面向对象第4天

[学习目标 1](#_Toc522292247)

[综合案例：学生信息管理系统 2](#_Toc522292248)

[1、面向对象开发流程 2](#_Toc522292249)

[2、单例设计模式(三私一公) 2](#_Toc522292250)

[3、数据库工具类(./libs/Db.class.php) 3](#_Toc522292251)

[4、连接数据库的公共文件(./conn.php) 4](#_Toc522292252)

[5、显示学生信息列表(./list.php) 4](#_Toc522292253)

[6、删除学生信息(./delete.php) 5](#_Toc522292254)

[7、创建分页类文件 ./libs/Pager.class.php 7](#_Toc522292255)

[工厂设计模式 7](#_Toc522292256)

[1、什么是工厂设计模式 7](#_Toc522292257)

[2、工厂设计模式的要求 8](#_Toc522292258)

[3、实例：创建不同形状类的对象 8](#_Toc522292259)

[重载 8](#_Toc522292260)

[1、什么是重载 8](#_Toc522292261)

[2、属性重载 9](#_Toc522292262)

[（1）\_\_get()魔术方法 9](#_Toc522292263)

[（2）\_\_set()魔术方法 10](#_Toc522292264)

[（3）\_\_isset()魔术方法 11](#_Toc522292265)

[（4）\_\_unset()魔术方法 11](#_Toc522292266)

[3、方法重载 12](#_Toc522292267)

[（1）\_\_call()魔术方法 12](#_Toc522292268)

[（2）\_\_callStatic()魔术方法 12](#_Toc522292269)

[变量序列化 13](#_Toc522292270)

[1、什么是变量序列化 13](#_Toc522292271)

[2、序列化函数serialize() 13](#_Toc522292272)

[3、反序列化函数unserialize() 13](#_Toc522292273)

[4、对象序列化 14](#_Toc522292274)

[5、对象反序列化 14](#_Toc522292275)

[常用的魔术常量 15](#_Toc522292276)

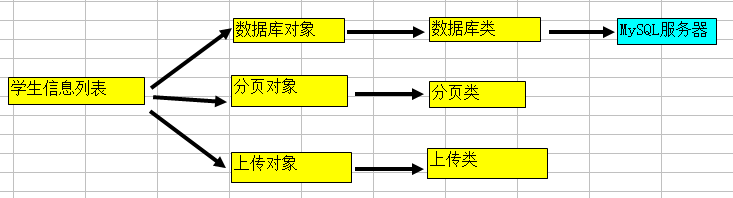
# 学习目标

* 综合案例：学生信息管理
* 工厂设计模式
* 重载
* 静态延时绑定
* 变量序列化
* 常用的魔术常量

# 综合案例：学生信息管理系统

## 1、面向对象开发流程

* 面向过程是以过程(步骤)为中心的编程思想，面向对象是事务(对象)为中心的编程思想。
* 对象是专业对象，是一个功能方方面面的总和。例如：数据库对象、分页对象、图像处理等。
* 一个项目由若干个功能模块构成，包括：用户管理、新闻管理、产品管理、文章管理、学生管理等。
* 每个功能模块是一个对象，包括：用户对象、新闻对象、产品对象、文章对象、学生对象等。
* 每个对象对应一个类：包括：用户类、新闻类、产品类、文章类、学生类等。
* 当然，每个模块还有一些公共对象：数据库对象、分页对象、上传对象、图像处理、验证码对象等。



## 2、单例设计模式(三私一公)

* 一个类永远只能创建一个对象，不管用任何方法都无法创建第2个对象。该对象大家共享。
* 一私：私有的静态的保存对象的属性。
* 一私：私有的构造方法，阻止类外new对象。
* 一私：私有的克隆方法，阻止类外clone对象。
* 一公：公共的静态的创建对象的方法。

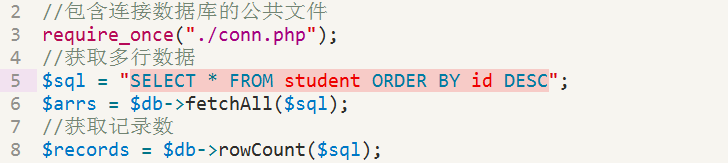
## 3、数据库工具类(./libs/Db.class.php)



## 4、连接数据库的公共文件(./conn.php)



## 5、显示学生信息列表(./list.php)







## 6、删除学生信息(./delete.php)

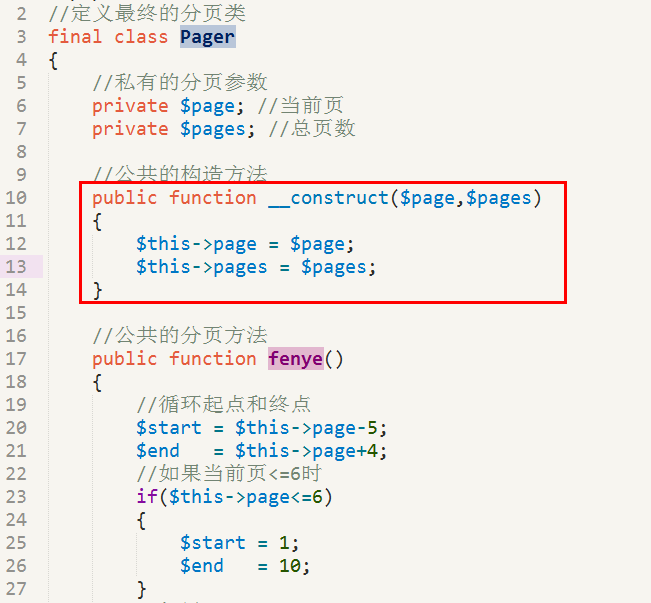
**（1）列表页list.php**



**（2）删除页面delete.php**



## 7、创建分页类文件 ./libs/Pager.class.php



# 工厂设计模式

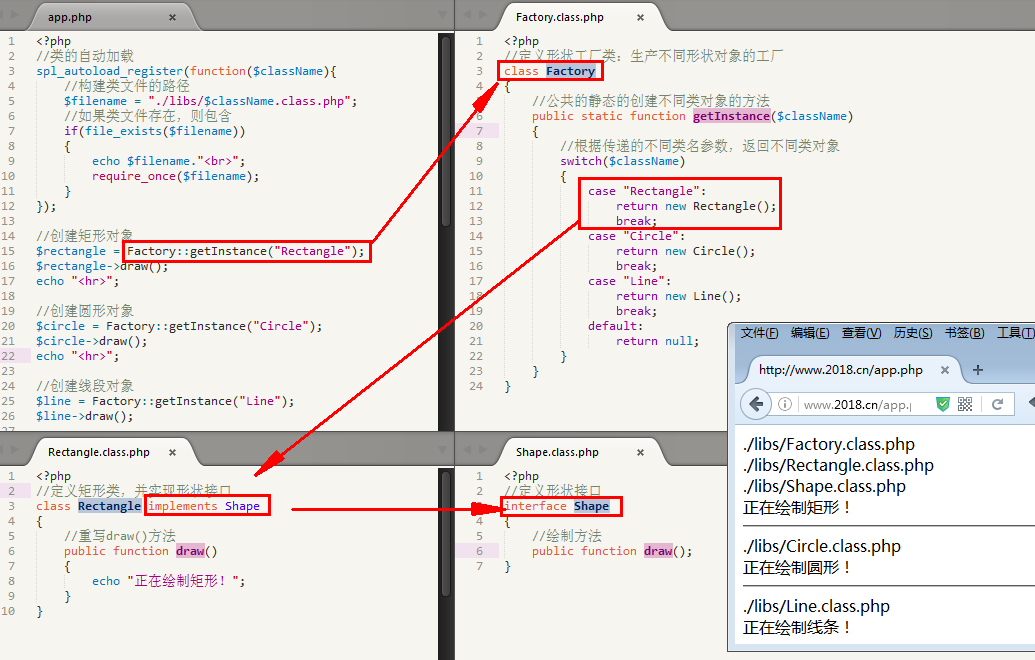
## 1、什么是工厂设计模式

* 根据传递不同的类名参数，返回不同类的对象；
* 工厂模式，就是生产各种的不同类的对象；
* 工厂模式，改变了在类外使用new关键字创建对象的方式，改成了在工厂类中创建类的对象。
* 在类的外部我们无法控制类的行为，但在类内部自己可以控制类的行为。

## 2、工厂设计模式的要求

* 一般情况下，定义一个工厂类；
* 工厂类中的方法，应该是公共的静态的方法；
* 该方法功能：就是根据传递的不同参数，去创建不同的类实例；

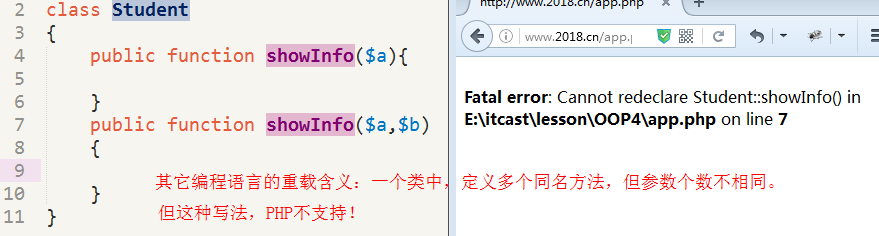
## 3、实例：创建不同形状类的对象

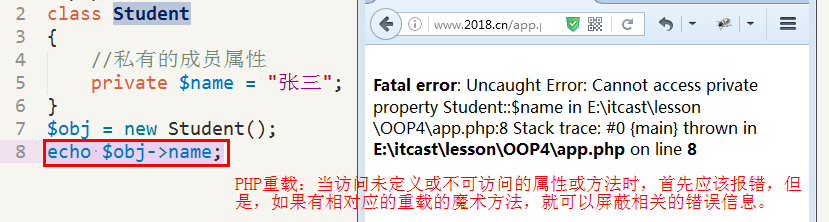


# 重载

## 1、什么是重载

* 在其它编程语言中，面向对象的重载是指，方法有相同的名称，但是参数列表不相同的情形，但PHP不支持同名函数或同名方法。
* PHP所提供的"重载"（overloading）是指动态地"创建"类属性和方法，我们是通过魔术方法来实现的。
* 当调用当前环境下未定义或不可见的类属性或方法时，重载方法会被调用。
* 所有的重载方法都必须被声明为 public。
* 属性重载只能在对象中进行。在静态方式中，这些魔术方法将不会被调用。所以这些方法都不能被 声明为 static。
* 这些魔术方法的参数都不能通过引用传递。





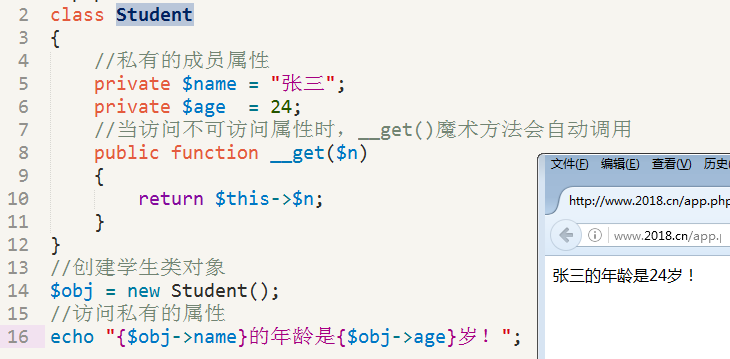


## 2、属性重载

### （1）\_\_get()魔术方法

描述：读取不可访问属性的值时，\_\_get() 会被调用。

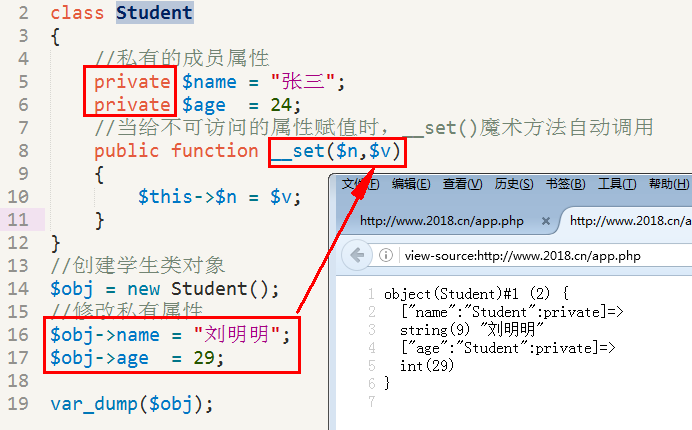
语法：**public mixed \_\_get ( string $name )**



### （2）\_\_set()魔术方法

描述：在给不可访问属性赋值时，\_\_set() 会被调用。

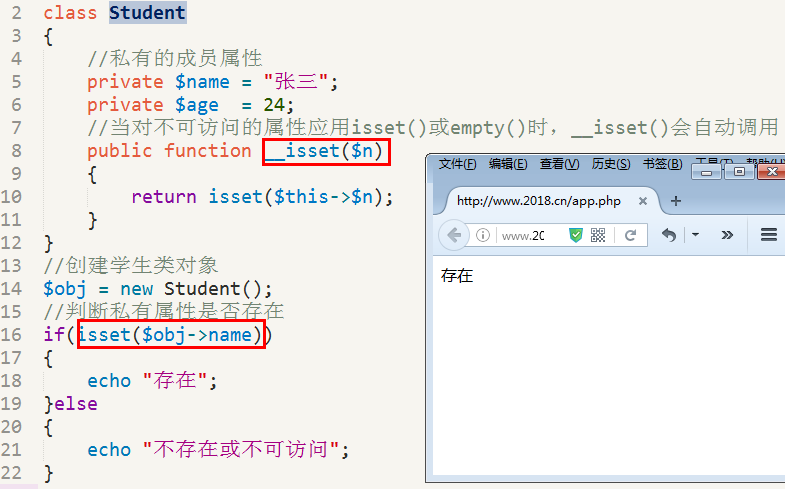
语法：**public void \_\_set ( string $name , mixed $value )**



### （3）\_\_isset()魔术方法

描述：当对不可访问属性调用 isset() 或 empty() 时，\_\_isset()会被调用。

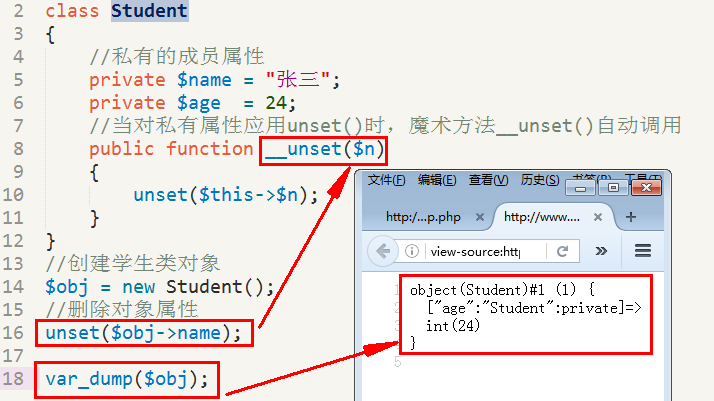
语法：**public bool \_\_isset ( string $name )**



### （4）\_\_unset()魔术方法

描述：当对不可访问属性调用 unset() 时，\_\_unset()会被调用。

语法：**public void \_\_unset ( string $name )**

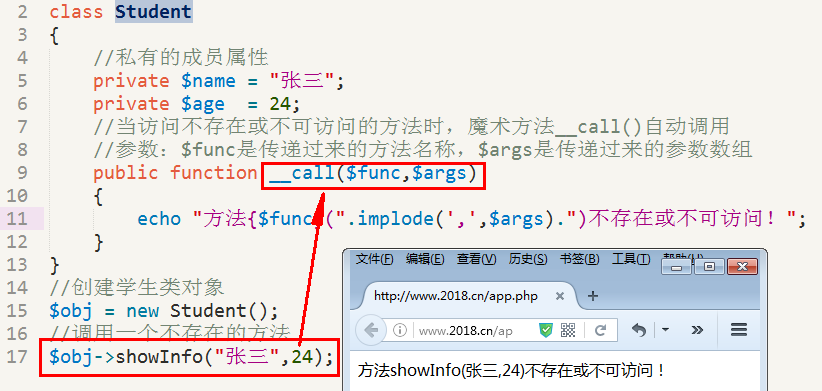


## 3、方法重载

### （1）\_\_call()魔术方法

描述：在对象中调用一个不可访问方法时，\_\_call() 会被调用。

语法：**public mixed \_\_call ( string $name , array $arguments )**



### （2）\_\_callStatic()魔术方法

描述：用静态方式中调用一个不可访问方法时，\_\_callStatic() 会被调用。

语法：**public static mixed \_\_callStatic ( string $name , array $arguments )**



# 变量序列化

## 1、什么是变量序列化

* 序列化是将变量转换为可保存或传输的字符串的过程；
* 反序列化就是在适当的时候把这个字符串再转化成原来的变量使用；
* 这两个过程结合起来，可以轻松地存储和传输数据，使程序更具维护性；
* 序列化有利于存储或传递 PHP 的值，同时不丢失其类型和结构。

## 2、序列化函数serialize()

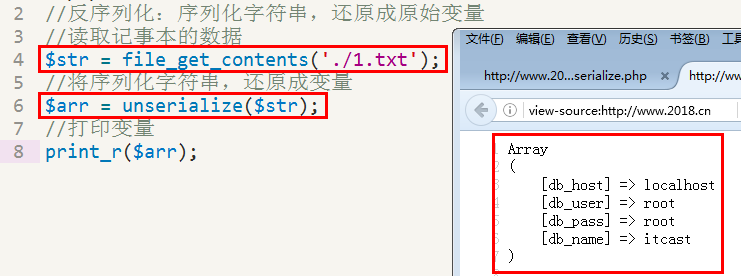
* 描述：产生一个可存储的值的表示；
* 语法：**string serialize ( mixed $value )**
* 参数：$value可以是任何类型，除了资源外；
* 返回：返回序列化之后的字符串，可以存储于任何地方。





## 3、反序列化函数unserialize()

* 描述：从已存储的表示中创建 PHP 的值
* 语法：**mixed unserialize ( string $str )**
* 说明：对单一的已序列化的变量进行操作，将其转换回 PHP 的值。
* 参数：$str为序列化后的字符串；
* 返回：返回的是转换之后的值，可为 integer、float、string、array或 object。 如果传递的字符串不可序列化，则返回 FALSE，并产生一个 E\_NOTICE。



## 4、对象序列化

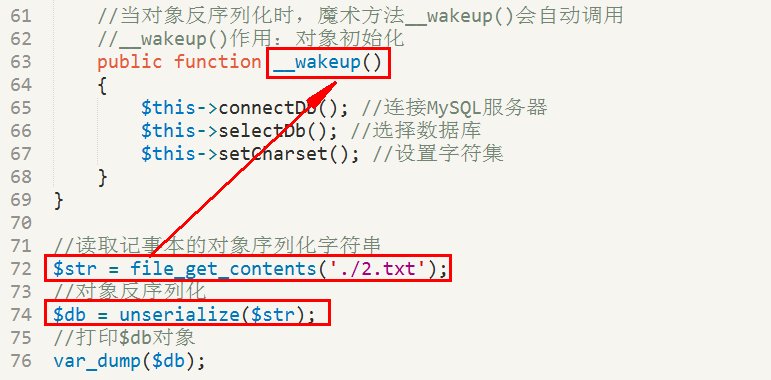
* 对象的序列化过程，与其它变量数据一样；
* 对象序列化的内容只能包含成员属性；
* 当序列化对象时，serialize()函数会检查类中是否存在一个魔术方法 \_\_sleep()。如果存在，该方法会先被调用，然后才执行序列化操作。此功能可以用于清理对象，并返回一个包含对象中所有应被序列化的变量名称的数组。





## 5、对象反序列化

* 对象的反序列化过程，与其它变量数据一样；
* 当对象反序列化时，unserialize()函数会检查类中是否存在一个\_\_wakeup()方法。如果存在，则会先调用 \_\_wakeup 方法，预先准备对象需要的资源。 \_\_wakeup() 经常用在反序列化操作中，进行一些初始化操作，例如重新建立数据库连接，或执行其它初始化操作。



# 常用的魔术常量

* \_\_LINE\_\_：当前行号
* \_\_FILE\_\_：当前文件
* \_\_DIR\_\_：当前目录
* \_\_FUNCTION\_\_：当前函数
* \_\_CLASS\_\_：当前类
* \_\_METHOD\_\_：当前方法
* \_\_NAMESPACE\_\_：当前命名空间