Làm Việc Với Form

Form là thành phần quan trọng của bất kỳ trang web nào. Là cầu nối giữa người dùng và hệ thống, cho phép thu thập và xử lý thông tin. Chúng ta sẽ tìm hiểu các thành phần form, cách xác thực dữ liệu, và làm thế nào để tạo trải nghiệm người dùng tốt nhất.

Tại sao form lại quan trọng?

- Thu thập thông tin từ người dùng
- Tạo tương tác giữa người dùng và ứng dụng
- Cho phép đăng ký, đăng nhập và các chức năng thiết yếu khác

Nội Dung Khóa Học

1

Tổng quan về Form

- Các thành phần cơ bản của form
- HTML5 và các thuộc tính mới
- Các loại input và validation
- Cách tạo form thân thiện với người dùng

2

Xử lý dữ liệu Form

- Phương thức GET và POST
- Xử lý an toàn dữ liệu nhập vào
- Validation dữ liệu phía server
- Xử lý upload file và dữ liệu phức tạp

3

Hiển thị dữ liệu

- Phản hồi thông báo cho người dùng
- Hiển thị lỗi và thông báo thành công
- Lưu trữ và hiển thị dữ liệu từ cơ sở dữ liệu
- Ajax và hiển thị không làm mới trang

4

Bài tập thực hành

- Xây dựng form đăng ký/đăng nhập
- Tạo form đặt hàng với validation
- Xây dựng hệ thống feedback

1. Form Trong Web Development

Form được sử dụng trong nhiều trường hợp như đăng ký tài khoản, đăng nhập, tìm kiếm, đặt hàng, gửi phản hồi và nhiều mục đích khác.

Các Loại Form Phổ Biến

- Form đăng ký và đăng nhập
- Form tìm kiếm và lọc dữ liệu
- Form liên hệ và phản hồi
- Form thanh toán và đặt hàng
- Form upload file

Các Thành Phần Cơ Bản

Một form HTML thường bao gồm các thành phần như:

- Trường văn bản (text field)
- Trường mật khẩu (password field)
- Checkbox và radio button
- Menu dropdown (select)
- Textarea cho văn bản dài
- Nút gửi (submit button)

Xử Lý Form Với PHP

PHP cung cấp hai mảng toàn cục chính để xử lý dữ liệu form:

- **$_GET**: Xử lý dữ liệu được gửi bằng method GET
- **$_POST**: Xử lý dữ liệu được gửi bằng method POST

2. Cấu Trúc Cơ Bản Của Form HTML

```
<form action="process.php" method="POST">
    <label for="name">Họ tên:</label>
    <input type="text" id="name" name="name">

    <label for="email">Email:</label>
    <input type="email" id="email" name="email">

    <input type="submit" value="Gửi">
</form>
```

Các thành phần quan trọng: **action** (tệp xử lý form), **method** (phương thức gửi dữ liệu), **input** (trường nhập liệu) và **name** (định danh cho PHP).

Giải Thích Chi Tiết Về Thuộc Tính Form

- **action**: Xác định URL của trang web hoặc tệp script sẽ xử lý dữ liệu form sau khi người dùng submit. Trong ví dụ trên, "process.php" sẽ nhận và xử lý dữ liệu.
- **method**: Xác định phương thức HTTP để gửi dữ liệu đến máy chủ:
  - **GET**: Dữ liệu được gửi thông qua URL, có thể nhìn thấy được và có giới hạn kích thước. Thích hợp cho tìm kiếm hoặc dữ liệu không nhạy cảm.
  - **POST**: Dữ liệu được gửi ngầm, không hiển thị trong URL, không giới hạn kích thước. Thích hợp cho dữ liệu nhạy cảm hoặc tải lên tệp.

Các Thành Phần Phổ Biến Trong Form

Ngoài các thành phần đã thấy trong ví dụ, form còn có thể chứa nhiều loại trường khác:

- **label**: Tạo nhãn cho trường input, cải thiện khả năng tiếp cận và UX.
- **fieldset** và **legend**: Nhóm các trường liên quan và tạo tiêu đề cho nhóm đó.
- **textarea**: Cho phép nhập văn bản nhiều dòng, thích hợp cho mô tả, bình luận.
- **select** và **option**: Tạo menu thả xuống cho người dùng lựa chọn.
- **button**: Tạo nút có thể tùy chỉnh hành vi (submit, reset, hoặc javascript).

Thuộc Tính Quan Trọng Của Input

Mỗi trường input cần các thuộc tính sau để hoạt động hiệu quả với PHP:

- **type**: Xác định loại dữ liệu (text, email, password, number, date...)
- **name**: Cực kỳ quan trọng khi xử lý form với PHP, định danh trường trong mảng $_POST hoặc $_GET.
- **id**: Kết nối input với label và cho phép JavaScript truy cập trường.
- **value**: Giá trị mặc định hoặc giá trị hiện tại của trường.
- **required**: Đánh dấu trường bắt buộc phải điền trước khi submit.
- **placeholder**: Văn bản gợi ý trong trường khi nó trống.

Xử Lý Form Trong PHP

Sau khi form được gửi đến máy chủ, PHP có thể truy cập dữ liệu thông qua:

```
// Với method="POST"
$name = $_POST['name'];
$email = $_POST['email'];

// Với method="GET"
$name = $_GET['name'];
$email = $_GET['email'];
```

3. Các Loại Input Trong Form

Input Text

Trường nhập văn bản đơn dòng, sử dụng cho tên, địa chỉ, số điện thoại, v.v.

```
<input type="text" name="username" placeholder="Nhập tên người dùng">
```

Có thể thêm thuộc tính **required**, **minlength**, **maxlength** để kiểm tra dữ liệu.

Input Password

Tương tự như text nhưng ẩn ký tự người dùng nhập vào bằng dấu chấm hoặc sao (*).

```
<input type="password" name="password">
```

Thường kết hợp với pattern để kiểm tra độ mạnh của mật khẩu.

Radio Buttons

Cho phép người dùng chọn một lựa chọn từ một nhóm các tùy chọn.

```
<input type="radio" name="gender" value="male"> Nam
<input type="radio" name="gender" value="female"> Nữ
```

Các radio cùng một nhóm phải có cùng thuộc tính name.

Checkboxes

Cho phép người dùng chọn nhiều tùy chọn cùng lúc.

```
<input type="checkbox" name="hobby[]" value="music"> Âm nhạc
<input type="checkbox" name="hobby[]" value="sports"> Thể thao
```

Thêm dấu [] trong name để PHP nhận dạng là mảng giá trị.

Select Dropdown

Tạo danh sách thả xuống cho người dùng lựa chọn.

```
<select name="city">
  <option value="hn">Hà Nội</option>
  <option value="hcm">TP HCM</option>
</select>
```

Thêm thuộc tính **multiple** để cho phép chọn nhiều giá trị.

Textarea

Trường nhập văn bản nhiều dòng, thích hợp cho nội dung dài.

```
<textarea name="message" rows="5" cols="30"></textarea>
```

Thuộc tính **rows** và **cols** xác định kích thước hiển thị.

Input File

Cho phép người dùng tải lên tệp từ thiết bị của họ.

```
<input type="file" name="document">
```

Thêm thuộc tính **accept** để giới hạn loại tệp và **multiple** để tải nhiều tệp.

Input Date

Trường chọn ngày tháng với định dạng chuẩn.

```
<input type="date" name="birthday">
```

Các biến thể khác: **datetime-local**, **month**, **week**, **time**.

Input Number

Trường nhập số với các nút tăng/giảm.

```
<input type="number" name="quantity" min="1" max="10">
```

Thuộc tính **step** xác định bước nhảy giữa các giá trị.

4. Phương Thức GET và POST

Phương Thức GET

- Tham số hiển thị trên thanh địa chỉ dưới dạng cặp key-value sau dấu "?"
- Giới hạn dung lượng (~2048 ký tự)
- Có thể bookmark và cache
- Không an toàn cho dữ liệu nhạy cảm
- Thích hợp tìm kiếm, lọc dữ liệu, phân trang

Ví dụ URL với GET:

```
example.com/search?
keyword=laptop&price=5000000&brand=dell
```

Phương Thức POST

- Dữ liệu gửi trong thân HTTP request
- Gửi được dữ liệu lớn, tệp tin
- Không thể bookmark hoặc cache
- An toàn phù hợp cho mật khẩu, thông tin tài khoản, thông tin thanh toán, đăng nhập, đăng ký, tải lên tệp, cập nhật dữ liệu

Mã PHP xử lý POST:

```
<?php
if ($_SERVER["REQUEST_METHOD"] == "POST")
{
   $username = $_POST["username"];
   $password = $_POST["password"];
   // Xử lý đăng nhập
}
?>
```

So Sánh Chi Tiết

Tiêu chí	GET	POST
Bảo mật	Thấp (dữ liệu hiện trên URL)	Cao hơn (dữ liệu ẩn trong request)
Tốc độ (không đáng kể)	Nhanh hơn (với dữ liệu nhỏ)	Chậm hơn
Lưu trữ	Lưu trong lịch sử trình duyệt	Không lưu trong lịch sử
Sử dụng cho	Đọc dữ liệu, tìm kiếm	Tạo, cập nhật, xóa dữ liệu

5. Xử Lý Form Với Phương Thức GET

Phương thức GET dữ liệu được gửi như một phần của URL, giúp dễ dàng theo dõi.

```
<!-- form.html -->
<form action="process.php" method="GET">
    <input type="text" name="ten" placeholder="Nhập tên">
    <input type="submit" value="Gửi">
</form>

<?php
// process.php
if(isset($_GET['ten'])) {
    $ten = $_GET['ten'];
    echo "Xin chào, " . $ten;
}
?>
```

URL kết quả sẽ có dạng: process.php?ten=NguyenVanA

Xử lý nhiều tham số với GET

```
<!-- form-multi.html -->
<form action="process-multi.php" method="GET">
    <input type="text" name="ten" placeholder="Nhập tên">
    <input type="number" name="tuoi" placeholder="Nhập tuổi">
    <select name="thanhpho">
        <option value="hanoi">Hà Nội</option>
        <option value="hcm">TP. Hồ Chí Minh</option>
    </select>
    <input type="submit" value="Gửi">
</form>

<?php
// process-multi.php
if(isset($_GET['ten']) && isset($_GET['tuoi']) && isset($_GET['thanhpho'])) {
    $ten = $_GET['ten'];
    $tuoi = $_GET['tuoi'];
    $thanhpho = $_GET['thanhpho'];

    echo "Xin chào " . $ten . ", " . $tuoi . " tuổi từ " .
        ($thanhpho == "hanoi" ? "Hà Nội" : "TP. Hồ Chí Minh");
}
?>
```

URL kết quả sẽ có dạng: process-multi.php?ten=NguyenVanA&tuoi=25&thanhpho=hanoi

Xác thực và làm sạch dữ liệu GET

```
<?php
// Xác thực dữ liệu
if(isset($_GET['ten']) && !empty($_GET['ten'])) {
    // Làm sạch dữ liệu
    $ten = htmlspecialchars($_GET['ten']);
    echo "Xin chào, " . $ten;
} else {
    echo "Vui lòng nhập tên của bạn!";
}
?>
```

6. Xử Lý Form Với Phương Thức POST

Phương thức POST dữ liệu được gửi không hiển thị trong URL và không có giới hạn về kích thước dữ liệu.

```
<!-- form.html -->
<form action="process.php" method="POST">
 <input type="text" name="ten" placeholder="Nhập tên">
 <input type="password" name="matkhau" placeholder="Mật khẩu">
 <input type="submit" value="Đăng nhập">
</form>

<?php
// process.php
if(isset($_POST['ten']) && isset($_POST['matkhau'])) {
 $ten = $_POST['ten'];
 $matkhau = $_POST['matkhau'];
 echo "Đăng nhập thành công, " . $ten;
}
?>
```

Lưu ý bảo mật:

Mặc dù POST an toàn hơn GET, bạn vẫn cần xử lý dữ liệu đầu vào để tránh các lỗ hổng bảo mật như:

- SQL Injection: Sử dụng Prepared Statements
- XSS (Cross-Site Scripting): Lọc dữ liệu với htmlspecialchars()

Ví dụ về xử lý dữ liệu an toàn:

```
<?php
// Xử lý dữ liệu đầu vào an toàn
if(isset($_POST['ten']) && isset($_POST['matkhau'])) {
 $ten = htmlspecialchars($_POST['ten']);
 $matkhau = password_hash($_POST['matkhau'], PASSWORD_DEFAULT);

 // Tiếp tục xử lý...
}
?>
```

7. Biến $_REQUEST Trong PHP

Biến $_REQUEST là một biến toàn cục PHP sử dụng để thu thập dữ liệu form sau khi gửi form HTML.

$_REQUEST có thể thu thập dữ liệu gửi bằng cả phương thức GET và POST.

Cách $_REQUEST hoạt động

$_REQUEST là một mảng kết hợp chứa nội dung của $_GET, $_POST và $_COOKIE. Điều này có nghĩa là nó có thể xử lý dữ liệu từ URL, form và cookie cùng một lúc.

Khi gửi form, PHP sẽ tự động điền dữ liệu vào biến $_REQUEST, cho phép truy cập các giá trị thông qua tên trường form.

```
<?php
$ten = $_REQUEST['ten'];
echo $ten;
?>
```

So sánh với $_GET và $_POST

Trong khi $_REQUEST kết hợp cả hai phương thức, $_GET và $_POST mang những đặc điểm riêng:

- $_GET: Chỉ thu thập dữ liệu từ URL (method="get")
- $_POST: Thu thập dữ liệu từ form (method="post")
- $_REQUEST: Thu thập từ cả hai, nhưng xử lý chậm hơn và tiềm ẩn rủi ro bảo mật

Ví dụ hoàn chỉnh

```
<!-- form.html -->
<form action="process.php" method="POST">
  <input type="text" name="ten" placeholder="Nhập tên">
  <input type="email" name="email" placeholder="Email">
  <input type="submit" value="Gửi">
</form>

<?php
// process.php
if(isset($_REQUEST['ten']) && isset($_REQUEST['email'])) {
  $ten = $_REQUEST['ten'];
  $email = $_REQUEST['email'];

  // Xử lý dữ liệu
  echo "Xin chào " . $ten . ", email của bạn là: " . $email;
}
?>
```

8. Kiểm Tra Dữ Liệu Form

Kiểm tra tồn tại dữ liệu với isset()

Xác định xem biến đã được khởi tạo và có giá trị khác NULL hay không

```
<?php
// Kiểm tra nếu biến đã được gửi từ form
if(isset($_POST['username'])) {
    // Xử lý dữ liệu username
    echo "Username đã được nhập: " . $_POST['username'];
} else {
    echo "Vui lòng nhập username";
}
?>
```

Kiểm tra form đã submit hay chưa

Sử dụng các điều kiện kiểm tra phương thức submit hoặc trường ẩn

```
<?php
// Phương pháp 1: Kiểm tra nút submit
if(isset($_POST['submit'])) {
    // Form đã được gửi
}

// Phương pháp 2: Sử dụng trường ẩn
if(isset($_POST['form_submitted']) && $_POST['form_submitted'] == '1') {
    // Form đã được gửi
}
?>
```

Validate dữ liệu đầu vào

Kiểm tra định dạng email, số điện thoại, độ dài mật khẩu, v.v.

```
<?php
// Kiểm tra email
$email = $_POST['email'];
if(!filter_var($email, FILTER_VALIDATE_EMAIL)) {
    echo "Email không hợp lệ";
}

// Kiểm tra độ dài mật khẩu
$password = $_POST['password'];
if(strlen($password) < 8) {
    echo "Mật khẩu phải có ít nhất 8 ký tự";
}

// Kiểm tra số điện thoại (VN format)
$phone = $_POST['phone'];
if(!preg_match('/^(0|\+84)(\d{9,10})$/', $phone)) {
    echo "Số điện thoại không hợp lệ";
}
?>
```

Lọc dữ liệu với filter_var()

Làm sạch dữ liệu đầu vào trước khi xử lý hoặc lưu trữ

```
<?php
// Lọc và xác thực email
$email = filter_var($_POST['email'], FILTER_SANITIZE_EMAIL);
if(filter_var($email, FILTER_VALIDATE_EMAIL)) {
    echo "Email hợp lệ: " . $email;
}

// Lọc và xác thực số nguyên
$age = filter_var($_POST['age'], FILTER_SANITIZE_NUMBER_INT);
if(filter_var($age, FILTER_VALIDATE_INT)) {
    echo "Tuổi hợp lệ: " . $age;
}

// Lọc và xác thực URL
$website = filter_var($_POST['website'], FILTER_SANITIZE_URL);
if(filter_var($website, FILTER_VALIDATE_URL)) {
    echo "URL hợp lệ: " . $website;
}
?>
```

Ví dụ hoàn chỉnh về xử lý form

```
<?php
// Khởi tạo biến lỗi
$errors = [];

// Kiểm tra form đã được submit chưa
if($_SERVER["REQUEST_METHOD"] == "POST") {

    // Kiểm tra tên
    if(empty($_POST["name"])) {
        $errors[] = "Tên là bắt buộc";
    } else {
        $name = filter_var($_POST["name"], FILTER_SANITIZE_STRING);
    }

    // Kiểm tra email
    if(empty($_POST["email"])) {
        $errors[] = "Email là bắt buộc";
    } else {
        $email = filter_var($_POST["email"], FILTER_SANITIZE_EMAIL);
        if(!filter_var($email, FILTER_VALIDATE_EMAIL)) {
            $errors[] = "Email không hợp lệ";
        }
    }

    // Hiển thị lỗi hoặc xử lý dữ liệu
    if(count($errors) > 0) {
        foreach($errors as $error) {
            echo $error . "
";
        }
    } else {
        echo "Form đã được xử lý thành công!";
        // Xử lý dữ liệu tiếp theo (lưu vào database, gửi email, v.v.)
    }
}
?>
```

9. Xử Lý Upload File Trong PHP

HTML Form Upload

```
<form action="upload.php" method="POST"
   enctype="multipart/form-data">
   <input type="file" name="fileToUpload">
   <input type="submit" value="Tải lên">
</form>
```

Thuộc tính **enctype="multipart/form-data"** là bắt buộc khi form có chức năng upload file. Nếu thiếu thuộc tính này, dữ liệu file sẽ không được gửi đến server.

Có thể thêm các thuộc tính khác cho input file:

- **multiple**: cho phép người dùng chọn nhiều file cùng lúc
- **accept**: giới hạn loại file có thể chọn (ví dụ: accept="image/*")

PHP Script Xử Lý

```
<?php
// upload.php
if(isset($_FILES['fileToUpload'])) {
 $target_dir = "uploads/";
 $target_file = $target_dir .
 basename($_FILES["fileToUpload"]["name"]);

 if(move_uploaded_file(
 $_FILES["fileToUpload"]["tmp_name"],
 $target_file)) {
 echo "File đã được tải lên thành công";
 } else {
 echo "Có lỗi xảy ra khi tải file";
 }
}
?>
```

Hàm **move_uploaded_file()** di chuyển file từ thư mục tạm thời vào thư mục lưu trữ.

Kiểm Tra Và Xác Thực File Upload

Để đảm bảo an toàn, bạn nên luôn kiểm tra kỹ lưỡng file trước khi lưu trữ:

Kiểm tra kích thước file

```
// Giới hạn file dưới 2MB
if($_FILES["fileToUpload"]["size"] > 2000000) {
   echo "File quá lớn!";
   $uploadOk = 0;
}
```

Kiểm tra loại file

```
$fileType = strtolower(pathinfo($target_file, PATHINFO_EXTENSION));
// Chỉ cho phép định dạng ảnh
if($fileType != "jpg" && $fileType != "png" && $fileType != "jpeg" && $fileType != "gif") {
   echo "Chỉ chấp nhận file JPG, JPEG, PNG & GIF.";
   $uploadOk = 0;
}
```

Kiểm tra lỗi upload

```
if($_FILES["fileToUpload"]["error"] > 0) {
   echo "Lỗi: " . $_FILES["fileToUpload"]["error"];
   $uploadOk = 0;
}
```

Xử Lý Upload Nhiều File

```
<input type="file" name="files[]" multiple>

<?php
foreach($_FILES['files']['tmp_name'] as $key => $tmp_name) {
   $file_name = $_FILES['files']['name'][$key];
   $file_tmp = $_FILES['files']['tmp_name'][$key];

   // Di chuyển từng file vào thư mục đích
   move_uploaded_file($file_tmp, "uploads/".$file_name);
}
?>
```

10. Mảng $_FILES

Khi upload file, PHP lưu thông tin trong mảng $_FILES với các thành phần:

1. **$_FILES['file']['name']** - Tên gốc của file được upload
2. **$_FILES['file']['type']** - Kiểu MIME của file (VD: image/jpeg)
3. **$_FILES['file']['size']** - Kích thước file tính bằng byte
4. **$_FILES['file']['tmp_name']** - Đường dẫn tạm thời lưu file trên server
5. **$_FILES['file']['error']** - Mã lỗi nếu có (0 nếu thành công)

Ví dụ về cách sử dụng mảng $_FILES

```
<?php
// Kiểm tra xem file đã được upload hay chưa
if(isset($_FILES['fileToUpload'])) {
    // Hiển thị thông tin chi tiết về file
    echo "Tên file: " . $_FILES['fileToUpload']['name'] . "<br>";
    echo "Kiểu: " . $_FILES['fileToUpload']['type'] . "<br>";
    echo "Kích thước: " . ($_FILES['fileToUpload']['size'] / 1024) . " KB<br>";
    echo "Lưu tạm tại: " . $_FILES['fileToUpload']['tmp_name'] . "<br>";

    // Kiểm tra lỗi
    if($_FILES['fileToUpload']['error'] === 0) {
        echo "Không có lỗi khi upload";
    } else {
        echo "Mã lỗi: " . $_FILES['fileToUpload']['error'];
    }
}
?>
```

Các mã lỗi thường gặp

Mảng $_FILES['file']['error'] có thể chứa các giá trị sau:

1. **UPLOAD_ERR_OK (0)** - Upload thành công, không có lỗi
2. **UPLOAD_ERR_INI_SIZE (1)** - File vượt quá giới hạn upload_max_filesize trong php.ini
3. **UPLOAD_ERR_FORM_SIZE (2)** - File vượt quá giới hạn MAX_FILE_SIZE trong form HTML
4. **UPLOAD_ERR_PARTIAL (3)** - File chỉ được upload một phần
5. **UPLOAD_ERR_NO_FILE (4)** - Không có file nào được upload
6. **UPLOAD_ERR_NO_TMP_DIR (6)** - Thiếu thư mục tạm thời
7. **UPLOAD_ERR_CANT_WRITE (7)** - Không thể ghi file vào ổ đĩa
8. **UPLOAD_ERR_EXTENSION (8)** - Upload bị dừng bởi một extension của PHP

Kiểm tra và xác thực file upload

```
<?php
// Kiểm tra kiểu file
$allowed_types = array('image/jpeg', 'image/png', 'image/gif');
if(!in_array($_FILES['fileToUpload']['type'], $allowed_types)) {
    die("Chỉ chấp nhận file ảnh (JPEG, PNG, GIF)");
}

// Kiểm tra kích thước file (giới hạn 2MB)
if($_FILES['fileToUpload']['size'] > 2 * 1024 * 1024) {
    die("File không được vượt quá 2MB");
}

// Kiểm tra phần mở rộng thực tế của file
$file_info = pathinfo($_FILES['fileToUpload']['name']);
$extension = strtolower($file_info['extension']);
$allowed_extensions = array('jpg', 'jpeg', 'png', 'gif');
if(!in_array($extension, $allowed_extensions)) {
    die("Phần mở rộng file không hợp lệ");
}

// Tạo tên file ngẫu nhiên để tránh ghi đè
$new_filename = uniqid() . '.' . $extension;
$target_file = "uploads/" . $new_filename;

// Di chuyển file từ thư mục tạm sang thư mục đích
if(move_uploaded_file($_FILES['fileToUpload']['tmp_name'], $target_file)) {
    echo "Upload thành công! File lưu tại: " . $target_file;
} else {
    echo "Có lỗi xảy ra khi lưu file!";
}
?>
```

Upload nhiều file cùng lúc

Để upload nhiều file, sử dụng input với thuộc tính multiple và tên có dạng mảng:

```
<!-- HTML Form để upload nhiều file -->
<form action="upload_multiple.php" method="POST" enctype="multipart/form-data">
    <input type="file" name="files[]" multiple>
    <input type="submit" value="Upload Files">
</form>

<?php
// Xử lý upload nhiều file
if(isset($_FILES['files'])) {
    // Đếm tổng số file được upload
    $total_files = count($_FILES['files']['name']);

    // Xử lý từng file một
    for($i = 0; $i < $total_files; $i++) {
        // Kiểm tra xem file hiện tại có lỗi không
        if($_FILES['files']['error'][$i] == 0) {
            $filename = $_FILES['files']['name'][$i];
            $filetype = $_FILES['files']['type'][$i];
            $filesize = $_FILES['files']['size'][$i];
            $temp_location = $_FILES['files']['tmp_name'][$i];

            // Xử lý file tại đây...
            echo "File " . ($i+1) . ": " . $filename . " đã được upload<br>";
        }
    }
}
?>
```

11. Xử Lý Nhiều Input Cùng Tên

Khi làm việc với form HTML, đôi khi chúng ta cần cho phép người dùng chọn nhiều giá trị cho cùng một trường dữ liệu. PHP cung cấp cách xử lý thuận tiện bằng cách sử dụng cú pháp mảng trong tên input.

1. Checkbox - Cho phép chọn nhiều

```
<!-- form.html -->
<form action="process.php" method="POST">
    <input type="checkbox" name="sothich[]" value="doc_sach"> Đọc sách
    <input type="checkbox" name="sothich[]" value="am_nhac"> Âm nhạc
    <input type="checkbox" name="sothich[]" value="the_thao"> Thể thao
    <input type="submit" value="Gửi">
</form>

<?php
// process.php
if(isset($_POST['sothich'])) {
    echo "Sở thích của bạn là: <br>";
    foreach($_POST['sothich'] as $sothich) {
        echo "- " . $sothich . "<br>";
    }
}
?>
```

Lưu ý dấu [] sau tên input để PHP nhận biết đây là mảng.

2. Select Multiple - Chọn nhiều tùy chọn

```
<!-- form với select multiple -->
<form action="process.php" method="POST">
    <select name="ngonngu[]" multiple>
        <option value="php">PHP</option>
        <option value="javascript">JavaScript</option>
        <option value="python">Python</option>
        <option value="java">Java</option>
    </select>
    <input type="submit" value="Gửi">
</form>

<?php
// Xử lý select multiple
if(isset($_POST['ngonngu'])) {
    echo "Ngôn ngữ bạn chọn: <br>";
    foreach($_POST['ngonngu'] as $ngonngu) {
        echo "- " . $ngonngu . "<br>";
    }
}
?>
```

3. Radio Buttons - Chỉ cho phép chọn một

Với radio buttons, mặc dù chỉ cho phép chọn một giá trị, nhưng chúng vẫn có cùng tên:

```
<form action="process.php" method="POST">
    <input type="radio" name="gioitinh" value="nam"> Nam
    <input type="radio" name="gioitinh" value="nu"> Nữ
    <input type="submit" value="Gửi">
</form>

<?php
// Xử lý radio button
if(isset($_POST['gioitinh'])) {
    echo "Giới tính: " . $_POST['gioitinh'];
}
?>
```

4. Xử lý mảng đa chiều

Bạn cũng có thể tạo mảng đa chiều trong form:

```
<form action="process.php" method="POST">
    <input type="text" name="sanpham[0][ten]" placeholder="Tên sản phẩm 1">
    <input type="number" name="sanpham[0][soluong]" placeholder="Số lượng">

    <input type="text" name="sanpham[1][ten]" placeholder="Tên sản phẩm 2">
    <input type="number" name="sanpham[1][soluong]" placeholder="Số lượng">

    <input type="submit" value="Gửi">
</form>

<?php
// Xử lý mảng đa chiều
if(isset($_POST['sanpham'])) {
    foreach($_POST['sanpham'] as $index => $sp) {
        echo "Sản phẩm " . ($index+1) . ": " . $sp['ten'] . " - Số lượng: " . $sp['soluong'] . "<br>";
    }
}
?>
```

5. Kiểm tra dữ liệu trước khi xử lý

Khi xử lý mảng từ form, luôn kiểm tra dữ liệu trước khi sử dụng:

```
<?php
if(isset($_POST['sothich']) && is_array($_POST['sothich'])) {
    // Lọc dữ liệu
    $sothich = array_map('htmlspecialchars', $_POST['sothich']);

    // Kiểm tra mảng rỗng
    if(count($sothich) > 0) {
        echo "Sở thích của bạn là: <br>";
        foreach($sothich as $item) {
            echo "- " . $item . "<br>";
        }
    } else {
        echo "Bạn chưa chọn sở thích nào";
    }
}
?>
```

12. Hiển Thị Dữ Liệu Với PHP

H

Thu thập dữ liệu

Lấy dữ liệu từ $_POST, $_GET hoặc $_REQUEST

- $_POST: Nhận dữ liệu từ form với method="post"
- $_GET: Nhận dữ liệu từ URL hoặc form với method="get"
- $_REQUEST: Nhận dữ liệu từ cả $_POST và $_GET

Xử lý dữ liệu

Lọc, kiểm tra và định dạng dữ liệu

- Sử dụng filter_var() để lọc dữ liệu
- Dùng isset() và empty() để kiểm tra tồn tại
- Chuyển đổi kiểu dữ liệu nếu cần thiết
- Xác thực dữ liệu trước khi xử lý

Hiển thị kết quả

Đưa dữ liệu vào HTML hoặc lưu vào database

- Sử dụng echo, print để hiển thị trực tiếp
- printf(), sprintf() để định dạng chuỗi
- Kết hợp với HTML để tạo giao diện
- Hoặc lưu vào database với MySQLi/PDO

```php
<?php
// Ví dụ đơn giản về quy trình xử lý dữ liệu
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    // Bước 1: Thu thập dữ liệu
    $ten = $_POST['ten'];
    $email = $_POST['email'];

    // Bước 2: Xử lý dữ liệu
    $ten = htmlspecialchars(trim($ten));
    $email = filter_var($email, FILTER_SANITIZE_EMAIL);

    if (empty($ten) || !filter_var($email, FILTER_VALIDATE_EMAIL)) {
        echo "Dữ liệu không hợp lệ!";
    } else {
        // Bước 3: Hiển thị kết quả
        echo "Xin chào " . $ten . "!<br>";
        echo "Email của bạn: " . $email;
    }
}
?>
```

Lưu ý: Bước xử lý dữ liệu là rất quan trọng để ngăn chặn các lỗ hổng bảo mật như SQL Injection hoặc Cross-site Scripting (XSS).

13. Phương Thức Hiển Thị Dữ Liệu

echo và print

```
<?php
$ten = $_POST['ten'];
echo "Xin chào " . $ten;
// hoặc
print "Xin chào " . $ten;
?>
```

echo nhận nhiều tham số (phân tách bằng dấu phẩy), print chỉ nhận một tham số.

```
<?php
// Nhiều tham số với echo
echo "Xin chào ", $ten, "! ", "Chào mừng bạn.";

// Với HTML
echo "<div class='thong-bao'>Xin chào $ten</div>";
?>
```

Cả hai đều hiển thị biến, chuỗi và HTML. Khi sử dụng trong HTML:

```
<div>
    <?php echo "Xin chào $ten"; ?>
    <p>Nội dung HTML thường</p>
    <?php echo "Tuổi: $tuoi"; ?>
</div>
```

printf và sprintf

```
<?php
$ten = $_POST['ten'];
$tuoi = $_POST['tuoi'];
printf("Tôi tên là %s, năm nay %d tuổi",
    $ten, $tuoi);

$chuoi = sprintf("Tôi tên là %s,
    năm nay %d tuổi", $ten, $tuoi);
echo $chuoi;
?>
```

Định dạng chuỗi trước khi hiển thị. Ký tự đặc biệt:

- **%s** - Chuỗi
- **%d** - Số nguyên
- **%f** - Số thực
- **%.2f** - Số thực với 2 chữ số thập phân

```
<?php
// Định dạng số thập phân
$tien = 1500000.75;
printf("Tổng tiền: %.2f VND", $tien);
// Kết quả: Tổng tiền: 1500000.75 VND

// Định dạng tiền tệ
setlocale(LC_MONETARY, 'vi_VN');
$tien_dinh_dang = money_format('%i', $tien);
echo $tien_dinh_dang;
?>
```

Các phương thức hiển thị khác

var_dump() và print_r()

```
<?php
$thong_tin = [
    'ten' => 'Nguyễn Văn A',
    'tuoi' => 30,
    'dia_chi' => 'Hà Nội'
];

// Hiển thị chi tiết về biến
var_dump($thong_tin);

// Hiển thị cấu trúc dễ đọc hơn
print_r($thong_tin);
?>
```

Hai hàm này hữu ích khi debug. var_dump() hiển thị chi tiết hơn, bao gồm kiểu dữ liệu và kích thước.

Sử dụng vòng lặp để hiển thị dữ liệu

```
<?php
$sinh_vien = [
    ['ten' => 'Nguyễn Văn A', 'diem' => 8.5],
    ['ten' => 'Trần Thị B', 'diem' => 9.0],
    ['ten' => 'Lê Văn C', 'diem' => 7.5]
];
?>

<table border="1">
    <tr><th>Họ Tên</th><th>Điểm</th></tr>
    <?php foreach($sinh_vien as $sv): ?>
    <tr>
        <td><?php echo $sv['ten']; ?></td>
        <td><?php echo $sv['diem']; ?></td>
    </tr>
    <?php endforeach; ?>
</table>
```

Luôn lọc dữ liệu người dùng trước khi hiển thị để tránh lỗ hổng XSS:

```
<?php
// Lọc dữ liệu trước khi hiển thị
$input = $_POST['comment'];
$safe_input = htmlspecialchars($input, ENT_QUOTES, 'UTF-8');
echo $safe_input;
?>
```

14. Hiển Thị Dữ Liệu Trong HTML

PHP cung cấp nhiều cách để hiển thị dữ liệu người dùng từ form HTML vào trang web. Dưới đây là một ví dụ cơ bản về cách xử lý và hiển thị thông tin từ form:

```
<?php
if(isset($_POST['submit'])) {
 $ten = $_POST['ten'];
 $email = $_POST['email'];
 $sdt = $_POST['sdt'];
?>

<div class="ket-qua">
 <h2>Thông tin đã nhập:</h2>
 <table>
 <tr>
 <td>Họ tên:</td>
 <td><?php echo $ten; ?></td>
 </tr>
 <tr>
 <td>Email:</td>
 <td><?php echo $email; ?></td>
 </tr>
 <tr>
 <td>Số điện thoại:</td>
 <td><?php echo $sdt; ?></td>
 </tr>
 </table>
</div>

<?php } ?>
```

Giải thích Mã

Đoạn mã trên thực hiện các bước sau:

1. Kiểm tra xem form đã được gửi chưa bằng cách kiểm tra biến $_POST['submit']
2. Thu thập dữ liệu từ form thông qua biến $_POST
3. Hiển thị dữ liệu bằng cách nhúng PHP vào HTML

Form HTML Tương Ứng

```
<form method="post" action="">
  <div>
    <label for="ten">Họ tên:</label>
    <input type="text" id="ten" name="ten" required>
  </div>
  <div>
    <label for="email">Email:</label>
    <input type="email" id="email" name="email" required>
  </div>
  <div>
    <label for="sdt">Số điện thoại:</label>
    <input type="tel" id="sdt" name="sdt">
  </div>
  <div>
    <button type="submit" name="submit">Gửi</button>
  </div>
</form>
```

Phương Pháp Hiển Thị Khác

Sử dụng biến trong chuỗi

```
<?php
// Cách 1: Dùng dấu ngoặc kép
echo "Xin chào $ten!";

// Cách 2: Dùng cú pháp nối chuỗi
echo 'Địa chỉ email: ' . $email;
?>
```

Hiển thị thông qua mảng

```
<?php
// Tạo mảng từ dữ liệu form
$thongTin = [
  'Họ tên' => $ten,
  'Email' => $email,
  'Số điện thoại' => $sdt
];

// Hiển thị mảng
foreach ($thongTin as $key => $value) {
  echo "<p>$key: $value</p>";
}
?>
```

Xử Lý Dữ Liệu Trước Khi Hiển Thị

Trước khi hiển thị dữ liệu nhập từ người dùng, cần thực hiện xử lý để đảm bảo an toàn:

```
<?php
// Làm sạch dữ liệu
$ten = trim(htmlspecialchars($_POST['ten']));
$email = filter_var($_POST['email'], FILTER_SANITIZE_EMAIL);
$sdt = preg_replace('/[^0-9]/', '', $_POST['sdt']);

// Kiểm tra dữ liệu
if (filter_var($email, FILTER_VALIDATE_EMAIL)) {
  echo "Email hợp lệ";
} else {
  echo "Email không hợp lệ";
}
?>
```

15. Form Tự Xử Lý

Form tự xử lý (self-processing form) là form HTML được xử lý bởi cùng một file PHP. Khi người dùng gửi form, dữ liệu được gửi lại đến chính trang hiện tại thay vì một trang xử lý riêng biệt.

Ví dụ cơ bản:

```
<?php
$message = '';
if ($_SERVER["REQUEST_METHOD"] == "POST") {
 $ten = htmlspecialchars($_POST['ten']);
 $message = "Xin chào, " . $ten;
}
?>

<!DOCTYPE html>
<html>
<head>
 <title>Form Tự Xử Lý</title>
</head>
<body>
 <h2>Form Liên Hệ</h2>

 <?php if ($message): ?>
 <div><?php echo $message; ?></div>
 <?php endif; ?>

 <form method="post" action="<?php echo htmlspecialchars($_SERVER["PHP_SELF"]); ?>">
 <label for="ten">Họ tên:</label>
 <input type="text" id="ten" name="ten" required>
 <input type="submit" value="Gửi">
 </form>
</body>
</html>
```

Giải thích code:

Phần PHP:

- `$_SERVER["REQUEST_METHOD"]`: Kiểm tra nếu form được gửi bằng phương thức POST
- `htmlspecialchars()`: Chuyển đổi ký tự đặc biệt thành entities HTML để ngăn chặn tấn công XSS
- `$_POST['ten']`: Lấy giá trị từ trường input có name="ten"
- `$_SERVER["PHP_SELF"]`: Tham chiếu đến file hiện tại, giúp form gửi dữ liệu về chính nó

Phần HTML:

- `method="post"`: Xác định phương thức gửi dữ liệu form
- `action="..."`: Chỉ định nơi dữ liệu form sẽ được gửi đến
- `<?php if ($message): ?>`: Điều kiện hiển thị thông báo chỉ khi có giá trị
- `required`: Thuộc tính HTML5 yêu cầu trường input phải được điền trước khi gửi

Ví dụ nâng cao hơn với xác thực form:

```
<?php
$tenErr = $emailErr = "";
$ten = $email = $message = "";

// Kiểm tra khi form được gửi đi
if ($_SERVER["REQUEST_METHOD"] == "POST") {
  // Xác thực tên
  if (empty($_POST["ten"])) {
    $tenErr = "Họ tên là bắt buộc";
  } else {
    $ten = htmlspecialchars($_POST["ten"]);
    // Kiểm tra tên chỉ chứa chữ cái và khoảng trắng
    if (!preg_match("/^[a-zA-ZÀ-ỹ ]*$/u", $ten)) {
      $tenErr = "Chỉ cho phép chữ cái và khoảng trắng";
    }
  }

  // Xác thực email
  if (empty($_POST["email"])) {
    $emailErr = "Email là bắt buộc";
  } else {
    $email = htmlspecialchars($_POST["email"]);
    // Kiểm tra email hợp lệ
    if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
      $emailErr = "Định dạng email không hợp lệ";
    }
  }

  // Nếu không có lỗi, hiển thị thông báo chào mừng
  if (empty($tenErr) && empty($emailErr)) {
    $message = "Xin chào, " . $ten . "! Chúng tôi sẽ liên hệ qua email: " . $email;
  }
}
?>
```

16. Giữ Giá Trị Form Sau Khi Submit

```
<?php
$ten = $email = $tuoi = $gioiTinh = $soThich = $quocTich = '';
$soThichArr = [];

if ($_SERVER["REQUEST_METHOD"] == "POST") {
    // Lấy và lưu trữ giá trị từ form
    $ten = $_POST['ten'] ?? '';
    $email = $_POST['email'] ?? '';
    $tuoi = $_POST['tuoi'] ?? '';
    $gioiTinh = $_POST['gioi_tinh'] ?? '';
    $soThichArr = $_POST['so_thich'] ?? [];
    $quocTich = $_POST['quoc_tich'] ?? '';

    // Xử lý dữ liệu form...
    // Kiểm tra lỗi, lưu vào database, etc.
}

// Hàm kiểm tra trạng thái selected cho select box
function isSelected($value, $selected) {
    return ($value === $selected) ? 'selected' : '';
}

// Hàm kiểm tra trạng thái checked cho checkboxes
function isChecked($value, $array) {
    return in_array($value, $array) ? 'checked' : '';
}
?>

<form method="post" action="<?php echo htmlspecialchars($_SERVER["PHP_SELF"]); ?>">
    <div>
        <label>Họ tên:</label>
        <input type="text" name="ten" value="<?php echo htmlspecialchars($ten); ?>">
    </div>

    <div>
        <label>Email:</label>
        <input type="email" name="email" value="<?php echo htmlspecialchars($email); ?>">
    </div>

    <div>
        <label>Tuổi:</label>
        <input type="number" name="tuoi" value="<?php echo htmlspecialchars($tuoi); ?>">
    </div>

    <div>
        <label>Giới tính:</label>
        <input type="radio" name="gioi_tinh" value="nam" <?php echo ($gioiTinh === 'nam') ? 'checked' : ''; ?
>> Nam
        <input type="radio" name="gioi_tinh" value="nu" <?php echo ($gioiTinh === 'nu') ? 'checked' : ''; ?>>
Nữ
        <input type="radio" name="gioi_tinh" value="khac" <?php echo ($gioiTinh === 'khac') ? 'checked' : ''; ?
>> Khác
    </div>

    <div>
        <label>Sở thích:</label>
        <input type="checkbox" name="so_thich[]" value="doc_sach" <?php echo isChecked('doc_sach',
$soThichArr); ?>> Đọc sách
        <input type="checkbox" name="so_thich[]" value="the_thao" <?php echo isChecked('the_thao',
$soThichArr); ?>> Thể thao
        <input type="checkbox" name="so_thich[]" value="am_nhac" <?php echo isChecked('am_nhac',
$soThichArr); ?>> Âm nhạc
    </div>

    <div>
        <label>Quốc tịch:</label>
        <select name="quoc_tich">
            <option value="">Chọn quốc tịch</option>
            <option value="vn" <?php echo isSelected('vn', $quocTich); ?>>Việt Nam</option>
            <option value="us" <?php echo isSelected('us', $quocTich); ?>>Hoa Kỳ</option>
            <option value="jp" <?php echo isSelected('jp', $quocTich); ?>>Nhật Bản</option>
        </select>
    </div>

    <input type="submit" value="Gửi">
</form>
```

Kỹ thuật xử lý các loại input

Đối với các loại input phức tạp:

Checkbox

Cần sử dụng mảng và kiểm tra xem giá trị có tồn tại trong mảng đã submit không:

```
isChecked('value', $arrayOfValues)
```

Radio buttons

So sánh giá trị hiện tại với giá trị đã chọn:

```
($value === $selectedValue) ? 'checked' : ''
```

Select boxes

Kiểm tra option nào đã được chọn:

```
isSelected('option_value', $selectedValue)
```

Xử lý an toàn dữ liệu

Khi hiển thị lại dữ liệu, luôn sử dụng `htmlspecialchars()` để tránh các lỗ hổng XSS:

```
value="<?php echo htmlspecialchars($value); ?>"
```

17. Redirect Sau Khi Xử Lý Form

Sau khi xử lý form thành công, việc chuyển hướng (redirect) người dùng đến trang khác để tránh việc tự gửi lại form khi làm mới trang.

Ví dụ cơ bản

```
<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    // Xử lý dữ liệu form
    $ten = $_POST['ten'];
    $email = $_POST['email'];

    // Lưu vào database hoặc xử lý dữ liệu
    // ...

    // Chuyển hướng sau khi xử lý thành công
    header("Location: thank_you.php");
    exit(); // Kết thúc script để đảm bảo redirect hoạt động
}
?>
```

Các loại redirect phổ biến

Redirect 302 (Tạm thời)

Đây là loại redirect mặc định khi sử dụng hàm header(). Thích hợp cho hầu hết các trường hợp chuyển hướng sau khi xử lý form.

```
header("Location: thank_you.php");
exit();
```

Redirect 301 (Vĩnh viễn)

Sử dụng khi URL đích là vĩnh viễn, giúp tối ưu SEO.

```
header("HTTP/1.1 301 Moved Permanently");
header("Location: new_page.php");
exit();
```

Redirect với tham số

Truyền thông tin bổ sung qua URL để hiển thị thông báo hoặc dữ liệu.

```
header("Location: thank_you.php?status=success&id=123");
exit();
```

Lưu ý quan trọng khi sử dụng redirect

- Luôn sử dụng hàm exit() sau khi gọi header() để ngăn chặn việc thực thi code tiếp theo
- Không được xuất bất kỳ nội dung nào trước khi gọi hàm header() - kể cả khoảng trắng
- Nếu cần truyền dữ liệu phức tạp giữa các trang, hãy sử dụng session

Ví dụ sử dụng session để truyền thông báo giữa các trang:

```
<?php
session_start();

if ($_SERVER["REQUEST_METHOD"] == "POST") {
    // Xử lý form
    // ...

    // Lưu thông báo vào session
    $_SESSION['message'] = "Đăng ký thành công!";

    // Chuyển hướng
    header("Location: thank_you.php");
    exit();
}
?>

<!-- Trong trang thank_you.php -->
<?php
session_start();
if (isset($_SESSION['message'])) {
    echo $_SESSION['message'];
    unset($_SESSION['message']); // Xóa thông báo sau khi hiển thị
}
?>
```

Bài Tập 1: Form Đăng Ký

Yêu Cầu:

1. Tạo form đăng ký với các trường: Họ tên, Email, Mật khẩu, Xác nhận mật khẩu, Giới tính, Sở thích (ít nhất 5 lựa chọn)
2. Thêm các trường bổ sung: Ngày sinh (sử dụng input type date), Địa chỉ, Số điện thoại, Ảnh đại diện (file upload)
3. Kiểm tra dữ liệu đầu vào: Email hợp lệ (định dạng xxx@xxx.xxx), mật khẩu khớp nhau và có ít nhất 8 ký tự, bao gồm chữ hoa, chữ thường và số
4. Hiển thị thông báo lỗi chi tiết cho từng trường không hợp lệ
5. Lưu thông tin đăng ký vào session hoặc cơ sở dữ liệu
6. Hiển thị trang xác nhận thông tin đăng ký thành công
7. Thêm chức năng gửi email xác nhận đến địa chỉ email đã đăng ký

Phương Pháp Kiểm Tra Dữ Liệu:

- Sử dụng HTML5 validation attributes (required, pattern, min, max)
- Kết hợp JavaScript để kiểm tra client-side trước khi gửi form
- Sử dụng PHP để kiểm tra server-side, đảm bảo an toàn dữ liệu

Bài Tập 2: Máy Tính Đơn Giản

Yêu Cầu

Tạo một máy tính đơn giản với hai ô input số và các nút cho phép thực hiện các phép tính cộng, trừ, nhân, chia.

- Thiết kế giao diện người dùng rõ ràng, trực quan
- Cho phép nhập hai số với các trường input
- Cung cấp các nút riêng biệt cho các phép toán (+, -, *, /)
- Hiển thị kết quả ngay sau khi nhấn nút phép toán

Kỹ Năng

Xử lý nhiều nút submit trong cùng một form, kiểm tra dữ liệu đầu vào là số, xử lý các trường hợp đặc biệt (chia cho 0).

- Kiểm tra và xác thực dữ liệu người dùng nhập vào
- Xử lý lỗi và hiển thị thông báo phù hợp
- Định dạng số thập phân và hiển thị đúng

Các Bước Thực Hiện

1. Thiết kế HTML form với các trường input và nút phép toán
2. Viết PHP để xử lý dữ liệu form
3. Kiểm tra dữ liệu đầu vào và thực hiện phép tính
4. Hiển thị kết quả và thông báo lỗi nếu có
5. Tối ưu giao diện người dùng

Kết Quả Mong Đợi

Sau khi hoàn thành, bạn sẽ có một máy tính đơn giản có thể:

- Nhận input từ người dùng và xác thực dữ liệu
- Thực hiện được các phép tính cơ bản
- Xử lý các trường hợp đặc biệt (chia cho 0)
- Hiển thị kết quả hoặc thông báo lỗi

Điểm nâng cao: Thêm lịch sử tính toán và khả năng xóa lịch sử.

Bài Tập 3: Upload Và Hiển Thị Ảnh

Upload file là một kỹ năng thiết yếu cho mọi web developer. Bài tập này giúp bạn nắm vững cách xử lý dữ liệu đa phương tiện từ người dùng, đảm bảo tính bảo mật và hiệu suất cho ứng dụng web.

Yêu Cầu:

1. Tạo form cho phép người dùng tải lên một tệp ảnh
2. Kiểm tra đó có phải là tệp ảnh hợp lệ không (jpg, png, gif)
3. Giới hạn kích thước tệp (không quá 2MB)
4. Lưu ảnh vào thư mục trên server
5. Hiển thị ảnh đã tải lên