Chương 3

Lập trình HĐT với PHP & PDO

CT275 – CÔNG NGHỆ WEB

Mục tiêu

Giới thiệu
phương pháp Lập trình hướng đối tượng trong PHP và
sử dụng PDO để truy xuất CSDL

Nội dung

- Lập trình hướng đối tượng căn bản
- Truy xuất CSDL quan hệ với PDO
- Giới thiệu Composer

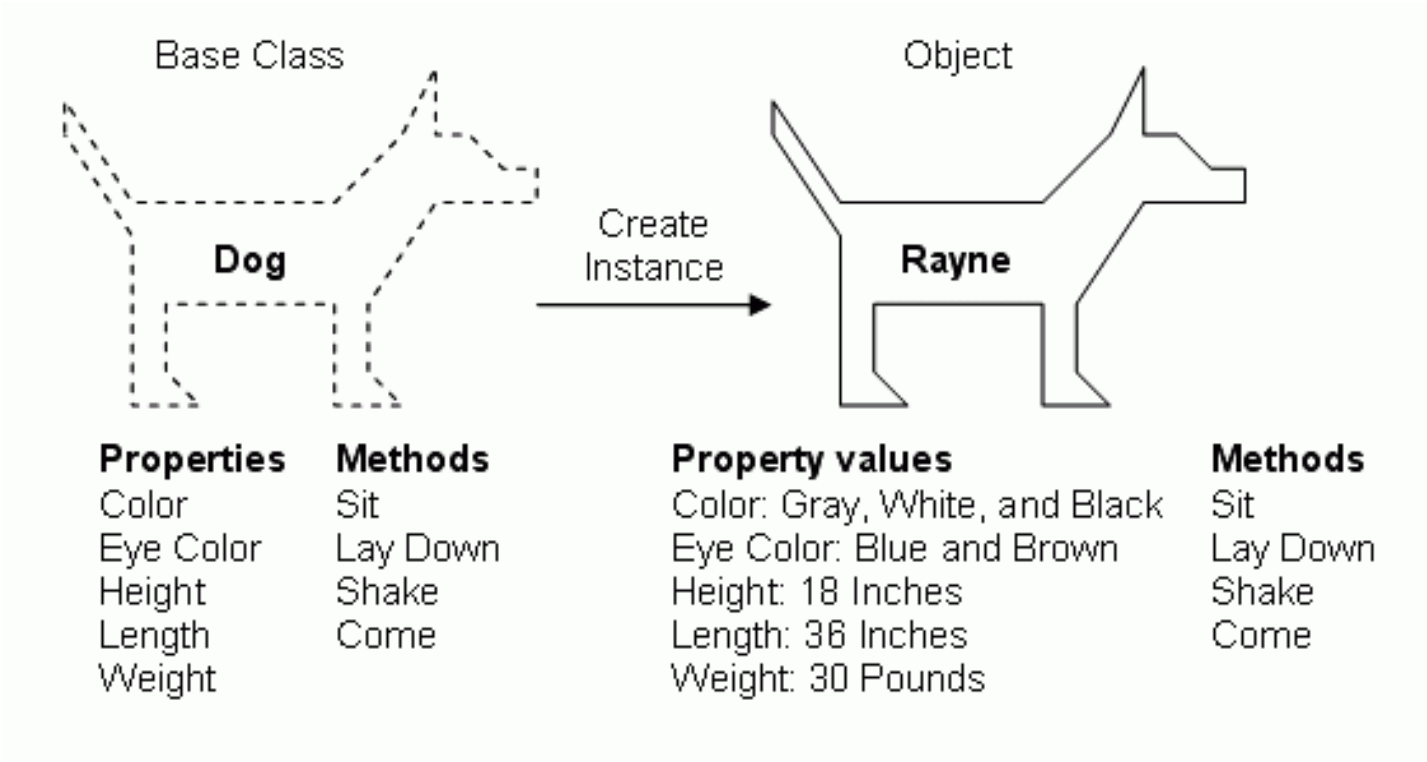
Lập trình Hướng đối tượng trong PHP

Object Oriented Programming (OOP) in PHP

Giới thiệu

- OOP trong PHP được hỗ trợ từ phiên bản PHP 5
- Các khái niệm cơ bản trong OOP:
  - **Đối tượng** (object): là một thực thể mà chương trình muốn đề cập đến, bao gồm 2 thành phần
    - Thuộc tính: lưu trữ các đặc điểm, trạng thái của đối tượng
    - Phương thức: thể hiện cho các hành vi, hoạt động mà đối tượng có thể thực hiện (khả năng của đối tượng)
  - **Lớp** (class): là một tập các đối tượng có cùng đặc điểm (hay là khuôn mẫu để tạo ra các đối tượng)

Giới thiệu


Base Class
Dog
Create
Instance
Object
Rayne
Properties
Color
Eye Color
Height
Length
Weight
Methods
Sit
Lay Down
Shake
Come
Property values
Color: Gray, White, and Black
Eye Color: Blue and Brown
Height: 18 Inches
Length: 36 Inches
Weight: 30 Pounds
Methods
Sit
Lay Down
Shake
Come

Tạo lớp và đối tượng

- Định nghĩa lớp:

```
class classname {  //classname là một định danh trong PHP
    //định nghĩa các thuộc tính/phương thức thành viên
}
```

  - **Thuộc tính**: được khai báo như các biến trong định nghĩa lớp với các từ khóa điều khiển truy cập: public, protected, private
  - **Các phương thức**: được tạo ra bằng các khai báo các hàm trong định nghĩa lớp
- Tạo đối tượng: `$varname = new <ClassName>([đối số]);`
- Truy xuất thành phần của đối tượng: `object->member`

Tạo lớp và đối tượng

- Ví dụ 1: một khai báo lớp đơn giản

```php
<?php
    class phpClass {
        //khai báo các thuộc tính
        var $var1;
        var $var2 = "constant string";

        //định nghĩa các phương thức
        function myfunc ($arg1, $arg2) {
            [..]
        }
        [..]
    }
?>
```

Tạo lớp và đối tượng

- Ví dụ 2: lớp Person

```php
<?php
  class Person {
    public $fullname = false;
    public $givenname = false;
    public $familyname = false;
    public $room = false;
    function getName() {
      if ($this->fullname !== false)
         return $this->fullname;
      if ($this->familyname!==false && $this->givenname!==false)
         return $this->givenname . ' ' . $this->familyname;
      return false;
    }
  }
?>
```

```php
<?php
$chuck = new Person();
$chuck->fullname = "Chuck Severance";
$chuck->room = "4429NQ";
print $chuck->getName() . "\n";
```

Chuck Severance

Hàm xây dựng (constructor)

- Là một hàm đặc biệt của lớp
  - Tự động được gọi khi đối tượng được tạo ra
  - Cú pháp: `function __construct (param_list) {...}`
- Mục đích: khởi tạo giá trị cho các dữ liệu thành viên
- Lưu ý:
  - PHP không cho phép chồng hàm khởi tạo (ĐN nhiều hàm khởi tạo trong cùng 1 lớp)
  - Tuy nhiên, PHP hỗ trợ tham số mặc định (ngược với Java)

Hàm xây dựng (constructor)

```php
<?php
class PartyAnimal {
    function __construct() {
      echo("Constructed\n");
    }

    function something() {
      echo("Something\n");
    }

    function __destruct() {
      echo("Destructed\n");
    }
}
```

```php
    echo("--One\n");
    $x = new PartyAnimal();
    echo("--Two\n");
    $y = new PartyAnimal();
    echo("--Three\n");
    $x->something();
    echo("--The End");
?>
```

```
--One
Constructed
--Two
Constructed
--Three
Something
--The End
```

Getter và Setter

- Thông thường, các thuộc tính được thiết đặt private hoặc protected
  - Không truy cập được từ bên ngoài lớp
  - Việc truy cập sẽ thông qua các hàm getter và setter
- Cú pháp:
  - **Getter**: `function __set($var, $value) { ... }`
  - **Setter**: `function __get($var) { ... }`
- Các hàm này được gọi là các "magic methods"
  - Không được gọi 2 cách tường minh mà sẽ tự động được gọi trong một ngữ cảnh cụ thể

Getter và Setter

```php
<?php
class BankAccount { private $balance;
  function __set($name, $value) {
     echo "Setting '$name' to '$value'\n";
     $this->$name = $value;
  }
  function __get($name) {
     echo "Getting '$name'\n";
     return $this->$name;
  }
}
?>
```

```php
$acc = new BankAccount();
$acc->balance = 50; // gọi __set()
echo $acc->balance; // gọi __get()
```

Các thành phần tĩnh (static)

- Là các thành phần của lớp (độc lập với các đối tượng)
- Được truy xuất trực tiếp thông qua tên lớp

```php
<?php
class Foo {
    private static $count = 0;
    function __construct() {
        static::$count++;
    }
    static function getCount() {
        return static::$count;
    }
}
?>
```

```php
Foo::getCount() // 0
$foo1 = new Foo();
$foo2 = new Foo();
Foo::getCount() // 2
```

Thừa kế

- Mục đích:
  - Tạo lớp mới từ các lớp có sẵn: sử dụng 1 lớp có sẵn, thừa kế các thành phần có sẵn và bổ sung thêm các thành phần mới
  - Tổ chức các lớp: dựa trên sự khác biệt và giống nhau giữa các lớp để tạo cây phân cấp lớp (class hierarchy)
- Lớp thừa kế: lớp con; lớp được thừa kế: lớp cha
- Truy cập đến các thành phần của lớp cha:

```
parent::property/method()
```

- Lớp con có thể khai báo chồng các t/phần của lớp cha

Thừa kế

```php
class Pet {
  public $name;
  function __construct($petName) {
    $this->name = $petName;
  }

  function play() {
    echo "<p>$this->name is playing.</p>";
  }
}

class Cat extends Pet {
  function play() {
    parent::play();
    echo "<p>$this->name is climbing.</p>";
  }
}
```

```php
class Dog extends Pet {
  function play() {
    parent::play();
    echo "<p>$this->name is fetching.</p>";
  }
}

$pet = new Pet('Rob');
$dog = new Dog('Satchel');
$cat = new Cat('Bucky');

$pet->play();
$dog->play();
$cat->play();
```

```
Rob is playing.
Satchel is playing.
Satchel is fetching.
Bucky is playing.
Bucky is climbing.
```

Truy xuất CSDL với PDO

Database access with PDO

PDO là gì?

- PDO = PHP Data Object
- Là một sự mở rộng của PHP cho việc truy xuất CSDL
- Tại sao nên dùng PDO:
  - Cung cấp giao diện nhất quán cho việc truy xuất CSDL, chi tiết cài đặt cho từng loại CSDL được định nghĩa trong các PDO driver
  - Cung cấp prepare statement và hướng đối tượng, cho phép các thao tác trên CSDL dễ dàng và hiệu quả hơn

Các hệ quản trị CSDL được hỗ trợ

- Microsoft SQL Server / Sybase
- Firebird / Interbase
- DB2 / INFORMIX (IBM)
- MySQL
- OCI (Oracle Call Interface)
- ODBC
- PostgreSQL
- SQLite

```
print_r(PDO::getAvailableDrivers());
```

Tạo nối kết

```php
try {
    $conn = new PDO($dsn, $user, $pass);
    $conn->setAttribute(PDO::ATTR_ERRMODE,
                        PDO::ERRMODE_EXCEPTION);

    // Sử dụng CSDL
    // ...
    // Đóng kết nối

    $conn = null;
} catch (PDOException $e) {
    // Xử lý lỗi
    echo "Error!: " . $e->getMessage() . "<br/>";
    die();
}
```

Data Source Name (DSN)

- Cú pháp: **`drivername:<driver-specific-stuff>`**
- Ví dụ:
  - mysql:host=name;dbname=dbname
  - odbc:odbc_dsn
  - oci:dbname=dbname;charset=charset
  - sqlite:/path/to/db/file
  - sqlite::memory:

Tạo và thực thi lệnh SQL

1) Chuẩn bị câu lệnh


```php
$conn = new PDO(...);
$stmt = $conn->prepare("INSERT INTO Users
        (username, password) VALUES (:username, :password)");
```

2) Nối kết tham số:

- Qua tên:

```php
$stmt->bindParam('username', $username);
$stmt->bindParam('password', $password);
```

- Qua chỉ số:

```php
$stmt->bindParam(1, $username);
$stmt->bindParam(2, $password);
```

3) Thực thi: `$stmt->execute();`

Truyền tham số bằng mảng

- Tham số của câu lệnh SQL có thể được truyền trực tiếp bằng mảng khi gọi hàm execute như sau:

```
$conn = new PDO(...);
$stmt = $conn->prepare("INSERT INTO Users
   (username, password) VALUES (:username, :password)");
$stmt->execute(array(
                    'username' => $username,
                    'password' => $password));
```

Truy vấn dữ liệu

- Sau khi thực hiện câu truy vấn (execute()), ta có thể lấy dữ liệu trả về từ câu truy vấn bằng hàm `fetch()` hoặc `fecth_All()`
- Các hàm này có thể trả về dữ liệu theo nhiều định dạng, được điều khiển bởi hàm `setFetchMode()`

```php
$conn = new PDO(...);
$stmt = $conn->prepare("SELECT * FROM Users");
$stmt->execute();

$rows = $stmt->fetchAll()/fetchAll(PDO::FETCH_OBJ);
```

Truy vấn dữ liệu

- Các định dạng trả về phổ biến của các lệnh fetch:
  - PDO::FETCH_ASSOC: dạng mảng, với key là tên column
  - PDO::FETCH_CLASS: dạng đối tượng, gán giá trị của từng trường cho từng thuộc tính của lớp
  - PSO::FETCH_OBJ: dạng đối tượng (anonymous), với tên thuộc tính của đối tượng là tên của column
  - PDO::FETCH_BOTH (default): dạng mảng, với key là tên của column và cả số thứ tự của column
  - PDO::FETCH_INTO: tương tự như FETCH_CLASS nhưng ghi vào 1 đối tượng có sẵn thay vì tạo đối tượng mới
  - …

Truy vấn dữ liệu – FETCH_ASSOC

```php
$conn = new PDO(...);
$stmt = $conn->query('SELECT name, addr, city from folks');
$stmt->setFetchMode(PDO::FETCH_ASSOC);

while($row = $stmt->fetch()) {
    echo $row['name'] . "\n";
    echo $row['addr'] . "\n";
    echo $row['city'] . "\n";
}
```

Truy vấn dữ liệu – FETCH_OBJ

```php
$conn = new PDO(...);
$stmt = $conn->query('SELECT name, addr, city from folks');
$stmt->setFetchMode(PDO::FETCH_OBJ);

while($row = $stmt->fetch()) {
    echo $row->name . "\n";
    echo $row->addr . "\n";
    echo $row->city . "\n";
}
```

Truy vấn dữ liệu – FETCH_CLASS

```php
$conn = new PDO(...);
$stmt = $conn->query('SELECT name, addr, city from folks');
$stmt->setFetchMode(PDO::FETCH_CLASS, 'App/Model/Person');

while($row = $stmt->fetch()) {
    echo $row->name . "\n";
    echo $row->addr . "\n";
    echo $row->city . "\n";
}
```

Giao dịch với PDO

- Giao dịch: tập các lệnh, mang tính nguyên tử

```php
$conn = new PDO(...);
...
try {
    $conn->beginTransaction();
    $conn->query("UPDATE …");
    $conn->query("UPDATE …");
    $conn->commit();
} catch (PDOException $e) {
    $conn->rollBack();
}
```

Một số hàm PDO thông dụng

- `$conn->lastInsertId();`
  Trả về ID của mẩu tin cuối cùng được chèn vào
- `$safe = $conn->quote($unsafe_string);`
  Tạo chuỗi "an toàn" trong câu truy vấn (PDO::prepare() sẽ tự gọi đến hàm này)
- `$rowEffected = $stmt->rowCount();`
  Trả về số dòng bị tác động bởi câu lệnh SQL cuối cùng

- http://www.mustbebuilt.co.uk/2012/10/16/pdo-cheatsheet/

Giới thiệu Composer

Introduction to Composer

Composer là gì?

- A per-project dependency manager that allows you to:
  - declare a consistent list of dependencies and versions for your application
  - share your libraries and make them discoverable using **packagist.org**

Dependency Manager for PHP

Các tính năng của Composer

- Download libraries/packages your project depends on
- Specify version of each dependency (using semantic versioning)
- Download dependencies of your dependencies
- Write autoloader file and class map of dependencies
- Create projects based on packages
- Can use multiple repositories
- …

Cài đặt Composer

- Install composer locally (composer.phar)

  http://www.getcomposer.org/composer.phar

- Install it globally

  curl -sS https://getcomposer.org/installer | php

  mv composer.phar /usr/local/bin/composer

- Windows Installer
- Requires PHP 5.3.2+

Cấu hình Composer – composer.json

- **composer.json** lưu cấu hình Composer của project
- Đặt trong thư mục gốc của ứng dụng

```
{
    "name": “ct257/my_project",
    "description": "My First Composer Project",
    "authors": [
    {
        "name": "Author of the Project",
        "email": "author@somedomain.com"
    }],
    "require": {
        "monolog/monolog": "1.1.2“
    }
}
```

Cấu hình Composer – `composer.json`

- Các thành phần cơ bản của **composer.json**:
  - **"name"**: tên dự án, có dạng vendor_name/package_name
  - **"description"**: mô tả của gói/dự án
  - **"author"**: tên tác giả của dự án (có thể có nhiều)
  - **"require"**: danh sách các thư viện cần thiết, bao gồm vendor name, pakage name và version
  - "**require-dev**": các thư viện chỉ cài khi **--dev** flag được chỉ định khi install hoặc không có **--no-dev** flag khi update
  - …

Cấu hình Composer – Versions

- Cấu trúc của một semantic version:
  - x.y.**Z** (Patch version) - Backwards compatible bug fixes introduced.
  - x.**Y**.z (Minor version) - New features introduced that are backwards compatible, or feature is deprecated
  - **X**.y.z (Major version) - New features introduced that are not backwards compatible
  - Number increase by one (i.e. 1.9.0, 1.10.0, 1.11.0)
  - An increase in one version resets the version to the right back to zero.

Cấu hình Composer – Versions

- Cấu hình version trong Composer:
  - Can request specific stability level: @dev, @alpha, @beta, @rc, @stable (default)
  - Can request specific commits via #(ref)

```
{
  "require": {
    "monolog/monolog": "1.0.*@beta",
    "acme/foo": "@dev",
    "elibyy/crypt": "master@b1fe5..."
  }
}
```

Cấu hình Composer – Versions

- Version chỉ định trong cấu hình Composer cũng có thể dưới dạng tương đối như sau:
  - X.Y.* - Match the latest stable minor version (i.e. 1.2.3 will be selected rather than 1.2.2)
  - >= X.Y.Z, < A.B.C - Match a range of versions (i.e. any stable version between 1.2.3 & 2.0.0)
  - ~X.Y.Z - Match next significant release (i.e. ~1.2 will match up to 1.9, ~1.1.3 will match 1.1.99)

Packagist.org

- Search for packages
- See package requirements & recommendations
- List of every version registered
- Dependency declaration for each version

Creating a Project

1) Change the working directory to /dev/myproject

```
cd ~/dev/myproject
```

2) Create a composer.json file

3) Create a project with declared dependencies

```
php composer.phar install
```
(*hoặc* `composer install`)

4) Lệnh install sẽ:
   - Download các thư viện
   - Tạo tập tin composer.lock và autoload.php

Tập tin `composer.lock`

- composer.lock: được tạo ra khi thực hiện lệnh `install`
- Lưu trữ thông tin chính xác về version của các thư viện đang được sử dụng
- Khi file này hiện diện trong thư mục của project, lệnh `install` sẽ bỏ qua tập tin cấu hình json và sử dụng tập tin này
- Tập tin này sẽ được cập nhật khi thực hiện lệnh `update`
- Tập tin composer.lock phải được commit lên kho chứa VCS

Tập tin composer.lock

- composer.json:

```
{
   "require" : {
      "aws/aws-sdk-php-zf2": "1.0.*"
   }
}
```

- composer.lock

```
{
   "require" : {
      "aws/aws-sdk-php-zf2": "1.0.1"
   }
}
```

Tập tin composer.lock

- All future clones of the project will install version 1.0.1
- To upgrade a dependency, run:

```
php composer.phar update
```

(hoặc `composer update`)

- Update command installs the latest version
- Changes the installed version on the lock file
- Lock file should be recommitted along with composer.json

Tập tin autoload.php

- Chứa các lệnh cần thiết để load các thư viện
- Để sử dụng được các thư viện, thêm lệnh require tập tin này:

```php
require "vendor/autoload.php";
```

hoặc

```php
require_once 'vendor/autoload.php';
```

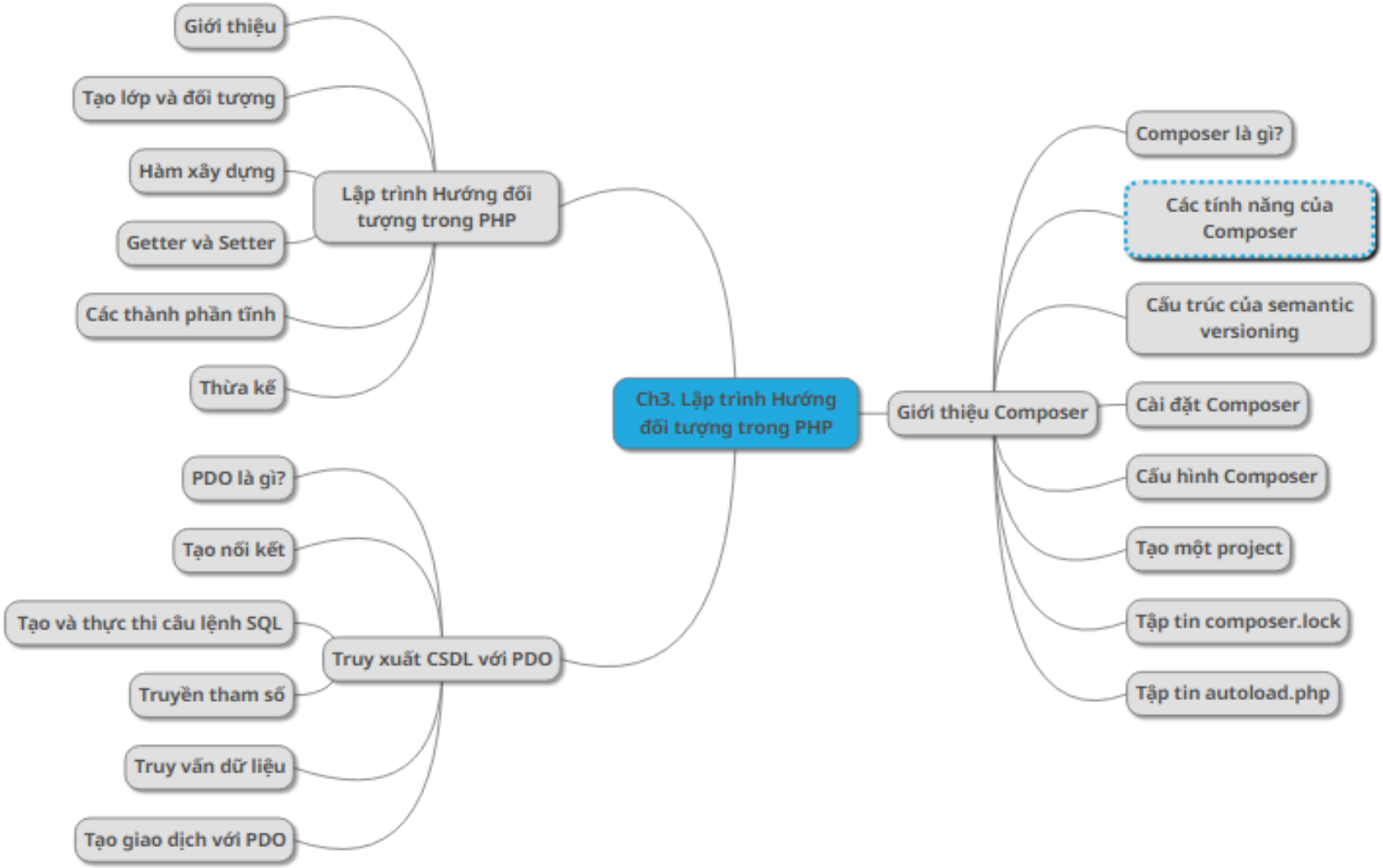
Sử dụng autoload.php

```php
require "vendor/autoload.php";

use Monolog\Logger;
use Monolog\Handler\StreamHandler;

// create a log channel
$log = new Logger('name');
$log->pushHandler(new StreamHandler('path/to/your.log',
                                    Logger::WARNING));

// add records to the log
$log->addWarning('Foo');
$log->addError('Bar');
```

Một số thư viện thông dụng

- **league/flysystem**: abstraction for local and remote filesystems
- **guzzlehttp/guzzle**: a PHP HTTP client
- **nesbot/carbon**: a simple PHP API extension for DateTime
- **gregwar/captcha**: a PHP Captcha library
- **league/plates**: A native PHP template system
- **mpdf/mpdf**: A PHP class to generate PDF files from HTML with Unicode/UTF-8 and CJK support
- **intervention/image**: a PHP image handling and manipulation library
- **swiftmailer/swiftmailer**: comprehensive mailing tools for PHP

Tóm tắt


Ch3. Lập trình Hướng đối tượng trong PHP
Lập trình Hướng đối tượng trong PHP
Giới thiệu
Tạo lớp và đối tượng
Hàm xây dựng
Getter và Setter
Các thành phần tĩnh
Thừa kế
Truy xuất CSDL với PDO
PDO là gì?
Tạo nối kết
Tạo và thực thi câu lệnh SQL
Truyền tham số
Truy vấn dữ liệu
Tạo giao dịch với PDO
Giới thiệu Composer
Composer là gì?
Các tính năng của Composer
Cấu trúc của semantic versioning
Cài đặt Composer
Cấu hình Composer
Tạo một project
Tập tin composer.lock
Tập tin autoload.php

Question?

CT275 – CÔNG NGHỆ WEB