

**HỌC VIỆN KỸ THUẬT MẬT MÃ**  
**KHOA ATTT**

**KẾT NỐI VÀ TRUY VẤN CSDL, PHIÊN VÀ  
COOKIE**

# NỘI DUNG

**1. Kết nối và truy vấn CSDL**

**2. Phiên và cookie**



# KẾT NỐI VÀ TRUY VẤN CSDL

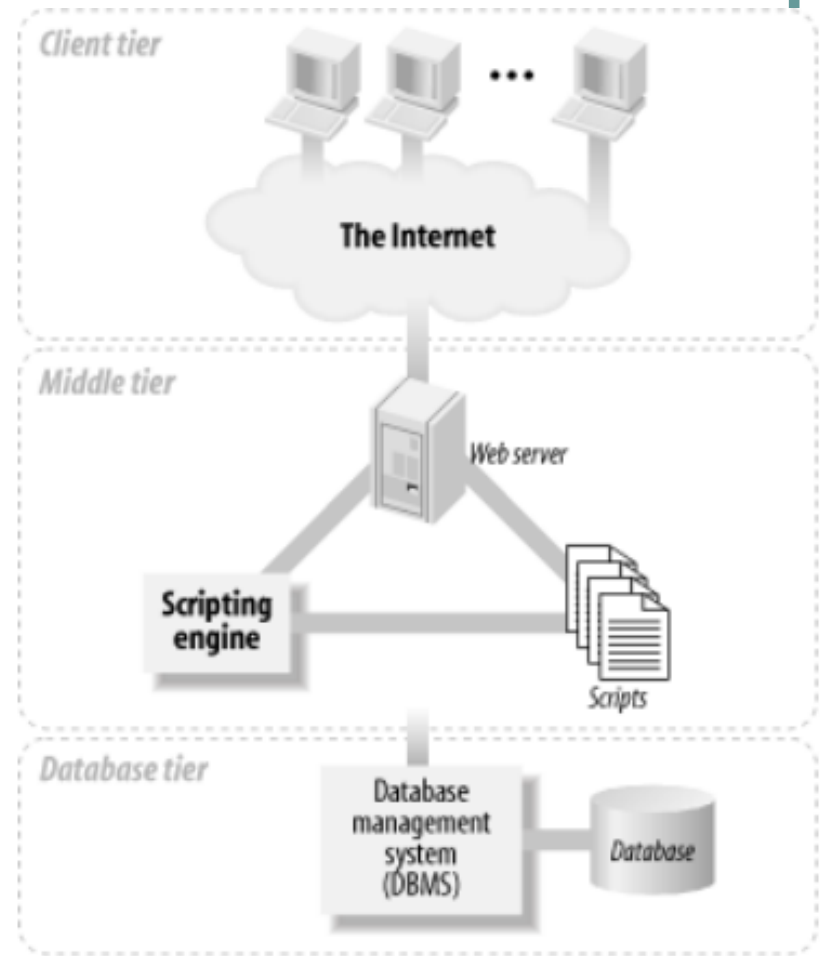
**1. Mô hình ba tầng**

**2. Thao tác với cơ sở dữ liệu**



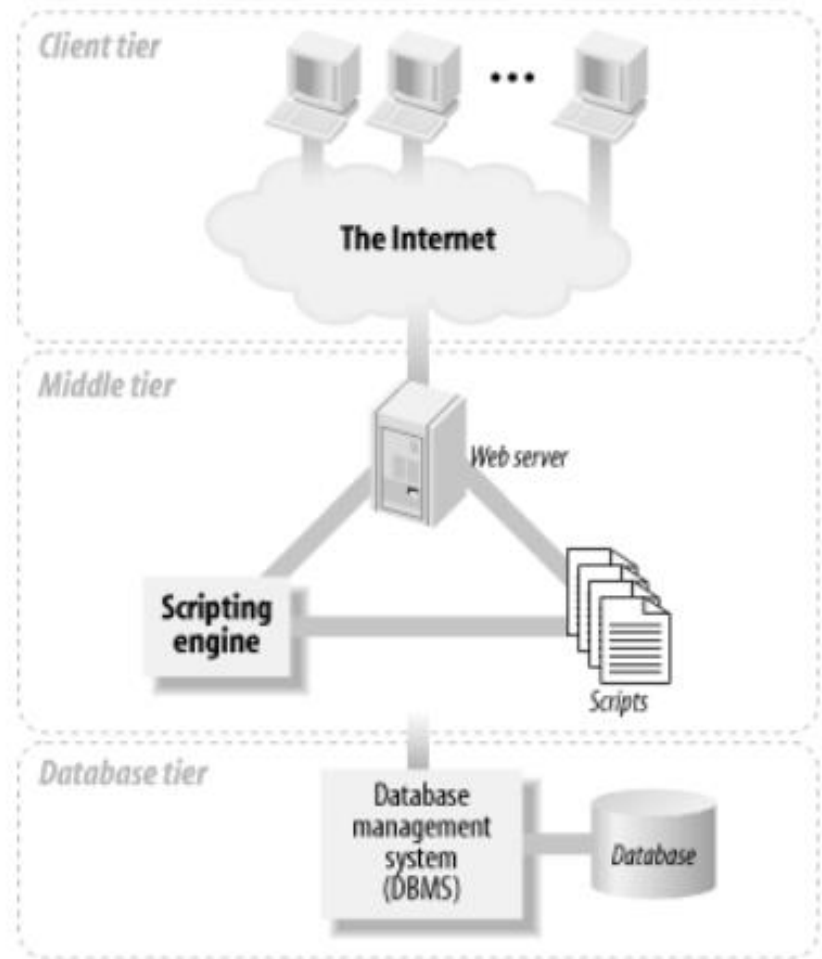
# Mô hình ba tầng

- **Tầng khách:** trình diễn và tương tác với người dùng
- **Tầng giữa:** thực hiện các logic của ứng dụng
- **Tầng CSDL:** bao gồm hệ quản trị CSDL, CSDL của ứng dụng

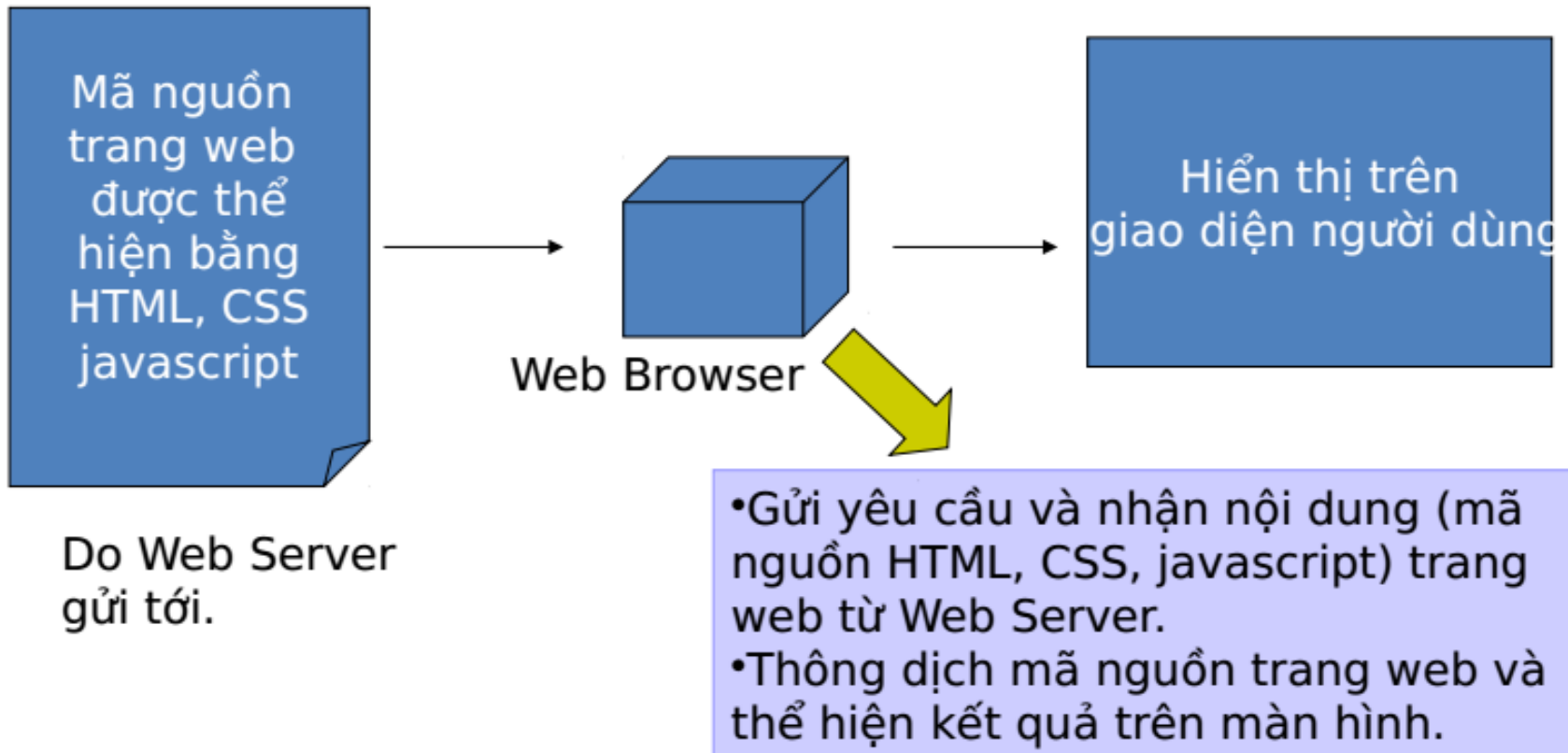


# Tầng khách

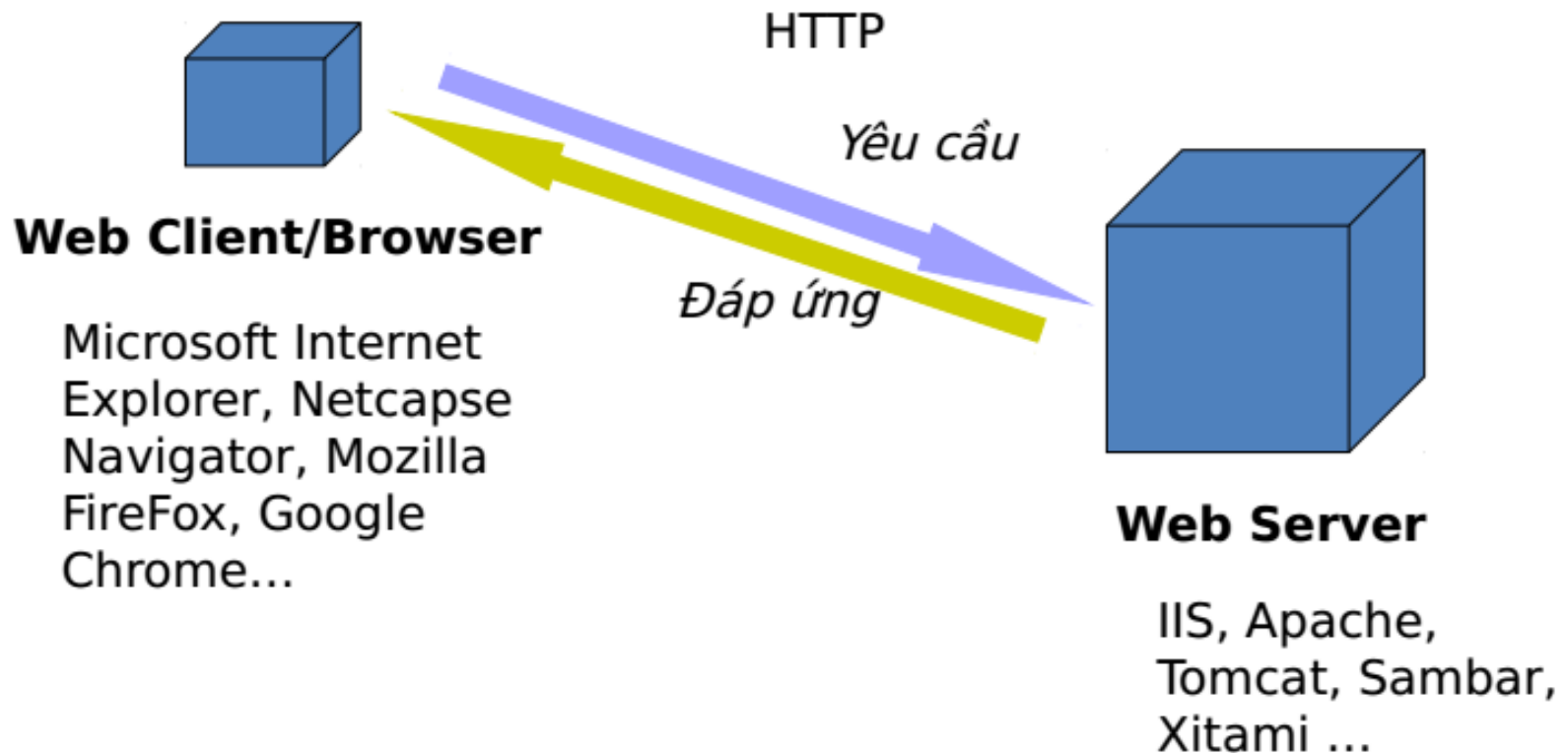
- Thường là trình duyệt web, nhưng có thể là web crawler, web proxy
- Gửi HTTP Request lên Web server và nhận HTTP Response từ Web server
- Trình diễn nội dung web và tương tác với người dùng



# Web client



# Web client/server



# Tầng giữa

- Các trang web/kịch bản được viết bằng các ngôn ngữ kịch bản khác nhau và HTML
- Web server
  - Nhận yêu cầu từ web client
  - Yêu cầu các chương trình dịch chạy các kịch bản/trang động để sinh ra nội dung web (HTML, javascript, css)
  - Gửi nội dung web cho web client
- Chương trình dịch
  - Chạy các kịch bản được viết từ các ngôn ngữ khác nhau (PHP, ASP, ASP.NET, C#, Java, ...) để sinh ra nội dung web
  - Gửi truy vấn đến Hệ quản trị CSDL (tầng dưới) và nhận dữ liệu nếu cần



# Tầng CSDL

- CSDL cho ứng dụng
- Hệ quản trị CSDL quản lý CSDL của ứng dụng
  - Nhận truy vấn từ các chương trình dịch tầng giữa, thao tác CSDL và trả kết quả cho chương trình yêu cầu

# Ví dụ mẫu

- Quản lý sinh viên với các chức năng sau:
  - Hiển thị danh sách sinh viên. Thực hiện phân trang danh sách. Cho người dùng tùy chỉnh số dòng/trang
  - Thêm, cập nhật, xóa sinh viên.

# KẾT NỐI VÀ TRUY VẤN CSDL

**1.** Mô hình ba tầng

**2.** Thao tác với cơ sở dữ liệu



# Kết nối PHP - MySQL

- Kết nối PHP & MySQL là bước cơ bản khi muốn thao tác đến CSDL.
- PHP sử dụng 2 thư viện là MySQLi và PDO.
  - MySQLi có cấu trúc và cách viết tương tự như MySQL. MySQLi có 2 lựa chọn cách viết khác nhau:
    - Theo kiểu thủ tục: viết giống như MySQL chỉ khác là thay đổi mysql thành mysqli.
    - Theo kiểu hướng đối tượng
  - PDO (PHP Data Object) là một thư viện có sẵn trong php giúp tương tác với các hệ quản trị CSDL khác như SQLite , Microsoft SQL Server, PostgreSQL...

# Kết nối MySQLi

- Kiểu hướng đối tượng

```
<?php
$username = "user_tintuc"; // Khai báo username
$password = "123456";      // Khai báo password
$server   = "localhost";  // Khai báo server
$dbname   = "tintuc";      // Khai báo database

// Kết nối database tintuc
$connect = new mysqli($server, $username, $password, $dbname);

//Nếu kết nối bị lỗi thì xuất báo lỗi và thoát.
if ($connect->connect_error) {
    die("Không kết nối :" . $connect->connect_error);
    exit();
}

echo "Khi kết nối thành công sẽ tiếp tục dòng code bên dưới đây."
?>
```

# Kết nối MySQLi

- Kiểu thủ tục

```
<?php
$username = "user_tintuc"; // Khai báo username
$password = "123456";      // Khai báo password
$server    = "localhost";  // Khai báo server
$dbname     = "tintuc";     // Khai báo database

// Kết nối database tintuc
$connect = mysqli_connect($server, $username, $password, $dbname);

//Nếu kết nối bị lỗi thì xuất báo lỗi và thoát.
if (!$connect) {
    die("Không kết nối : " . mysqli_connect_error());
    exit();
}
echo "Khi kết nối thành công sẽ tiếp tục dòng code bên dưới đây."
?>
```

# PHP insert dữ liệu vào MySQL

- Sử dụng các câu lệnh MySQL để thêm record (hàng) cho table, các câu lệnh này được viết bên trong PHP.
- Các bước thực hiện:
  - Kết nối database và table.
  - Xử lý dữ liệu.
  - Đóng database.
- Cấu trúc insert:

```
<?php  
INSERT INTO tên_table (tên_cột1, tên_cột2 tên_cột3,...)  
VALUES (giá_trị1, giá_trị2, giá_trị3,...)  
?>
```

# PHP insert dữ liệu vào MySQL

- Kiểu hướng đối tượng:

```
<?php
$username = "user_tintuc"; // Khai báo username
$password = "123456";      // Khai báo password
$server   = "localhost";  // Khai báo server
$dbname   = "tintuc";      // Khai báo database

// Kết nối database tintuc
$connect = new mysqli($server, $username, $password, $dbname);

//Nếu kết nối bị lỗi thì xuất báo lỗi và thoát.
if ($connect->connect_error) {
    die("Không kết nối : " . $connect->connect_error);
    exit();
}

//Code xử lý, insert dữ liệu vào table
$sql = "INSERT INTO tin_xahoi (title, date, description, content)
VALUES ('Tin hot', '2016-10-24', 'Đây là mô tả cho tin hot', 'Đây là nội dung của tin hot')";

if ($connect->query($sql) === TRUE) {
    echo "Thêm dữ liệu thành công";
} else {
    echo "Error: " . $sql . "<br>" . $connect->error;
}

//Đóng database
$connect->close();
?>
```



# PHP insert dữ liệu vào MySQL

- Kiểu thủ tục:

```
<?php
$username = "user_tintuc"; // Khai báo username
$password = "123456";      // Khai báo password
$server = "localhost";    // Khai báo server
$dbname = "tintuc";        // Khai báo database

// Kết nối database tintuc
$connect = mysqli_connect($server, $username, $password, $dbname);

//Nếu kết nối bị lỗi thì xuất báo lỗi và thoát.
if (!$connect) {
    die("Không kết nối : " . mysqli_connect_error());
    exit();
}

//Code xử lý, insert dữ liệu vào table
$sql = "INSERT INTO tin_xahoi (title, date, description, content)
VALUES ('Tin hot', '2016-10-24', 'Đây là mô tả cho tin hot', 'Đây là nội dung của tin hot')";

if (mysqli_query($connect, $sql)) {
    echo "Thêm dữ liệu thành công";
} else {
    echo "Error: " . $sql . "<br>" . mysqli_error($connect);
}

//Đóng database
mysqli_close($connect);
?>
```

# PHP xem dữ liệu MySQL

- Khi đã có dữ liệu thì việc kế tiếp phải xem coi dữ liệu đó hiển thị như thế nào, hoặc dữ liệu hiển thị cho người dùng thấy.
- Các bước thực hiện:
  - Kết nối database và table.
  - Xử lý dữ liệu.
  - Đóng database.
- Cấu trúc view:

```
<?php  
SELECT tên_cột FROM tên_table  
?>
```

# PHP xem dữ liệu MySQL

- Kiểu hướng đối tượng:

```
<?php
$username = "user_tintuc"; // Khai báo username
$password = "123456";      // Khai báo password
$server   = "localhost";   // Khai báo server
$dbname    = "tintuc";      // Khai báo database

// Kết nối database tintuc
$connect = new mysqli($server, $username, $password, $dbname);

//Nếu kết nối bị lỗi thì xuất báo lỗi và thoát.
if ($connect->connect_error) {
    die("Không kết nối :" . $connect->connect_error);
    exit();
}

//Code xử lý, insert dữ liệu vào table
$sql      = "SELECT * FROM tin_xahoi";
$ket_qua = $connect->query($sql);

//Nếu kết quả kết nối không được thì xuất báo lỗi và thoát
if (!$ket_qua) {
    die("Không thể thực hiện câu lệnh SQL: " . $connect->connect_error);
    exit();
}
```

```
//Dùng vòng lặp while truy xuất các phần tử trong table
while ($row = $ket_qua->fetch_array(MYSQLI_ASSOC)) {
    echo "<p>ID: " . $row['id'] . "</p>";
    echo "<p>Tiêu đề: " . $row['title'] . "</p>";
    echo "<p>Ngày: " . $row['date'] . "</p>";
    echo "<p>Mô tả: " . $row['description'] . "</p>";
    echo "<p>Nội dung: " . $row['content'] . "</p>";
    echo "<hr>";
}

//Đóng kết nối database tintuc
$connect->close();
?>
```

# PHP xem dữ liệu MySQL

- Kiểu thủ tục:

```
<?php
$username = "user_tintuc"; // Khai báo username
$password = "123456";      // Khai báo password
$server   = "localhost";   // Khai báo server
$dbname    = "tintuc";      // Khai báo database

// Kết nối database tintuc
$connect = mysqli_connect($server, $username, $password, $dbname);

//Nếu kết nối bị lỗi thì xuất báo lỗi và thoát.
if (!$connect) {
    die("Không kết nối :". mysqli_connect_error());
    exit();
}
```

```
//Code xử lý, truy vấn xem dữ liệu
$sql = "SELECT * FROM tin_xahoi";
$ket_qua = mysqli_query($connect,$sql);

if (!$ket_qua) {
    die("Không thể thực hiện câu lệnh SQL: " . mysqli_error($connect));
    exit();
}

//Dùng vòng lặp while truy xuất các phần tử trong table
while ($row = mysqli_fetch_array($ket_qua)) {
    echo "<p>ID: " . $row['id'] . "</p>";
    echo "<p>Tiêu đề: " . $row['title'] . "</p>";
    echo "<p>Ngày: " . $row['date'] . "</p>";
    echo "<p>Mô tả: " . $row['description'] . "</p>";
    echo "<p>Nội dung: " . $row['content'] . "</p>";
    echo "<hr>";
}

//Đóng database
mysqli_close($connect);
?>
```

# PHP cập nhật dữ liệu vào MySQL

- Một khi muốn thay đổi dữ liệu đã được thêm trước đó, thì cần cập nhật lại dữ liệu với câu lệnh UPDATE có trong MySQL.
- Các bước thực hiện:
  - Kết nối database và table.
  - Xác định record (hàng) cần cập nhật.
  - Xử lý dữ liệu.
  - Đóng database.
- Cấu trúc update:

```
<?php
UPDATE tên_table
SET tên_cột1=giá_trị1, tên_cột2=giá_trị2,...
WHERE tên_cột=giá_trị_cột_cần_update
?>
```

# PHP cập nhật dữ liệu vào MySQL

- Kiểu hướng đối tượng:

```
<?php
$username = "user_tintuc"; // Khai báo username
$password = "123456";      // Khai báo password
$server   = "localhost";   // Khai báo server
$dbname   = "tintuc";      // Khai báo database

// Kết nối database tintuc
$connect = new mysqli($server, $username, $password, $dbname);

//Nếu kết nối bị lỗi thì xuất báo lỗi và thoát.
if ($connect->connect_error) {
    die("Không kết nối :" . $connect->connect_error);
    exit();
}

//Code xử lý, update dữ liệu vào table dựa theo điều kiện WHERE tại id = 1
$sql = "UPDATE tin_xahoi SET title='Học có trẻ đầu', content='Đây là nội dung của bài viết Học có trẻ đầu' WHERE id=1";

//Nếu kết quả kết nối không được thì xuất báo lỗi và thoát
if ($connect->query($sql) === TRUE) {
    echo "Dữ liệu đã được update";
} else {
    echo "Lỗi update: " . $connect->error;
}

//Đóng kết nối database tintuc
$connect->close();
?>
```

# PHP cập nhật dữ liệu vào MySQL

- Kiểu thủ tục:

```
<?php
$username = "user_tintuc"; // Khai báo username
$password = "123456";      // Khai báo password
$server   = "localhost";   // Khai báo server
$dbname    = "tintuc";      // Khai báo database

// Kết nối database tintuc
$connect = mysqli_connect($server, $username, $password, $dbname);

//Nếu kết nối bị lỗi thì xuất báo lỗi và thoát.
if (!$connect) {
    die("Không kết nối : " . mysqli_connect_error());
    exit();
}

//Code xử lý, update dữ liệu vào table dựa theo điều kiện WHERE tại id = 1
$sql = "UPDATE tin_xahoi SET title='Học có trẻ đầu1', content='Đây là nội dung của bài viết Học có trẻ đầu' WHERE id=1";

//Nếu kết quả kết nối không được thì xuất báo lỗi và thoát
if (mysqli_query($connect, $sql)) {
    echo "Dữ liệu đã được update";
} else {
    echo "Lỗi update: " . mysqli_error($connect);
}

//Đóng database
mysqli_close($connect);
?>
```

# MySQL delete

- Khi cần xóa dữ liệu nào đó không cần thiết, sử dụng câu lệnh DELETE trong MySQL để thực hiện việc xóa.
- Các bước thực hiện:
  - Kết nối database và table.
  - Xác định record (hàng) cần xóa.
  - Xử lý dữ liệu.
  - Đóng database.
- Cấu trúc delete:

```
<?php  
DELETE FROM tên_table  
WHERE tên_cột=giá_trị_cột_cần_xóa  
?>
```



# MySQL delete

- Kiểu hướng đối tượng:

```
<?php
$username = "user_tintuc"; // Khai báo username
$password = "123456";      // Khai báo password
$server   = "localhost";   // Khai báo server
$dbname   = "tintuc";       // Khai báo database

// Kết nối database tintuc
$connect = new mysqli($server, $username, $password, $dbname);

//Nếu kết nối bị lỗi thì xuất báo lỗi và thoát.
if ($connect->connect_error) {
    die("Không kết nối :" . $connect->connect_error);
    exit();
}

//Code xử lý, xóa record dữ liệu của table dựa theo điều kiện WHERE tại id = 1
$sql = "DELETE FROM tin_xahoi WHERE id=1";

//Nếu kết quả kết nối không được thì xuất báo lỗi và thoát
if ($connect->query($sql) === TRUE) {
    echo "Dữ liệu đã được xóa";
} else {
    echo "Lỗi delete: " . $connect->error;
}

//Đóng kết nối database tintuc
$connect->close();
?>
```

# MySQL delete

- Kiểu thủ tục:

```
<?php
$username = "user_tintuc"; // Khai báo username
$password = "123456";      // Khai báo password
$server   = "localhost";   // Khai báo server
$dbname   = "tintuc";       // Khai báo database

// Kết nối database tintuc
$connect = mysqli_connect($server, $username, $password, $dbname);

//Nếu kết nối bị lỗi thì xuất báo lỗi và thoát.
if (!$connect) {
    die("Không kết nối :" . mysqli_connect_error());
    exit();
}

//Code xử lý, xóa record dữ liệu của table dựa theo điều kiện WHERE tại id = 1
$sql = "DELETE FROM tin_xahoi WHERE id=1";

//Nếu kết quả kết nối không được thì xuất báo lỗi và thoát
if (mysqli_query($connect, $sql)) {
    echo "Dữ liệu đã được xóa";
} else {
    echo "Lỗi delete: " . mysqli_error($connect);
}

//Đóng database
mysqli_close($connect);
?>
```

# PHP insert dữ liệu vào MySQL thông qua form

- Kết hợp với form để insert dữ liệu. Dữ liệu có thể là `_GET` hay `_POST`.
- Các bước thực hiện:
  - Tạo form insert dữ liệu.
  - Kết nối database và table.
  - Lấy dữ liệu post từ form
  - Xử lý dữ liệu.
  - Đóng database.

# PHP insert dữ liệu vào MySQL thông qua form

- Kiểu hướng đối tượng:

Tiêu đề:

Ngày tháng:  

Mô tả:

Nội dung:

```
<form action="" method="post">
  <table>
    <tr>
      <th>Tiêu đề:</th>
      <td><input type="text" name="title" value=""></td>
    </tr>

    <tr>
      <th>Ngày tháng:</th>
      <td><input type="date" name="date" value=""></td>
    </tr>

    <tr>
      <th>Mô tả:</th>
      <td><input type="text" name="description" value=""></td>
    </tr>

    <tr>
      <th>Nội dung:</th>
      <td><textarea cols="30" rows="7" name="content"></textarea></td>
    </tr>
  </table>
  <button type="submit">Gửi</button>
</form>
```

# PHP insert dữ liệu vào MySQL thông qua form

- Kiểu hướng đối tượng:

```
<?php
$username = "user_tintuc"; // Khai báo username
$password = "123456";      // Khai báo password
$server = "localhost";    // Khai báo server
$dbname = "tintuc";        // Khai báo database

// Kết nối database tintuc
$connect = new mysqli($server, $username, $password, $dbname);

//Nếu kết nối bị lỗi thì xuất báo lỗi và thoát.
if ($connect->connect_error) {
    die("Không kết nối : " . $connect->connect_error);
    exit();
}

//Khai báo giá trị ban đầu, nếu không có thì khi chưa submit câu lệnh insert sẽ báo lỗi
$title = "";
$date = "";
$description = "";
$content = "";
```

```
//Lấy giá trị POST từ form vừa submit
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    if(isset($_POST["title"])) { $title = $_POST['title']; }
    if(isset($_POST["date"])) { $date = $_POST['date']; }
    if(isset($_POST["description"])) { $description = $_POST['description']; }
    if(isset($_POST["content"])) { $content = $_POST['content']; }

    //Code xử lý, insert dữ liệu vào table
    $sql = "INSERT INTO tin_xahoi (title, date, description, content)
    VALUES ('$title', '$date', '$description', '$content')";

    if ($connect->query($sql) === TRUE) {
        echo "Thêm dữ liệu thành công";
    } else {
        echo "Error: " . $sql . "<br>" . $connect->error;
    }
}

//Đóng database
$connect->close();
?>
```

# PHP insert dữ liệu vào MySQL thông qua form

- Kiểu thủ tục:

```
<?php
$username = "user_tintuc"; // Khai báo username
$password = "123456";      // Khai báo password
$server = "localhost";    // Khai báo server
$dbname = "tintuc";        // Khai báo database

// Kết nối database tintuc
$connect = mysqli_connect($server, $username, $password, $dbname);

// Nếu kết nối bị lỗi thì xuất báo lỗi và thoát.
if (!$connect) {
    die("Không kết nối : " . mysqli_connect_error());
    exit();
}

// Khai báo giá trị ban đầu, nếu không có thì khi chưa submit câu lệnh insert sẽ báo lỗi
$title = "";
$date = "";
$description = "";
$content = "";
```

```
// Lấy giá trị POST từ form vừa submit
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    if(isset($_POST["title"])) { $title = $_POST['title']; }
    if(isset($_POST["date"])) { $date = $_POST['date']; }
    if(isset($_POST["description"])) { $description = $_POST['description']; }
    if(isset($_POST["content"])) { $content = $_POST['content']; }

    // Code xử lý, insert dữ liệu vào table
    $sql = "INSERT INTO tin_xahoi (title, date, description, content)
    VALUES ('$title', '$date', '$description', '$content')";

    if (mysqli_query($connect, $sql)) {
        echo "Thêm dữ liệu thành công";
    } else {
        echo "Error: " . $sql . "<br>" . mysqli_error($connect);
    }
}

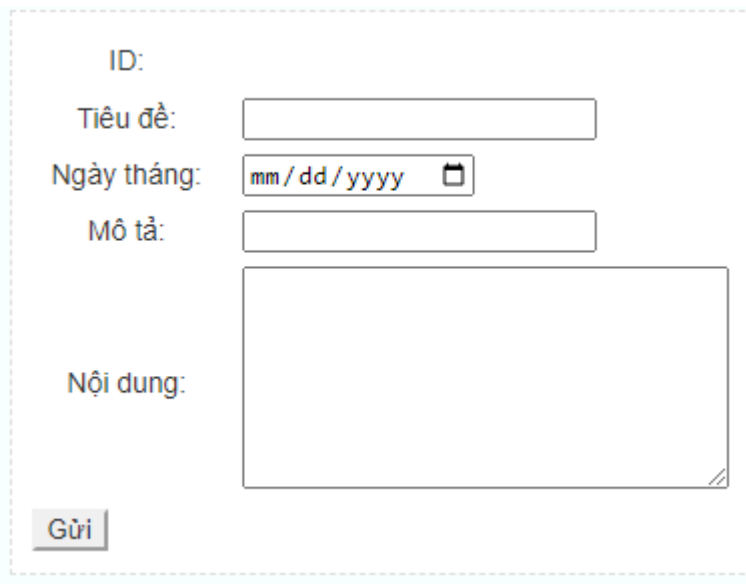
// Đóng database
mysqli_close($connect);
?>
```

# MySQL form update

- Truyền tham số xác định record (hàng) để cập nhật dữ liệu.
- Các bước thực hiện:
  - Tạo liên kết cập nhật dữ liệu từ trang view.php
  - Khi Click liên kết sẽ truyền tham số sang trang update.php
  - Dựa theo tham số này, đổ dữ liệu vào các giá trị form tương ứng trong trang update.php, submit form tới trang xử lý process.php
  - Kết nối database và table ở trang xử lý.
  - Xử lý dữ liệu dựa theo tham số được gửi.
  - Đóng database.


# MySQL form update

- Kiểu hướng đối tượng, kiểu thủ tục: tự tham khảo



ID:

Tiêu đề:

Ngày tháng:  

Mô tả:

Nội dung:



# Câu lệnh được chuẩn bị trước

- **Prepared Statement** được sử dụng khi cần thực hiện câu lệnh lặp lại nhiều lần để tăng hiệu năng
- Chuẩn bị: Tạo mẫu câu lệnh, sử dụng dấu ? làm tham số.
  - `$stmt = $db->prepare("INSERT INTO CountryLanguage VALUES (?, ?, ?, ?)");`
- Buộc các tham số
  - `$stmt->bind_param('sssd', $code, $language, $official, $percent);`
- Đặt giá trị các tham số
  - `$code = 'DEU'; $language = 'Bavarian'; $official = "F"; $percent = 11.2;`
- Thực thi
  - `$stmt->execute();`
- Lấy kết quả truy vấn
  - `$res = $stmt->get_result();`
- Đóng
  - `$stmt->close();`

## Kiểu tham số

i – số nguyên  
d – số thực  
s – chuỗi ký tự  
b - blob

# Thủ tục lưu trữ

- **Stored Procedures** được sử dụng để tăng hiệu năng
- Cập nhật  

```
PROCEDURE p(IN id_val INT) BEGIN  
  INSERT INTO test(id) VALUES(id_val);  
END;
```

```
if (!$db->query("CALL p(1)"))  
  echo "CALL failed: ". $db->error;
```

# Thủ tục lưu trữ

- Truy vấn

```
PROCEDURE p() BEGIN SELECT id FROM test; SELECT id + 1  
FROM test; END;
```

```
if (!$db->multi_query("CALL p()")) {  
    echo "CALL failed: (" . $db->errno . ") " . $mysqli->error;  
}
```

```
do {  
    if ($res = $db->store_result()) {  
        ...  
    } else {  
        if ($db->errno) {  
            echo "Store failed: (" . $db->errno . ") " . $db->error;  
        }  
    }  
} while ($db->more_results() && $db->next_result());
```

# Thủ tục lưu trữ

- Sử dụng lệnh chuẩn bị sẵn

```
PROCEDURE p() READS SQL DATA BEGIN SELECT id FROM test; SELECT id + 1 FROM test; END;
```

```
if (!$db->multi_query("CALL p()")) {  
    echo "CALL failed: (" . $db->errno . ") " . $mysqli->error;  
}
```

```
if (!$stmt = $db->prepare("CALL p()")) {  
    echo "Prepare failed: (" . $db->errno . ") " . $db->error;  
}
```

```
if (!$stmt->execute()) {  
    echo "Execute failed: (" . $stmt->errno . ") " . $stmt->error;  
}
```

```
do {  
    if ($res = $stmt->get_result()) {  
        ...  
    } else {  
        if ($stmt->errno) {  
            echo "Store failed: (" . $stmt->errno . ") " . $stmt->error;  
        }  
    }  
} while ($stmt->more_results() && $stmt->next_result());
```

# Giao tác

// tắt tự động ủy thác

```
$db->autocommit(FALSE);
```

//thực hiện nhiều lệnh trong giao tác

```
$db->query(...);
```

```
$db->query(...);
```

```
$db->query(...);
```

//thực hiện ủy thác

```
$db->commit();
```

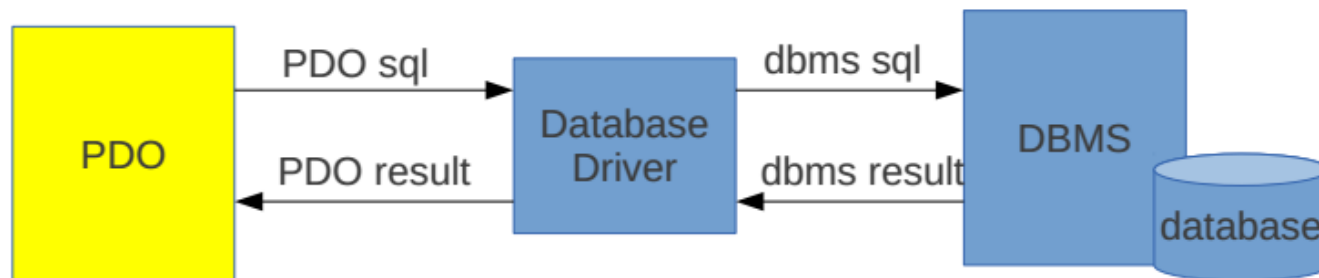
# Thao tác với CSDL trong PHP thông qua PDO

# PHP Data Objects

- PDO cung cấp một giao diện chuẩn hóa, nhất quán để thao tác với CSDL sử dụng PHP
  - Dễ chuyển đổi CSDL
  - Viết mã hiệu quả và an toàn
  - PDO có thể được sử dụng để lập trình theo cả hướng thủ tục và theo hướng đối tượng.

# Nguyên lý hoạt động

- Ứng với mỗi CSDL khác nhau PDO sẽ sử dụng các loại driver khác nhau để thao tác với CSDL.
  - Các lệnh SQL được truyền vào PDO sẽ được các driver này chuyển sang câu lệnh SQL tương ứng với CSDL đang được sử dụng.
- ⇒ Chính vì vậy khi sử dụng PDO có thể dễ dàng chuyển sang một hệ quản trị CSDL khác mà không cần phải viết lại mã.





# Kết nối CSDL

- Cú pháp:

```
$db = new PDO($dsn, $username, $password,  
[$options]);
```

- Tên nguồn dữ liệu (dsn - database source name) khác nhau với các hệ QTCSDL

- MySQL: “mysql:host=...; dbname=...”
- PostgreSQL: “pgsql:host=...; dbname=...”
- MS SQL: “sqlsrv:Server=...;Database=...”
- ...

# Xử lý ngoại lệ

```
try {  
    //Sử dụng PDO  
}  
  
// Catch any errors  
catch(PDOException $ex) {  
    echo $e->getMessage();  
    exit();  
}
```

# Các hàm trong PDO

1. `PDO::query` – Chạy câu lệnh sql (thường là select), trả về đối tượng chứa dữ liệu
2. `PDOStatement::fetch` – Lấy ra 1 dòng dữ liệu, kết quả là 1 array
3. `PDOStatement::fetchAll` – Lấy ra tất cả các dòng dữ liệu , kết quả là 1 array
4. `PDOStatement::fetchObject` – Lấy ra 1 dòng dữ liệu row and returns it as an object.
5. `PDO::exec` – Chạy câu lệnh sql, (thường là insert, update, delete) , trả về số dòng ảnh hưởng
6. `PDOStatement::rowCount` – Trả về số dòng ảnh hưởng bởi câu lệnh sql vừa chạy. Dùng để đếm số dòng dữ liệu lấy được chẳng hạn
7. `PDO::prepare` – Tạo đối tượng statement để chuẩn bị thực thi sau đó bởi hàm execute
8. `PDOStatement::execute` – thực thi statement đã tạo trước
9. `PDO::lastInsertId` – trả về id của record mới vừa chèn vào table
10. `PDO::commit` – cho tác dụng các thực thi trong transaction
11. `PDO::rollBack` – Thu hồi lại transaction

# Cập nhật CSDL

- Thực thi insert, update, delete
  - `$rows = $db->exec("UPDATE ...");`
- Lấy id (tự tăng) của bản ghi vừa được thêm
  - `$db->lastInsertId();`

# Cập nhật CSDL (tiếp)

- Thực thi insert, update, delete với lệnh chuẩn bị trước
  - `$stmt = $db->prepare("DELETE FROM table WHERE id=:id and name LIKE ?");`
  - `$stmt->bindValue(':id', $id);`
  - `$stmt->bindValue(2, "%$search%");`
  - `$stmt->execute();`
  - `$affected_rows = $stmt->rowCount();`

# Truy vấn CSDL

- Thực hiện truy vấn
  - `$stmt = $db->query('SELECT * FROM ...');`
- Duyệt các bản ghi
  - `while($row = $stmt->fetch(PDO::FETCH_ASSOC)) {  
    echo $row['field1'].' '.$row['field2'];  
}`
- Đếm số bản ghi
  - `$row_count = $stmt->rowCount();`

# Truy vấn CSDL (tiếp)

- Thực hiện truy vấn với lệnh chuẩn bị trước
  - `$stmt = $db->prepare("SELECT * FROM table WHERE id=? AND name=?");`  
`$stmt->bindValue(1, $id);`  
`$stmt->bindValue(2, $name);`  
`$stmt->execute();`
- Duyệt các bản ghi
  - `while($row = $stmt->fetch(PDO::FETCH_ASSOC)) {`  
`echo $row['field1'].' '.$row['field2'];`  
`}`
- Đếm số bản ghi
  - `$row_count = $stmt->rowCount();`

# Giao tác

```
try {  
    $db->beginTransaction();  
  
    $db->exec("SOME QUERY");  
  
    $stmt = $db->prepare("SOME OTHER QUERY?");  
    $stmt->execute(array($value));  
  
    $stmt = $db->prepare("YET ANOTHER QUERY??");  
    $stmt->execute(array($value2, $value3));  
  
    $db->commit();  
} catch(PDOException $ex) {  
    $db->rollBack();  
    echo $ex->getMessage();  
}
```



# Ví dụ Phân trang hiển thị

- Hiển thị các bản ghi ứng với trang hiện tại

- PHP

```
$firstRow = $currentPage*$rowsPerPage;  
if (!mysql_data_seek($result, $firstRow))  showerror( );  
for ( $i= 0; (($i< $rowsPerPage) &&  
($row = mysql_fetch_array($result)) ); $i++) {  
    echo "<tr><td>";  
    echo $row["thuoctinh"];  
    echo "</td></tr>";  
}
```

# Ví dụ Phân trang hiển thị

- Thêm liên kết Trang trước
- PHP

```
if ($currentPage == 0) echo "Trang trước";  
else {  
    echo "<a href = \"?currentPage=";  
    echo (currentPage-1);  
    echo "\">";  
    echo "Trang trước";  
    echo "</a>";  
}
```

# Ví dụ Phân trang hiển thị

- Thêm liên kết Trang sau

- PHP

```
$numPage =  
floor(mysql_num_rows($result)/$rowsPerPage);  
if (mysql_num_rows($result) % $rowsPerPage != 0)  
$numPage++;  
if ($currentPage == $numPage-1) echo "Trang sau";  
else {  
    echo "<a href = \"?currentPage=";  
    echo ($currentPage+1);  
    echo "\">";  
    echo "Trang sau";  
    echo "</a>";  
}
```

# Ví dụ Phân trang hiển thị

- Thêm liên kết Trang đầu
- PHP

```
if ($currentPage == 0) echo "Trang đầu";  
else {  
    echo "<a href = \"?currentPage=0\">";  
    echo "Trang đầu";  
    echo "</a>";  
}
```

# Ví dụ Phân trang hiển thị

- Thêm liên kết Trang cuối
- PHP

```
$numPage = floor(mysql_num_rows($result)/$rowsPerPage);  
if (mysql_num_rows($result) % $rowsPerPage != 0)  
$numPage++;  
if ($currentPage == $numPage-1) echo "Trang cuối";  
else {  
    echo "<a href = \"?currentPage=";  
    echo ($numPage-1);  
    echo "\">";  
    echo "Trang cuối";  
    echo "</a>";  
}
```

# Ví dụ Phân trang hiển thị

- Thêm liên kết số trang

- PHP

```
$numPage = floor(mysql_num_rows($result)/$rowsPerPage);  
if (mysql_num_rows($result) % $rowsPerPage != 0)  
$numPage++;  
for ($i = 0; $i < $numPage; $i++)  
    if ($i == $currentPage) echo ($i+1);  
    else {  
        echo "<a href = \"?currentPage=";  
        echo $i;  
        echo "\">";  
        echo (i+1);  
        echo "</a> ";  
    }  
}
```

# Kiến trúc ghi vào CSDL

Trang danh sách

-----  
Hiển thị thông báo  
(nếu có)

Hiển thị danh sách  
phân trang

Chọn thêm  
Chọn sửa  
Chọn xóa

Form  
Method="POST"  
hành động,  
ma,...

Trang cập nhật

-----  
Đọc dữ liệu đối tượng  
cần sửa từ CSDL

Form tạo giao diện  
nhập, sửa  
Method="POST"  
hành động, ma,...

CSDL

-----  
Thêm  
Sửa  
Xóa  
...

CSDL

# Thêm, xóa, cập nhật bản ghi

```
if ((mysql_query ($sql)) && (($c =  
mysql_affected_rows( )) > 0))  
    echo $c.“ bản ghi đã được cập nhật”;  
else  
    showerror( );
```

**Lưu ý:** Hàm `int mysql_insert_id([resource connection])` dùng để lấy định danh tự tăng của bản ghi vừa được thêm



# Mẫu trang tạo form cập nhật

//Nếu là cập nhật thì load giá trị bản ghi từ CSDL để đưa vào form

//Tạo form nhập

//Kiểm tra hợp thức phía client

//Yêu cầu server kiểm tra hợp thức phía server và kiểm tra trùng mã (nếu cần)

# Mẫu trang cập nhật CSDL

```
$note = "";  
$input = clean($input);  
$insertQuery = "lệnh sql được xây dựng theo $input";  
if ((mysql_query ($insertQuery, $connection)) && (($c  
= mysql_affected_rows( )) > 0)) {  
    $note = "Thông báo đã thêm/cập nhật/xóa được  
bao nhiêu bản ghi";  
} else {  
    $note = "Thông báo không thêm/cập nhật/xóa  
được";  
}  
header("Location: list.php? note=".$note);
```

# Trang hiển thị DS được chỉnh sửa để thông báo kết quả cập nhật

```
if (!empty($_GET["note"])) {  
    echo $_GET["note"];  
}  
//mã xử lý còn lại để hiển thị danh sách  
theo trang
```

# Upload tệp và lưu vào CSDL

- Tạo form upload tệp

```
<form enctype="multipart/form-data"
action="page.php" method="post">
  <input name="userfile" type="file">
  <br><input type="submit">
</form>
```

# Upload tệp và lưu vào CSDL

- Nhận tệp

// Tệp đã được upload?

```
if (is_uploaded_file($userfile)) {
```

// Mở tệp

```
$file = fopen($userfile, "r");
```

// Đọc nội dung

```
$fileContents = fread($file, filesize($userfile));
```

// Xử lý các ký tự đặc biệt bằng cách thêm \ trước chúng

```
$fileContents = addslashes($fileContents);
```

```
} else $fileContents = NULL;
```

# Upload tệp và lưu vào CSDL

- Lưu nội dung tệp vào CSDL

```
$insertQuery = "INSERT INTO TableName  
VALUES (... , \"\".$fileContents . \"\")";
```

# Đọc và hiển thị tệp ảnh đã lưu vào CSDL

- Tệp `imgdisp.php` hiển thị ảnh

```
$data = @ mysql_fetch_array($result);
if (!empty($data["map"])) {
    // Xuất dữ liệu ra GIF MIME
    header("Content-Type: image/gif");
    // Xuất dữ liệu ảnh
    echo $data["map"];
}
```
- Tệp sử dụng

```
echo "<img src=\"imgdisp.php?p=" . $p. "\">";
```

# Vấn đề truy cập đồng thời

- Nhiều người truy cập CSDL đồng thời có thể dẫn đến các tình huống sau:
  - *Mất cập nhật:* Người dùng A đọc giá trị từ CSDL. Người dùng B cũng đọc giá trị từ CSDL và cập nhật ngay lập tức. Người dùng A cập nhật, ghi đè giá trị của người B
  - *Đọc sai:* Người dùng A cập nhật giá trị. Người dùng B đọc giá trị đã cập nhật. Người dùng A undo lại thao tác => Giá trị B đọc được không còn đúng
  - *Tính tổng sai:* Người dùng A đang tính tổng thì người dùng B thay đổi giá trị một số mục
  - *Đọc giá trị không thể lặp:* A đọc giá trị, B thay đổi giá trị, A đọc lại thấy giá trị khác



# Xử lý truy cập đồng thời

- Khóa bảng cần thao tác
  - Kiểu khóa READ: cho người dùng khác được đọc nhưng không được ghi
  - Kiểu khóa WRITE: không cho người dùng khác đọc hay ghi
- Thực hiện truy vấn
- Thực hiện cập nhật
- Mở khóa bảng

# Ví dụ Xử lý cập nhật đồng thời

//Khóa các bảng cần thao tác

```
$query = "LOCK TABLES items READ, orders WRITE, customer READ";
```

```
if (!mysql_query($query, $connection)) showerror();
```

// Thực hiện truy vấn

```
$query = "SELECT SUM(price*qty) from FROM items, orders, customer  
WHERE customer.cust_id = orders.cust_id AND orders.order_id =  
items.order_id AND items.cust_id = orders.cust_id AND orders.order_id = $orderId  
AND customer.cust_id = $custId";
```

```
if (!($result = mysql_query($query, $connection))) showerror( );
```

```
$row = mysql_fetch_array($result);
```

//rồi cập nhật

```
if ($row["SUM(price*qty)"] > $minimum) {
```

```
    $query = "UPDATE orders SET discount = $discount WHERE cust_id = $custId  
AND order_id = $orderId";
```

```
    if (!mysql_query($query, $connection)) showerror();
```

```
}
```

// Mở khóa các bảng

```
$query = "UNLOCK TABLES";
```

# Lưu ý khi khóa các bảng

- Sau khi khóa xong phải mở khóa các bảng
- Không cần khóa các bảng nếu chỉ thực hiện một truy vấn.

# NỘI DUNG

**1. Kết nối và truy vấn CSDL**

**2. Phiên và cookie**



# Trạng thái của ứng dụng

- HTTP là giao thức phi trạng thái
  - Mỗi yêu cầu (request) được xử lý độc lập. Không yêu cầu server nhớ trạng thái của các xử lý trước
- Ứng dụng có thể cần nhớ trạng thái
  - Khi xử lý trên nhiều trang, cần sự tương tác phức tạp
    - Ví dụ: chuyển qua nhiều trang khác nhau để chọn nhiều mặt hàng đưa vào giỏ hàng
  - Cần tính cá nhân hóa
    - Ví dụ: phải biết người dùng nào đang sử dụng để cung cấp nội dung phù hợp

# Các phương pháp lưu trạng thái

- Lưu trạng thái ở trình khách
  - Sử dụng cookie
- Lưu trạng thái ở trình phục vụ
  - Sử dụng phiên (session)

# Cookie

- Một cookie là mẫu tin (*tên, giá trị, ...*)
  - Server gửi cookie cho client (trong đáp ứng cho yêu cầu trước)
  - Client nhớ cookie và gửi cookie cho server trong các yêu cầu sau
  - Server xử lý theo cookie nhận được

# Tương tự cookie

- Việc tạo và sử dụng cookie giống như sử dụng sổ y bạ
  - Server == Bác sỹ
  - Client == Bệnh nhân
  - Cookie == Nội dung được ghi trong sổ y bạ
  - Bác sỹ ghi bệnh lý và đơn thuốc vào sổ y bạ. Bệnh nhân cầm sổ y bạ về và đem sổ y bạ đến tại các lần khám sau.



# Client gửi cookie cho server

- Client nhớ cookie và gửi cookie cho server bằng đặt thuộc tính Cookie trong tiêu đề HTTP Request, ví dụ

*GET /nextPage.htm HTTP/1.1*

*Host: www.example.com*

***Cookie: food=choco; tasty=strawberry***

# Các thành phần của cookie

- Các thành phần

- *key=<value>*
- *Expires=<date>*
- *Max-Age=<non-zero-digit>*
- *Domain=<domain-value>*
- *Path=<path-value>*
- *Secure*
- *HttpOnly*

- Ví dụ

- *tasty=strawberry; Expires=Wed, 21 Oct 2017 07:28:00 GMT; Secure;*

# Server tạo và gửi cookie

- `int setcookie(string name, [string value], [int expire], [string path], [string domain], [bool secure], [bool httponly])`
  - *name*: tên cookie
  - *value*: giá trị của cookie
  - *expire*: thời điểm cookie hết hạn
  - *path*: đường dẫn trình duyệt sẽ gửi cookie
  - *domain*: tên miền trình duyệt sẽ gửi cookie
  - *secure*: chỉ truyền cookie qua kết nối an toàn
  - *httponly*: trình duyệt không được truy cập cookie bằng javascript

# Server nhận và xử lý cookie

- Cookie được lưu trong mảng `$_COOKIE`
- Nếu bật *register\_globals* (trong php.ini), biến có tên cookie được khởi tạo
- Ví dụ
  - `$tasty= $_COOKIE*“tasty”+;`

# Client truy cập cookie bằng javascript

- Nếu httponly = false

- Ví dụ

```
<script type="text/javascript">  
    console.log(document.cookie);  
    document.cookie = "amount=3";  
    console.log(document.cookie);  
</script>
```

# Ví dụ sử dụng cookie

```
<?php
if (!isset($_COOKIE["guideshown"])) { //Truy cập trang lần đầu
?>
    <div id="guide">
        Chào mừng quý vị ghé thăm trang của chúng tôi.
        Quý vị vui lòng giành ít thời gian xem bản giới thiệu.<br><br>
        <button id="closeguide">Đóng</button>
    </div>
    <script>
        document.getElementById("closeguide").onclick = function() {
            document.getElementById("guide").style.display = "none";
        };
    </script>
<?php
    setcookie("guideshown", "shown");
}
?>
```

# Vấn đề của cookie

- Kích thước request tăng khi bao hàm cookie
- Tính an ninh không cao
- Tính riêng tư bị vi phạm do trình duyệt nhớ cookie

# Phiên

- Phiên (session) là cuộc thoại (dialogue/conversation) giữa trình khách và trình phục vụ.
- Server lưu các biến phiên và gửi cho client định danh phiên. Định danh phiên được server tạo ngẫu nhiên.
- Trình duyệt bao gồm định danh phiên trong các requests sau, server ánh xạ định danh phiên sang các biến phiên
- Phiên phải có thời gian hết hạn (timeout).
- Các biến phiên bị hủy khi ngắt kết nối hoặc hết hạn phiên



# Phiên

- HTTP là giao thức không có trạng thái. Do đó, nếu không có một cơ chế đặc biệt để duy trì trạng thái đăng nhập thì người dùng sẽ phải thực hiện xác thực mỗi khi thực hiện một truy vấn HTTP.
- ⇒ Ứng dụng điển hình của session là để duy trì trạng thái đăng nhập của người dùng.

# Phiên

- Session là một tập hợp có thể mở rộng các biến được lưu trong một file với tên ngẫu nhiên, khó đoán trên máy chủ. Mỗi session được xác định bởi một định danh, gọi là session ID – cũng là một giá trị khó đoán.
- Giá trị của session ID được truyền qua lại giữa web server và web client để duy trì phiên làm việc.
  - Giá trị session ID thường được truyền dưới dạng cookie và theo mặc định không được web client ghi nhớ, vì thế, khi đóng web client (trình duyệt) thì giá trị này trên máy người dùng sẽ mất, người dùng không thể tiếp tục phiên làm việc cũ.
  - Trong trường hợp trình duyệt không hỗ trợ cookie hoặc tính năng cookie bị vô hiệu hóa thì session ID có thể được truyền đi dưới dạng tham số của URL.

# Tương tự biến phiên

- Việc tạo và sử dụng biến phiên giống như sử dụng sổ theo dõi bệnh nhân
  - Server == Bác sỹ
  - Client == Bệnh nhân
  - Phiên == Nội dung được ghi trong sổ theo dõi
  - Bác sỹ ghi bệnh lý và đơn thuốc vào sổ theo dõi, cấp cho bệnh nhân thẻ khám chữa bệnh (định danh phiên) nhưng không đưa sổ cho bệnh nhân. Bệnh nhân cầm thẻ về và đem thẻ đến tại các lần khám sau.

# Đăng ký và sử dụng biến phiên

- Khởi động phiên
  - `session_start( );`
- Sử dụng biến phiên
  - `$_SESSION*“svName”+;`
- Hủy phiên
  - `session_destroy( );`

# Ví dụ sử dụng biến phiên

```
<?php
if (!isset($_SESSION["guideshown"])) { //Truy cập trang lần đầu trong phiên
?>
    <div id="guide">
        Chào mừng quý vị ghé thăm trang của chúng tôi.
        Quý vị vui lòng dành ít thời gian xem bản giới thiệu.<br><br>
        <button id="closeguide">Đóng</button>
    </div>
    <script>
        document.getElementById("closeguide").onclick = function() {
            document.getElementById("guide").style.display = "none";
        };
    </script>
<?php
$_SESSION["guideshown"] = "shown";
}
?>
```

# Khi nào nên/không nên sử dụng biến phiên

## ● Nên

- *Tăng hiệu năng*: Thực hiện tính toán phức tạp một lần, lưu kết quả trong biến phiên, sử dụng kết quả nhiều lần
- *Cần chuỗi các tương tác*: Người dùng cần nhập liệu trên nhiều giao diện khác nhau, nếu cần có thể quay về giao diện trước để sửa dữ liệu đã được nhập ở giao diện trước
- *Kết quả trung gian*: Nhiều kết quả trung gian nên được ghi nhớ cho các tính toán tiếp sau
- *Cá nhân hóa*: Lưu định danh người dùng ở dạng biến phiên, căn cứ vào định danh người dùng để cung cấp nội dung phù hợp

# Khi nào nên/không nên sử dụng biến phiên

- Không nên

- *Lưu trữ trên server*: Nếu lạm dụng sử dụng biến phiên, server sẽ phải dành nhiều bộ nhớ để lưu
- *An toàn*: Hacker có thể lợi dụng phiên để thực hiện các tấn công

The End!