PHP Intermediate Experience

By - Aung Khant Kyaw

OOP

- Object-Oriented Programming (OOP) is for software design that revolves around objects and classes instead of logic and functions.

- Focuses on what the developer wants to manipulate rather than how to manipulate

Benefits

- Code reusability
- Scalability(handle increased workloads, data, or user demands while maintaining performance)
- Efficiency

Concerns

- Overemphasizing the data
- Ignoring computation and algorithm components
- Complexity of writing OOP code

Inheritance – uses extends keyword to allow subclass to inherit variables and functions

Encapsulation – uses private/protected to restrict direct access

Abstraction – provides an empty body function

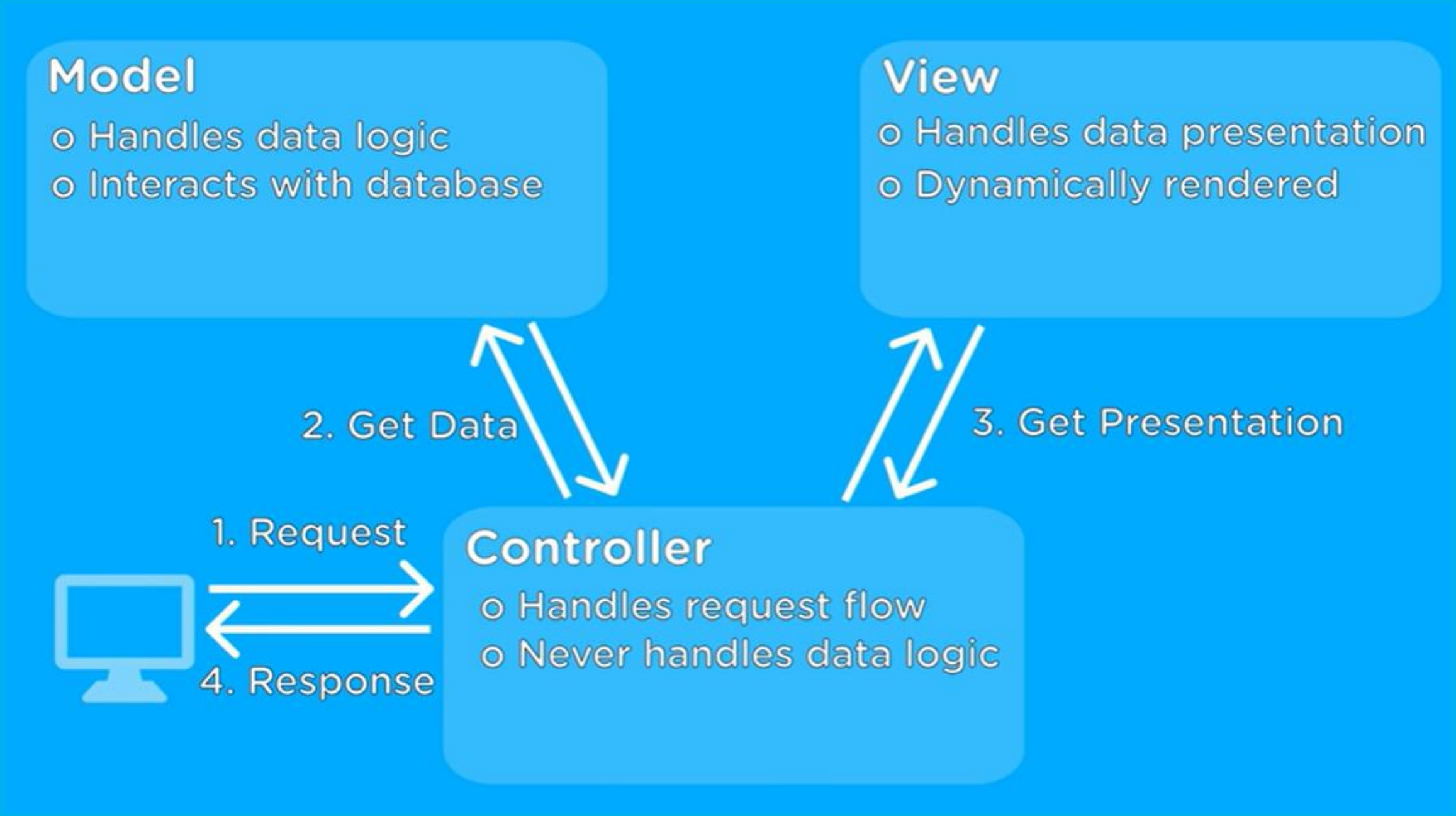
Polymorphism – allows rewriting of function from parent

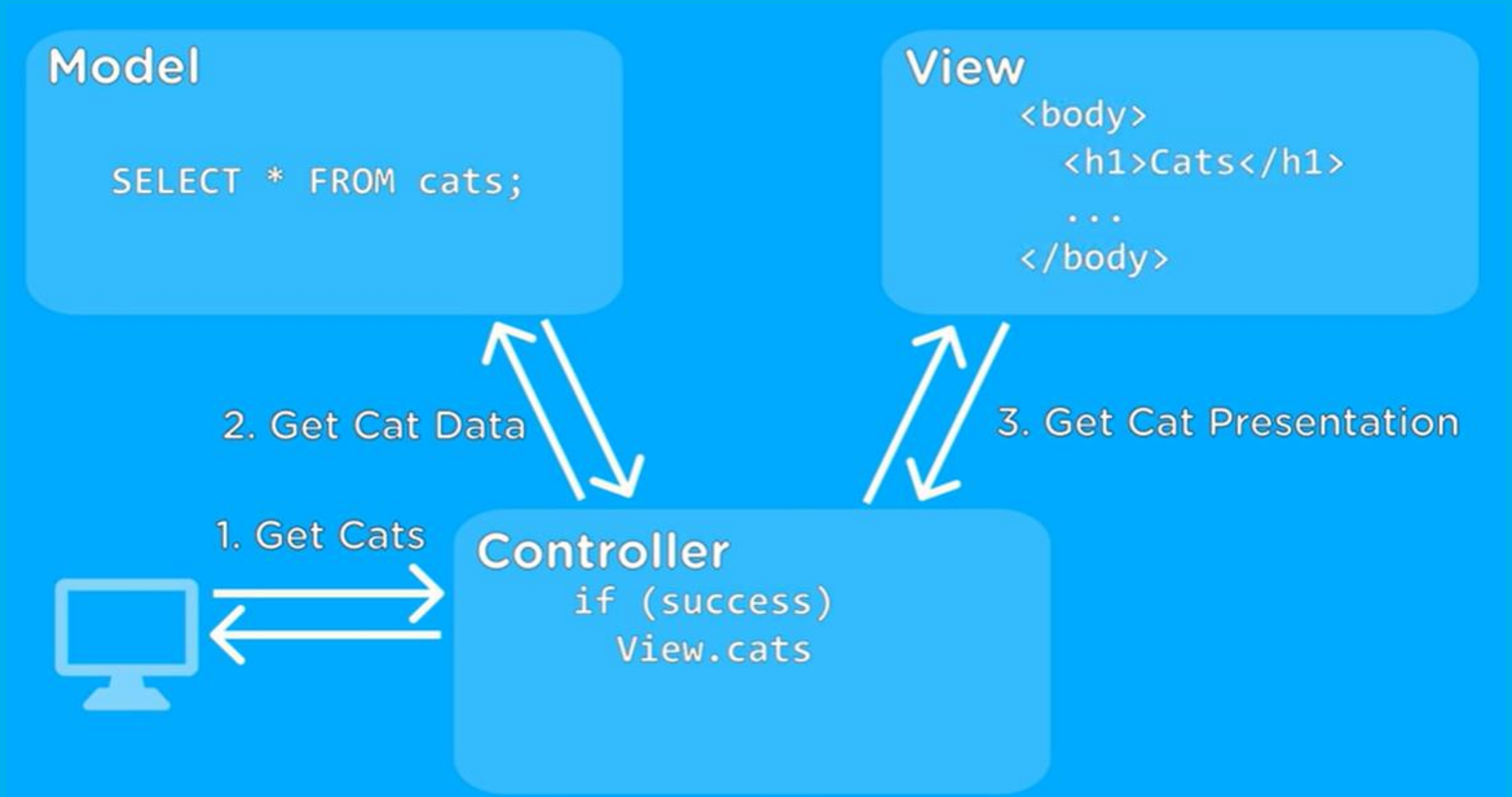
```php
trait Logging {
    public function log($message) {
        echo "Log: $message\n";
    }
}

trait Emailing {
    public function sendEmail($to, $message) {
        echo "Email sent to $to: $message\n";
    }
}

class User {
    use Logging, Emailing;

    public function register() {
        $this->log("User registered.");
        $this->sendEmail("user@example.com", "Welcome to our website!");
    }
}

$user = new User();
$user->register();
```

Namespace – prevent naming collisions
Must be at the beginning of php file

Autoloading – load required files automatically without repeatedly using include() or require().

-composer dump-autoload -o


Model
o Handles data logic
o Interacts with database
View
o Handles data presentation
o Dynamically rendered
2. Get Data
3. Get Presentation
1. Request
4. Response
Controller
o Handles request flow
o Never handles data logic


Model
SELECT * FROM cats;
View
<body>
<h1>Cats</h1>
...
</body>
2. Get Cat Data
3. Get Cat Presentation
1. Get Cats
Controller
if (success)
View.cats

SQL Injection

-inject malicious SQL code into a system's SQL query.

-can lead to unauthorized access to a database, data theft

- To prevent SQL Injection---
- Instead of directly embedding user input in SQL queries, use parameterized queries or prepared statements.

```
$stmt = $pdo->prepare("SELECT * FROM users WHERE username = :username");
$stmt->bindParam(':username', $username, PDO::PARAM_STR);
$stmt->execute();
```

-Inject malicious scripts into user browser.

-Steals user cookies/private info stored in the browser.

-Mostly performed using javascript And HTML.

- To prevent XSS ---
- Always escape user-generated input

```php
$userInput = '<script>alert("XSS attack")</script>';
echo htmlspecialchars($userInput, ENT_QUOTES, 'UTF-8');
```

```php
$userInput = 'alert("XSS attack")';
echo 'var data = ' . json_encode($userInput) . ';';
```

```php
header("X-Content-Type-Options: nosniff");
header("X-Frame-Options: DENY");
```

What are Cross-site request forgery (CSRF) attacks?

-One click attack or session riding

-can lead to actions such as—

1. Changing settings
2. Making purchases
3. Money transfers
4. Altering data

- To prevent CSRF---
- 1. Use anti-CSRF tokens in your forms.
- 2. Set ‘Samesite’ attribute on cookies

```php
<?php
session_start();
$token = bin2hex(random_bytes(32)); // Generate a random token
$_SESSION['csrf_token'] = $token;
?>
```

```php
setcookie('my_cookie', 'value', ['samesite' => 'Strict']);
```

Thank u for listening