

Web Application Development

Interacting with a Database

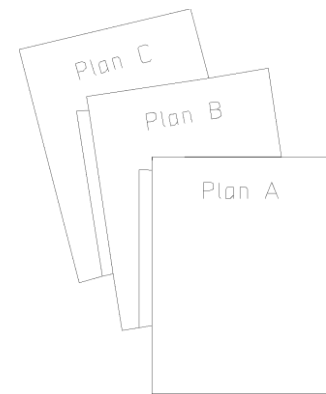


If people are not making mistakes, they are not trying new things.

- Walter C Wright Jr

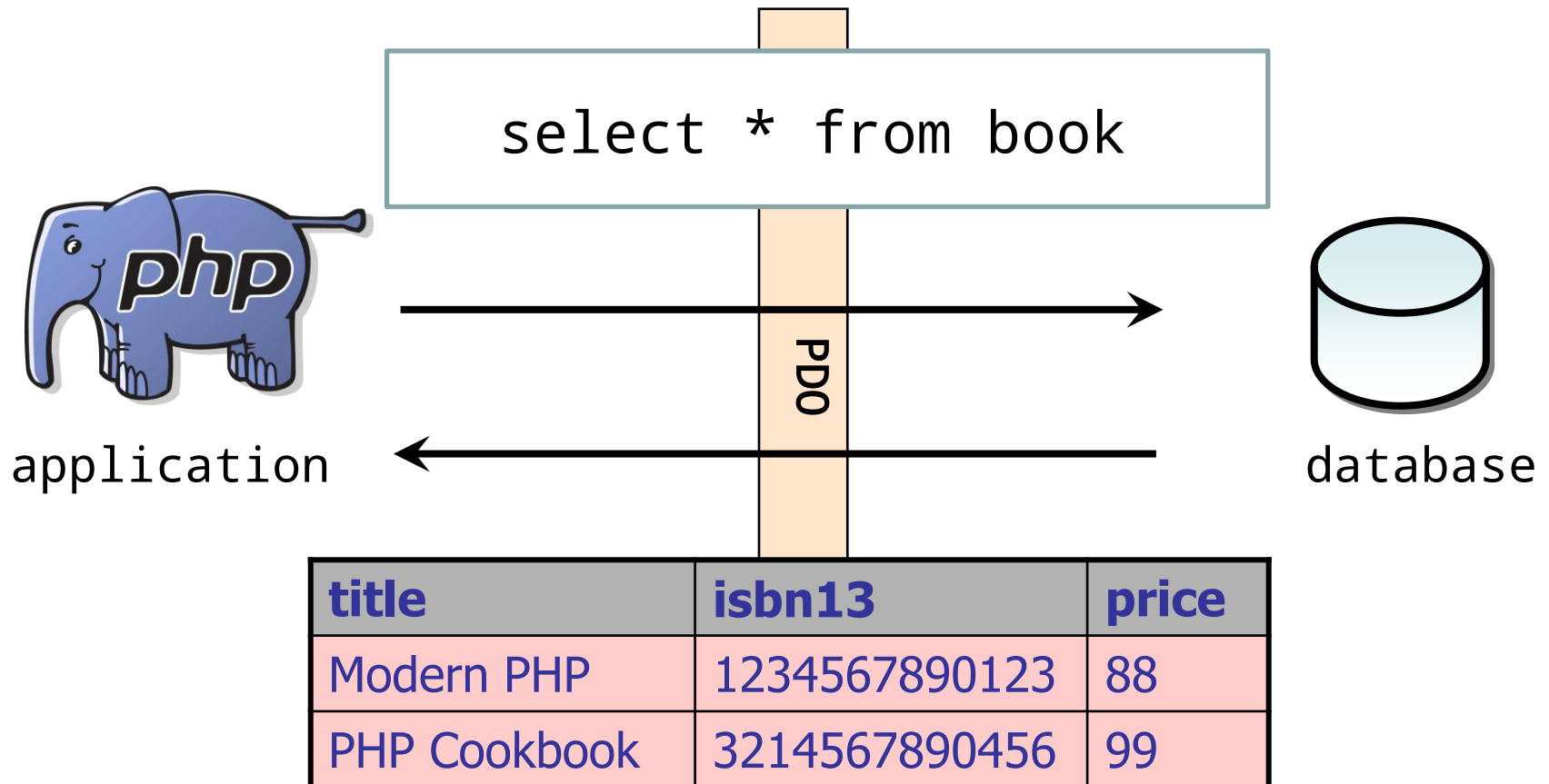
Overview

- Objective
 - Learn how to store and manipulate data in MySQL using PHP Data Objects (PDO)
- Content
 - What is PDO?
 - Using PDO to perform Create-Read-Update-Delete (CRUD) operations
- After this module, you should be able to
 - Use PDO to perform CRUD operations



Use of PDO

- It allows a PHP code to send SQL queries to the database and receive the results



Step 1: Establishing the connection

- Data source name (DSN)
 - Database type
 - Host
 - Database name
 - Port
- Create a PDO object
 - Specify DSN, username, and password

```
$dsn = "mysql:host=localhost;dbname=week6;port=3306";  
$pdo = new PDO($dsn, "root", "");
```

Step 2: Prepare the statement object

- A PDOStatement object allows you to communicate with a database

```
$isbn13 = '1234567890123';  
$sql = 'select * from book where isbn13 = :isbn13';  
  
$stmt = $pdo->prepare($sql);  
$stmt->setFetchMode(PDO::FETCH_ASSOC);  
$stmt->bindParam(':isbn13', $isbn13, PDO::PARAM_STR);  
$stmt->execute();  
  
if($row = $stmt->fetch()) {  
    echo $row['title'] . '|' . $row['isbn13']  
        . '|' . $row['price'];  
}
```

Step 3: Specify return data format

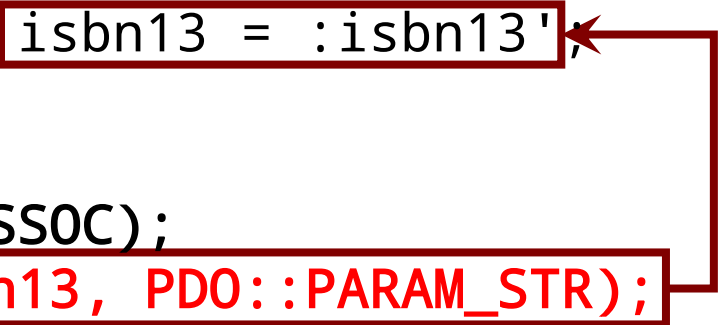
- PDO::FETCH_ASSOC:
 - Each matching DB record is retrieved as an associate array with column names as keys

```
$isbn13 = '1234567890123';  
$sql = 'select * from book where isbn13=:isbn13';  
  
$stmt = $pdo->prepare($sql);  
$stmt->setFetchMode(PDO::FETCH_ASSOC);  
$stmt->bindParam(':isbn13', $isbn13, PDO::PARAM_STR);  
$stmt->execute();  
  
if($row = $stmt->fetch()) {  
    echo $row['title'] . '|' . $row['isbn13']  
        . '|' . $row['price'];  
}
```

Step 4: Substitute values for the placeholders

- Each :XX specified in the SQL string is a value that you have to fill in using the bindParam method

```
$isbn13 = '1234567890123';  
$sql = 'select * from book where isbn13 = :isbn13';  
  
$stmt = $pdo->prepare($sql);  
$stmt->setFetchMode(PDO::FETCH_ASSOC);  
$stmt->bindParam(':isbn13', $isbn13, PDO::PARAM_STR);  
$stmt->execute();  
  
if($row = $stmt->fetch()) {  
    echo $row['title'] . '|' . $row['isbn13']  
        . '|' . $row['price'];  
}
```

A red box highlights the placeholder ':isbn13' in the SQL string. A red arrow points from this box to another red box that highlights the bindParam method call '\$stmt->bindParam(':isbn13', \$isbn13, PDO::PARAM_STR);' in the code below.

PDO class constants

Constants	Description
PDO::PARAM_BOOL	Represents a boolean data type
PDO::PARAM_NULL	Represents the SQL NULL data type
PDO::PARAM_INT	Represents the SQL INTEGER data type
PDO::PARAM_STR	Represents the SQL CHAR, VARCHAR, or other string data type.
PDO::PARAM_LOB	Represents the SQL large object data type

Step 5: Send the query to the server

```
$isbn13 = '1234567890123';  
$sql = 'select * from book where isbn13 = :isbn13';  
  
$stmt = $pdo->prepare($sql);  
$stmt->setFetchMode(PDO::FETCH_ASSOC);  
$stmt->bindParam(':isbn13', $isbn13, PDO::PARAM_STR);  
$stmt->execute();  
  
if($row = $stmt->fetch()) {  
    echo $row['title'] . '|' . $row['isbn13']  
        . '|' . $row['price'];  
}
```

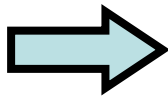
Step 5: Process the data

- Fetch each matching row as an associative array
- Retrieve the value using the column name

```
$stmt->execute();
```

```
if($row = $stmt->fetch()) {  
    echo $row['title'] . '|' . $row['isbn13'] .  
        '|' . $row['price'];  
}
```

```
$stmt.fetch()
```



Key	Value
title	Modern PHP
isbn13	1234567890123
price	88

Step 4: Free up resources

- Close the connection to the database.

```
$stmt->closeCursor();  
$pdo = null;
```

Issuing Insert/Delete/Update queries

```
$sql = 'insert into book (title, isbn13, price)
      values (:title, :isbn13, :price)';
```

```
$title = "Java";
$isbn13 = "999999999999999";
$price = 105;
```

```
$stmt = $pdo->prepare($sql);
$stmt->bindParam(':title', $title, PDO::PARAM_STR);
$stmt->bindParam(':isbn13', $isbn13, PDO::PARAM_STR);
$stmt->bindParam(':price', $price, PDO::PARAM_INT);
```

```
// $isAddOK stores true or false
$isAddOK = $stmt->execute();
```

```
$stmt->closeCursor();
$pdo = null;
```

Exercise 1: First try

- Given:
 - setup.sql

- To do:
 - Import setup.sql using phpMyAdmin. It will import all tables needed for this exercise and the following ones.
 - Put code given in slides 4, 5, 11 into display.php. Delete any pptx special symbols.

Exercise 2: Display table contents

- Given:
 - display.php (incomplete)
- To do:
 - Modify display.php such that it displays the data stored in the person table of the week6 database

Name	Gender	Age
Amy	F	28
Bill	M	18
Charles	M	17
Doraemon	F	32

Exercise 3: Insert new content

- Given:
 - add-view.html
 - add.php (incomplete)

- Implement add.php such that
 - If add is successful, print 'Person added'
 - if add is unsuccessful, print 'Person is not added'

PHP INItalization File

- Used to configure the parameters and initial settings
- Never expose sensitive configuration files in public
 - Never do this: <https://example.com/database.ini>
 - Place it outside the publicly accessible document root on the server

PHP INIInitialization File

- Format

```
key1 = value1
```

- Example

```
; lines that starts with ; are comments  
; file name is usually xx.ini e.g. database.ini  
host = localhost  
username = root  
password =  
dbname = week6
```

PHP INIInitialization File

- Reading in PHP code
 - Read the ini file and store it in an associative array
 - Get the individual value out

```
$conf = parse_ini_file($_SERVER["DOCUMENT_ROOT"] .  
                        "../private/database.ini");  
$dsn = "mysql:host={$conf['host']};".  
        dbname={$conf['dbname']}" ;  
$pdo = new PDO($dsn,$conf['username'],$conf['password']);
```

ConnectionManager class

- Don't Repeat Yourself (DRY) Rule

-- ConnectionManager.php --

```
<?php
class ConnectionManager {

    public function getConnection() {
        // store the ini file outside of www folder
        $conf = parse_ini_file($_SERVER["DOCUMENT_ROOT"] .
                                "../private/database.ini");
        $dsn="mysql:host={$conf['host']};" .
            "dbname={$conf['dbname']}";
        return new PDO($dsn,$conf['username'],$conf['password']);
    }
}
```

Using a Data Access Object (DAO) class

- Avoid writing PDO code in multiple php pages
- A DAO class captures reusable PDO code
 - Grouped into logical unit
 - E.g., BookDAO, PersonDAO, EmployeeDAO, etc.

-- BookDAO.php --

```
<?php
class BookDAO {
    public function retrieveAll() {
        $sql = 'SELECT * FROM book';
        $connMgr = new ConnectionManager();
        $pdo = $connMgr->getConnection();
        $stmt = $pdo->prepare($sql);
        $stmt->setFetchMode(PDO::FETCH_ASSOC);
        $stmt->execute();
        $result = array();
        while($row = $stmt->fetch()) {
            $result[] = new Book($row['title'], $row['isbn13'],
                                $row['price']);
        }
        $stmt->closeCursor();
        $pdo = null;
        return $result;
    }
}
?>
```

Exercise 4: DAO

- Given:
 - ConnectionManager.php
 - PersonDAO.php (incomplete)
 - display.php (incomplete)
 - add.php (incomplete)
 - add-view.html, Person.php, autoload.php
 - database.ini
- To do:
 - Put database.ini at a safe location (e.g., wamp64/private/database.ini)
 - Modify the argument of parse_ini_file (if needed)
 - Put other files in a suitable folder under www

Exercise 4: DAO

- To do (con't):
 - Implement **retrieveAll()** and **add(\$person)** in PersonDAO.php
 - Use PersonDAO to implement **display.php**
 - Produce the same output as Exercise 1
 - Use PersonDAO to implement **add.php**
 - Produce the same output as Exercise 2

Exercise 5: DAO (More Practice)

- Reusing database.ini from exercise 4 + the following files in ex5 folder:

- ConnectionManager.php
- CourseDAO.php
- Course.php
- display.php (incomplete)
- add.php (incomplete)
- add-view.php
- autoload.php

- Complete the ones marked as incomplete

Key Points

- PHP-DB interaction
 - Connect to DB
 - Specify DSN
 - Instantiate PDO object
 - Prepare statement
 - prepare, bindParam, setFetchMode
 - Execute statement
 - execute, fetch
 - Free up resources
 - closeCursor, \$pdo = null

- Initialization file
 - .ini file
 - `key1 = value1`
 - *parse_ini_file(...)*
- ConnectionManager
- Data Access Object (DAO)

