MẢNG VÀ CHUỖI TRONG PHP

Nội dung bài giảng

1. Mảng trong PHP

Các loại mảng, cách tạo và thao tác với mảng

2. Chuỗi trong PHP

Khai báo và các hàm xử lý chuỗi thông dụng

3. Thực hành

Demo và ví dụ thực tế về mảng và chuỗi

4. Bài tập

Các bài tập củng cố kiến thức

1. Mảng (Array) trong PHP

Mảng là một cấu trúc dữ liệu đặc biệt cho phép lưu trữ nhiều giá trị trong một biến duy nhất. PHP hỗ trợ ba loại mảng chính:

Mảng chỉ số (Indexed)

Sử dụng số nguyên làm chỉ số để truy cập các phần tử, thường bắt đầu từ 0.

Mảng kết hợp (Associative)

Sử dụng các khóa có tên để truy cập vào các giá trị tương ứng thay vì chỉ số số.

Mảng đa chiều (Multidimensional)

Mảng chứa một hoặc nhiều mảng con, cho phép lưu trữ dữ liệu phức tạp hơn.

Trong PHP, mảng được sử dụng rộng rãi để lưu trữ, quản lý và xử lý dữ liệu hiệu quả, đặc biệt khi làm việc với lượng dữ liệu lớn hoặc phức tạp.

2.2 Mảng chỉ số (Indexed Array)

Mảng chỉ số sử dụng số nguyên làm chỉ số, bắt đầu từ 0.

```
// Cách khai báo mảng chỉ số
$fruits = array("Táo", "Chuối", "Cam");

// Cú pháp ngắn gọn (PHP 5.4+)
$fruits = ["Táo", "Chuối", "Cam"];

// Truy cập phần tử mảng
echo $fruits[0]; // Hiển thị: Táo
echo $fruits[1]; // Hiển thị: Chuối
```

Mảng chỉ số giống như một danh sách các phần tử được đánh số theo thứ tự, bắt đầu từ 0 cho phần tử đầu tiên.

Khi làm việc với mảng chỉ số, ta có thể dễ dàng duyệt qua các phần tử bằng vòng lặp, đặc biệt là vòng lặp for hoặc foreach. Điều này rất hữu ích khi xử lý các danh sách dữ liệu có cùng kiểu.

Mảng kết hợp (Associative Array)

Mảng kết hợp sử dụng các khóa do người dùng đặt tên để xác định và truy cập các phần tử.

1. Khái niệm chính

Mảng kết hợp cho phép bạn sử dụng các chuỗi có ý nghĩa làm khóa thay vì các số, giúp mã nguồn dễ đọc và dễ hiểu hơn.

2. Cú pháp

Có thể khai báo bằng từ khóa array() hoặc sử dụng cú pháp ngắn gọn với dấu ngoặc vuông [] (PHP 5.4+).

3. Truy cập dữ liệu

Truy cập phần tử bằng cách sử dụng khóa trong cặp dấu ngoặc vuông, ví dụ: $student["name"].

```
// Khai báo mảng kết hợp
$student = array(
    "name" => "Nguyễn Văn A",
    "age" => 20,
    "major" => "Công nghệ thông tin"
);

// Cú pháp ngắn gọn
$student = [
    "name" => "Nguyễn Văn A",
    "age" => 20,
    "major" => "Công nghệ thông tin"
];

// Truy cập phần tử
echo $student["name"]; // Nguyễn Văn A
```

Lưu ý: Mảng kết hợp rất hữu ích khi làm việc với dữ liệu có cấu trúc như thông tin người dùng, cấu hình hay các tham số.

3. Mảng đa chiều (Multidimensional Array)

Mảng đa chiều là mảng chứa một hoặc nhiều mảng bên trong nó.

3.1. Mảng hai chiều

```
// Mảng hai chiều
$students = [
 ["Nguyễn Văn A", "CNTT", 8.5],
 ["Trần Thị B", "Kế toán", 9.0],
 ["Lê Văn C", "Marketing", 7.5]
];

// Truy cập phần tử
echo $students[0][0]; // Nguyễn Văn A
echo $students[1][2]; // 9.0
```

3.2. Mảng kết hợp đa chiều

```
// Mảng kết hợp đa chiều
$classes = [
 "CNTT" => [
 "students" => 40,
 "teacher" => "Trần Văn D"
 ],
 "Marketing" => [
 "students" => 35,
 "teacher" => "Lê Thị E"
 ]
];

echo $classes["CNTT"]["teacher"]; // Trần Văn D
```

Duyệt qua các phần tử của mảng

1. Sử dụng vòng lặp for

```
$fruits = ["Táo", "Chuối", "Cam", "Xoài"];
$length = count($fruits);

for($i = 0; $i < $length; $i++) {
    echo $fruits[$i] . "
";
}
```

2. Sử dụng vòng lặp foreach

```
// Cho mảng chỉ số
foreach($fruits as $fruit) {
    echo $fruit . "
";
}

// Cho mảng kết hợp
$student = ["name" => "An", "age" => 20];
foreach($student as $key => $value) {
    echo "$key: $value
";
}
```

Các hàm thao tác với mảng

1. count()

Đếm số phần tử trong mảng. Hàm này rất hữu ích khi cần biết kích thước của mảng trước khi xử lý.

```
$fruits = ["Táo", "Chuối", "Cam"];
$length = count($fruits); // Kết quả: 3

// Sử dụng với mảng đa chiều
$multiArray = [
  [1, 2],
  [3, 4, 5]
];
$count = count($multiArray); // Kết quả: 2
```

2. array_push()

Thêm một hoặc nhiều phần tử vào cuối mảng. Hàm này sẽ trả về số lượng phần tử của mảng sau khi thêm.

```
$fruits = ["Táo", "Chuối"];
array_push($fruits, "Cam", "Xoài");
// Kết quả: $fruits = ["Táo", "Chuối", "Cam", "Xoài"]

// Cách thay thế
$fruits[] = "Dứa"; // Thêm 1 phần tử
```

3. array_pop()

Xóa và trả về phần tử cuối cùng của mảng. Hàm này sẽ làm thay đổi mảng gốc bằng cách loại bỏ phần tử cuối.

```
$fruits = ["Táo", "Chuối", "Cam"];
$lastFruit = array_pop($fruits);
// $lastFruit = "Cam"
// $fruits = ["Táo", "Chuối"]

// Có thể sử dụng trong vòng lặp để rút ngắn mảng
while(count($fruits) > 0) {
    $item = array_pop($fruits);
    // xử lý $item
}
```

4. array_merge()

Kết hợp hai hoặc nhiều mảng thành một mảng mới. Nếu các mảng có cùng key, giá trị của mảng sau sẽ ghi đè lên giá trị của mảng trước.

```
$fruits1 = ["Táo", "Chuối"];
$fruits2 = ["Cam", "Xoài"];
$allFruits = array_merge($fruits1, $fruits2);
// $allFruits = ["Táo", "Chuối", "Cam", "Xoài"]

// Với mảng kết hợp
$info1 = ["name" => "An", "age" => 20];
$info2 = ["age" => 21, "class" => "CNTT"];
$result = array_merge($info1, $info2);
// $result = ["name" => "An", "age" => 21, "class" => "CNTT"]
```

Sắp xếp mảng trong PHP

PHP cung cấp nhiều hàm sắp xếp mảng cho các trường hợp khác nhau. Mỗi hàm sắp xếp đều trả về TRUE khi thành công và FALSE khi thất bại.

Các hàm sắp xếp phổ biến:

1. sort()

Sắp xếp mảng theo thứ tự tăng dần. Chỉ số sẽ được đánh lại từ 0.

```
$fruits = ["Cam", "Ổi", "Táo", "Bưởi"];
sort($fruits);
// Kết quả: $fruits = ["Bưởi", "Cam", "Ổi", "Táo"]
```

2. rsort()

Sắp xếp mảng theo thứ tự giảm dần. Chỉ số sẽ được đánh lại từ 0.

```
$numbers = [5, 3, 8, 1, 9];
rsort($numbers);
// Kết quả: $numbers = [9, 8, 5, 3, 1]
```

3. asort()

Sắp xếp mảng kết hợp theo giá trị tăng dần, giữ nguyên khóa.

```
$ages = ["An" => 20, "Bình" => 18, "Cường" => 25];
asort($ages);
// Kết quả: $ages = ["Bình" => 18, "An" => 20, "Cường" => 25]
```

4. ksort()

Sắp xếp mảng kết hợp theo khóa tăng dần, giữ nguyên mối quan hệ khóa-giá trị.

```
$scores = ["PHP" => 85, "HTML" => 90, "CSS" => 78];
ksort($scores);
// Kết quả: $scores = ["CSS" => 78, "HTML" => 90, "PHP" => 85]
```

5. arsort()

Sắp xếp mảng kết hợp theo giá trị giảm dần, giữ nguyên khóa.

```
$scores = ["PHP" => 85, "HTML" => 90, "CSS" => 78];
arsort($scores);
// Kết quả: $scores = ["HTML" => 90, "PHP" => 85, "CSS" => 78]
```

6. krsort()

Sắp xếp mảng kết hợp theo khóa giảm dần, giữ nguyên mối quan hệ khóa-giá trị.

```
$products = ["SP001" => "Máy tính", "SP010" => "Điện thoại", "SP005" => "Máy in"];
krsort($products);
// Kết quả: $products = ["SP010" => "Điện thoại", "SP005" => "Máy in", "SP001" => "Máy tính"]
```

Lưu ý: Các hàm sắp xếp trong PHP sẽ làm thay đổi trực tiếp mảng gốc. Nếu muốn giữ nguyên mảng gốc, cần tạo một bản sao trước khi sắp xếp.

Ví dụ thực tế về mảng

Quản lý danh sách sinh viên:

```
$students = [
    [
        "id" => "SV001",
        "name" => "Nguyễn Văn An",
        "scores" => [8, 7.5, 9]
    ],
    [
        "id" => "SV002",
        "name" => "Trần Thị Bình",
        "scores" => [9, 8.5, 8]
    ],
    [
        "id" => "SV003",
        "name" => "Lê Văn Cường",
        "scores" => [7, 7, 8]
    ]
];

// Tính điểm trung bình của từng sinh viên
foreach($students as $student) {
    $avg = array_sum($student["scores"]) /
          count($student["scores"]);
    echo $student["name"] . ": " .
         number_format($avg, 1) . "\n";
}
```

Giải thích chi tiết mã nguồn:

1. Tạo mảng đa chiều lưu trữ thông tin sinh viên:

- Mảng `$students` là mảng chính (mảng ngoài) chứa các mảng con
- Mỗi mảng con là một mảng kết hợp với các khóa: "id", "name" và "scores"
- Khóa "scores" lại chứa một mảng chỉ số lưu điểm số của sinh viên

2. Duyệt qua mảng sinh viên với vòng lặp foreach:

- `foreach($students as $student)`: Duyệt qua từng phần tử (sinh viên) trong mảng `$students`
- Mỗi lần lặp, biến `$student` sẽ chứa thông tin của một sinh viên

3. Tính điểm trung bình:

- `array_sum($student["scores"])`: Tính tổng các điểm số của sinh viên
- `count($student["scores"])`: Đếm số lượng điểm số
- Chia tổng điểm cho số lượng điểm để tính điểm trung bình
- `number_format($avg, 1)`: Định dạng điểm trung bình với 1 chữ số thập phân

4. Hiển thị kết quả:

- Nối tên sinh viên với điểm trung bình bằng toán tử nối chuỗi (.)
- Sử dụng `echo` để hiển thị kết quả
- \n: Ký tự xuống dòng giúp mỗi sinh viên hiển thị trên một dòng

Các thao tác mở rộng với mảng sinh viên:

1. Truy xuất thông tin của một sinh viên cụ thể:

```
// Lấy thông tin sinh viên đầu tiên
$firstStudent = $students[0];
echo "Sinh viên: " . $firstStudent["name"] . ", Mã số: " . $firstStudent["id"];

// Lấy điểm số thứ 2 của sinh viên thứ 3
$thirdStudentSecondScore = $students[2]["scores"][1];
echo "Điểm số thứ 2 của " . $students[2]["name"] . ": " . $thirdStudentSecondScore;
```

2. Thêm thông tin mới cho sinh viên:

```
// Thêm thông tin lớp cho sinh viên đầu tiên
$students[0]["class"] = "PHP123";

// Thêm một điểm số mới cho sinh viên thứ hai
$students[1]["scores"][] = 8.5;
```

3. Sắp xếp danh sách sinh viên theo điểm trung bình:

```
// Tính điểm trung bình và lưu vào mảng
foreach($students as &$student) {
    $student["average"] = array_sum($student["scores"]) / count($student["scores"]);
}

// Sắp xếp giảm dần theo điểm trung bình
usort($students, function($a, $b) {
    return $b["average"] <=> $a["average"];
});
```

Chuỗi (String) trong PHP

Khái niệm chuỗi

Chuỗi trong PHP là một chuỗi các ký tự, như "Hello World". PHP cung cấp nhiều cách để khai báo và thao tác với chuỗi.

Khai báo chuỗi

PHP cung cấp nhiều cách để khai báo chuỗi như sử dụng dấu nháy đơn, nháy kép, heredoc và nowdoc.

Ứng dụng xử lý chuỗi

Xử lý chuỗi là một trong những tác vụ phổ biến nhất trong lập trình web, từ xử lý dữ liệu form, hiển thị nội dung đến thao tác với văn bản.

Khai báo chuỗi trong PHP

PHP hỗ trợ 4 cách để khai báo chuỗi:

1. Dấu nháy đơn ('...')

Hiển thị chính xác những gì được viết, không phân tích biến.

```
$name = 'Nguyễn Văn A';
$text = 'Xin chào $name'; // Kết quả: Xin chào $name
```

2. Dấu nháy kép ("...")

Cho phép phân tích biến bên trong chuỗi.

```
$name = "Nguyễn Văn A";
$text = "Xin chào $name"; // Kết quả: Xin chào Nguyễn Văn A
```

3. Cú pháp Heredoc (<<)

Hỗ trợ chuỗi nhiều dòng và phân tích biến.

```
$name = "Nguyễn Văn A";
$text = <<
```

4. Cú pháp Nowdoc (<<<'EOT' ... EOT;)

Hỗ trợ chuỗi nhiều dòng, không phân tích biến.

```
$name = "Nguyễn Văn A";
$text = <<<'EOT'
Xin chào $name!
Chuỗi này sẽ không
phân tích biến bên trong.
EOT;
```

3. Nối chuỗi trong PHP

Trong PHP, toán tử nối chuỗi là dấu chấm (.). Sử dụng để kết hợp hai hoặc nhiều chuỗi lại.

```
// 3.1. Nối chuỗi cơ bản
$firstName = "Nguyễn";
$lastName = "An";
$fullName = $firstName . " " . $lastName;
echo $fullName; // Hiển thị: Nguyễn An

// 3.2. Nối chuỗi và gán
$greeting = "Xin chào ";
$greeting .= "các bạn!";
echo $greeting; // Hiển thị: Xin chào các bạn!

// 3.3. Nối trong chuỗi nháy kép
echo "Tôi tên là {$firstName} {$lastName}.";
// Hiển thị: Tôi tên là Nguyễn An.
```

1. Tìm độ dài chuỗi trong PHP

Hàm strlen() được sử dụng để đếm số ký tự trong một chuỗi. Lưu ý rằng với chuỗi Unicode (như tiếng Việt), nên sử dụng hàm mb_strlen() để đếm chính xác.

```
// Đếm độ dài chuỗi ASCII
$text = "Hello";
echo strlen($text); // Hiển thị: 5

// Đếm độ dài chuỗi Unicode
$text = "Xin chào";
echo strlen($text); // Không chính xác với UTF-8
echo mb_strlen($text, 'UTF-8'); // Chính xác: 8
```

Tìm kiếm trong chuỗi

strpos()

Tìm vị trí xuất hiện đầu tiên của một chuỗi trong chuỗi khác

```
$pos = strpos("Hello world", "world");
// $pos = 6
```

stripos()

Giống strpos() nhưng không phân biệt hoa thường

```
$pos = stripos("Hello World", "world");
// $pos = 6
```

strrpos()

Tìm vị trí xuất hiện cuối cùng của một chuỗi

```
$pos = strrpos("Hello world world", "world");
// $pos = 12
```

str_contains()

Kiểm tra xem chuỗi có chứa chuỗi con không (PHP 8+)

```
$check = str_contains("Hello world", "world");
// $check = true
```

Khi tìm kiếm trong chuỗi tiếng Việt, nên sử dụng các hàm mb_strpos(), mb_stripos(), mb_strrpos() để xử lý đúng các ký tự Unicode.

Các hàm xử lý chuỗi phổ biến

Cắt và trích xuất chuỗi

- substr() - Trích xuất một phần của chuỗi
- trim() - Loại bỏ khoảng trắng (hoặc ký tự khác) từ đầu và cuối chuỗi
- ltrim() - Loại bỏ khoảng trắng từ đầu chuỗi
- rtrim() - Loại bỏ khoảng trắng từ cuối chuỗi

```
// Ví dụ minh họa
$text = "  Học PHP cơ bản  ";
echo substr($text, 5, 3); // Kết quả: PHP
echo trim($text);        // Kết quả: Học PHP cơ bản
echo ltrim($text);       // Kết quả: Học PHP cơ bản
echo rtrim($text);       // Kết quả:   Học PHP cơ bản
```

Thay đổi chuỗi

- str_replace() - Thay thế tất cả các lần xuất hiện của chuỗi tìm kiếm bằng chuỗi thay thế
- strtolower() - Chuyển chuỗi thành chữ thường
- strtoupper() - Chuyển chuỗi thành chữ hoa
- ucfirst() - Chuyển ký tự đầu tiên của chuỗi thành chữ hoa
- ucwords() - Chuyển ký tự đầu tiên của mỗi từ thành chữ hoa

```
// Ví dụ minh họa
$text = "Học php Cơ bản";
echo str_replace("php", "PHP", $text); // Kết quả: Học PHP Cơ bản
echo strtolower($text);               // Kết quả: học php cơ bản
echo strtoupper($text);               // Kết quả: HỌC PHP CƠ BẢN
echo ucfirst("xin chào");             // Kết quả: Xin chào
echo ucwords("xin chào việt nam");    // Kết quả: Xin Chào Việt Nam
```

Định dạng chuỗi

- sprintf() - Trả về chuỗi được định dạng
- printf() - Hiển thị chuỗi được định dạng
- nl2br() - Chèn thẻ HTML br trước tất cả các dòng mới
- number_format() - Định dạng số với hàng nghìn được nhóm

```
// Ví dụ minh họa
$name = "Minh";
$age = 25;
$price = 1000000;

// Định dạng chuỗi với các tham số
$text = sprintf("Tôi tên là %s, %d tuổi.", $name, $age);
echo $text;  // Kết quả: Tôi tên là Minh, 25 tuổi.

// Chuyển đổi xuống dòng thành thẻ br
$note = "Dòng 1\nDòng 2";
echo nl2br($note);  // Kết quả: Dòng 1
Dòng 2

// Định dạng số
echo number_format($price); // Kết quả: 1,000,000
```

2.2 Ví dụ thực tế về xử lý chuỗi

Xử lý tên người dùng

```
function formatName($fullName) {
    // Chuyển thành chữ thường
    $name = strtolower($fullName);

    // Chuyển ký tự đầu mỗi từ thành hoa
    $name = ucwords($name);

    // Loại bỏ khoảng trắng thừa
    $name = trim($name);

    return $name;
}

$input = "  nguyễn VĂN an  ";
echo formatName($input); // Nguyễn Văn An
```

Tạo slug cho URL

```
function createSlug($string) {
 // Chuyển sang chữ thường
 $string = strtolower($string);

 // Thay thế ký tự đặc biệt
 $string = preg_replace('/[^a-z0-9\s-]/', '', $string);

 // Thay khoảng trắng bằng dấu gạch ngang
 $string = preg_replace('/[\s]+/', '-', $string);

 return $string;
}

$title = "Học PHP Cơ Bản";
echo createSlug($title); // hoc-php-co-ban
```

Demo: Xử lý dữ liệu từ file CSV

```
// Giả lập dữ liệu CSV
$csvData = "id,name,score\n1,An,85\n2,Bình,92\n3,Cường,78";

// Chuyển thành mảng các dòng
$rows = explode("\n", $csvData);

// Lấy tiêu đề
$headers = explode(",", $rows[0]);

// Xử lý từng dòng dữ liệu
$students = [];
for ($i = 1; $i < count($rows); $i++) {
    $data = explode(",", $rows[$i]);
    $student = [
        $headers[0] => $data[0],
        $headers[1] => $data[1],
        $headers[2] => $data[2]
    ];
    $students[] = $student;
}

// Hiển thị dữ liệu
foreach ($students as $student) {
    echo "Sinh viên: " . $student['name'] .
        ", Điểm: " . $student['score'] . "
";
}
```

Trong ví dụ này, chúng ta đã kết hợp kỹ thuật xử lý chuỗi và mảng để phân tích dữ liệu định dạng CSV. Đây là một tác vụ phổ biến khi nhập dữ liệu từ file hoặc xử lý dữ liệu từ các hệ thống khác.

2.3 Xây dựng hệ thống đăng nhập

```
// Dữ liệu người dùng (thực tế sẽ lưu trong cơ sở dữ liệu)
$users = [
 "admin" => [
 "password" => "admin123",
 "fullname" => "Quản trị viên"
 ],
 "user1" => [
 "password" => "pass123",
 "fullname" => "Nguyễn Văn An"
 ],
 "user2" => [
 "password" => "secure456",
 "fullname" => "Trần Thị Bình"
 ]
];

// Mô phỏng dữ liệu đăng nhập
$username = "user1";
$password = "pass123";

// Kiểm tra đăng nhập
if (array_key_exists($username, $users) &&
 $users[$username]["password"] === $password) {

 $fullname = $users[$username]["fullname"];
 echo "Đăng nhập thành công! Xin chào $fullname.";
} else {
 echo "Tên đăng nhập hoặc mật khẩu không đúng.";
}
```

Dữ liệu người dùng

Lưu trữ thông tin về người dùng (tên đăng nhập, mật khẩu, họ tên) trong một mảng kết hợp.

Dữ liệu đăng nhập

Thông tin đăng nhập được gửi từ form (mô phỏng bằng biến trong ví dụ).

Xác thực

Kiểm tra sự tồn tại của tên đăng nhập và tính chính xác của mật khẩu.

2.4.1 Bài tập thực hành

Bài tập 1: Bài tập về mảng

Viết chương trình PHP để tạo một mảng chứa thông tin về 5 sản phẩm (tên, giá, số lượng). Sau đó tính tổng giá trị của tất cả sản phẩm và hiển thị sản phẩm có giá trị cao nhất.

Hướng dẫn chi tiết:

1. Tạo một mảng đa chiều với cấu trúc: `$products = array(array("name" => "Tên sản phẩm", "price" => giá, "quantity" => số lượng), ...);`
2. Sử dụng vòng lặp `foreach` để duyệt qua mảng sản phẩm và tính tổng giá trị: `giá × số lượng` cho mỗi sản phẩm
3. Tạo biến `$totalValue` để lưu tổng giá trị của tất cả sản phẩm
4. Sử dụng biến `$maxProduct` để lưu thông tin sản phẩm có giá trị cao nhất
5. So sánh giá trị của từng sản phẩm để tìm sản phẩm có giá trị cao nhất
6. Hiển thị kết quả bằng lệnh `echo`

2.4.2 Bài tập thực hành

Bài tập 2: Bài tập về chuỗi

Viết hàm PHP nhận vào một chuỗi và trả về chuỗi đã được chuẩn hóa (loại bỏ khoảng trắng thừa, chuyển ký tự đầu câu thành chữ hoa, các ký tự khác thành chữ thường).

Hướng dẫn chi tiết:

1. Tạo hàm `normalizeString($inputString)` nhận vào một chuỗi
2. Sử dụng `trim()` để loại bỏ khoảng trắng ở đầu và cuối chuỗi
3. Sử dụng `preg_replace()` để thay thế nhiều khoảng trắng liên tiếp bằng một khoảng trắng
4. Sử dụng `strtolower()` để chuyển toàn bộ chuỗi thành chữ thường
5. Tách chuỗi thành các câu bằng cách sử dụng `explode()` với dấu chấm (.)
6. Duyệt qua mỗi câu và sử dụng `ucfirst()` để viết hoa ký tự đầu tiên
7. Nối các câu lại với nhau bằng `implode()`
8. Trả về chuỗi đã được chuẩn hóa

2.4.3 Bài tập thực hành

Bài tập 3: Bài tập kết hợp

Tạo một form đăng ký với các trường: họ tên, email, mật khẩu, xác nhận mật khẩu. Viết mã PHP để xác thực dữ liệu nhập vào (kiểm tra định dạng email, độ dài mật khẩu, mật khẩu trùng khớp) và hiển thị thông báo phù hợp.

Hướng dẫn chi tiết:

1. Tạo file HTML với form có method="POST" và action chỉ đến file PHP xử lý
2. Thêm các trường input: họ tên (text), email (email), mật khẩu (password), xác nhận mật khẩu (password) và nút submit
3. Tạo file PHP để xử lý dữ liệu từ form:
4. Kiểm tra phương thức gửi dữ liệu có phải là POST không
5. Lấy dữ liệu từ $_POST và lưu vào biến
6. Tạo mảng $errors để lưu các lỗi
7. Kiểm tra trường họ tên có trống không
8. Kiểm tra email có đúng định dạng không bằng `filter_var($email, FILTER_VALIDATE_EMAIL)`
9. Kiểm tra độ dài mật khẩu (ít nhất 8 ký tự)
10. Kiểm tra hai mật khẩu có trùng khớp không
11. Nếu không có lỗi, hiển thị thông báo đăng ký thành công
12. Nếu có lỗi, hiển thị danh sách lỗi và giữ lại các giá trị đã nhập (trừ mật khẩu)

2.4.4 Bài tập thực hành

Bài tập 4: Xử lý dữ liệu JSON

Viết chương trình PHP để đọc và xử lý dữ liệu từ một file JSON chứa thông tin sinh viên, sau đó tính điểm trung bình và sắp xếp sinh viên theo thứ tự điểm giảm dần.

Hướng dẫn chi tiết:

1. Tạo file JSON `students.json` chứa mảng các đối tượng sinh viên với các thuộc tính: id, name, scores (mảng điểm các môn)
2. Sử dụng `file_get_contents()` để đọc nội dung file JSON
3. Sử dụng `json_decode()` để chuyển đổi chuỗi JSON thành mảng PHP
4. Tạo hàm tính điểm trung bình từ mảng điểm
5. Duyệt qua mảng sinh viên, tính và thêm thuộc tính `average` cho mỗi sinh viên
6. Sử dụng hàm `usort()` với hàm so sánh tùy chỉnh để sắp xếp sinh viên theo điểm trung bình giảm dần
7. Hiển thị danh sách sinh viên đã sắp xếp với định dạng dễ đọc

2.4.5 Bài tập thực hành

Bài tập 5: Quản lý sản phẩm với CSV

Tạo một ứng dụng quản lý sản phẩm sử dụng file CSV để lưu trữ dữ liệu, cho phép thêm, sửa, xóa và tìm kiếm sản phẩm.

Hướng dẫn chi tiết:

1. Tạo file CSV `products.csv` với các cột: id, name, price, quantity, category
2. Viết các hàm để đọc và ghi dữ liệu vào file CSV
3. Tạo lớp `ProductManager` với các phương thức:
4. `getAllProducts()`: đọc và trả về tất cả sản phẩm
5. `addProduct($data)`: thêm sản phẩm mới vào file
6. `updateProduct($id, $data)`: cập nhật thông tin sản phẩm
7. `deleteProduct($id)`: xóa sản phẩm
8. `searchProducts($keyword)`: tìm kiếm sản phẩm theo tên hoặc danh mục
9. Tạo giao diện HTML/PHP đơn giản để tương tác với các chức năng
10. Thêm xác thực dữ liệu nhập vào và xử lý lỗi

2.4.6 Bài tập thực hành

Bài tập 6: Làm việc với API

Viết ứng dụng PHP kết nối với một API công khai (như OpenWeatherMap, COVID-19 API, hoặc REST Countries), lấy dữ liệu và hiển thị theo yêu cầu.

Hướng dẫn chi tiết:

1. Đăng ký để lấy API key (nếu cần) từ dịch vụ bạn chọn
2. Sử dụng `curl` hoặc `file_get_contents()` để gửi yêu cầu đến API
3. Xử lý phản hồi JSON từ API bằng `json_decode()`
4. Tạo các hàm xử lý dữ liệu để trích xuất thông tin cần thiết
5. Thiết kế giao diện để hiển thị dữ liệu một cách trực quan
6. Thêm tính năng tìm kiếm hoặc lọc dữ liệu (ví dụ: tìm kiếm thời tiết theo thành phố, lọc quốc gia theo châu lục)
7. Lưu trữ kết quả vào bộ nhớ đệm (cache) để giảm số lượng yêu cầu API
8. Xử lý các lỗi kết nối và phản hồi từ API