Financiado por la Unión Europea
NextGenerationEU
GOBIERNO DE ESPAÑA
MINISTERIO DE EDUCACIÓN, FORMACIÓN PROFESIONAL Y DEPORTES
Plan de Recuperación, Transformación y Resiliencia
Junta de Andalucía
Consejería de Desarrollo Educativo y Formación Profesional

Unit 4. Access to databases through PHP

I.E.S. Zaidín Vergeles (Granada)

PHP Databases Access

Think about a commercial website that allows the company to collect and manage information about its visitors and their purchases. What is the most appropriate data structure to store all this information? Why? Give an example.

PHP Databases Access

Search the Internet for the most commonly used database management systems for web environments nowadays and briefly describe their characteristics.

PHP Databases Access

XAMPP makes it easy to connect to databases through ***MySQL***. We just need to start this service to begin to work and, by pressing the *Admin* button, we will enter the ***phpMyAdmin*** tool, to create and modify databases.

PHP Databases Access

Use the *XAMPP* ***phpMyAdmin*** tool to:

- Design a small database.
- Insert data into it and perform a search through the corresponding buttons.
- Perform operations similar to the previous ones through the SQL button.

PHP Databases Access

Using the *XAMPP* ***phpMyAdmin*** tool, do these activities:

- Create a database called *yourstore*.
- Import the file *YourStore.sql* into it.

PHP Databases Access

PHP allows you to work with *MySQL* through one of these two extensions:

- *MySQLi* (*MySQL* improved).
- *PDO* (*PHP Data Objects*).

PHP Databases Access

Both extensions involve **object management**, although *MySQLi* presents an option based solely on functions.

PHP Databases Access

MySQLi is only oriented towards *MySQL* but *PDO* can also serve other database management systems, making compatibility easier.

PHP Databases Access

To work with a *MySQL* database, the first step is to **establish a connection with the server** that stores it.

PHP Databases Access

Driver	Connection through…
MySQLi (Procedural)	*mysqli_connect($server,$user,$password,$Database);*
MySQLi (Object-oriented)	*new mysqli($server,$user,$password,$Database);*
PDO (Object-oriented)	*new PDO("mysql:host=$server;dbname=$Database", $user, $password);*

PHP Databases Access

MySQLi (procedural) connection

```
<?php
        $server = "localhost";
        $user = "root";
        $password = "";
        $dataBase = "yourstore";

       $conn = mysqli_connect($server,$user,$password,$dataBase);
?>
```

PHP Databases Access

MySQLi (objects) connection

```
<?php
        $server = "localhost";
        $user = "root";
        $password = "";
        $dataBase = "yourstore";

       $conn = new mysqli($server,$user,$password,$dataBase);
?>
```

PHP Databases Access

PDO connection

```
<?php
        $dsn = "mysql:host=localhost;dbname=yourstore";
        $user = "root";
        $password = "";

         $connection = new PDO($dsn,$user,$password);
?>
```

PHP Databases Access

PDO connection

```
<?php
        $dsn = "mysql:host=localhost;dbname=yourstore";
        $user = "root";
        $password = "";

       try {
          $connection = new PDO($dsn,$user,$password);
          $connection->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
       }
       catch (PDOException $exception){
           echo "Connection failed", $exception->getmessage();
       }
?>
```

PHP Databases Access

DSN is the acronym for ***Data Source Name***, so it contains the name of the server and the database.

PHP Databases Access

PDO has the advantage of having a class that allows you to handle *exceptions* when accessing a database.

An *exception* can be any problem that causes the interruption of the execution of a program.

PHP Databases Access

The ***try - catch*** structure allows the script to catch exceptions if they occur and act accordingly.

```
try {
   Code that we want to execute under normal circumstances
}
catch (PDOException $exception){
    Code that will be executed if an exception occurs
}
```

PHP Databases Access

Try the above code in the following ways and see what happens:

- As it is written exactly.
- With an error in the database name.
- Typing the database name correctly but adding an error in some other connection parameter.

PHP Databases Access

To finish working with a database, the connection to the server must be closed.
This process varies depending on the extension we are using.

PHP Databases Access

Driver	Disconnection through…
MySQLi (Procedural)	*mysqli_close($connection);*
MySQLi (Object-oriented)	*$connection->close();*
PDO(Object-oriented)	*$connection=null;*

PHP Databases Access

TRY IT!

```
<?php
    $server = "localhost";
    $user = "root";
    $password = "";

    try {
        $connection =  new PDO("mysql:host=$server", $user, $password);
        $connection->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
        $sql = "CREATE DATABASE Library";
        $connection->query($sql);
        echo "Library database has been created successfully";
    }
    catch (PDOException $exception){
        echo "The connection failed.", $exception->getmessage();
    }
?>
```

PHP Databases Access

Answer these questions:

- What database has been included when opening the connection? Why?
- What is the function of the *query* method in the *PDO* class?

PHP Databases Access

Driver	Execution of a sentence through…
MySQLi (Procedural)	*mysqli_close($connection,$sentence);*
MySQLi (Object-oriented)	*$connection->query($sentence);*
PDO (Object-oriented)	*$connection->query($sentence);*

PHP Databases Access

Using *MySQL PDO* driver, write the code to performance these operations:

- Create a database called *Library*.
- Include these three tables as well as their main columns and keys.
  - *Books*
  - *Readers*
  - *Lendings*
- Add several rows to each table.

PHP Databases Access

When managing a database we can use ***prepared statements*** in which we do not specify the data but that provide us with multiple advantages, such as:

- More efficiency.
- Cleaner and easier to maintain code.
- Security, as they protect against SQL injection.

PHP Databases Access

Search the internet for the meaning of ***"SQL injection"*** and try and explain an example.

PHP Databases Access

Once we have written a ***PDO prepared statement***, we will add sentences for these other operations:

1. Associate parameters.
2. Assign values.
3. Execute the sentences with these values.

PHP Databases Access


```php
<?php
    $dsn = "mysql:host=localhost;dbname=Library";
    $user = "root";
    $password = "";

    try {
       $connection =  new PDO($dsn, $user, $password);
       $connection->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

       $sql = "INSERT INTO Books (BookId,ISBN,Title,Author,Publisher) VALUES (?,?,?,?,?);";
       $statement = $connection->prepare($sql);

       $statement->bindParam(1, $bookid);
       $statement->bindParam(2, $isbn);
       $statement->bindParam(3, $title);
       $statement->bindParam(4, $author);
       $statement->bindParam(5, $publisher);
```

PHP Databases Access


```
    $bookid = 961;
    $isbn = “978-0099435488”;
    $title = "Wilt";
    $author = "Tom Sharpe";
    $publisher = "Arrow Books, Limited";
     $statement->execute();

    $bookid = 894;
    $isbn = “978-0751552188”;
    $title = "A Place to Call Home";
    $author = "Carole Matthews";
    $publisher = "Little, Brown Book Group";
     $statement->execute();
    }

  catch (PDOException $exception){
     echo "The connection failed. ", $exception->getmessage();
    }
  $connection =  null;
?>
```

PHP Databases Access

Answer these questions:

- Did you notice any changes in the Books table?
- What are the purposes of these methods in the *PDO* class?
  - *prepare()*
  - *bindParam()*
  - *execute()*

PHP Databases Access


```
<?php
    $dsn = "mysql:host=localhost;dbname=YourStore";
    $user = "root";
    $password = "";

    try {
      $connection =  new PDO($dsn, $user, $password);
      $connection->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

      $sql = "INSERT INTO customers (CustomerId, Name, Contact, Position, City, Country, Telephone) VALUES
     (:CustomerId,:Name,:Contact,:Position,:City,:Country,:Telephone);";
      $statement = $connection->prepare($sql);

      $statement->bindParam(':CustomerId', $customerid);
      $statement->bindParam(':Name', $name);
      $statement->bindParam(':Contact', $contact);
      $statement->bindParam(':Position', $position);
      $statement->bindParam(':City', $city);
      $statement->bindParam(':Country', $country);
      $statement->bindParam(':Telephone', $telephone);
```

PHP Databases Access

TRY IT!

```
    $customerid = 'PERCA';
    $name = 'Pérez Calero';
    $contact = 'Antonio Pérez';
    $position = 'Comercial';
    $city = 'Granada';
    $country = 'España';
    $telephone = '632147852';
    $statement->execute();

    $customerid = 'RONCA';
    $name = 'Roncero Carmona';
    $contact = 'Jesús Roncero';
    $position = 'Comercial';
    $city = 'Sevilla';
    $country = 'España';
    $telephone = '615945782';
    $statement->execute();

     }

    catch (PDOException $exception){
        echo "The connection failed. ", $exception->getmessage();
     }
    $connection =  null;
?>
```

PHP Databases Access

What differences can you find between both ways of using prepared statements?

PHP Databases Access

It is also possible to avoid the use of the *bindParam()* method by passing an **associative array** with the names and the variables that contain their values to *execute()*.

PHP Databases Access

TRY IT!

```
<?php
    $dsn = "mysql:host=localhost;dbname=YourStore";
    $user = "root";
    $password = "";

    try {
       $connection =  new PDO($dsn, $user, $password);
       $connection->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

       $sql = "INSERT INTO customers (CustomerId, Name, Contact, Position, City, Country, Telephone) VALUES
        (:CustomerId,:Name,:Contact,:Position,:City,:Country,:Telephone);";
       $statement = $connection->prepare($sql);

       $customerid = 'TOSER';
       $name = 'Toledo Servant';
       $contact = 'Juan Toledo';
       $position = 'Sales';
       $city = 'Granada';
       $country = Spain;
       $telephone = '632147852';
       $statement->execute(array(':CustomerId' => $customerid, ':Name' => $name, ':Contact' => $contact, ':Position' => $position,':City' =>
       $city, ':Country' => $country, ':Telephone' => $telephone));
```

PHP Databases Access


```
    $customerid = 'RONCA';
    $name = 'Roncero Carmona';
    $contact = 'Jesús Roncero';
    $position = 'Comercial';
    $city = 'Sevilla';
    $country = 'España';
    $telephone = '615945782';
    $statement->execute();
     $statement->execute(array(':CustomerId' => $customerid, ':Name' => $name, ':Contact' =>
     $contact, ':Position' => $position,':City' => $city, ':Country' => $country, ':Telephone' =>
     $telephone));
      }

    catch (PDOException $exception){
        echo "The connection failed. ", $exception->getmessage();
      }
    $connection =  null;
?>
```

PHP Databases Access

Use a different way to insert data into each of the tables in your Library database.

Web Application Implementation

PHP Databases Access


```
<?php
    $dsn = "mysql:host=localhost;dbname=YourStore";
    $user = "root";
    $password = "";

    try {
      $connection =  new PDO($dsn, $user, $password);
      $connection->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

      $sql = "SELECT * FROM customers WHERE City = ? ORDER BY Name";
      $statement = $connection->prepare($sql);

      $statement->bindParam(1, $city);
      $city = 'Londres';
      $statement->execute();

      echo "<table><caption>Customers in “, $city, “</caption>";
      echo  "<tr>  <th>ID</th>  <th>NAME</th>  <th>CONTACT</th>  <th>POSITION</th>  <th>TELEPHONE</th>
      </tr>";
```

PHP Databases Access

TRY IT!

```
        while ($tuple = $statement->fetch()){
            echo "<tr>";
            echo "<td>",$tuple['CustomerId'] , "</td>";
            echo "<td>",$tuple['Name'] , "</td>";
            echo "<td>",$tuple['Contact'] , "</td>";
            echo "<td>",$tuple['Position'] , "</td>";
            echo "<td>",$tuple['Telephone'] , "</td>";
            echo "</tr>";
        }
        echo "</table>";
    }

    catch (PDOException $exception){
        echo "The connection failed. ", $exception->getmessage();
    }
    $connection =  null;
?>
```

PHP Databases Access

Answer these questions:

- What is the purpose of the ***fetch()*** method?
- What is the meaning of the condition in the *while* sentence?
- Why is it convenient to use a prepared sentence in this code?

PHP Databases Access

Write a script to deliver a list of books published by *Arrow Books, Limited* in your Library database.

PHP Databases Access

As you know, the tables in a database are connected through their primary and foreign keys, which allows you to obtain more data in queries.

How do we establish relationships between tables in a query?

PHP Databases Access

Write a script to display the books borrowed by the reader whose DNI is *78945632D*.

PHP Databases Access

What happens if we add these texts at the end of the *SELECT* statement in the previous code?

- *LIMIT 4*
- *LIMIT 5 OFFSET 1*
- *LIMIT 3, 2*

PHP Databases Access

Add these two lines before the *while* statement in the previous code and tell what the *fetch()* method delivers.

```
$tuple = $statement->fetch();
var_dump($tuple);
```

PHP Databases Access

What happens if you modify the *fetch()* method in these ways?

- *$tuple = $statement->fetch(PDO::FETCH_ASSOC);*
- *$tuple = $statement->fetch(PDO::FETCH_NUM);*
- *$tuple = $statement->fetch(PDO::FETCH_BOTH);*

Which of them works like *fetch()*?

PHP Databases Access


```
<?php
   class Customer{
        public $CustomerId;
        public $Name;
        public $Contact;
        public $Position;
        public $City;
        public $Country;
        public $Telephone;
    }


    $dsn = "mysql:host=localhost;dbname=YourStore";
    $user = "root";
    $password = "";
```

PHP Databases Access


```
try {
        $connection =  new PDO($dsn, $user, $password);
        $connection->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

        $sql = "SELECT * FROM customers";
        $statement = $connection->query($sql);
        $customer = $statement->fetchObject('Customer');
        var_dump($customer);
  }

  catch (PDOException $exception){
     echo "The connection failed. ", $exception->getmessage();
   }
  $connection =  null;
?>
```

PHP Databases Access

Answer these questions:

- What parameter do we pass to the ***fetchObject()*** method?
- What does this method return?

PHP Databases Access

Insert in the try clause of the previous script, the necessary code to generate a list with the name and telephone number of the clients in Berlín, ordered by name.

PHP Databases Access


```
<?php

    $dsn = "mysql:host=localhost;dbname=YourStore";
    $user = "root";
    $password = "";

    try {
        $connection =  new PDO($dsn, $user, $password);
        $connection->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

        $sql = "SELECT * FROM customers ORDER BY Name";
        $statement = $connection->query($sql);
        echo "<table><caption>Customers</caption>";
        echo "<tr> <th>ID</th> <th>NAME</th> <th>CONTACT</th> <th>POSITION</th>
        <th>TELEPHONE</th> </tr>";
        $customers = $statement->fetchAll(PDO::FETCH_ASSOC);
```

PHP Databases Access


```
        foreach ($customers as $customer){
            echo "<tr>";
            echo "<td>",$customer['CustomerId'] , "</td>";
            echo "<td>",$customer['Name'] , "</td>";
            echo "<td>",$customer['Contact'] , "</td>";
            echo "<td>",$customer['Position'] , "</td>";
            echo "<td>",$customer['Telephone'] , "</td>";
            echo "</tr>";
        }
        echo "</table>";
    }

    catch (PDOException $exception){
        echo "The connection failed. ", $exception->getmessage();
    }
    $connection =  null;
?>
```

PHP Databases Access

Answer these questions:

- What is the purpose of the ***fetchAll()*** method?
- What is the meaning of the ***PDO::FETCH_ASSOC*** parameter?
- What control statement do we use to access the data?

PHP Databases Access

Write a script that, using the ***fetchAll()*** method, generates a list with the title and author of each of the books in your *Library* database.

PHP Databases Access

TRY IT!

```
<?php

    $dsn = "mysql:host=localhost;dbname=YourStore";
    $user = "root";
    $password = "";

   try {
         $connection =  new PDO($dsn, $user, $password);
          $connection->setAttribute(PDO::ATTR_ERRMODE,
          PDO::ERRMODE_EXCEPTION);

        $sql = "SELECT * FROM customers ORDER BY Name";
        $statement = $connection->query($sql);
        echo "<ul>";
```

PHP Databases Access

TRY IT!

```
    foreach ($statement->fetchAll(PDO::FETCH_COLUMN,2) as $name){
          echo "<li>", $name, “</li>”;
        }
        echo "</ul>";
    }

  catch (PDOException $exception){
     echo "The connection failed. ", $exception->getmessage();
   }
  $connection =  null;
?>
```

PHP Databases Access

Answer these questions:

- What do we get when adding the ***PDO::FETCH_COLUMN*** parameter to the ***fetchAll()*** method?
- What is the second parameter that we need to write in this case?

PHP Databases Access

Using the previous method, write a script that generates a list of the names of the items in the *YourStore* database.

PHP Databases Access

Answer these questions:

- Could you write the above two scripts using the ***fetch()*** method, instead of ***fetchAll()***?
- When should each be used?

PHP Databases Access


```php
<?php
    $dsn = "mysql:host=localhost;dbname=Library";
    $user = "root";
    $password = "";

    try {
        $connection =  new PDO($dsn, $user, $password);
        $connection->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

        $sql = "UPDATE readers SET Telephone=? WHERE ReaderId=?;";
        $statement = $connection->prepare($sql);

        $statement->bindParam(1, $tel);
        $statement->bindParam(2, $id);
```

PHP Databases Access

TRY IT!

```
        $tel = 655998877;
        $id = 4;
        $statement->execute();
        echo "The reader with id ", $id, " has this new telephone number ", $tel, ".";

    }

    catch (PDOException $exception){
        echo "The connection failed. ", $exception->getmessage();
    }
    $connection =  null;
?>
```

PHP Databases Access

Write a script for each of these updates:

- Change the address of the reader whose id is 5 to Spring, 12 - 6º D.
- Assign the current date to the return date of the lending number 4. (Use the ***date()*** function)
- Write the phone number 654987123 to the reader with DNI 23456789D.

PHP Databases Access


```php
<?php
    $dsn = "mysql:host=localhost;dbname=Library";
    $user = "root";
    $password = "";

    try {
        $connection =  new PDO($dsn, $user, $password);
        $connection->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

        $sql = "DELETE FROM readers WHERE ReaderId=?;";
        $statement = $connection->prepare($sql);

        $statement->bindParam(1, $id);
```

PHP Databases Access

TRY IT!

```
        $id = 4;
        $statement->execute();
         echo "The reader with id ", $id, " has been deleted.";

    }

   catch (PDOException $exception){
      echo "The connection failed. ", $exception->getmessage();
    }
   $connection =  null;
?>
```

PHP Databases Access

Write scripts to delete:

- All the books written by *Tom Sharp*.
- All the lendings to the reader with id *100*.