**目 录**

[第一章 ：类和对象的声明 3](#_Toc17565)

[面向对象(Object oritened) 3](#_Toc28369)

[与面向对象相关的单词 3](#_Toc17287)

[面向对象和面向过程的区分 3](#_Toc27845)

[类和对象的概念 3](#_Toc17435)

[类和对象的关系 3](#_Toc10589)

[类和对象的组成 3](#_Toc19064)

[类的声明 3](#_Toc31858)

[类文件的书写规则 3](#_Toc20175)

[类的声明规则 4](#_Toc26183)

[成员属性书写规则 4](#_Toc24037)

[成员方法书写规则 4](#_Toc16367)

[类的使用 4](#_Toc30292)

[实例化对象 4](#_Toc22537)

[对象成员属性的使用 4](#_Toc5357)

[对象成员方法的使用 4](#_Toc9768)

[$this关键字 4](#_Toc4337)

[第二章：面向对象的特性 5](#_Toc21149)

[魔术方法 5](#_Toc12085)

[构造方法 5](#_Toc32132)

[析构方法 5](#_Toc19328)

[操作私有成员的魔术方法 5](#_Toc18240)

[面向对象的三大特性 6](#_Toc12694)

[封装特性 6](#_Toc28104)

[继承特性 7](#_Toc19399)

[常见关键字 8](#_Toc11638)

[对象的串行化 10](#_Toc15122)

[与串行化相关的魔术方法 10](#_Toc18744)

[其他魔术方法 10](#_Toc19718)

[抽象类 11](#_Toc32078)

[抽象方法 11](#_Toc9741)

[方法体 11](#_Toc30975)

[抽象方法作用 11](#_Toc27341)

[抽象类特征 11](#_Toc7911)

[接口 11](#_Toc18873)

[接口的特征 11](#_Toc2256)

[抽象类和接口的区别 12](#_Toc3600)

[多态设计模式 12](#_Toc29491)

[单例设计模式 12](#_Toc28627)

[PDO 13](#_Toc5582)

[学习PDO理由 13](#_Toc12413)

[开启PDO 13](#_Toc23035)

[try...catch 语法 13](#_Toc21133)

[PDO类 13](#_Toc9385)

[PDO操作MySQL数据库 14](#_Toc27149)

[PDOStatement操作数据库 14](#_Toc22819)

# ：类和对象的声明

## 面向对象(Object oritened)

* 面向对象的语法的学习
* 面向对象的编程思想的学习

## 与面向对象相关的单词

* OO -- 面向对象 Object oritened
* OOA -- 面向对象的分析方法 Object oritened Analysis
* OOD -- 面向对象的设计方法 Object oritened Design
* OOI -- 面向对象的实现方法 Object oritened implementation
* OOP -- 面向对象的程序设计(OOA->OOD-OOI) Object oritened programming

## 面向对象和面向过程的区分

### 类和对象的概念

* 类：所有事物的操作流程就是一个类，类是一种思想或者未实现的东西。
* 对象：所有事物的实体表现都是对象不仅仅是概念上的，所有事物的操作流程的实施过程就是一个对象。

### 类和对象的关系

* 对象是由类产生的一个实例的事物或者操作流程，这个过程叫做实例化。
* 类是由多个对象抽象或者说想像总结得来的，这个抽象总结的过程叫做抽化

### 类和对象的组成

* 成员属性：是对特性的描述，在面向对象中叫做成员属性。
* 成员方法：是对其功能的描述，在面向对象中叫做成员方法(就是函数)

## 类的声明

### 类文件的书写规则

* 函数的文件名：xxx.func.php
* 类的文件名：xxx.class.php(单类文件)、现在可以不用加class
* 类的文件名一般与类名相同。
* 类的驼峰命名法是每一个单词的首字母大写，那么意味着类文件名的首字母也要大写。

### 类的声明规则

* 类必须以class关键字开始。
* class关键字之后跟类名，类名通常和文件名相同。
* 类名之后直接跟着{}。

### 成员属性书写规则

* 声明成员属性必须使用关键字var 或者public，private，protected。
* 声明成员属性时，可以赋值，也可以不赋值。
* 声明成员属性时，属性值不能赋值一个函数值。
* 成员属性的最大作用是为了给成员方法提供一个桥梁的作用。

### 成员方法书写规则

* 成员方法的书写其实和函数一样。
* 声明成员方法时，需要使用function关键字。
* 声明成员方法可以传入参数也可以不传入参数。
* 声明成员方法时，方法允许存在返回值。
* 同时需要注意：类中只能出现成员属性和成员方法，不允许出现其他的代码，否则会有语法错误。

## 类的使用

一般情况下，类是无法直接使用的，需要将类实例化对象之后方可使用。

### 实例化对象

* 格式1： 对象变量 = new类名;
* 格式2： 对象变量 = new类名();

### 对象成员属性的使用

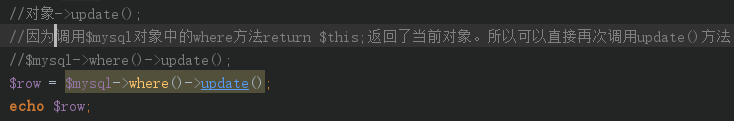
* 对象变量->成员属性

### 对象成员方法的使用

* 对象变量->成员方法名();

### $this关键字

* $this作用：在当前对象的成员方法中调用当前对象的成员属性。
* $this表示当前的对象。
* 同时需要注意：如果代码中具有$this，那么该类必须实例化才能使用。



# 第二章：面向对象的特性

## 魔术方法

在PHP中以两个下划线开头的方法(一种特殊的函数)称之为魔术方法。如果希望PHP调用这些魔术方法，首先必须在类中定义，否则PHP不会执行未创建的魔术方法

注意：PHP将所有以\_\_开头的类方法保留为魔术方法，所以在定义类方法时，除了魔术方法，建议不要以\_\_为前缀.

### 构造方法

* 构造方法：\_\_construct()
* 触发时机：在对象实例化的时候自动触发
* 作用：初始化成员属性
* 参数：可以有，也可以没有，取决于设定和逻辑
* 返回值：没有
* 注意：如果构造方法具有参数且参数没有默认值，在实例化对象的时候必须在类名()内添加实参值。
* 了解：在PHP版本5之前和类名一样的函数也称之为构造方法

### 析构方法

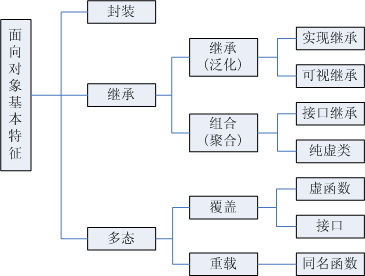
* 析构方法：\_\_destruct()
* 触发时机：在销毁对象的时候自动触发(页面执行完毕时)
* 作用：回收对象使用过程中的资源
* 参数：没有
* 返回值：没有

### 操作私有成员的魔术方法

* \_\_get()方法
* 触发时机：访问私有成员属性的时候自动触发
* 功能：1.防止报错。2.为私有成员属性访问提供后门
* 参数: 1个访问私有成员属性的名称
* 返回值：可以有，可以没有
* \_\_set()方法
* 触发时机:对私有成员属性进行设置的时候自动触发
* 功能：1.屏蔽错误。2.位私有成员属性设置新值
* 参数：2个。1个是私有成员属性名，1个是要设置的新值
* 返回值：无
* \_\_isset()方法
* 触发时机：对私有成员属性进行isset检测时自动触发
* 功能：代替对象外部的isset检测返回结果
* 参数：1个私有成员属性名
* 返回值：一般返回isset(私有属性)的结果
* \_\_unset()方法
* 触发时机：在对象外部对私有成员属性进行unset操作时自动触发
* 功能：代替对象外部的unset操作
* 参数：1个私有成员属性名
* 返回值：无

## 面向对象的三大特性

封装、继承和多态



### 封装特性

在一个对象产生之后，对象的部分成员属性和成员方法逻辑是是不允许在对象外部调用或者使用的，这就是封装特性的由来

* 封装的三个关键字

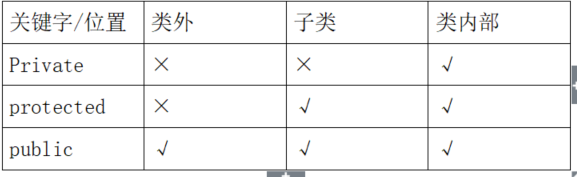
private 私有的

* 被private修饰的成员属性不能够在类的外部进行访问，可以在类的内部进行访问
* 被private修饰的成员方法也不能够在类的外部进行访问，可以在类的内部进行访问
* var和private只能选取一个

protected 受保护的

* 和private类似，但差别在于，private不能被继承而protected可以别继承

public 公共的



* 封装等级
* 子类同名的方法封装等级必须大于或者等于父类的封装等级

private = 1；protected = 2; public = 3;

* 基类中private封装的方法，派生类中能用private、protected、 public来封装。
* 基类中protected封装的方法，派生类中能用protected、public来封装。
* 基类中public封装的方法，派生类中只能用public来封装。
* var 是封装关键字，但是他不是一个标准的封装关键字，一般仅用于测试和学习

### 继承特性

* 继承概念
* 继承的修饰词extends
* 父类：被其他类继承的就是父类，也叫做基类。
* 子类：继承其他类的类就是子类，也叫做派生类。
* 继承格式

class 类名 extends 父类名{

里面可以写独有的子类方法

}

* 继承特点
* 子类继承父类既具有父类中所有的成员属性和方法(私有的除外)
* 子类继承父类之后可以具有子类独有的属性和方法
* 在继承时，子类中允许出现和父类同名的方法，这种情况不会冲突，而是子类的方法会覆盖父类的方法，这种方式叫做重载(重新加载)
* 子类在重写父类方法时，要求参数个数和参数类型必须和父类的方法一致
* 魔术方法也可以被子类继承
* 如果父类具有构造方法，子类也需要构造方法进行构造成员属性，此时需要在子类中重载父类的构造方法，并且在构造方法中使用parent::\_\_construct();进行调用父类的构造方法，构造继承而来的父类的成员属性。也可以使用类名::\_\_construct()来调用父类的构造方法(该方法不推荐)
* 继承代码的时候不要瞎继承，一定要有逻辑关系
* php继承的特点和其他语言略有不同。PHP是单继承的语言，其他语言可能是多继承

### 常见关键字

* final 关键字
* final 英文意思 最终的 最后的
* 作用：
* 用来修饰成员方法。禁止在子类中重载父类的方法
* 用来修饰类。被final修饰的类无法被继承
* const 关键字
* 作用：用于在类中声明常量，这种常量是类中常量不归对象所有，而是归类所有。
* 需要注意：
* 在类中声明常量使用const关键字
* 声明常量时需要给常量赋值
* 声明常量归类所有而不是归任何对象所有
* 类中常量的访问方式
* 类外部：

类名::常量名

* 类内部：

类名::常量名

self::常量名

* 需要记住：

self 表示当前类

$this 表示当前对象



* static 关键字
* 在类中使用static声明的成员都是静态成员。
* 静态成员的特征是：不归对象所有，而是归类所有。
* static关键字，既可以修饰成员属性，也可以修饰成员方法。
* static关键字，声明的成员都是归类所有，而不是归对象所有，在没有对象的情况下，可以通过类来访问。
* 能声明静态的成员就声明静态的成员(效率高)。
* 具有动态成员的方法不可以声明静态成员方法。
* 如果方法中没有$this存在，那么这个方式实际上已经是一个静态方法(php版本大于5.3以后要求写上static关键字)
* 静态成员访问：
* 类内部

成员属性

类名::$静态成员属性名

self::$静态成员属性名

成员方法

类名::静态成员方法名();

self::静态成员方法名();

* 类外部

成员属性

类名::$静态成员属性名

成员方法

类名::静态成员方法名();

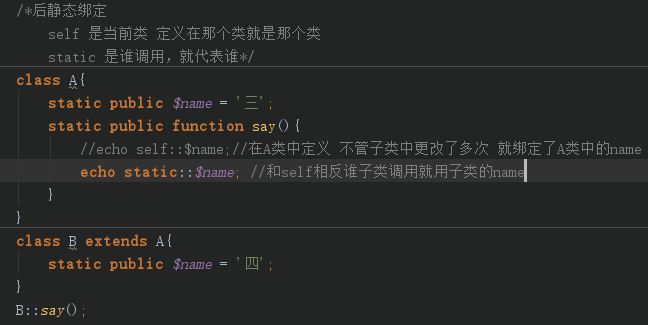
对象->静态成员方法名();（一般不用）



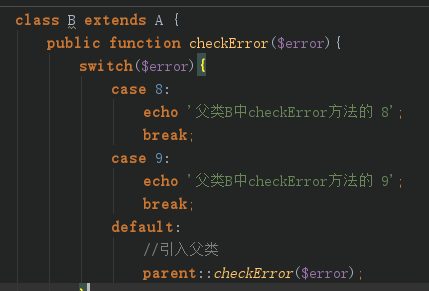
* 后静态绑定：

Self 定义在哪里，就代表那个类，即时继承也代表父类，

Static 谁调用，就代表谁。

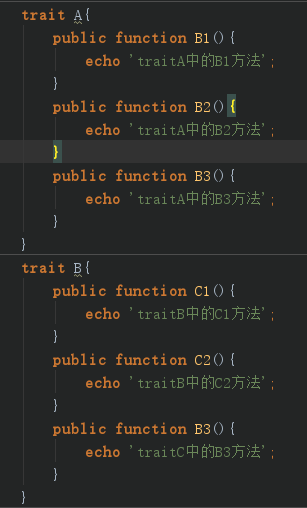
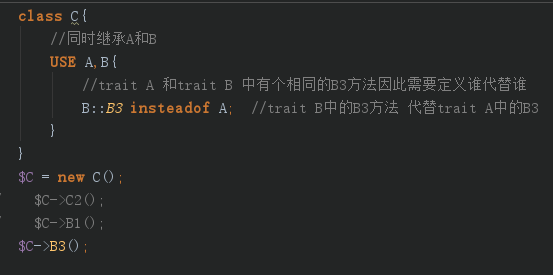


* Parent::方法();
* 在extends过后想在子类中更新该方法（父类的方法不全，要在子类中更新，但有不能在将父类的方法重写一遍）时采用parent::方法()；



* trait
* 作用：是为了解决PHP单继承的问题。
* 用法：trait A{} trait B{} 代替class 来写 class C中用USE A,B{}来同时继承A和B，如果trait A{} 和trait B{} 中同时都有一个相同的方法时，需要在USE A,B{}中来定义谁的优先级高（谁代替谁）

B::方法名() insteadof A 即 trait B中的方法代替trait A 中的方法。 也可用as 方法给其中一个起一个别名（查手册）

### 对象的串行化

将对象变成一个字符串的过程就是对象串行化。

* 作用
* 方便长久的保存对象。
* 方便在网络上进行传输。

### 与串行化相关的魔术方法

* \_\_sleep() 方法
* 触发时机：在对象进行串行化操作时自动触发。
* 功能：指定需要串行化的成员属性。
* 参数：无。
* 返回值：需要返回一个数组类型，每个元素都是需要串行化的成员属性名。
* \_\_wakeup() 方法
* 触发时机：在对象进行反串行化时自动触发
* 功能：反串行化时自动修改部分成员属性值
* 参数：无
* 返回值：无
* 串行化操作：字符串变量 = serialize(对象变量);
  + serialize 串行化可以规定对象串行化时那些属性保留
* 反串行化操作：对象变量 = unserialize(字符串变量);

### 其他魔术方法

* \_\_tostring() 方法
* 触发时机：对一个对象进行echo操作(当做字符串处理)时自动触发
* 功能：简化操作或者为对象输出值
* 参数：无
* 返回值：必须有，并且必须为字符串类型。
* \_\_invoke() 方法
  + 触发时机：对一个对象进行函数操作时自动触发
  + 功能：
  + 参数：无
  + 返回值：无
* \_\_clone() 方法
* 触发时机：对对象进行克隆操作时自动触发
* 功能：修改克隆之后的对象的部分成员属性。
* 参数：无
* 返回值：无
* 注意：对象的赋值默认是引用赋值，不能通过赋值操作获取一个新的对象，获取新对象的操作应该使用克隆操作或者new
* 克隆格式：新变量 = clone 对象变量
* \_\_call() 方法
* 触发时机：调用类中非公有或不存在的方法时(私有或不存在)，自动触发
* 功能：1.屏蔽错误。2.为不存在的方法定义一个功能
* 参数：2个 一个是不存在的方法名，一个传入不存在方法的所有参数列表
* 返回值：可有可无
* \_\_autoload() 方法
* 触发时机：当页面需要一个类而该类不存在时自动触发
* 功能：实现类的自动加载
* 参数：一个参数(不存在的类名)
* 返回值：无
* \_\_callStatic() 方法
* 触发时机:访问非公或不存在的静态成员方法时触发
* 细节：该魔术方法的本身就是static静态的，如果存在同名的普通方法，该方法就不会触发

## 抽象类

具有抽象方法的类就是抽象类。

### 抽象方法

没有方法体的方法就是抽象方法。

### 方法体

方法后面花括号内就是方法体。

### 抽象方法作用

协同工作、制定标准（领导用的~）

### 抽象类特征

* 抽象方法就是没有方法体的方法，方法体相当于函数体，抽象方法前，必须使用 abstract 关键字，且方法后不能写花括号({})，但是要写分号(;)结束。
* 如果类中有抽象方法，那么这个类前面也必须使用abstract 关键字进行修饰。
* 抽象类中可以具有非抽象方法。
* 抽象方法可以有参数，但不能有返回值。要实现该抽象方法，必须传入同样的参数。
* 抽象类不能直接使用(不能被实例化)，必须由其他类继承且完全实现出所有的抽象方法，才可以使用。抽象类主要是给其他类进行继承操作。
* 抽象类可以继承抽象类。
* 抽象类中定义抽象方法时，抽象方法不可以使用private关键字。

## 接口

如果一个类没有任何的非抽象方法(所有的方法都是抽象方法)，且不存在常量之外的任何成员属性(没有成员属性,常量除外)，那么这个类就是接口。

### 接口的特征

* 声明一个接口不使用class 关键字而是使用interface关键字。
* 接口中的抽象方法不能接abstract 关键字。
* 接口只能用public关键字，不可以使用protected 或者private
* 接口中不允许有任何成员属性，除了常量。
* 接口的继承不是使用extends，而是使用implements关键字。
* extends 只能继承一个普通类或者是一个抽象类，而implements可以同时继承多个接口，从而模拟多继承。

### 抽象类和接口的区别

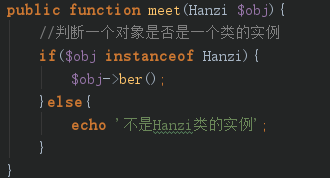
* 抽象类的成员可以具有访问级别，而接口的成员全部为public级别。
* 抽象类可以包含成员属性，而接口只可以有常量。
* 抽象类可以继承接口，而接口不能继承抽象类。
* 抽象类的成员具有具体的实现，而接口不行。
* 抽象的子类可以选择性实现基类的抽象方法，而接口的子类必须实现。

### 多态设计模式

就是不同的人做相同的事得到不同的结果

* instanceof 运算符

判断是不是一个类的实例化



### 单例设计模式

最终的结果只能实例化一个对象

* 比如：（A）



（B）不用单例模式时，该类调用一次就成生一个空间初始化一次静态变量。而单例设计是利用静态变量只能初始化一次且不用new来调用的特性，如果没有实例化那么静态变量是空就实例化一次，并且将实例化的对象赋值给静态属性。如果判断一次静态成员有属性就将原来的返回出去。



## 命名空间

作用：是为了解决 常量 函数 类的名称相同的问题

关键字：namespace(设定) use(选择)

### namespace

命名空间之前不能包括任何内容(namespace 之前)

1. 不能有变量的定义
2. html、空格
3. 文件的编码不能是有BOM格

### use

use 命名空间\类名

* + 1. 有些调用系统类(如：PDO等)因在跟目录下，所以可直接USE PDO;
    2. 在其他命名空间下，如果没有use定义，可杂实例化时将路径加上（）。
    3. 补充

## PDO

PDO一是PHP数据对象（PHP Data Object）的缩写，它是数据库抽象层的一种。

### 学习PDO理由

* 以模块的方式直接打入系统，执行效率高、防止数据注入。
* 官方推荐使用PDO作为数据库抽象层。
* PHP版本从5.3之后都是默认开启PDO。

### 开启PDO

PDO是一个模块，因此需要时还必须开启。

* 找到配置文件php.ini

Extension=php\_pdo\_mysql.dll (PDO所支持的数据库) 修改开启。

* 5.3版本之前的
* 查看 Extension=php\_pdo.dll 是否开启，当开启后才代表支持PDO。
* 重启apache服务器。
* 检测phpinfo中对应模块是否开启。

### try...catch 语法

PDO是建立在try...catch语法之上

* 格式：

try{ //尝试执行某代码

}catch(){

}



### 异常处理Exception类

Exception 是一个系统类，在错误发生时，把每一个错误看成一个对象，单独输出错误。其核心就是错误代码和正常业务逻辑分离

* throw new Exception
* throw new Exception(‘错误提示信息’，错误编码);
* 定义报错
* getMessage() 报错时定义的错误提示信息
* getCode() 报错时定义的错误编码
* getFile() 报错时的文件名
* getLine() 报错的错误行码
* 错误处理函数
  + set\_exception\_handler() 设置由哪个函数来处理异常 先设置哪个函数处理在抛出否知会报错
  + restore\_exception\_handler() 将异常交给原来的处理函数

### PDO类

* [PDO::beginTransaction](mk:@MSITStore:C:\\WINDOWS\\system32\\p.chm::/res/pdo.begintransaction.html) — 启动一个事务

启动事务回滚。启动前需要将自动提交关闭(setAttlribute()中设置)。

* PDO::commit 提交一个事物
* PDO::rollBack 回滚一个事物

使用完事物回滚都，需要在将自动提交开启。

* [PDO::\_\_construct](mk:@MSITStore:C:\\WINDOWS\\system32\\p.chm::/res/pdo.construct.html) — 创建一个表示数据库连接的 PDO 实例
* [PDO::errorCode](mk:@MSITStore:C:\\WINDOWS\\system32\\p.chm::/res/pdo.errorcode.html) — 错误代码
* [PDO::errorInfo](mk:@MSITStore:C:\\WINDOWS\\system32\\p.chm::/res/pdo.errorinfo.html) — Fetch extended error information associated with the last operation on the database handle —错误信息
* [PDO::exec](mk:@MSITStore:C:\\WINDOWS\\system32\\p.chm::/res/pdo.exec.html) — 执行一条 SQL 语句(增、删、改)，并返回受影响的行数
* [PDO::getAttribute](mk:@MSITStore:C:\\WINDOWS\\system32\\p.chm::/res/pdo.getattribute.html) — 取回一个数据库连接的属性
* PDO::ATTR\_AUTOCOMMIT 检测是否开启自动提交
* PDO::ATTR\_CASE 获取大小写转换设置
* PDO::ATTR\_CLIENT\_VERSION 获取客户端信息
* PDO::ATTR\_CONNECTION\_STATUS 获取链接状态
* PDO::ATTR\_DRIVER\_NAME 获取驱动名称
* PDO::ATTR\_ERRMODE 获取错误模式
* PDO::ATTR\_ORACLE\_NULLS 获取ORACLE空值转换设置
* PDO::ATTR\_PERSISTENT 获取当前连接是否为持久链接
* PDO::ATTR\_PREFETCH 获取数据的预计大小（MYSQL不支持）
* PDO::ATTR\_SERVER\_INFO 查看服务器信息
* PDO::ATTR\_SERVER\_VERSION 查看服务器的版本
* PDO::ATTR\_TIMEOUT 获取超时设置（MYSQL 不支持）
* [PDO::getAvailableDrivers](mk:@MSITStore:C:\\WINDOWS\\system32\\p.chm::/res/pdo.getavailabledrivers.html) — 获取PDO支持的数据库类型
* [PDO::inTransaction](mk:@MSITStore:C:\\WINDOWS\\system32\\p.chm::/res/pdo.intransaction.html) — 检查是否在一个事务内
* [PDO::lastInsertId](mk:@MSITStore:C:\\WINDOWS\\system32\\p.chm::/res/pdo.lastinsertid.html) — 返回最后插入行的ID或序列值
* [PDO::prepare](mk:@MSITStore:C:\\WINDOWS\\system32\\p.chm::/res/pdo.prepare.html) — Prepares a statement for execution and returns a statement object —准备SQL语句 返回pdo预处理对象。
* [PDO::query](mk:@MSITStore:C:\\WINDOWS\\system32\\p.chm::/res/pdo.query.html) — Executes an SQL statement, returning a result set as a PDOStatement object —执行查询SQL语句
* [PDO::quote](mk:@MSITStore:C:\\WINDOWS\\system32\\p.chm::/res/pdo.quote.html) — Quotes a string for use in a query. —字符串转译
* [PDO::setAttribute](mk:@MSITStore:C:\\WINDOWS\\system32\\p.chm::/res/pdo.setattribute.html) — 设置属性
* 设置属性需要两个参数，详情可查手册

PDO::ATTR\_CASE：强制列名为指定的大小写。

* PDO::CASE\_LOWER：强制列名小写。 (强制转小写)
* PDO::CASE\_NATURAL：保留数据库驱动返回的列名。 (默认，不转换)
* PDO::CASE\_UPPER：强制列名大写。 (强制转大写)

PDO::ATTR\_ERRMODE：错误报告。

* + - PDO::ERRMODE\_SILENT： 仅设置错误代码。(寂静模式、推荐使用)
    - 什么叫做寂静模式：
    - 有错误不输出
    - PDO::ERRMODE\_WARNING: 引发 [E\_WARNING](mk:@MSITStore:C:\\手册\\php_5.4.chm::/res/) 错误(使用系统警告的方式报错，不推荐使用该方式，会在网页中直接显示错误)
    - PDO::ERRMODE\_EXCEPTION: 抛出 [exceptions](mk:@MSITStore:C:\\手册\\php_5.4.chm::/res/class.pdoexception.html) 异常。(有错误后抛出异常模式，推荐使用)

PDO::ATTR\_AUTOCOMMIT 自动提交

### PDO操作MySQL数据库

* 链接数据库
* new PDO($dsn,’root’,’密码’);

//实例化PDO类,中的构造方法会链接数据库需要传入对应的参数。

$dsn data source name 数据原名 需要在手册中查找

* 判读数据库是否链接成功
* catch(PDOException $e)

//这里使用的是PDO预处理类，该类会自动抛出，因此不用try手动抛出，try只要出现错误就会自动跳过来执行catch。

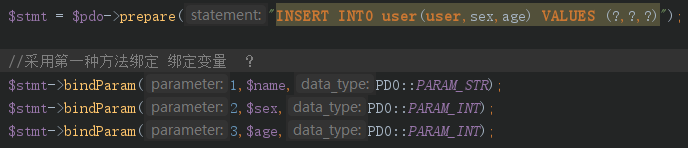
* 选择数据库
* 数据库的选择在链接数据库的时候$dsn 里面的dbname=’数据库名’;
* 设置字符集
* 设置字符集就好比向数据库发送一条命令，这里需要使用PDO中的exec()方法
* 对象->exec(“SET NAMES utf8”);
* 准备SQL语句
* 判断并且处理结果
* 增、删、改时使用exec();方法来执行，且会返回受影响的行数。
* 查询时使用query();方法来执行
* 关闭数据库
* PDO会自动关闭数据库，这里不需要自己关闭。

### PDOStatement操作数据库

该类为PDO的预处理类使用该类不需要实例化，采用PDO::prepare准备SQL语句，其返回值就是PDO的预处理对象。(其实不用记这么多用到那个查手册就行)

* [PDOStatement::bindColumn](mk:@MSITStore:C:\\WINDOWS\\system32\\p.chm::/res/pdostatement.bindcolumn.html) — 绑定一列到一个 PHP 变量
* [PDOStatement::bindParam](mk:@MSITStore:C:\\WINDOWS\\system32\\p.chm::/res/pdostatement.bindparam.html) — 绑定一个参数到指定的变量名
* [PDOStatement::bindValue](mk:@MSITStore:C:\\WINDOWS\\system32\\p.chm::/res/pdostatement.bindvalue.html) — 把一个值绑定到一个参数
* [PDOStatement::closeCursor](mk:@MSITStore:C:\\WINDOWS\\system32\\p.chm::/res/pdostatement.closecursor.html) — 关闭游标，使语句能再次被执行。
* [PDOStatement::columnCount](mk:@MSITStore:C:\\WINDOWS\\system32\\p.chm::/res/pdostatement.columncount.html) — 返回结果集中的列数
* [PDOStatement::debugDumpParams](mk:@MSITStore:C:\\WINDOWS\\system32\\p.chm::/res/pdostatement.debugdumpparams.html) — 打印一条 SQL 预处理命令
* [PDOStatement::errorCode](mk:@MSITStore:C:\\WINDOWS\\system32\\p.chm::/res/pdostatement.errorcode.html) — 获取跟上一次语句句柄操作相关的 SQLSTATE
* [PDOStatement::errorInfo](mk:@MSITStore:C:\\WINDOWS\\system32\\p.chm::/res/pdostatement.errorinfo.html) — 获取跟上一次语句句柄操作相关的扩展错误信息
* [PDOStatement::execute](mk:@MSITStore:C:\\WINDOWS\\system32\\p.chm::/res/pdostatement.execute.html) — 执行一条预处理语句
* [PDOStatement::fetch](mk:@MSITStore:C:\\WINDOWS\\system32\\p.chm::/res/pdostatement.fetch.html) — 从结果集中获取下一行
* [PDOStatement::fetchAll](mk:@MSITStore:C:\\WINDOWS\\system32\\p.chm::/res/pdostatement.fetchall.html) — 返回一个包含结果集中所有行的数组
* [PDOStatement::fetchColumn](mk:@MSITStore:C:\\WINDOWS\\system32\\p.chm::/res/pdostatement.fetchcolumn.html) — 从结果集中的下一行返回单独的一列。
* [PDOStatement::fetchObject](mk:@MSITStore:C:\\WINDOWS\\system32\\p.chm::/res/pdostatement.fetchobject.html) — 获取下一行并作为一个对象返回。
* [PDOStatement::getAttribute](mk:@MSITStore:C:\\WINDOWS\\system32\\p.chm::/res/pdostatement.getattribute.html) — 检索一个语句属性
* [PDOStatement::getColumnMeta](mk:@MSITStore:C:\\WINDOWS\\system32\\p.chm::/res/pdostatement.getcolumnmeta.html) — 返回结果集中一列的元数据
* [PDOStatement::nextRowset](mk:@MSITStore:C:\\WINDOWS\\system32\\p.chm::/res/pdostatement.nextrowset.html) — 在一个多行集语句句柄中推进到下一个行集
* [PDOStatement::rowCount](mk:@MSITStore:C:\\WINDOWS\\system32\\p.chm::/res/pdostatement.rowcount.html) — 返回受上一个 SQL 语句影响的行数
* [PDOStatement::setAttribute](mk:@MSITStore:C:\\WINDOWS\\system32\\p.chm::/res/pdostatement.setattribute.html) — 设置一个语句属性
* [PDOStatement::setFetchMode](mk:@MSITStore:C:\\WINDOWS\\system32\\p.chm::/res/pdostatement.setfetchmode.html) — 为语句设置默认的获取模式。
* 预处理中的三种绑定方法：
* 第一种：变量问号绑定

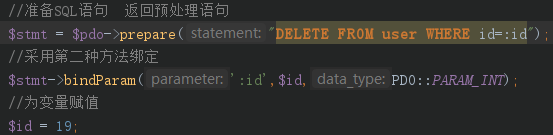
变量用问号占位，然后调用预处理对象中的bindParam()方法来绑定



(增)

* 第二种：变量冒号绑定

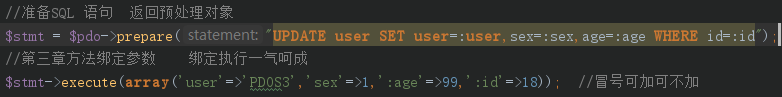
变量用冒号和随意名占位，然后调用预处理对象中的bindParam()方法绑定



(删)

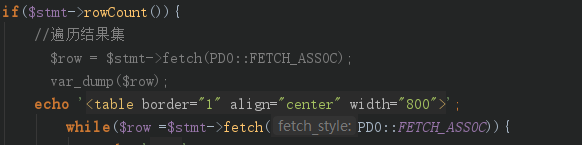
* 第三种：冒号数组绑定

和第二种一样用冒号和随意名占位，然后调用预处理对象中的execute()绑定执行一气呵成

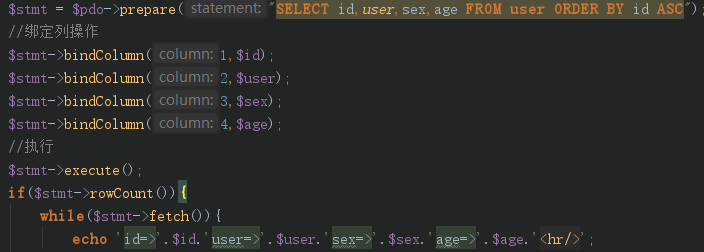


(改)

* 预处理中查的三种方法：
* 第一种：使用fetch()方法 参数看手册



* + - 第二种：使用fetch()加绑定的方法



* 第三种：使用fetchAll()方法



### 大数据库存储

* 图片的存储就是典型的大数据库存储，这里图片存储数据库采用的是二进制存储，建表是的数据类型要选着BOLB类型。
* 首先要在链接抛出异常之后，采用file\_get\_contents();读取图片的信息，然后和其他数据存储一样，将其存入数据库。
* 显示图片到浏览器是不要忘记，用header();通知浏览器相应的图片类型。