es fa sol)
                // Aquesta funció és cridada cada vegada que es fa una iteració (1 volta, 1 registre en SQL)
                function beginChildren() :void { // Afegim :void per evitar el missatge "deprecated". Indiquem que no retorna res
                    echo "<tr>"; // Creem la fila
                }

                //Creem una funció anomenada endChildren L'objectiu de la funció és tancar la fila creada per la funció endChildren
                // Aquesta funció la crida automàticament per l'objecte RecursiveIteratorIterator (no cal cridar-la, es fa sol)
                // Aquesta funció és cridada cada vegada que es fa una iteració (1 volta, 1 registre en SQL)
                function endChildren() :void { // Afegim :void per evitar el missatge "deprecated"
                    echo "</tr>" . "\n"; // Tanquem la fila
                }

                // Resum del Flux de l'Iteració
                // 1. Inici de l'Iteració (Per a la primera fila): Es crida beginChildren() --> Imprimeix <tr>.
                // 2. Iteració de la Primera Columna (id): Es crida current() --> Imprimeix <td ...>1</td>.
                // 3. Iteració de la Segona Columna (firstname): Es crida current() --> Imprimeix <td ...>John</td>.
                // 4. Iteració de la Tercera Columna (lastname): Es crida current() --> Imprimeix <td ...>Doe</td>.
                // 5. Fi de l'Iteració de la Fila: Es crida endChildren() --> Imprimeix </tr>\n.
                // 6. Repetició (Per a la següent fila), tornant al punt 1.


            }
```

```
            $servername = "127.0.0.1";
            $username = "root";
            $password = "fjeclot";
            $dbname = "myDBPDO";

            try {
                // Definim un nou objecte de la classe PDO amb els atributs: host al que ens connectarem, el nom d'usuari i contrasenya.
                $conn = new PDO("mysql:host=$servername;dbname=$dbname", $username, $password);

                // A través de la funció setAttribute, agafem el tipus d'error en cas de que n'hi hagi algun
                $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

                // Definim una preparació en la variable $stmt.
                $stmt = $conn->prepare("SELECT id, firstname, lastname FROM MyGuests");

                // Executem la preparació
                $stmt->execute();

                // Imprimim la taula fent servir la crida automàtica a les funcions de la classe RecursiveArrayIterator
                $result = $stmt->setFetchMode(PDO::FETCH_ASSOC);
                foreach(new TableRows(new RecursiveArrayIterator($stmt->fetchAll())) as $k=>$v) {
                    // new RecursiveArrayIterator($stmt->fetchAll()) converteix els registres en iteratius
                    echo $v; // Imprimeix tota la fila HTML ja processada per les funcions
                }
            } catch(PDOException $e) {                      // En cas de que hi hagi un error, PHP llença una PDOException, i la variable $e agafa aquesta excepció
                echo "Error: " . $e->getMessage();          // Si no s'executa correctament, imprimeix el missatge d'error de PDOException
            }
            $conn = null;
            echo "</table>";
        ?>

        <h1>Codi en PHP</h1>
        <?php
        show_source("ex7.php");
        ?>
    </body>
</html>
```

Codi

```
<!DOCTYPE html>
<html lang="ca">
    <head>
        <title>Exemple Seleccio Dades Guillem Serrat</title>
                    <meta  http-equiv="Content-type"  content="text/html;
charset=UTF-8">
    </head>
    <body>
        <?php
                        echo  "<table  style='border:  solid  1px  black;'>";
// Crea la taula
                                                                        echo
"<tr><th>Id</th><th>Firstname</th><th>Lastname</th></tr>";       // Crea
una fila per les capçeleres de la taula

                            //  Creem  una  classe  que  és  filla  de
RecursiveIteratorIterator,  una  classe  de  PHP  que  permet  recórrer
estructures d'arrays
            class TableRows extends RecursiveIteratorIterator {
                    // Sempre es crida al constructor 1 vegada abans de
començar a iterar sobre un array
                function __construct($it) {
                    parent::__construct($it, self::LEAVES_ONLY);
                        // Amb l'operador LEAVES_ONLY ens assegurem que
únicament agafa el valor i res més
                }

                // Creem una funció anomenada current. L'objectiu de la
funció és crear una cel·la i agafar el valor d'un registre SQL.
                // Aquesta funció la crida automàticament per l'objecte
RecursiveIteratorIterator (no cal cridar-la, es fa sol)
                    function current() :string { // Afegim :string per
evitar el missatge "deprecated". Indiquem que retorna un String
                        return "<td style='width:150px;border:1px solid
black;'>" . parent::current(). "</td>";
                }

                // Creem una funció anomenada beginChildren. L'objectiu
de la funció és crear una nova fila.
                // Aquesta funció la crida automàticament per l'objecte
RecursiveIteratorIterator (no cal cridar-la, es fa sol)
```

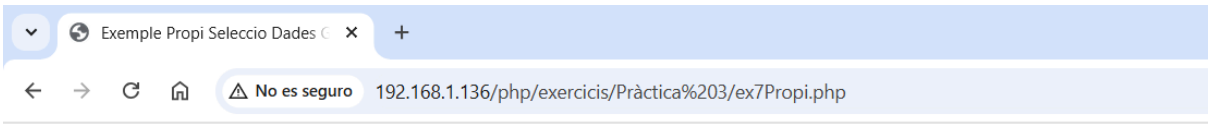
```
                // Aquesta funció és cridada cada vegada que es fa una
iteració (1 volta, 1 registre en SQL)
                function beginChildren() :void { // Afegim :void per
evitar el missatge "deprecated". Indiquem que no retorna res
                    echo "<tr>"; // Creem la fila
                }

                //Creem una funció anomenada endChildren L'objectiu de
la funció és tancar la fila creada per la funció endChildren
                // Aquesta funció la crida automàticament per l'objecte
RecursiveIteratorIterator (no cal cridar-la, es fa sol)
                // Aquesta funció és cridada cada vegada que es fa una
iteració (1 volta, 1 registre en SQL)
                    function endChildren() :void { // Afegim :void per
evitar el missatge "deprecated"
                    echo "</tr>" . "\n"; // Tanquem la fila
                }

                // Resum del Flux de l'Iteració
                // 1. Inici de l'Iteració (Per a la primera fila): Es
crida beginChildren() --> Imprimeix <tr>.
                    // 2. Iteració de la Primera Columna (id): Es crida
current() --> Imprimeix <td ...>1</td>.
                    // 3. Iteració de la Segona Columna (firstname): Es
crida current() --> Imprimeix <td ...>John</td>.
                    // 4. Iteració de la Tercera Columna (lastname): Es
crida current() --> Imprimeix <td ...>Doe</td>.
                        // 5. Fi de l'Iteració de la Fila: Es crida
endChildren() --> Imprimeix </tr>\n.
                    // 6. Repetició (Per a la següent fila), tornant al
punt 1.


            }

            $servername = "127.0.0.1";
            $username = "root";
            $password = "fjeclot";
            $dbname = "myDBPDO";

            try {
```

```
                    // Definim un nou objecte de la classe PDO amb els
atributs: host al que ens connectarem, el nom d'usuari i contrasenya.
                                                        $conn    =    new
PDO("mysql:host=$servername;dbname=$dbname", $username, $password);

                  // A través de la funció setAttribute, agafem el tipus
d'error en cas de que n'hi hagi algun
                                    $conn->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);

                // Definim una preparació en la variable $stmt.
                  $stmt = $conn->prepare("SELECT id, firstname, lastname
FROM MyGuests");

                // Executem la preparació
                $stmt->execute();

                  // Imprimim la taula fent servir la crida automàtica a
les funcions de la classe RecursiveArrayIterator
                $result = $stmt->setFetchMode(PDO::FETCH_ASSOC);
                                              foreach(new   TableRows(new
RecursiveArrayIterator($stmt->fetchAll())) as $k=>$v) {
                        // new RecursiveArrayIterator($stmt->fetchAll())
converteix els registres en iteratius
                            echo $v; // Imprimeix tota la fila HTML ja
processada per les funcions
                }
             } catch(PDOException $e) {                        // En cas
de que hi hagi un error, PHP llença una PDOException, i la variable $e
agafa aquesta excepció
                  echo "Error: " . $e->getMessage();           // Si no
s'executa correctament, imprimeix el missatge d'error de PDOException
            }
            $conn = null;
            echo "</table>";
        ?>

        <h1>Codi en PHP</h1>
        <?php
        show_source("ex7.php");
        ?>
    </body>
```

```
</html>
```

Codi Propi

Resultat



Id	Firstname	Lastname
1	Guillem	guillem@example.com
2	Faiez	Faiez@example.com
3	Guillem	guillem@example.com
4	Faiez	faiez@example.com
5	Jordi	microsoftTheBest@binefa.com
6	Guillem	guillem@example.com
7	Faiez	faiez@example.com
8	Jordi	microsoftTheBest@binefa.com

Codi en PHP

```php
<!DOCTYPE html>
<html lang="ca">
    <head>
        <title>Exemple Propi Seleccio Dades Guillem Serrat</title>
        <meta http-equiv="Content-type" content="text/html; charset=UTF-8">
    </head>
    <body>
        <?php
            echo "<table style='border: solid 1px black;'>";                    // Crea la taula
            echo "<tr><th>Id</th><th>Firstname</th><th>Lastname</th></tr>";     // Crea una fila per les capçeleres de la taula

            // Creem una classe que és filla de RecursiveIteratorIterator, una classe de PHP que permet recórrer estructures d'arrays
            class TableRows extends RecursiveIteratorIterator {
                // Sempre es crida al constructor 1 vegada abans de començar a iterar sobre un array
                function __construct($it) {
                    parent::__construct($it, self::LEAVES_ONLY);
                    // Amb l'operador LEAVES_ONLY ens assegurem que únicament agafa el valor i res més
                }

                // Creem una funció anomenada current. L'objectiu de la funció és crear una cel·la i agafar el valor d'un registre SQL.
                // Aquesta funció la crida automàticament per l'objecte RecursiveIteratorIterator (no cal cridar-la, es fa sol)
                function current() :string { // Afegim :string per evitar el missatge "deprecated". Indiquem que retorna un String
                    return "<td style='width:150px;border:1px solid black;'>" . parent::current(). "</td>";
                }

                // Creem una funció anomenada beginChildren. L'objectiu de la funció és crear una nova fila.
                // Aquesta funció la crida automàticament per l'objecte RecursiveIteratorIterator (no cal cridar-la, es fa sol)
                // Aquesta funció és cridada cada vegada que es fa una iteració (1 volta, 1 registre en SQL)
                function beginChildren() :void { // Afegim :void per evitar el missatge "deprecated". Indiquem que no retorna res
                    echo "<tr>"; // Creem la fila
                }


                //Creem una funció anomenada endChildren L'objectiu de la funció és tancar la fila creada per la funció endChildren
                // Aquesta funció la crida automàticament per l'objecte RecursiveIteratorIterator (no cal cridar-la, es fa sol)
                // Aquesta funció és cridada cada vegada que es fa una iteració (1 volta, 1 registre en SQL)
                function endChildren() :void { // Afegim :void per evitar el missatge "deprecated"
                    echo "</tr>" . "\n"; // Tanquem la fila
                }

                // Resum del Flux de l'Iteració
                // 1. Inici de l'Iteració (Per a la primera fila): Es crida beginChildren() --> Imprimeix <tr>.
                // 2. Iteració de la Primera Columna (id): Es crida current() --> Imprimeix <td ...>1</td>.
                // 3. Iteració de la Segona Columna (firstname): Es crida current() --> Imprimeix <td ...>John</td>.
                // 4. Iteració de la Tercera Columna (lastname): Es crida current() --> Imprimeix <td ...>Doe</td>.
                // 5. Fi de l'Iteració de la Fila: Es crida endChildren() --> Imprimeix </tr>\n.
                // 6. Repetició (Per a la següent fila), tornant al punt 1.


            }
```

```
            $servidor = "127.0.0.1";
            $usuari = "root";
            $contrasenya = "fjeclot";
            $nomDB = "dbGserrat";
            $nomTaula = "Clients";

            try {
                // Definim un nou objecte de la classe PDO amb els atributs: host al que ens connectarem, el nom d'usuari i contrasenya.
                $connexio = new PDO("mysql:host=$servidor;dbname=$nomDB", $usuari, $contrasenya);

                // A través de la funció setAttribute, agafem el tipus d'error en cas de que n'hi hagi algun
                $connexio->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

                // Definim una preparació en la variable $consultaPreparada.
                $consultaPreparada = $connexio->prepare("SELECT id, nom, mail FROM $nomTaula");

                // Executem la preparació
                $consultaPreparada->execute();

                // Imprimim la taula fent servir la crida automàtica a les funcions de la classe RecursiveArrayIterator
                $resultat = $consultaPreparada->setFetchMode(PDO::FETCH_ASSOC);
                foreach(new TableRows(new RecursiveArrayIterator($consultaPreparada->fetchAll())) as $k=>$v) {
                    // new RecursiveArrayIterator($consultaPreparada->fetchAll()) converteix els registres en iteratius
                    echo $v; // Imprimeix tota la fila HTML ja processada per les funcions
                }
            } catch(PDOException $e) {                                          // En cas de que hi hagi un error, PHP llença una PDOException, i la variable $e agafa aquesta excepció
                echo "No s'ha pogut mostrar les dades" . "<br>" . $e->getMessage();          // Si no s'executa correctament, imprimeix el missatge d'error de PDOException
            }
            $connexio = null;                                                           // Tanca la connexió amb la BBDD
            echo "</table>";
        ?>

        <h1>Codi en PHP</h1>
        <?php
        show_source("ex7Propi.php");
        ?>
    </body>
</html>
```

Codi

```
<!DOCTYPE html>
<html lang="ca">
    <head>
        <title>Exemple Propi Seleccio Dades Guillem Serrat</title>
                    <meta  http-equiv="Content-type"  content="text/html;
charset=UTF-8">
    </head>
    <body>
        <?php
                        echo  "<table  style='border:  solid  1px  black;'>";
// Crea la taula
                                                                        echo
"<tr><th>Id</th><th>Firstname</th><th>Lastname</th></tr>";       // Crea
una fila per les capçeleres de la taula

                              //  Creem  una  classe  que  és  filla  de
RecursiveIteratorIterator,  una  classe  de  PHP  que  permet  recórrer
estructures d'arrays
            class TableRows extends RecursiveIteratorIterator {
                    // Sempre es crida al constructor 1 vegada abans de
començar a iterar sobre un array
                function __construct($it) {
                    parent::__construct($it, self::LEAVES_ONLY);
                        // Amb l'operador LEAVES_ONLY ens assegurem que
únicament agafa el valor i res més
                }

                // Creem una funció anomenada current. L'objectiu de la
funció és crear una cel·la i agafar el valor d'un registre SQL.
                // Aquesta funció la crida automàticament per l'objecte
RecursiveIteratorIterator (no cal cridar-la, es fa sol)
                    function current() :string { // Afegim :string per
evitar el missatge "deprecated". Indiquem que retorna un String
                        return "<td style='width:150px;border:1px solid
black;'>" . parent::current(). "</td>";
                }

                // Creem una funció anomenada beginChildren. L'objectiu
de la funció és crear una nova fila.
                // Aquesta funció la crida automàticament per l'objecte
RecursiveIteratorIterator (no cal cridar-la, es fa sol)
```

```
                // Aquesta funció és cridada cada vegada que es fa una
iteració (1 volta, 1 registre en SQL)
                function beginChildren() :void { // Afegim :void per
evitar el missatge "deprecated". Indiquem que no retorna res
                    echo "<tr>"; // Creem la fila
                }

                //Creem una funció anomenada endChildren L'objectiu de
la funció és tancar la fila creada per la funció endChildren
                // Aquesta funció la crida automàticament per l'objecte
RecursiveIteratorIterator (no cal cridar-la, es fa sol)
                // Aquesta funció és cridada cada vegada que es fa una
iteració (1 volta, 1 registre en SQL)
                    function endChildren() :void { // Afegim :void per
evitar el missatge "deprecated"
                    echo "</tr>" . "\n"; // Tanquem la fila
                }

                // Resum del Flux de l'Iteració
                // 1. Inici de l'Iteració (Per a la primera fila): Es
crida beginChildren() --> Imprimeix <tr>.
                    // 2. Iteració de la Primera Columna (id): Es crida
current() --> Imprimeix <td ...>1</td>.
                    // 3. Iteració de la Segona Columna (firstname): Es
crida current() --> Imprimeix <td ...>John</td>.
                    // 4. Iteració de la Tercera Columna (lastname): Es
crida current() --> Imprimeix <td ...>Doe</td>.
                        // 5. Fi de l'Iteració de la Fila: Es crida
endChildren() --> Imprimeix </tr>\n.
                    // 6. Repetició (Per a la següent fila), tornant al
punt 1.


            }

            $servidor = "127.0.0.1";
            $usuari = "root";
            $contrasenya = "fjeclot";
            $nomDB = "dbGserrat";
            $nomTaula = "Clients";

            try {
```

```
                    // Definim un nou objecte de la classe PDO amb els
atributs: host al que ens connectarem, el nom d'usuari i contrasenya.
                                                        $connexio    =    new
PDO("mysql:host=$servidor;dbname=$nomDB", $usuari, $contrasenya);

                // A través de la funció setAttribute, agafem el tipus
d'error en cas de que n'hi hagi algun
                                $connexio->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);

                            // Definim una preparació en la variable
$consultaPreparada.
                    $consultaPreparada = $connexio->prepare("SELECT id,
nom, mail FROM $nomTaula");

                // Executem la preparació
                $consultaPreparada->execute();

                // Imprimim la taula fent servir la crida automàtica a
les funcions de la classe RecursiveArrayIterator
                                                                $resultat    =
$consultaPreparada->setFetchMode(PDO::FETCH_ASSOC);
                                                foreach(new   TableRows(new
RecursiveArrayIterator($consultaPreparada->fetchAll())) as $k=>$v) {
                                                                    //    new
RecursiveArrayIterator($consultaPreparada->fetchAll())  converteix  els
registres en iteratius
                            echo $v; // Imprimeix tota la fila HTML ja
processada per les funcions
                }
                                            }   catch(PDOException   $e)    {
// En cas de que hi hagi un error, PHP llença una PDOException, i la
variable $e agafa aquesta excepció
                    echo "No s'ha pogut mostrar les dades" . "<br>" .
$e->getMessage();             // Si no s'executa correctament, imprimeix
el missatge d'error de PDOException
            }
                                                    $connexio    =    null;
// Tanca la connexió amb la BBDD
            echo "</table>";
        ?>
```

```
        <h1>Codi en PHP</h1>
        <?php
        show_source("ex7Propi.php");
        ?>
    </body>
</html>
```

Exemple de l'ordre WHERE (Ex8)

Explicació de l'exemple

L'objectiu d'aquest exemple és, mitjançant una selecció de dades, filtrar-les per només mostrar aquelles que compleixin la condició

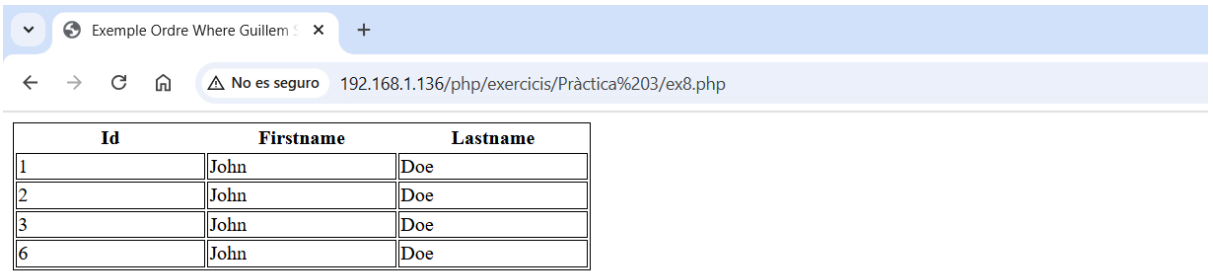
Aprofitant la base de l'exemple 7 d'obtenció de dades, per tal de filtrar-les mitjançant una condició únicament cal incloure l'ordre WHERE dins de la instrucció SQL, seguit de la condició

```
$stmt = $conn->prepare("SELECT id, firstname, lastname FROM MyGuests WHERE lastname='Doe'");
```

En aques cas únicament obtindrem aquells registres que en el camp "lastname" tingui el valor "Doe"

Codi W3Schools

Resultat



Id	Firstname	Lastname
1	John	Doe
2	John	Doe
3	John	Doe
6	John	Doe

Codi en PHP

```
<!DOCTYPE html>
<html lang="ca">
    <head>
        <title>Exemple Ordre Where Guillem Serrat</title>
        <meta http-equiv="Content-type" content="text/html; charset=UTF-8">
    </head>
    <body>
        <?php
            echo "<table style='border: solid 1px black;'>";                    // Crea la taula
            echo "<tr><th>Id</th><th>Firstname</th><th>Lastname</th></tr>";     // Crea una fila per les capçaleres de la taula

            // Creem una classe que és filla de RecursiveIteratorIterator, una classe de PHP que permet recórrer estructures d'arrays
            class TableRows extends RecursiveIteratorIterator {
                // Sempre es crida al constructor 1 vegada abans de començar a iterar sobre un array
                function __construct($it) {
                    parent::__construct($it, self::LEAVES_ONLY);
                    // Amb l'operador LEAVES_ONLY ens assegurem que únicament agafa el valor i res més
                }

                // Creem una funció anomenada current. L'objectiu de la funció és crear una cel·la i agafar el valor d'un registre SQL.
                // Aquesta funció la crida automàticament per l'objecte RecursiveIteratorIterator (no cal cridar-la, es fa sol)
                function current() :string { // Afegim :string per evitar el missatge "deprecated". Indiquem que retorna un String
                    return "<td style='width:150px;border:1px solid black;'>" . parent::current(). "</td>";
                }

                // Creem una funció anomenada beginChildren. L'objectiu de la funció és crear una nova fila.
                // Aquesta funció la crida automàticament per l'objecte RecursiveIteratorIterator (no cal cridar-la, es fa sol)
                // Aquesta funció és cridada cada vegada que es fa una iteració (1 volta, 1 registre en SQL)
                function beginChildren() :void { // Afegim :void per evitar el missatge "deprecated". Indiquem que no retorna res
                    echo "<tr>"; // Creem la fila
                }


                //Creem una funció anomenada endChildren L'objectiu de la funció és tancar la fila creada per la funció endChildren
                // Aquesta funció la crida automàticament per l'objecte RecursiveIteratorIterator (no cal cridar-la, es fa sol)
                // Aquesta funció és cridada cada vegada que es fa una iteració (1 volta, 1 registre en SQL)
                function endChildren() :void { // Afegim :void per evitar el missatge "deprecated"
                    echo "</tr>" . "\n"; // Tanquem la fila
                }

                // Resum del Flux de l'Iteració
                // 1. Inici de l'Iteració (Per a la primera fila): Es crida beginChildren() --> Imprimeix <tr>.
                // 2. Iteració de la Primera Columna (id): Es crida current() --> Imprimeix <td ...>1</td>.
                // 3. Iteració de la Segona Columna (firstname): Es crida current() --> Imprimeix <td ...>John</td>.
                // 4. Iteració de la Tercera Columna (lastname): Es crida current() --> Imprimeix <td ...>Doe</td>.
                // 5. Fi de l'Iteració de la Fila: Es crida endChildren() --> Imprimeix </tr>\n.
                // 6. Repetició (Per a la següent fila), tornant al punt 1.


            }
```

```
            $servername = "127.0.0.1";
            $username = "root";
            $password = "fjeclot";
            $dbname = "myDBPDO";

            try {
                // Definim un nou objecte de la classe PDO amb els atributs: host al que ens connectarem, el nom d'usuari i contrasenya.
                $conn = new PDO("mysql:host=$servername;dbname=$dbname", $username, $password);

                // A través de la funció setAttribute, agafem el tipus d'error en cas de que n'hi hagi algun
                $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

                // Definim una preparació en la variable $stmt.
                $stmt = $conn->prepare("SELECT id, firstname, lastname FROM MyGuests WHERE lastname='Doe'");

                // Executem la preparació
                $stmt->execute();

                // Imprimim la taula fent servir la crida automàtica a les funcions de la classe RecursiveArrayIterator
                $result = $stmt->setFetchMode(PDO::FETCH_ASSOC);
                foreach(new TableRows(new RecursiveArrayIterator($stmt->fetchAll())) as $k=>$v) {
                    // new RecursiveArrayIterator($stmt->fetchAll()) converteix els registres en iteratius
                    echo $v; // Imprimeix tota la fila HTML ja processada per les funcions
                }
            } catch(PDOException $e) {                      // En cas de que hi hagi un error, PHP llença una PDOException, i la variable $e agafa aquesta excepció
                echo "Error: " . $e->getMessage();          // Si no s'executa correctament, imprimeix el missatge d'error de PDOException
            }
            $conn = null;
            echo "</table>";
        ?>

        <h1>Codi en PHP</h1>
        <?php
        show_source("ex8.php");
        ?>
    </body>
</html>
```

Codi

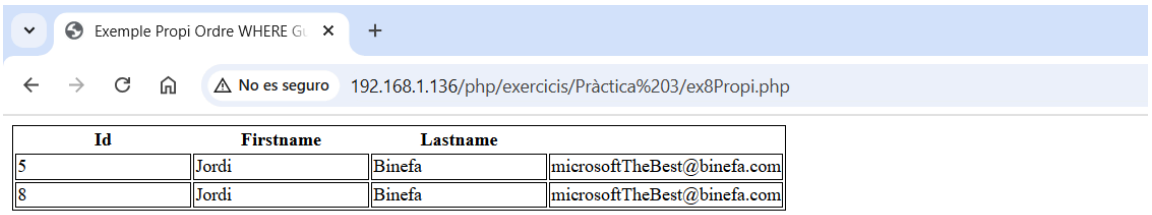
```
<!DOCTYPE html>
<html lang="ca">
    <head>
        <title>Exemple Ordre Where Guillem Serrat</title>
                    <meta  http-equiv="Content-type"  content="text/html;
charset=UTF-8">
    </head>
    <body>
        <?php
                        echo  "<table  style='border:  solid  1px  black;'>";
// Crea la taula
                                                                        echo
"<tr><th>Id</th><th>Firstname</th><th>Lastname</th></tr>";       // Crea
una fila per les capçeleres de la taula

                              //  Creem  una  classe  que  és  filla  de
RecursiveIteratorIterator,  una  classe  de  PHP  que  permet  recórrer
estructures d'arrays
            class TableRows extends RecursiveIteratorIterator {
                    // Sempre es crida al constructor 1 vegada abans de
començar a iterar sobre un array
                function __construct($it) {
                    parent::__construct($it, self::LEAVES_ONLY);
                        // Amb l'operador LEAVES_ONLY ens assegurem que
únicament agafa el valor i res més
                }

                // Creem una funció anomenada current. L'objectiu de la
funció és crear una cel·la i agafar el valor d'un registre SQL.
                // Aquesta funció la crida automàticament per l'objecte
RecursiveIteratorIterator (no cal cridar-la, es fa sol)
                    function current() :string { // Afegim :string per
evitar el missatge "deprecated". Indiquem que retorna un String
                        return "<td style='width:150px;border:1px solid
black;'>" . parent::current(). "</td>";
                }

                // Creem una funció anomenada beginChildren. L'objectiu
de la funció és crear una nova fila.
                // Aquesta funció la crida automàticament per l'objecte
RecursiveIteratorIterator (no cal cridar-la, es fa sol)
```

```
                // Aquesta funció és cridada cada vegada que es fa una
iteració (1 volta, 1 registre en SQL)
                function beginChildren() :void { // Afegim :void per
evitar el missatge "deprecated". Indiquem que no retorna res
                    echo "<tr>"; // Creem la fila
                }

                //Creem una funció anomenada endChildren L'objectiu de
la funció és tancar la fila creada per la funció endChildren
                // Aquesta funció la crida automàticament per l'objecte
RecursiveIteratorIterator (no cal cridar-la, es fa sol)
                // Aquesta funció és cridada cada vegada que es fa una
iteració (1 volta, 1 registre en SQL)
                    function endChildren() :void { // Afegim :void per
evitar el missatge "deprecated"
                    echo "</tr>" . "\n"; // Tanquem la fila
                }

                // Resum del Flux de l'Iteració
                // 1. Inici de l'Iteració (Per a la primera fila): Es
crida beginChildren() --> Imprimeix <tr>.
                    // 2. Iteració de la Primera Columna (id): Es crida
current() --> Imprimeix <td ...>1</td>.
                    // 3. Iteració de la Segona Columna (firstname): Es
crida current() --> Imprimeix <td ...>John</td>.
                    // 4. Iteració de la Tercera Columna (lastname): Es
crida current() --> Imprimeix <td ...>Doe</td>.
                        // 5. Fi de l'Iteració de la Fila: Es crida
endChildren() --> Imprimeix </tr>\n.
                    // 6. Repetició (Per a la següent fila), tornant al
punt 1.

            }

            $servername = "127.0.0.1";
            $username = "root";
            $password = "fjeclot";
            $dbname = "myDBPDO";

            try {
                    // Definim un nou objecte de la classe PDO amb els
atributs: host al que ens connectarem, el nom d'usuari i contrasenya.
```

```
                                                    $conn    =    new
PDO("mysql:host=$servername;dbname=$dbname", $username, $password);

                // A través de la funció setAttribute, agafem el tipus
d'error en cas de que n'hi hagi algun
                                    $conn->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);

                // Definim una preparació en la variable $stmt.
                 $stmt = $conn->prepare("SELECT id, firstname, lastname
FROM MyGuests WHERE lastname='Doe'");

                // Executem la preparació
                $stmt->execute();

                 // Imprimim la taula fent servir la crida automàtica a
les funcions de la classe RecursiveArrayIterator
                $result = $stmt->setFetchMode(PDO::FETCH_ASSOC);
                                                foreach(new   TableRows(new
RecursiveArrayIterator($stmt->fetchAll())) as $k=>$v) {
                        // new RecursiveArrayIterator($stmt->fetchAll())
converteix els registres en iteratius
                            echo $v; // Imprimeix tota la fila HTML ja
processada per les funcions
                }
            } catch(PDOException $e) {                          // En cas
de que hi hagi un error, PHP llença una PDOException, i la variable $e
agafa aquesta excepció
                 echo "Error: " . $e->getMessage();           // Si no
s'executa correctament, imprimeix el missatge d'error de PDOException
            }
            $conn = null;
            echo "</table>";
        ?>

        <h1>Codi en PHP</h1>
        <?php
        show_source("ex8.php");
        ?>
    </body>
</html>
```

Codi Propi

Resultat



Id	Firstname	Lastname	
5	Jordi	Binefa	microsoftTheBest@binefa.com
8	Jordi	Binefa	microsoftTheBest@binefa.com

Codi en PHP

```php
<!DOCTYPE html>
<html lang="ca">
    <head>
        <title>Exemple Propi Ordre WHERE Guillem Serrat</title>
        <meta http-equiv="Content-type" content="text/html; charset=UTF-8">
    </head>
    <body>
        <?php
            echo "<table style='border: solid 1px black;'>";                    // Crea la taula
            echo "<tr><th>Id</th><th>Firstname</th><th>Lastname</th></tr>";     // Crea una fila per les capceleres de la taula

            // Creem una classe que és filla de RecursiveIteratorIterator, una classe de PHP que permet recórrer estructures d'arrays
            class TableRows extends RecursiveIteratorIterator {
                // Sempre es crida al constructor 1 vegada abans de començar a iterar sobre un array
                function __construct($it) {
                    parent::__construct($it, self::LEAVES_ONLY);
                    // Amb l'operador LEAVES_ONLY ens assegurem que únicament agafa el valor i res més
                }

                // Creem una funció anomenada current. L'objectiu de la funció és crear una cel·la i agafar el valor d'un registre SQL.
                // Aquesta funció la crida automàticament per l'objecte RecursiveIteratorIterator (no cal cridar-la, es fa sol)
                function current() :string { // Afegim :string per evitar el missatge "deprecated". Indiquem que retorna un String
                    return "<td style='width:150px;border:1px solid black;'>" . parent::current(). "</td>";
                }

                // Creem una funció anomenada beginChildren. L'objectiu de la funció és crear una nova fila.
                // Aquesta funció la crida automàticament per l'objecte RecursiveIteratorIterator (no cal cridar-la, es fa sol)
                // Aquesta funció és cridada cada vegada que es fa una iteració (1 volta, 1 registre en SQL)
                function beginChildren() :void { // Afegim :void per evitar el missatge "deprecated". Indiquem que no retorna res
                    echo "<tr>"; // Creem la fila
                }

                //Creem una funció anomenada endChildren L'objectiu de la funció és tancar la fila creada per la funció endChildren
                // Aquesta funció la crida automàticament per l'objecte RecursiveIteratorIterator (no cal cridar-la, es fa sol)
                // Aquesta funció és cridada cada vegada que es fa una iteració (1 volta, 1 registre en SQL)
                function endChildren() :void { // Afegim :void per evitar el missatge "deprecated"
                    echo "</tr>" . "\n"; // Tanquem la fila
                }


                // Resum del Flux de l'Iteració
                // 1. Inici de l'Iteració (Per a la primera fila): Es crida beginChildren() --> Imprimeix <tr>.
                // 2. Iteració de la Primera Columna (id): Es crida current() --> Imprimeix <td ...>1</td>.
                // 3. Iteració de la Segona Columna (firstname): Es crida current() --> Imprimeix <td ...>John</td>.
                // 4. Iteració de la Tercera Columna (lastname): Es crida current() --> Imprimeix <td ...>Doe</td>.
                // 5. Fi de l'Iteració de la Fila: Es crida endChildren() --> Imprimeix </tr>\n.
                // 6. Repetició (Per a la següent fila), tornant al punt 1.


            }

            $servidor = "127.0.0.1";
            $usuari = "root";
            $contrasenya = "fjeclot";
            $nomDB = "dbGserrat";
            $nomTaula = "Clients";

            try {
                // Definim un nou objecte de la classe PDO amb els atributs: host al que ens connectarem, el nom d'usuari i contrasenya.
                $connexio = new PDO("mysql:host=$servidor;dbname=$nomDB", $usuari, $contrasenya);

                // A través de la funció setAttribute, agafem el tipus d'error en cas de que n'hi hagi algun
                $connexio->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

                // Definim una preparació en la variable $consultaPreparadat.
                $consultaPreparadat = $connexio->prepare("SELECT id, nom, cognom, mail FROM $nomTaula WHERE mail LIKE '%microsoft%'");

                // Executem la preparació
                $consultaPreparadat->execute();

                // Imprimim la taula fent servir la crida automàtica a les funcions de la classe RecursiveArrayIterator
                $resultat = $consultaPreparadat->setFetchMode(PDO::FETCH_ASSOC);
                foreach(new TableRows(new RecursiveArrayIterator($consultaPreparadat->fetchAll())) as $k=>$v) {
                    // new RecursiveArrayIterator($consultaPreparadat->fetchAll()) converteix els registres en iteratius
                    echo $v; // Imprimeix tota la fila HTML ja processada per les funcions
                }
            } catch(PDOException $e) {                    // En cas de que hi hagi un error, PHP llença una PDOException, i la variable $e agafa aquesta excepció
                echo "No s'ha pogut mostrar les dades" . "<br>" . $e->getMessage();          // Si no s'executa correctament, imprimeix el missatge d'error de PDOException
            }
            $connexio = null;
            echo "</table>";
        ?>

        <h1>Codi en PHP</h1>
        <?php
        show_source("ex8Propi.php");
        ?>
    </body>
</html>
```

Codi

```
<!DOCTYPE html>
<html lang="ca">
    <head>
        <title>Exemple Propi Ordre WHERE Guillem Serrat</title>
                    <meta  http-equiv="Content-type"  content="text/html;
charset=UTF-8">
    </head>
    <body>
        <?php
                        echo  "<table  style='border:  solid  1px  black;'>";
// Crea la taula
                                                                        echo
"<tr><th>Id</th><th>Firstname</th><th>Lastname</th></tr>";       // Crea
una fila per les capçeleres de la taula

                              //  Creem  una  classe  que  és  filla  de
RecursiveIteratorIterator,  una  classe  de  PHP  que  permet  recórrer
estructures d'arrays
            class TableRows extends RecursiveIteratorIterator {
                    // Sempre es crida al constructor 1 vegada abans de
començar a iterar sobre un array
                function __construct($it) {
                    parent::__construct($it, self::LEAVES_ONLY);
                        // Amb l'operador LEAVES_ONLY ens assegurem que
únicament agafa el valor i res més
                }

                // Creem una funció anomenada current. L'objectiu de la
funció és crear una cel·la i agafar el valor d'un registre SQL.
                // Aquesta funció la crida automàticament per l'objecte
RecursiveIteratorIterator (no cal cridar-la, es fa sol)
                    function current() :string { // Afegim :string per
evitar el missatge "deprecated". Indiquem que retorna un String
                        return "<td style='width:150px;border:1px solid
black;'>" . parent::current(). "</td>";
                }

                // Creem una funció anomenada beginChildren. L'objectiu
de la funció és crear una nova fila.
                // Aquesta funció la crida automàticament per l'objecte
RecursiveIteratorIterator (no cal cridar-la, es fa sol)
```

```
                // Aquesta funció és cridada cada vegada que es fa una
iteració (1 volta, 1 registre en SQL)
                function beginChildren() :void { // Afegim :void per
evitar el missatge "deprecated". Indiquem que no retorna res
                    echo "<tr>"; // Creem la fila
                }

                //Creem una funció anomenada endChildren L'objectiu de
la funció és tancar la fila creada per la funció endChildren
                // Aquesta funció la crida automàticament per l'objecte
RecursiveIteratorIterator (no cal cridar-la, es fa sol)
                // Aquesta funció és cridada cada vegada que es fa una
iteració (1 volta, 1 registre en SQL)
                    function endChildren() :void { // Afegim :void per
evitar el missatge "deprecated"
                    echo "</tr>" . "\n"; // Tanquem la fila
                }

                // Resum del Flux de l'Iteració
                // 1. Inici de l'Iteració (Per a la primera fila): Es
crida beginChildren() --> Imprimeix <tr>.
                    // 2. Iteració de la Primera Columna (id): Es crida
current() --> Imprimeix <td ...>1</td>.
                    // 3. Iteració de la Segona Columna (firstname): Es
crida current() --> Imprimeix <td ...>John</td>.
                    // 4. Iteració de la Tercera Columna (lastname): Es
crida current() --> Imprimeix <td ...>Doe</td>.
                        // 5. Fi de l'Iteració de la Fila: Es crida
endChildren() --> Imprimeix </tr>\n.
                    // 6. Repetició (Per a la següent fila), tornant al
punt 1.


            }

            $servidor = "127.0.0.1";
            $usuari = "root";
            $contrasenya = "fjeclot";
            $nomDB = "dbGserrat";
            $nomTaula = "Clients";

            try {
```

```
                    // Definim un nou objecte de la classe PDO amb els
atributs: host al que ens connectarem, el nom d'usuari i contrasenya.
                                                        $connexio    =    new
PDO("mysql:host=$servidor;dbname=$nomDB", $usuari, $contrasenya);

                  // A través de la funció setAttribute, agafem el tipus
d'error en cas de que n'hi hagi algun
                                $connexio->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);

                              // Definim una preparació en la variable
$consultaPreparadat.
                    $consultaPreparadat = $connexio->prepare("SELECT id,
nom, cognom, mail FROM $nomTaula WHERE mail LIKE '%microsoft%'");

                // Executem la preparació
                $consultaPreparadat->execute();

                  // Imprimim la taula fent servir la crida automàtica a
les funcions de la classe RecursiveArrayIterator
                                                                $resultat    =
$consultaPreparadat->setFetchMode(PDO::FETCH_ASSOC);
                                                foreach(new   TableRows(new
RecursiveArrayIterator($consultaPreparadat->fetchAll())) as $k=>$v) {
                                                                    //   new
RecursiveArrayIterator($consultaPreparadat->fetchAll())  converteix  els
registres en iteratius
                              echo  $v;  // Imprimeix tota la fila HTML ja
processada per les funcions
                }
             } catch(PDOException $e) {                        // En cas
de que hi hagi un error, PHP llença una PDOException, i la variable $e
agafa aquesta excepció
                    echo "No s'ha pogut mostrar les dades" . "<br>" .
$e->getMessage();            // Si no s'executa correctament, imprimeix
el missatge d'error de PDOException
            }
            $connexio = null;
            echo "</table>";
        ?>

        <h1>Codi en PHP</h1>
```

```
        <?php
        show_source("ex8Propi.php");
        ?>
    </body>
</html>
```

Exemple de l'ordre ORDER BY (Ex9)

Explicació de l'exemple

L'objectiu d'aquest exemple és realitzar una consulta SQL amb l'ordre ORDER BY.

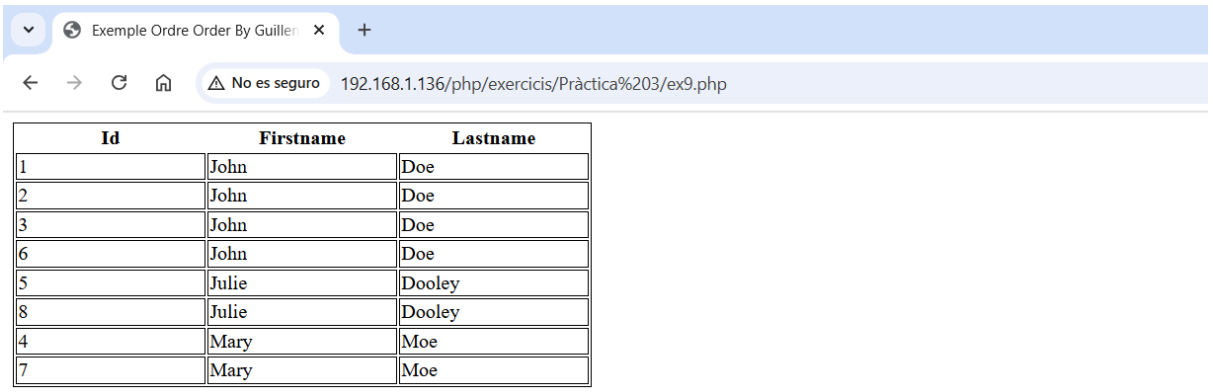
Al igual que amb l'exemple anterior, comptant amb la base de l'exemple 7, únicament cal modificar la instrucció SQL i indicar per quin camp volem ordenar els registres

```
$stmt = $conn->prepare("SELECT id, firstname, lastname FROM MyGuests ORDER BY lastname");
```

En aquest exemple estem ordenant els registres alfabèticament pel camp lastname

Codi W3Schools

Resultat



Id	Firstname	Lastname
1	John	Doe
2	John	Doe
3	John	Doe
6	John	Doe
5	Julie	Dooley
8	Julie	Dooley
4	Mary	Moe
7	Mary	Moe

Codi en PHP

```php
<!DOCTYPE html>
<html lang="ca">
    <head>
        <title>Exemple Ordre Order By Guillem Serrat</title>
        <meta http-equiv="Content-type" content="text/html; charset=UTF-8">
    </head>
    <body>
        <?php
            echo "<table style='border: solid 1px black;'>";                    // Crea la taula
            echo "<tr><th>Id</th><th>Firstname</th><th>Lastname</th></tr>";     // Crea una fila per les capçeleres de la taula

            // Creem una classe que és filla de RecursiveIteratorIterator, una classe de PHP que permet recórrer estructures d'arrays
            class TableRows extends RecursiveIteratorIterator {
                // Sempre es crida al constructor 1 vegada abans de començar a iterar sobre un array
                function __construct($it) {
                    parent::__construct($it, self::LEAVES_ONLY);
                    // Amb l'operador LEAVES_ONLY ens assegurem que únicament agafa el valor i res més
                }

                // Creem una funció anomenada current. L'objectiu de la funció és crear una cel·la i agafar el valor d'un registre SQL.
                // Aquesta funció la crida automàticament per l'objecte RecursiveIteratorIterator (no cal cridar-la, es fa sol)
                function current() :string { // Afegim :string per evitar el missatge "deprecated". Indiquem que retorna un String
                    return "<td style='width:150px;border:1px solid black;'>" . parent::current(). "</td>";
                }


                // Creem una funció anomenada beginChildren. L'objectiu de la funció és crear una nova fila.
                // Aquesta funció la crida automàticament per l'objecte RecursiveIteratorIterator (no cal cridar-la, es fa sol)
                // Aquesta funció és cridada cada vegada que es fa una iteració (1 volta, 1 registre en SQL)
                function beginChildren() :void { // Afegim :void per evitar el missatge "deprecated". Indiquem que no retorna res
                    echo "<tr>"; // Creem la fila
                }

                //Creem una funció anomenada endChildren L'objectiu de la funció és tancar la fila creada per la funció endChildren
                // Aquesta funció la crida automàticament per l'objecte RecursiveIteratorIterator (no cal cridar-la, es fa sol)
                // Aquesta funció és cridada cada vegada que es fa una iteració (1 volta, 1 registre en SQL)
                function endChildren() :void { // Afegim :void per evitar el missatge "deprecated"
                    echo "</tr>" . "\n"; // Tanquem la fila
                }

                // Resum del Flux de l'Iteració
                // 1. Inici de l'Iteració (Per a la primera fila): Es crida beginChildren() --> Imprimeix <tr>.
                // 2. Iteració de la Primera Columna (id): Es crida current() --> Imprimeix <td ...>1</td>.
                // 3. Iteració de la Segona Columna (firstname): Es crida current() --> Imprimeix <td ...>John</td>.
                // 4. Iteració de la Tercera Columna (lastname): Es crida current() --> Imprimeix <td ...>Doe</td>.
                // 5. Fi de l'Iteració de la Fila: Es crida endChildren() --> Imprimeix </tr>\n.
                // 6. Repetició (Per a la següent fila), tornant al punt 1.


            }
```

```
            $servername = "127.0.0.1";
            $username = "root";
            $password = "fjeclot";
            $dbname = "myDBPDO";

            try {
                // Definim un nou objecte de la classe PDO amb els atributs: host al que ens connectarem, el nom d'usuari i contrasenya.
                $conn = new PDO("mysql:host=$servername;dbname=$dbname", $username, $password);

                // A través de la funció setAttribute, agafem el tipus d'error en cas de que n'hi hagi algun
                $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

                // Definim una preparació en la variable $stmt.
                $stmt = $conn->prepare("SELECT id, firstname, lastname FROM MyGuests ORDER BY lastname");

                // Executem la preparació
                $stmt->execute();

                // Imprimim la taula fent servir la crida automàtica a les funcions de la classe RecursiveArrayIterator
                $result = $stmt->setFetchMode(PDO::FETCH_ASSOC);
                foreach(new TableRows(new RecursiveArrayIterator($stmt->fetchAll())) as $k=>$v) {
                    // new RecursiveArrayIterator($stmt->fetchAll()) converteix els registres en iteratius
                    echo $v; // Imprimeix tota la fila HTML ja processada per les funcions
                }
            } catch(PDOException $e) {                      // En cas de que hi hagi un error, PHP llença una PDOException, i la variable $e agafa aquesta excepció
                echo "Error: " . $e->getMessage();          // Si no s'executa correctament, imprimeix el missatge d'error de PDOException
            }
            $conn = null;
            echo "</table>";
        ?>

        <h1>Codi en PHP</h1>
        <?php
        show_source("ex9.php");
        ?>
    </body>
</html>
```

Codi

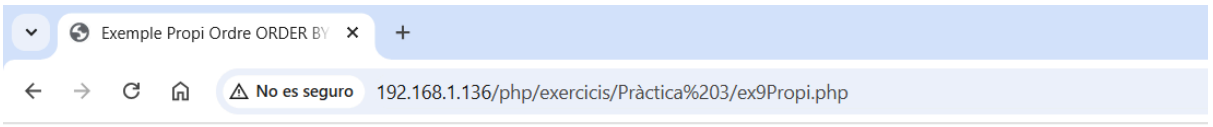
```
<!DOCTYPE html>
<html lang="ca">
    <head>
        <title>Exemple Ordre Order By Guillem Serrat</title>
                    <meta  http-equiv="Content-type"  content="text/html;
charset=UTF-8">
    </head>
    <body>
        <?php
                        echo  "<table  style='border:  solid  1px  black;'>";
// Crea la taula
                                                                        echo
"<tr><th>Id</th><th>Firstname</th><th>Lastname</th></tr>";       // Crea
una fila per les capçeleres de la taula

                              //  Creem  una  classe  que  és  filla  de
RecursiveIteratorIterator,  una  classe  de  PHP  que  permet  recórrer
estructures d'arrays
            class TableRows extends RecursiveIteratorIterator {
                    // Sempre es crida al constructor 1 vegada abans de
començar a iterar sobre un array
                function __construct($it) {
                    parent::__construct($it, self::LEAVES_ONLY);
                        // Amb l'operador LEAVES_ONLY ens assegurem que
únicament agafa el valor i res més
                }

                // Creem una funció anomenada current. L'objectiu de la
funció és crear una cel·la i agafar el valor d'un registre SQL.
                // Aquesta funció la crida automàticament per l'objecte
RecursiveIteratorIterator (no cal cridar-la, es fa sol)
                    function current() :string { // Afegim :string per
evitar el missatge "deprecated". Indiquem que retorna un String
                        return "<td style='width:150px;border:1px solid
black;'>" . parent::current(). "</td>";
                }

                // Creem una funció anomenada beginChildren. L'objectiu
de la funció és crear una nova fila.
                // Aquesta funció la crida automàticament per l'objecte
RecursiveIteratorIterator (no cal cridar-la, es fa sol)
```

```
                // Aquesta funció és cridada cada vegada que es fa una
iteració (1 volta, 1 registre en SQL)
                function beginChildren() :void { // Afegim :void per
evitar el missatge "deprecated". Indiquem que no retorna res
                    echo "<tr>"; // Creem la fila
                }

                //Creem una funció anomenada endChildren L'objectiu de
la funció és tancar la fila creada per la funció endChildren
                // Aquesta funció la crida automàticament per l'objecte
RecursiveIteratorIterator (no cal cridar-la, es fa sol)
                // Aquesta funció és cridada cada vegada que es fa una
iteració (1 volta, 1 registre en SQL)
                    function endChildren() :void { // Afegim :void per
evitar el missatge "deprecated"
                    echo "</tr>" . "\n"; // Tanquem la fila
                }

                // Resum del Flux de l'Iteració
                // 1. Inici de l'Iteració (Per a la primera fila): Es
crida beginChildren() --> Imprimeix <tr>.
                    // 2. Iteració de la Primera Columna (id): Es crida
current() --> Imprimeix <td ...>1</td>.
                    // 3. Iteració de la Segona Columna (firstname): Es
crida current() --> Imprimeix <td ...>John</td>.
                    // 4. Iteració de la Tercera Columna (lastname): Es
crida current() --> Imprimeix <td ...>Doe</td>.
                        // 5. Fi de l'Iteració de la Fila: Es crida
endChildren() --> Imprimeix </tr>\n.
                    // 6. Repetició (Per a la següent fila), tornant al
punt 1.

            }

            $servername = "127.0.0.1";
            $username = "root";
            $password = "fjeclot";
            $dbname = "myDBPDO";

            try {
                    // Definim un nou objecte de la classe PDO amb els
atributs: host al que ens connectarem, el nom d'usuari i contrasenya.
```

```
                                                    $conn    =    new
PDO("mysql:host=$servername;dbname=$dbname", $username, $password);

                // A través de la funció setAttribute, agafem el tipus
d'error en cas de que n'hi hagi algun
                                    $conn->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);

                // Definim una preparació en la variable $stmt.
                 $stmt = $conn->prepare("SELECT id, firstname, lastname
FROM MyGuests ORDER BY lastname");

                // Executem la preparació
                $stmt->execute();

                 // Imprimim la taula fent servir la crida automàtica a
les funcions de la classe RecursiveArrayIterator
                $result = $stmt->setFetchMode(PDO::FETCH_ASSOC);
                                                foreach(new   TableRows(new
RecursiveArrayIterator($stmt->fetchAll())) as $k=>$v) {
                        // new RecursiveArrayIterator($stmt->fetchAll())
converteix els registres en iteratius
                            echo $v; // Imprimeix tota la fila HTML ja
processada per les funcions
                }
            } catch(PDOException $e) {                          // En cas
de que hi hagi un error, PHP llença una PDOException, i la variable $e
agafa aquesta excepció
                 echo "Error: " . $e->getMessage();           // Si no
s'executa correctament, imprimeix el missatge d'error de PDOException
            }
            $conn = null;
            echo "</table>";
        ?>

        <h1>Codi en PHP</h1>
        <?php
        show_source("ex9.php");
        ?>
    </body>
</html>
```

Codi Propi

Resultat



Id	Firstname	Lastname	
5	Jordi	Binefa	microsoftTheBest@binefa.com
8	Jordi	Binefa	microsoftTheBest@binefa.com
2	Faiez	Mehmood	Faiez@example.com
4	Faiez	Mehmood	faiez@example.com
7	Faiez	Mehmood	faiez@example.com
1	Guillem	Serrat	guillem@example.com
3	Guillem	Serrat	guillem@example.com
6	Guillem	Serrat	guillem@example.com

Codi en PHP

```php
<!DOCTYPE html>
<html lang="ca">
    <head>
        <title>Exemple Propi Ordre ORDER BY Guillem Serrat</title>
        <meta http-equiv="Content-type" content="text/html; charset=UTF-8">
    </head>
    <body>
        <?php
            echo "<table style='border: solid 1px black;'>";                    // Crea la taula
            echo "<tr><th>Id</th><th>Firstname</th><th>Lastname</th></tr>";     // Crea una fila per les capçaleres de la taula

            // Creem una classe que és filla de RecursiveIteratorIterator, una classe de PHP que permet recórrer estructures d'arrays
            class TableRows extends RecursiveIteratorIterator {
                // Sempre es crida al constructor 1 vegada abans de començar a iterar sobre un array
                function __construct($it) {
                    parent::__construct($it, self::LEAVES_ONLY);
                    // Amb l'operador LEAVES_ONLY ens assegurem que únicament agafa el valor i res més
                }

                // Creem una funció anomenada current. L'objectiu de la funció és crear una cel·la i agafar el valor d'un registre SQL.
                // Aquesta funció la crida automàticament per l'objecte RecursiveIteratorIterator (no cal cridar-la, es fa sol)
                function current() :string { // Afegim :string per evitar el missatge "deprecated". Indiquem que retorna un String
                    return "<td style='width:150px;border:1px solid black;'>" . parent::current(). "</td>";
                }

                // Creem una funció anomenada beginChildren. L'objectiu de la funció és crear una nova fila.
                // Aquesta funció la crida automàticament per l'objecte RecursiveIteratorIterator (no cal cridar-la, es fa sol)
                // Aquesta funció és cridada cada vegada que es fa una iteració (1 volta, 1 registre en SQL)
                function beginChildren() :void { // Afegim :void per evitar el missatge "deprecated". Indiquem que no retorna res
                    echo "<tr>"; // Creem la fila
                }


                //Creem una funció anomenada endChildren L'objectiu de la funció és tancar la fila creada per la funció endChildren
                // Aquesta funció la crida automàticament per l'objecte RecursiveIteratorIterator (no cal cridar-la, es fa sol)
                // Aquesta funció és cridada cada vegada que es fa una iteració (1 volta, 1 registre en SQL)
                function endChildren() :void { // Afegim :void per evitar el missatge "deprecated"
                    echo "</tr>" . "\n"; // Tanquem la fila
                }

                // Resum del Flux de l'Iteració
                // 1. Inici de l'Iteració (Per a la primera fila): Es crida beginChildren() --> Imprimeix <tr>.
                // 2. Iteració de la Primera Columna (id): Es crida current() --> Imprimeix <td ...>1</td>.
                // 3. Iteració de la Segona Columna (firstname): Es crida current() --> Imprimeix <td ...>John</td>.
                // 4. Iteració de la Tercera Columna (lastname): Es crida current() --> Imprimeix <td ...>Doe</td>.
                // 5. Fi de l'Iteració de la Fila: Es crida endChildren() --> Imprimeix </tr>\n.
                // 6. Repetició (Per a la següent fila), tornant al punt 1.
```

```
            $servidor = "127.0.0.1";
            $usuari = "root";
            $contrasenya = "fjeclot";
            $nomDB = "dbGserrat";
            $nomTaula = "Clients";

            try {
                // Definim un nou objecte de la classe PDO amb els atributs: host al que ens connectarem, el nom d'usuari i contrasenya.
                $connexio = new PDO("mysql:host=$servidor;dbname=$nomDB", $usuari, $contrasenya);

                // A través de la funció setAttribute, agafem el tipus d'error en cas de que n'hi hagi algun
                $connexio->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

                // Definim una preparació en la variable $consultaPreparada.
                $consultaPreparada = $connexio->prepare("SELECT id, nom, cognom, mail FROM $nomTaula ORDER BY cognom");

                // Executem la preparació
                $consultaPreparada->execute();

                // Imprimim la taula fent servir la crida automàtica a les funcions de la classe RecursiveArrayIterator
                $resultat = $consultaPreparada->setFetchMode(PDO::FETCH_ASSOC);
                foreach(new TableRows(new RecursiveArrayIterator($consultaPreparada->fetchAll())) as $k=>$v) {
                    // new RecursiveArrayIterator($consultaPreparada->fetchAll()) converteix els registres en iteratius
                    echo $v; // Imprimeix tota la fila HTML ja processada per les funcions
                }
            } catch(PDOException $e) {                      // En cas de que hi hagi un error, PHP llença una PDOException, i la variable $e agafa aquesta excepció
                echo "No s'ha pogut mostrar les dades" . "<br>" . $e->getMessage();          // Si no s'executa correctament, imprimeix el missatge d'error de PDOException
            }
            $connexio = null;
            echo "</table>";
        ?>

        <h1>Codi en PHP</h1>
        <?php
        show_source("ex9Propi.php");
        ?>
    </body>
</html>
```

Codi

```
<!DOCTYPE html>
<html lang="ca">
    <head>
        <title>Exemple Propi Ordre ORDER BY Guillem Serrat</title>
                    <meta  http-equiv="Content-type"  content="text/html;
charset=UTF-8">
    </head>
    <body>
        <?php
                        echo  "<table  style='border:  solid  1px  black;'>";
// Crea la taula
                                                                        echo
"<tr><th>Id</th><th>Firstname</th><th>Lastname</th></tr>";       // Crea
una fila per les capçeleres de la taula

                              //  Creem  una  classe  que  és  filla  de
RecursiveIteratorIterator,  una  classe  de  PHP  que  permet  recórrer
estructures d'arrays
            class TableRows extends RecursiveIteratorIterator {
                    // Sempre es crida al constructor 1 vegada abans de
començar a iterar sobre un array
                function __construct($it) {
                    parent::__construct($it, self::LEAVES_ONLY);
                        // Amb l'operador LEAVES_ONLY ens assegurem que
únicament agafa el valor i res més
                }

                // Creem una funció anomenada current. L'objectiu de la
funció és crear una cel·la i agafar el valor d'un registre SQL.
                // Aquesta funció la crida automàticament per l'objecte
RecursiveIteratorIterator (no cal cridar-la, es fa sol)
                    function current() :string { // Afegim :string per
evitar el missatge "deprecated". Indiquem que retorna un String
                        return "<td style='width:150px;border:1px solid
black;'>" . parent::current(). "</td>";
                }

                // Creem una funció anomenada beginChildren. L'objectiu
de la funció és crear una nova fila.
                // Aquesta funció la crida automàticament per l'objecte
RecursiveIteratorIterator (no cal cridar-la, es fa sol)
```

```
                // Aquesta funció és cridada cada vegada que es fa una
iteració (1 volta, 1 registre en SQL)
                function beginChildren() :void { // Afegim :void per
evitar el missatge "deprecated". Indiquem que no retorna res
                    echo "<tr>"; // Creem la fila
                }

                //Creem una funció anomenada endChildren L'objectiu de
la funció és tancar la fila creada per la funció endChildren
                // Aquesta funció la crida automàticament per l'objecte
RecursiveIteratorIterator (no cal cridar-la, es fa sol)
                // Aquesta funció és cridada cada vegada que es fa una
iteració (1 volta, 1 registre en SQL)
                    function endChildren() :void { // Afegim :void per
evitar el missatge "deprecated"
                    echo "</tr>" . "\n"; // Tanquem la fila
                }

                // Resum del Flux de l'Iteració
                // 1. Inici de l'Iteració (Per a la primera fila): Es
crida beginChildren() --> Imprimeix <tr>.
                    // 2. Iteració de la Primera Columna (id): Es crida
current() --> Imprimeix <td ...>1</td>.
                    // 3. Iteració de la Segona Columna (firstname): Es
crida current() --> Imprimeix <td ...>John</td>.
                    // 4. Iteració de la Tercera Columna (lastname): Es
crida current() --> Imprimeix <td ...>Doe</td>.
                        // 5. Fi de l'Iteració de la Fila: Es crida
endChildren() --> Imprimeix </tr>\n.
                    // 6. Repetició (Per a la següent fila), tornant al
punt 1.


            }

            $servidor = "127.0.0.1";
            $usuari = "root";
            $contrasenya = "fjeclot";
            $nomDB = "dbGserrat";
            $nomTaula = "Clients";

            try {
```

```
                    // Definim un nou objecte de la classe PDO amb els
atributs: host al que ens connectarem, el nom d'usuari i contrasenya.
                                                        $connexio    =    new
PDO("mysql:host=$servidor;dbname=$nomDB", $usuari, $contrasenya);

                  // A través de la funció setAttribute, agafem el tipus
d'error en cas de que n'hi hagi algun
                                $connexio->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);

                              // Definim una preparació en la variable
$consultaPreparada.
                    $consultaPreparada = $connexio->prepare("SELECT id,
nom, cognom, mail FROM $nomTaula ORDER BY cognom");

                // Executem la preparació
                $consultaPreparada->execute();

                  // Imprimim la taula fent servir la crida automàtica a
les funcions de la classe RecursiveArrayIterator
                                                                $resultat    =
$consultaPreparada->setFetchMode(PDO::FETCH_ASSOC);
                                                foreach(new   TableRows(new
RecursiveArrayIterator($consultaPreparada->fetchAll())) as $k=>$v) {
                                                                    //   new
RecursiveArrayIterator($consultaPreparada->fetchAll())  converteix  els
registres en iteratius
                              echo $v; // Imprimeix tota la fila HTML ja
processada per les funcions
                }
             } catch(PDOException $e) {                        // En cas
de que hi hagi un error, PHP llença una PDOException, i la variable $e
agafa aquesta excepció
                    echo "No s'ha pogut mostrar les dades" . "<br>" .
$e->getMessage();            // Si no s'executa correctament, imprimeix
el missatge d'error de PDOException
            }
            $connexio = null;
            echo "</table>";
        ?>

        <h1>Codi en PHP</h1>
```

```
        <?php
        show_source("ex9Propi.php");
        ?>
    </body>
</html>
```

Exemple d'esborrament de dades (Ex10)

Explicació de l'exemple

L'objectiu d'aquest exemple és esborrar un registre dins la taula feta servir en els exemples anteriors

Per inserir dades a una taula, ho farem de la mateixa manera que per crear una taula, inserir un registre o qualsevol altre instrucció SQL . Establirem la instrucció SQL dins d'una variable, i mitjançant el mètode exec() executarem la instrucció.

Per esborrar un registre d'una taula, la instrucció és la següent:

```
$sql = "DELETE FROM MyGuests WHERE id=3";
```

Codi W3Schools

Resultat

Record deleted successfully

Codi en PHP

```php
<!DOCTYPE html>
<html lang="ca">
    <head>
        <title>Exemple Esborrament Dades Guillem Serrat</title>
        <meta http-equiv="Content-type" content="text/html; charset=UTF-8">
    </head>
    <body>
        <?php
            $servername = "127.0.0.1";
            $username = "root";
            $password = "fjeclot";
            $dbname = "myDBPDO";

            try {
                // Definim un nou objecte de la classe PDO amb els atributs: host al que ens connectarem, el nom d'usuari i contrasenya.
                $conn = new PDO("mysql:host=$servername;dbname=$dbname", $username, $password);

                // A través de la funció setAttribute, agafem el tipus d'error en cas de que n'hi hagi algun
                $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

                // Definim una variable que és l'ordre que s'executara, la qual és esborrar el registre amb ID = 3. La BBDD en la qual es treballarà està especificat a la connexió
                $sql = "DELETE FROM MyGuests WHERE id=3";

                // Dins de la connexió, executem la variable sql (que és l'ordre vista anteriorment)
                $conn->exec($sql);
                echo "Record deleted successfully";

            } catch(PDOException $e) {                  // En cas de que hi hagi un error, PHP llença una PDOException, i la variable $e agafa aquesta excepció
            echo $sql . "<br>" . $e->getMessage();      // Si no es crea correctament, imprimeix el missatge d'error de PDOException
            }

            $conn = null;                               // Tanca la connexió amb la BBDD
            ?>

        <h1>Codi en PHP</h1>
        <?php
        show_source("ex10.php");
        ?>
    </body>
</html>
```

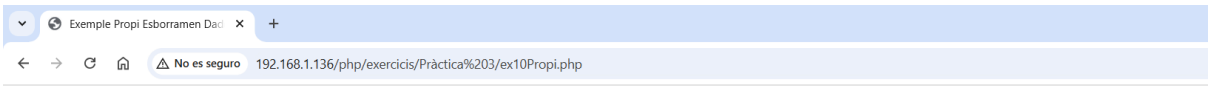
Codi

```
<!DOCTYPE html>
<html lang="ca">
    <head>
        <title>Exemple Esborrament Dades Guillem Serrat</title>
                    <meta  http-equiv="Content-type"  content="text/html;
charset=UTF-8">
    </head>
    <body>
        <?php
            $servername = "127.0.0.1";
            $username = "root";
            $password = "fjeclot";
            $dbname = "myDBPDO";

            try {
                    // Definim un nou objecte de la classe PDO amb els
atributs: host al que ens connectarem, el nom d'usuari i contrasenya.
                                                        $conn    =    new
PDO("mysql:host=$servername;dbname=$dbname", $username, $password);

                // A través de la funció setAttribute, agafem el tipus
d'error en cas de que n'hi hagi algun
                                    $conn->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);

                // Definim una variable que és l'ordre que s'executara,
la qual és esborrar el registre amb ID = 3. La BBDD en la qual es
treballarà està especificat a la connexió
                $sql = "DELETE FROM MyGuests WHERE id=3";

                 // Dins de la connexió, executem la variable sql (que
és l'ordre vista anteriorment)
                $conn->exec($sql);
                echo "Record deleted successfully";

            } catch(PDOException $e) {                    // En cas de
que hi hagi un error, PHP llença una PDOException, i la variable $e
agafa aquesta excepció
                echo $sql . "<br>" . $e->getMessage();       // Si no es
crea correctament, imprimeix el missatge d'error de PDOException
            }
```

```
            $conn = null;                                   // Tanca la
connexió amb la BBDD
        ?>

        <h1>Codi en PHP</h1>
        <?php
        show_source("ex10.php");
        ?>
    </body>
</html>
```

Codi Propi

Resultat



Registres esborrats correctament

Codi en PHP

```php
<!DOCTYPE html>
<html lang="ca">
    <head>
        <title>Exemple Propi Esborramen Dades Guillem Serrat</title>
        <meta http-equiv="Content-type" content="text/html; charset=UTF-8">
    </head>
    <body>
        <?php
            $servidor = "127.0.0.1";
            $usuari = "root";
            $contrasenya = "fjeclot";
            $nomDB = "dbGserrat";
            $nomTaula = "Clients";

            try {
                // Definim un nou objecte de la classe PDO amb els atributs: host al que ens connectarem, el nom d'usuari i contrasenya.
                $connexio = new PDO("mysql:host=$servidor;dbname=$nomDB", $usuari, $contrasenya);

                // A través de la funció setAttribute, agafem el tipus d'error en cas de que n'hi hagi algun
                $connexio->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

                // Definim una variable que és l'ordre que s'executara, la qual és esborrar el registre amb ID = 3. La BBDD en la qual es treballarà està especificat a la connexió
                $instruccioSQL = "DELETE FROM $nomTaula WHERE id=6";
                $instruccioSQL2 = "DELETE FROM $nomTaula WHERE id=2";

                // Dins de la connexió, executem la variable sql (que és l'ordre vista anteriorment)
                $connexio->exec($instruccioSQL);
                $connexio->exec($instruccioSQL2);
                echo "Registres esborrats correctament";

            } catch(PDOException $e) {                  // En cas de que hi hagi un error, PHP llença una PDOException, i la variable $e agafa aquesta excepció
            echo "Algun registre no s'ha esborrat correctament" . "<br>" . $e->getMessage();      // Si no es crea correctament, imprimeix el missatge d'error de PDOException
            }

            $connexio = null;                           // Tanca la connexió amb la BBDD
            ?>

        <h1>Codi en PHP</h1>
        <?php
        show_source("ex10Propi.php");
        ?>
```

Codi

```
<!DOCTYPE html>
<html lang="ca">
    <head>
        <title>Exemple Propi Esborramen Dades Guillem Serrat</title>
                    <meta  http-equiv="Content-type"  content="text/html;
charset=UTF-8">
    </head>
    <body>
        <?php
            $servidor = "127.0.0.1";
            $usuari = "root";
            $contrasenya = "fjeclot";
            $nomDB = "dbGserrat";
            $nomTaula = "Clients";

            try {
                    // Definim un nou objecte de la classe PDO amb els
atributs: host al que ens connectarem, el nom d'usuari i contrasenya.
                                                    $connexio    =    new
PDO("mysql:host=$servidor;dbname=$nomDB", $usuari, $contrasenya);

                // A través de la funció setAttribute, agafem el tipus
d'error en cas de que n'hi hagi algun
                            $connexio->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);

                // Definim una variable que és l'ordre que s'executara,
la qual és esborrar el registre amb ID = 3. La BBDD en la qual es
treballarà està especificat a la connexió
                $instruccioSQL = "DELETE FROM $nomTaula WHERE id=6";
                $instruccioSQL2 = "DELETE FROM $nomTaula WHERE id=2";

                // Dins de la connexió, executem la variable sql (que
és l'ordre vista anteriorment)
                $connexio->exec($instruccioSQL);
                $connexio->exec($instruccioSQL2);
                echo "Registres esborrats correctament";

            } catch(PDOException $e) {                    // En cas de
que hi hagi un error, PHP llença una PDOException, i la variable $e
agafa aquesta excepció
```

```
                echo "Algun registre no s'ha esborrat correctament" .
"<br>"  .  $e->getMessage();          //  Si  no  es  crea  correctament,
imprimeix el missatge d'error de PDOException
            }

             $connexio = null;                                  // Tanca la
connexió amb la BBDD
            ?>

        <h1>Codi en PHP</h1>
        <?php
        show_source("ex10Propi.php");
        ?>
    </body>
</html>
```

Exemple d'actualització de dades (Ex11)

Explicació de l'exemple

L'objectiu d'aquest exemple és actualitzar un registre dins la taula feta servir en els exemples anteriors

Per actualitzar un registre, ho farem de la mateixa manera que per crear una taula, inserir un registre o qualsevol altre instrucció SQL . Establirem la instrucció SQL dins d'una variable, i mitjançant el mètode exec() executarem la instrucció.

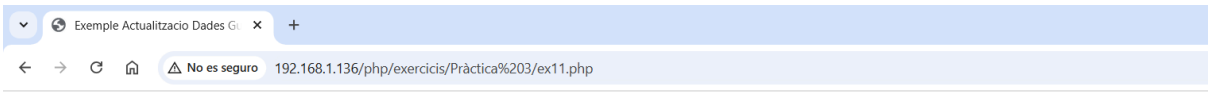
Per esborrar un registre d'una taula, la instrucció és la següent:

```
$sql = "UPDATE MyGuests SET lastname='Doe' WHERE id=2";
```

També farem servir una declaració preparada per obtenir els seus beneficis (https://www.php.net/manual/en/pdo.transactions.php)

Codi W3Schools

Resultat



0 records UPDATED successfully

Codi en PHP

```php
<!DOCTYPE html>
<html lang="ca">
    <head>
        <title>Exemple Actualitzacio Dades Guillem Serrat</title>
        <meta http-equiv="Content-type" content="text/html; charset=UTF-8">
    </head>
    <body>
        <?php
            $servername = "127.0.0.1";
            $username = "root";
            $password = "fjeclot";
            $dbname = "myDBPDO";

            try {
                // Definim un nou objecte de la classe PDO amb els atributs: host al que ens connectarem, el nom d'usuari i contrasenya.
                $conn = new PDO("mysql:host=$servername;dbname=$dbname", $username, $password);

                // A través de la funció setAttribute, agafem el tipus d'error en cas de que n'hi hagi algun
                $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

                // Definim una variable que és l'ordre que s'executara, actualitzar el lastname del registre amb ID = 2. La BBDD en la qual es treballarà està especificat a la connexió
                $sql = "UPDATE MyGuests SET lastname='Doe' WHERE id=2";

                // Definim una preparació, en aquest cas, amb valors fixos (la variable $sql que ja té definida l'ordre)
                $stmt = $conn->prepare($sql);

                // Executem la preparació (l'ordre de la variable $sql)
                $stmt->execute();

                // Funció rowCount() retorna el nombre de files afectades per l'operació. Per tant, es comptaran les files afectades per la preparació ($stmt), que és l'UPDATE d'una fila.
                // En cas de que es faci l'UPDATE, rowCount() valdrà 1, en cas que no es faci (ja que ja té aquell valor), valdrà 0.
                echo $stmt->rowCount() . " records UPDATED successfully";

            } catch(PDOException $e) {                      // En cas de que hi hagi un error, PHP llença una PDOException, i la variable $e agafa aquesta excepció
                echo $sql . "<br>" . $e->getMessage();      // Si no es crea correctament, imprimeix el missatge d'error de PDOException
            }

            $conn = null;                                   // Tanca la connexió amb la BBDD
        ?>

        <h1>Codi en PHP</h1>
        <?php
        show_source("ex11.php");
        ?>
    </body>
</html>
```

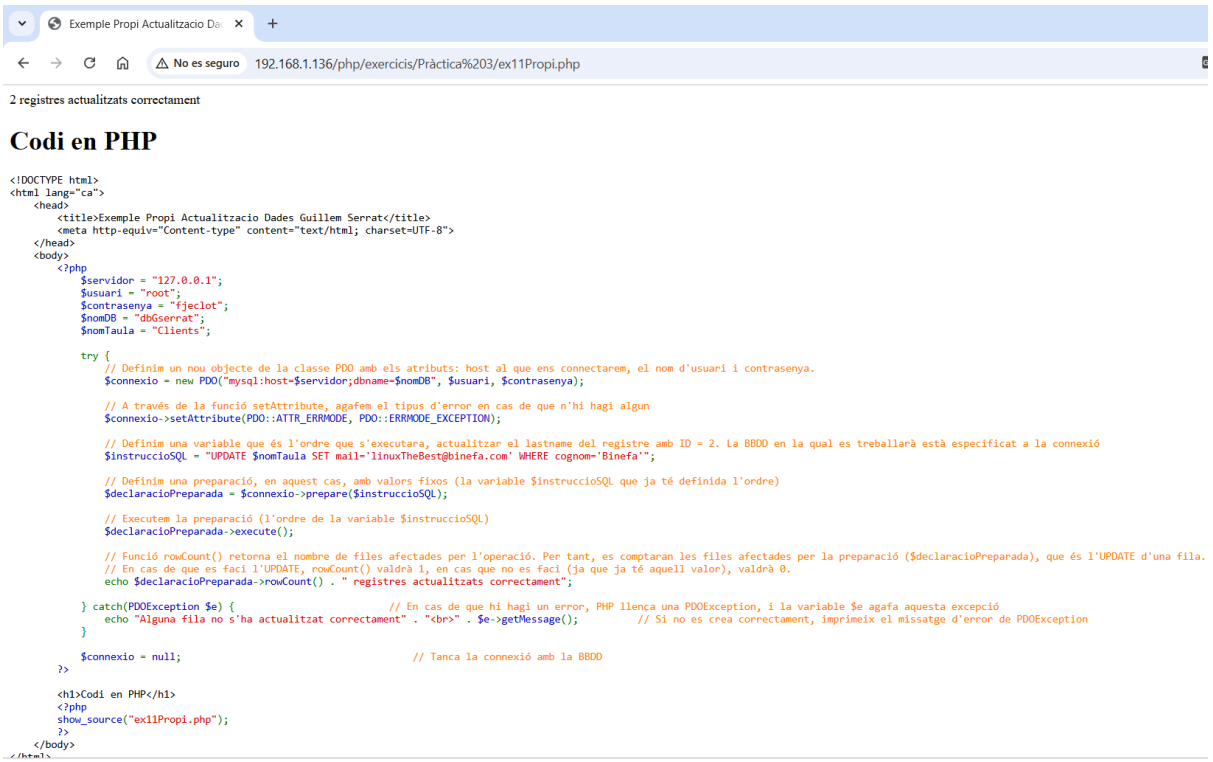
Codi

```
<!DOCTYPE html>
<html lang="ca">
    <head>
        <title>Exemple Actualitzacio Dades Guillem Serrat</title>
                    <meta  http-equiv="Content-type"  content="text/html;
charset=UTF-8">
    </head>
    <body>
        <?php
            $servername = "127.0.0.1";
            $username = "root";
            $password = "fjeclot";
            $dbname = "myDBPDO";

            try {
                    // Definim un nou objecte de la classe PDO amb els
atributs: host al que ens connectarem, el nom d'usuari i contrasenya.
                                                        $conn    =    new
PDO("mysql:host=$servername;dbname=$dbname", $username, $password);

                // A través de la funció setAttribute, agafem el tipus
d'error en cas de que n'hi hagi algun
                                    $conn->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);

                // Definim una variable que és l'ordre que s'executara,
actualitzar el lastname del registre amb ID = 2. La BBDD en la qual es
treballarà està especificat a la connexió
                $sql = "UPDATE MyGuests SET lastname='Doe' WHERE id=2";

                // Definim una preparació, en aquest cas, amb valors
fixos (la variable $sql que ja té definida l'ordre)
                $stmt = $conn->prepare($sql);

                // Executem la preparació (l'ordre de la variable $sql)
                $stmt->execute();

                    // Funció rowCount() retorna el nombre de files
afectades per l'operació. Per tant, es comptaran les files afectades
per la preparació ($stmt), que és l'UPDATE d'una fila.
```

```
                // En cas de que es faci l'UPDATE, rowCount() valdrà 1,
en cas que no es faci (ja que ja té aquell valor), valdrà 0.
                        echo $stmt->rowCount() . " records UPDATED
successfully";

            } catch(PDOException $e) {                          // En
cas de que hi hagi un error, PHP llença una PDOException, i la variable
$e agafa aquesta excepció
                echo $sql . "<br>" . $e->getMessage();          // Si
no es crea correctament, imprimeix el missatge d'error de PDOException
            }

            $conn = null;                                       //
Tanca la connexió amb la BBDD
        ?>

        <h1>Codi en PHP</h1>
        <?php
        show_source("ex11.php");
        ?>
    </body>
</html>
```

Codi Propi

Resultat




```php
<!DOCTYPE html>
<html lang="ca">
    <head>
        <title>Exemple Propi Actualitzacio Dades Guillem Serrat</title>
        <meta http-equiv="Content-type" content="text/html; charset=UTF-8">
    </head>
    <body>
        <?php
            $servidor = "127.0.0.1";
            $usuari = "root";
            $contrasenya = "fjeclot";
            $nomDB = "dbGserrat";
            $nomTaula = "Clients";

            try {
                // Definim un nou objecte de la classe PDO amb els atributs: host al que ens connectarem, el nom d'usuari i contrasenya.
                $connexio = new PDO("mysql:host=$servidor;dbname=$nomDB", $usuari, $contrasenya);

                // A través de la funció setAttribute, agafem el tipus d'error en cas de que n'hi hagi algun
                $connexio->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

                // Definim una variable que és l'ordre que s'executara, actualitzar el lastname del registre amb ID = 2. La BBDD en la qual es treballarà està especificat a la connexió
                $instruccioSQL = "UPDATE $nomTaula SET mail='linuxTheBest@binefa.com' WHERE cognom='Binefa'";

                // Definim una preparació, en aquest cas, amb valors fixos (la variable $instruccioSQL que ja té definida l'ordre)
                $declaracioPreparada = $connexio->prepare($instruccioSQL);

                // Executem la preparació (l'ordre de la variable $instruccioSQL)
                $declaracioPreparada->execute();

                // Funció rowCount() retorna el nombre de files afectades per l'operació. Per tant, es comptaran les files afectades per la preparació ($declaracioPreparada), que és l'UPDATE d'una fila.
                // En cas de que es faci l'UPDATE, rowCount() valdrà 1, en cas que no es faci (ja que ja té aquell valor), valdrà 0.
                echo $declaracioPreparada->rowCount() . " registres actualitzats correctament";

            } catch(PDOException $e) {                          // En cas de que hi hagi un error, PHP llença una PDOException, i la variable $e agafa aquesta excepció
                echo "Alguna fila no s'ha actualitzat correctament" . "<br>" . $e->getMessage();          // Si no es crea correctament, imprimeix el missatge d'error de PDOException
            }

            $connexio = null;                                   // Tanca la connexió amb la BBDD
        ?>

        <h1>Codi en PHP</h1>
        <?php
        show_source("ex11Propi.php");
        ?>
    </body>
</html>
```

Codi

```
<!DOCTYPE html>
<html lang="ca">
    <head>
        <title>Exemple Propi Actualitzacio Dades Guillem Serrat</title>
                    <meta  http-equiv="Content-type"  content="text/html;
charset=UTF-8">
    </head>
    <body>
        <?php
            $servidor = "127.0.0.1";
            $usuari = "root";
            $contrasenya = "fjeclot";
            $nomDB = "dbGserrat";
            $nomTaula = "Clients";

            try {
                    // Definim un nou objecte de la classe PDO amb els
atributs: host al que ens connectarem, el nom d'usuari i contrasenya.
                                                        $connexio    =    new
PDO("mysql:host=$servidor;dbname=$nomDB", $usuari, $contrasenya);

                // A través de la funció setAttribute, agafem el tipus
d'error en cas de que n'hi hagi algun
                            $connexio->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);

                // Definim una variable que és l'ordre que s'executara,
actualitzar el lastname del registre amb ID = 2. La BBDD en la qual es
treballarà està especificat a la connexió
                                $instruccioSQL  =  "UPDATE  $nomTaula  SET
mail='linuxTheBest@binefa.com' WHERE cognom='Binefa'";

                // Definim una preparació, en aquest cas, amb valors
fixos (la variable $instruccioSQL que ja té definida l'ordre)
                                                    $declaracioPreparada    =
$connexio->prepare($instruccioSQL);

                    // Executem la preparació (l'ordre de la variable
$instruccioSQL)
                $declaracioPreparada->execute();
```

```
                    // Funció rowCount() retorna el nombre de files
afectades per l'operació. Per tant, es comptaran les files afectades
per la preparació ($declaracioPreparada), que és l'UPDATE d'una fila.
                // En cas de que es faci l'UPDATE, rowCount() valdrà 1,
en cas que no es faci (ja que ja té aquell valor), valdrà 0.
                    echo $declaracioPreparada->rowCount() . " registres
actualitzats correctament";

            } catch(PDOException $e) {                          // En
cas de que hi hagi un error, PHP llença una PDOException, i la variable
$e agafa aquesta excepció
                    echo "Alguna fila no s'ha actualitzat correctament" .
"<br>" . $e->getMessage();              // Si no es crea correctament,
imprimeix el missatge d'error de PDOException
            }

            $connexio = null;                                       //
Tanca la connexió amb la BBDD
        ?>

        <h1>Codi en PHP</h1>
        <?php
        show_source("ex11Propi.php");
        ?>
    </body>
</html>
```

Exemple de limitació de dades (Ex12)

Explicació de l'exemple

L'objectiu d'aquest exemple és mostrar registres de la base de dades amb un límit i indicant el número de registre pel qual començar

Per mostrar els registres d'una base de dades amb un límit i indicant per on començar, ho especifiquem en l'ordre SQL.

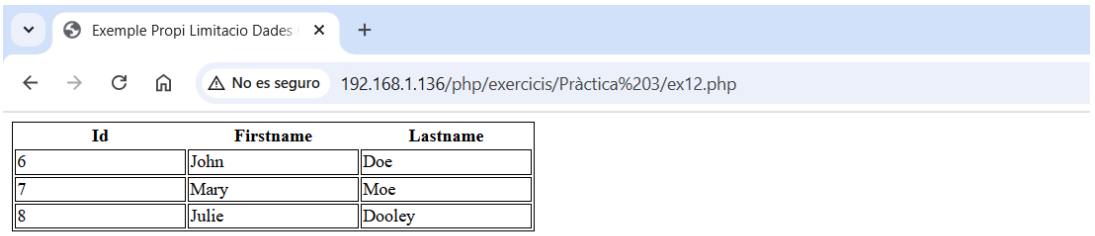
- Amb el paràmetre LIMIT indiquem el nombre de registres que volem que apareguin
- Amb el paràmetre OFFSET indiquem a partir de quin número de registre s'ha de mostrar
    - NO és el mateix número de registre que registre amb ID = n.
        - Si no s'esborra cap registre, el nº de registre i la ID coincidiran. Si esborrem el registre amb ID = 2, el registre nº2 serà el registre amb ID = 3

```
$stmt = $conn->prepare("SELECT id, firstname, lastname FROM MyGuests LIMIT 3 OFFSET 4");
```

En aquest cas, únicament mostrem 3 registres començant pel registre número 4.

Codi W3Schools

Resultat



Id	Firstname	Lastname
6	John	Doe
7	Mary	Moe
8	Julie	Dooley

Codi en PHP

```php
<!DOCTYPE html>
<html lang="ca">
    <head>
        <title>Exemple Propi Limitacio Dades Guillem Serrat</title>
        <meta http-equiv="Content-type" content="text/html; charset=UTF-8">
    </head>
    <body>
        <?php
            echo "<table style='border: solid 1px black;'>";                    // Crea la taula
            echo "<tr><th>Id</th><th>Firstname</th><th>Lastname</th></tr>";     // Crea una fila per les capçeleres de la taula

            // Creem una classe que és filla de RecursiveIteratorIterator, una classe de PHP que permet recórrer estructures d'arrays
            class TableRows extends RecursiveIteratorIterator {
                // Sempre es crida al constructor 1 vegada abans de començar a iterar sobre un array
                function __construct($it) {
                    parent::__construct($it, self::LEAVES_ONLY);
                    // Amb l'operador LEAVES_ONLY ens assegurem que únicament agafa el valor i res més
                }

                // Creem una funció anomenada current. L'objectiu de la funció és crear una cel·la i agafar el valor d'un registre SQL.
                // Aquesta funció la crida automàticament per l'objecte RecursiveIteratorIterator (no cal cridar-la, es fa sol)
                function current() :string { // Afegim :string per evitar el missatge "deprecated". Indiquem que retorna un String
                    return "<td style='width:150px;border:1px solid black;'>" . parent::current(). "</td>";
                }

                // Creem una funció anomenada beginChildren. L'objectiu de la funció és crear una nova fila.
                // Aquesta funció la crida automàticament per l'objecte RecursiveIteratorIterator (no cal cridar-la, es fa sol)
                // Aquesta funció és cridada cada vegada que es fa una iteració (1 volta, 1 registre en SQL)
                function beginChildren() :void { // Afegim :void per evitar el missatge "deprecated". Indiquem que no retorna res
                    echo "<tr>"; // Creem la fila
                }

                //Creem una funció anomenada endChildren L'objectiu de la funció és tancar la fila creada per la funció endChildren
                // Aquesta funció la crida automàticament per l'objecte RecursiveIteratorIterator (no cal cridar-la, es fa sol)
                // Aquesta funció és cridada cada vegada que es fa una iteració (1 volta, 1 registre en SQL)
                function endChildren() :void { // Afegim :void per evitar el missatge "deprecated"
                    echo "</tr>" . "\n"; // Tanquem la fila
                }


                // Resum del Flux de l'Iteració
                // 1. Inici de l'Iteració (Per a la primera fila): Es crida beginChildren() --> Imprimeix <tr>.
                // 2. Iteració de la Primera Columna (id): Es crida current() --> Imprimeix <td ...>1</td>.
                // 3. Iteració de la Segona Columna (firstname): Es crida current() --> Imprimeix <td ...>John</td>.
                // 4. Iteració de la Tercera Columna (lastname): Es crida current() --> Imprimeix <td ...>Doe</td>.
                // 5. Fi de l'Iteració de la Fila: Es crida endChildren() --> Imprimeix </tr>\n.
                // 6. Repetició (Per a la següent fila), tornant al punt 1.


            }

            $servername = "127.0.0.1";
            $username = "root";
            $password = "fjeclot";
            $dbname = "myDBPDO";

            try {
                // Definim un nou objecte de la classe PDO amb els atributs: host al que ens connectarem, el nom d'usuari i contrasenya.
                $conn = new PDO("mysql:host=$servername;dbname=$dbname", $username, $password);

                // A través de la funció setAttribute, agafem el tipus d'error en cas de que n'hi hagi algun
                $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

                // Definim una preparació en la variable $stmt.
                // En aquest cas, estem demanant únicament 3 registres començant pel registre nº4 (NO el registre amb ID=4, sinó el registre número 4)
                $stmt = $conn->prepare("SELECT id, firstname, lastname FROM MyGuests LIMIT 3 OFFSET 4");

                // Executem la preparació
                $stmt->execute();

                // Imprimim la taula fent servir la crida automàtica a les funcions de la classe RecursiveArrayIterator
                $result = $stmt->setFetchMode(PDO::FETCH_ASSOC);
                foreach(new TableRows(new RecursiveArrayIterator($stmt->fetchAll())) as $k=>$v) {
                    // new RecursiveArrayIterator($stmt->fetchAll()) converteix els registres en iteratius
                    echo $v; // Imprimeix tota la fila HTML ja processada per les funcions
                }
            } catch(PDOException $e) {                  // En cas de que hi hagi un error, PHP llença una PDOException, i la variable $e agafa aquesta excepció
                echo "Error: " . $e->getMessage();      // Si no s'executa correctament, imprimeix el missatge d'error de PDOException
            }
            $conn = null;
            echo "</table>";
        ?>

        <h1>Codi en PHP</h1>
        <?php
        show_source("ex12.php");
        ?>
    </body>
</html>
```

Codi

```
<!DOCTYPE html>
<html lang="ca">
    <head>
        <title>Exemple Propi Limitacio Dades Guillem Serrat</title>
                    <meta  http-equiv="Content-type"  content="text/html;
charset=UTF-8">
    </head>
    <body>
        <?php
                        echo  "<table  style='border:  solid  1px  black;'>";
// Crea la taula
                                                                        echo
"<tr><th>Id</th><th>Firstname</th><th>Lastname</th></tr>";       // Crea
una fila per les capçeleres de la taula

                              //  Creem  una  classe  que  és  filla  de
RecursiveIteratorIterator,  una  classe  de  PHP  que  permet  recórrer
estructures d'arrays
            class TableRows extends RecursiveIteratorIterator {
                    // Sempre es crida al constructor 1 vegada abans de
començar a iterar sobre un array
                function __construct($it) {
                    parent::__construct($it, self::LEAVES_ONLY);
                        // Amb l'operador LEAVES_ONLY ens assegurem que
únicament agafa el valor i res més
                }

                // Creem una funció anomenada current. L'objectiu de la
funció és crear una cel·la i agafar el valor d'un registre SQL.
                // Aquesta funció la crida automàticament per l'objecte
RecursiveIteratorIterator (no cal cridar-la, es fa sol)
                    function current() :string { // Afegim :string per
evitar el missatge "deprecated". Indiquem que retorna un String
                        return "<td style='width:150px;border:1px solid
black;'>" . parent::current(). "</td>";
                }

                // Creem una funció anomenada beginChildren. L'objectiu
de la funció és crear una nova fila.
                // Aquesta funció la crida automàticament per l'objecte
RecursiveIteratorIterator (no cal cridar-la, es fa sol)
```

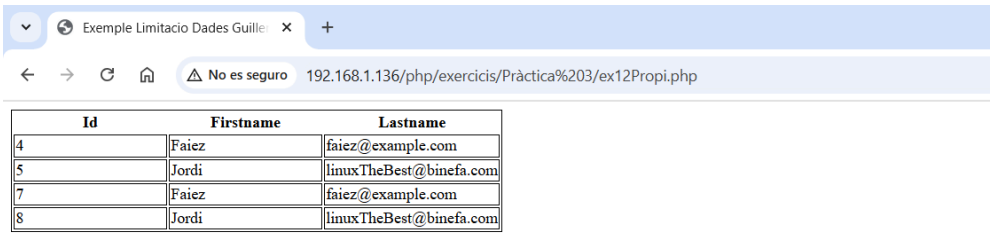
```
                // Aquesta funció és cridada cada vegada que es fa una
iteració (1 volta, 1 registre en SQL)
                function beginChildren() :void { // Afegim :void per
evitar el missatge "deprecated". Indiquem que no retorna res
                    echo "<tr>"; // Creem la fila
                }

                //Creem una funció anomenada endChildren L'objectiu de
la funció és tancar la fila creada per la funció endChildren
                // Aquesta funció la crida automàticament per l'objecte
RecursiveIteratorIterator (no cal cridar-la, es fa sol)
                // Aquesta funció és cridada cada vegada que es fa una
iteració (1 volta, 1 registre en SQL)
                    function endChildren() :void { // Afegim :void per
evitar el missatge "deprecated"
                    echo "</tr>" . "\n"; // Tanquem la fila
                }

                // Resum del Flux de l'Iteració
                // 1. Inici de l'Iteració (Per a la primera fila): Es
crida beginChildren() --> Imprimeix <tr>.
                    // 2. Iteració de la Primera Columna (id): Es crida
current() --> Imprimeix <td ...>1</td>.
                    // 3. Iteració de la Segona Columna (firstname): Es
crida current() --> Imprimeix <td ...>John</td>.
                    // 4. Iteració de la Tercera Columna (lastname): Es
crida current() --> Imprimeix <td ...>Doe</td>.
                        // 5. Fi de l'Iteració de la Fila: Es crida
endChildren() --> Imprimeix </tr>\n.
                    // 6. Repetició (Per a la següent fila), tornant al
punt 1.


            }

            $servername = "127.0.0.1";
            $username = "root";
            $password = "fjeclot";
            $dbname = "myDBPDO";

            try {
```

```
                    // Definim un nou objecte de la classe PDO amb els
atributs: host al que ens connectarem, el nom d'usuari i contrasenya.
                                                        $conn    =    new
PDO("mysql:host=$servername;dbname=$dbname", $username, $password);

                    // A través de la funció setAttribute, agafem el tipus
d'error en cas de que n'hi hagi algun
                                    $conn->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);

                // Definim una preparació en la variable $stmt.
                    // En aquest cas, estem demanant únicament 3 registres
començant pel registre nº4 (NO el registre amb ID=4, sinó el registre
número 4)
                    $stmt = $conn->prepare("SELECT id, firstname, lastname
FROM MyGuests LIMIT 3 OFFSET 4");

                // Executem la preparació
                $stmt->execute();

                    // Imprimim la taula fent servir la crida automàtica a
les funcions de la classe RecursiveArrayIterator
                $result = $stmt->setFetchMode(PDO::FETCH_ASSOC);
                                            foreach(new   TableRows(new
RecursiveArrayIterator($stmt->fetchAll())) as $k=>$v) {
                        // new RecursiveArrayIterator($stmt->fetchAll())
converteix els registres en iteratius
                            echo $v; // Imprimeix tota la fila HTML ja
processada per les funcions
                }
            } catch(PDOException $e) {                          // En cas
de que hi hagi un error, PHP llença una PDOException, i la variable $e
agafa aquesta excepció
                    echo "Error: " . $e->getMessage();           // Si no
s'executa correctament, imprimeix el missatge d'error de PDOException
            }
            $conn = null;
            echo "</table>";
        ?>

        <h1>Codi en PHP</h1>
        <?php
```

```
        show_source("ex12.php");
        ?>
    </body>
</html>
```

Codi Propi

Resultat



Id	Firstname	Lastname
4	Faiez	faiez@example.com
5	Jordi	linuxTheBest@binefa.com
7	Faiez	faiez@example.com
8	Jordi	linuxTheBest@binefa.com

Codi en PHP

```php
<!DOCTYPE html>
<html lang="ca">
    <head>
        <title>Exemple Limitacio Dades Guillem Serrat</title>
        <meta http-equiv="Content-type" content="text/html; charset=UTF-8">
    </head>
    <body>
        <?php
            echo "<table style='border: solid 1px black;'>";                    // Crea la taula
            echo "<tr><th>Id</th><th>Firstname</th><th>Lastname</th></tr>";     // Crea una fila per les capçeleres de la taula

            // Creem una classe que és filla de RecursiveIteratorIterator, una classe de PHP que permet recórrer estructures d'arrays
            class TableRows extends RecursiveIteratorIterator {
                // Sempre es crida al constructor 1 vegada abans de començar a iterar sobre un array
                function __construct($it) {
                    parent::__construct($it, self::LEAVES_ONLY);
                    // Amb l'operador LEAVES_ONLY ens assegurem que únicament agafa el valor i res més
                }

                // Creem una funció anomenada current. L'objectiu de la funció és crear una cel·la i agafar el valor d'un registre SQL.
                // Aquesta funció la crida automàticament per l'objecte RecursiveIteratorIterator (no cal cridar-la, es fa sol)
                function current() :string { // Afegim :string per evitar el missatge "deprecated". Indiquem que retorna un String
                    return "<td style='width:150px;border:1px solid black;'>" . parent::current(). "</td>";
                }

                // Creem una funció anomenada beginChildren. L'objectiu de la funció és crear una nova fila.
                // Aquesta funció la crida automàticament per l'objecte RecursiveIteratorIterator (no cal cridar-la, es fa sol)
                // Aquesta funció és cridada cada vegada que es fa una iteració (1 volta, 1 registre en SQL)
                function beginChildren() :void { // Afegim :void per evitar el missatge "deprecated". Indiquem que no retorna res
                    echo "<tr>"; // Creem la fila
                }

                //Creem una funció anomenada endChildren L'objectiu de la funció és tancar la fila creada per la funció endChildren
                // Aquesta funció la crida automàticament per l'objecte RecursiveIteratorIterator (no cal cridar-la, es fa sol)
                // Aquesta funció és cridada cada vegada que es fa una iteració (1 volta, 1 registre en SQL)
                function endChildren() :void { // Afegim :void per evitar el missatge "deprecated"
                    echo "</tr>" . "\n"; // Tanquem la fila
                }


                 // Resum del Flux de l'Iteració
                 // 1. Inici de l'Iteració (Per a la primera fila): Es crida beginChildren() --> Imprimeix <tr>.
                 // 2. Iteració de la Primera Columna (id): Es crida current() --> Imprimeix <td ...>1</td>.
                 // 3. Iteració de la Segona Columna (firstname): Es crida current() --> Imprimeix <td ...>John</td>.
                 // 4. Iteració de la Tercera Columna (lastname): Es crida current() --> Imprimeix <td ...>Doe</td>.
                 // 5. Fi de l'Iteració de la Fila: Es crida endChildren() --> Imprimeix </tr>\n.
                 // 6. Repetició (Per a la següent fila), tornant al punt 1.


            }

            $servidor = "127.0.0.1";
            $usuari = "root";
            $contrasenya = "fjeclot";
            $nomDB = "dbGserrat";
            $nomTaula = "Clients";

            try {
                // Definim un nou objecte de la classe PDO amb els atributs: host al que ens connectarem, el nom d'usuari i contrasenya.
                $connexio = new PDO("mysql:host=$servidor;dbname=$nomDB", $usuari, $contrasenya);

                // A través de la funció setAttribute, agafem el tipus d'error en cas de que n'hi hagi algun
                $connexio->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

                // Definim una preparació en la variable $consultaPreparada.
                $consultaPreparada = $connexio->prepare("SELECT id, nom, mail FROM $nomTaula LIMIT 5 OFFSET 2");

                // Executem la preparació
                $consultaPreparada->execute();

                // Imprimim la taula fent servir la crida automàtica a les funcions de la classe RecursiveArrayIterator
                $resultat = $consultaPreparada->setFetchMode(PDO::FETCH_ASSOC);
                foreach(new TableRows(new RecursiveArrayIterator($consultaPreparada->fetchAll())) as $k=>$v) {
                    // new RecursiveArrayIterator($consultaPreparada->fetchAll()) converteix els registres en iteratius
                    echo $v; // Imprimeix tota la fila HTML ja processada per les funcions
                }
            } catch(PDOException $e) {                                          // En cas de que hi hagi un error, Pl
                echo "No s'han progut extreure els resultats" . "<br>" . $e->getMessage();          // Si no s'executa correctament, imp
            }
            $connexio = null;                                                       // Tanca la connexió amb la BBDD
            echo "</table>";
        ?>

        <h1>Codi en PHP</h1>
        <?php
        show_source("ex12Propi.php");
        ?>
    </body>
</html>
```

Codi

```
<!DOCTYPE html>
<html lang="ca">
    <head>
        <title>Exemple Limitacio Dades Guillem Serrat</title>
                    <meta  http-equiv="Content-type"  content="text/html;
charset=UTF-8">
    </head>
    <body>
        <?php
                        echo  "<table  style='border:  solid  1px  black;'>";
// Crea la taula
                                                                        echo
"<tr><th>Id</th><th>Firstname</th><th>Lastname</th></tr>";       // Crea
una fila per les capçeleres de la taula

                            //  Creem  una  classe  que  és  filla  de
RecursiveIteratorIterator,  una  classe  de  PHP  que  permet  recórrer
estructures d'arrays
            class TableRows extends RecursiveIteratorIterator {
                    // Sempre es crida al constructor 1 vegada abans de
començar a iterar sobre un array
                function __construct($it) {
                    parent::__construct($it, self::LEAVES_ONLY);
                        // Amb l'operador LEAVES_ONLY ens assegurem que
únicament agafa el valor i res més
                }

                // Creem una funció anomenada current. L'objectiu de la
funció és crear una cel·la i agafar el valor d'un registre SQL.
                // Aquesta funció la crida automàticament per l'objecte
RecursiveIteratorIterator (no cal cridar-la, es fa sol)
                    function current() :string { // Afegim :string per
evitar el missatge "deprecated". Indiquem que retorna un String
                        return "<td style='width:150px;border:1px solid
black;'>" . parent::current(). "</td>";
                }

                // Creem una funció anomenada beginChildren. L'objectiu
de la funció és crear una nova fila.
                // Aquesta funció la crida automàticament per l'objecte
RecursiveIteratorIterator (no cal cridar-la, es fa sol)
```

```
                // Aquesta funció és cridada cada vegada que es fa una
iteració (1 volta, 1 registre en SQL)
                function beginChildren() :void { // Afegim :void per
evitar el missatge "deprecated". Indiquem que no retorna res
                    echo "<tr>"; // Creem la fila
                }

                //Creem una funció anomenada endChildren L'objectiu de
la funció és tancar la fila creada per la funció endChildren
                // Aquesta funció la crida automàticament per l'objecte
RecursiveIteratorIterator (no cal cridar-la, es fa sol)
                // Aquesta funció és cridada cada vegada que es fa una
iteració (1 volta, 1 registre en SQL)
                    function endChildren() :void { // Afegim :void per
evitar el missatge "deprecated"
                    echo "</tr>" . "\n"; // Tanquem la fila
                }

                // Resum del Flux de l'Iteració
                // 1. Inici de l'Iteració (Per a la primera fila): Es
crida beginChildren() --> Imprimeix <tr>.
                    // 2. Iteració de la Primera Columna (id): Es crida
current() --> Imprimeix <td ...>1</td>.
                    // 3. Iteració de la Segona Columna (firstname): Es
crida current() --> Imprimeix <td ...>John</td>.
                    // 4. Iteració de la Tercera Columna (lastname): Es
crida current() --> Imprimeix <td ...>Doe</td>.
                        // 5. Fi de l'Iteració de la Fila: Es crida
endChildren() --> Imprimeix </tr>\n.
                    // 6. Repetició (Per a la següent fila), tornant al
punt 1.


            }

            $servidor = "127.0.0.1";
            $usuari = "root";
            $contrasenya = "fjeclot";
            $nomDB = "dbGserrat";
            $nomTaula = "Clients";

            try {
```

```
                    // Definim un nou objecte de la classe PDO amb els
atributs: host al que ens connectarem, el nom d'usuari i contrasenya.
                                                        $connexio    =    new
PDO("mysql:host=$servidor;dbname=$nomDB", $usuari, $contrasenya);

                // A través de la funció setAttribute, agafem el tipus
d'error en cas de que n'hi hagi algun
                                $connexio->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);

                            // Definim una preparació en la variable
$consultaPreparada.
                    $consultaPreparada = $connexio->prepare("SELECT id,
nom, mail FROM $nomTaula LIMIT 5 OFFSET 2");

                // Executem la preparació
                $consultaPreparada->execute();

                // Imprimim la taula fent servir la crida automàtica a
les funcions de la classe RecursiveArrayIterator
                                                                $resultat    =
$consultaPreparada->setFetchMode(PDO::FETCH_ASSOC);
                                                foreach(new   TableRows(new
RecursiveArrayIterator($consultaPreparada->fetchAll())) as $k=>$v) {
                                                                    //    new
RecursiveArrayIterator($consultaPreparada->fetchAll())  converteix  els
registres en iteratius
                            echo $v; // Imprimeix tota la fila HTML ja
processada per les funcions
                }
                                            }   catch(PDOException   $e)   {
// En cas de que hi hagi un error, PHP llença una PDOException, i la
variable $e agafa aquesta excepció
                    echo "No s'han progut extreure els resultats" . "<br>"
. $e->getMessage();          // Si no s'executa correctament, imprimeix
el missatge d'error de PDOException
            }
                                                    $connexio    =    null;
// Tanca la connexió amb la BBDD
            echo "</table>";
        ?>
```

```
        <h1>Codi en PHP</h1>
        <?php
        show_source("ex12Propi.php");
        ?>
    </body>
</html>
```