php

会话控制，面向对象，数据传输

一、会话控制

什么是会话控制

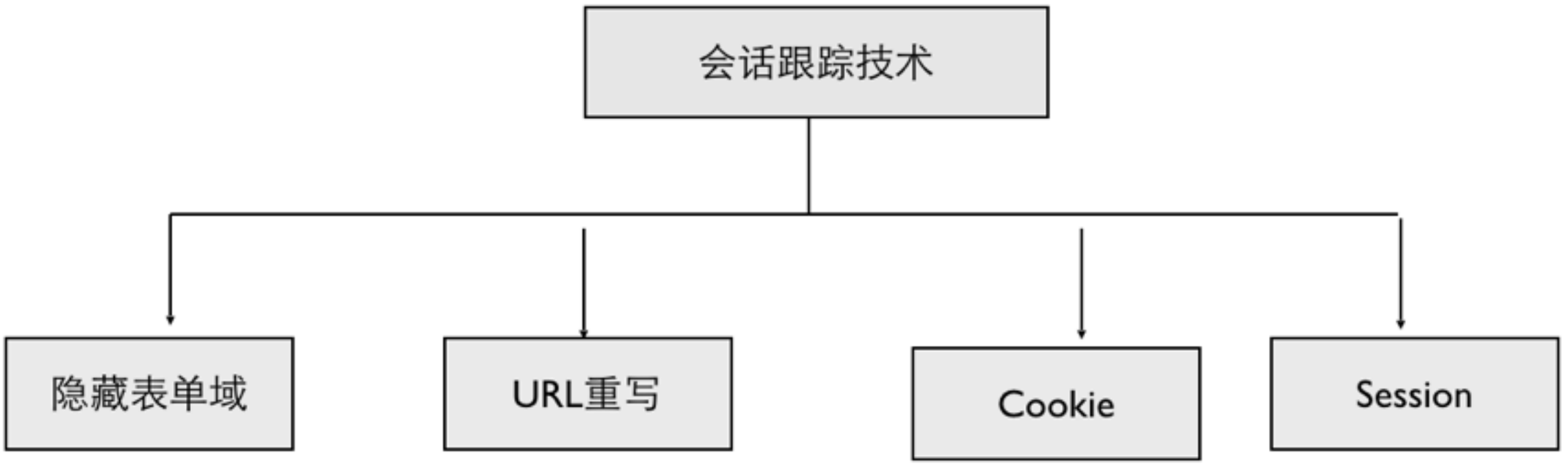
为了使得网站可以跟踪客户端与服务器之间的交互，保存和记忆每个用户生身份和信息，我们需要一种强有力的解决方案，这样就产生了会话控制。

HTTP是一个无状态的协议，此协议无法来维护两个事务之间的联系。

当一个用户请求一个页面后再请求另外一个页面时，HTTP无法告诉我们这两个请求是来自同一个人。

会话控制思想就是能够在网站中跟踪一个变量，我们可以跟踪变量，就可以做到对用户的支持，并根据授权和用户身份显示不同内容，不同页面。

会话控制技术



cookie

cookie是在http协议下，服务器或脚本可以维护客户端信息的一种方式。

cookie是web服务器保存在用户浏览器上的小甜饼(一个很小的文本文件)，她可以包含有关用户的信息，常用于保存用户名，密码，个性化设置，个人偏好记录等。

当用户访问服务器时，服务器可以设置和访问cookie的信息。

cookie保存在客户端，通常是IE或Firefox浏览器的cookie临时文件夹中，可以手动删除。

cookie是利用了网页代码中的HTTP头信息进行传递的，浏览器的每一次网页请求，都可以伴随Cookie传递。

注意：如果浏览器上cookie 太多，超过了系统所允许的范围，浏览器也会自动对它进行删除。

cookie工作原理

当客户访问某个基于PHP技术的网站时，在PHP中可以使用setcookie()函数生成一个cookie，系统经处理把这个cookie发送到客户端并保存在C:\Documents and Settings\用户名\Cookies目录下。

cookie是 HTTP头的一部分, 因此setcookie()函数必须在HTML本身的任何内容送到浏览器之前调用。这种限制与header()函数一样。

当客户再次访问该网站时，浏览器会自动把C:\Documents and Settings\用户名\Cookies目录下与该站点对应的cookie发送到服务器，服务器则把从客户端传来的cookie将自动地转化成一个PHP变量。在PHP5中，客户端发来的cookie将被转换成全局变量。可以通过$_COOKIE['xxx']读取。

设置cookie

bool setcookie(string name,[string value,[int expire,[string path,[string domain,[int secure]]]]]);

name　　设置cookie的名字.（必须）

value　　设置cookie的值.

expire　　设置cookie的过期时间和日期，用一个标准的Unix时间标记，可以用time()函数取得，以秒为单位.

可选参数path, domain, secure

path :服务器端的有效路径，设置为“/”表示这个域中所有目录都可以被访问读取

domain :设定cookie有效域名

secure :设置是否仅在https安全连接时才发送cookie到客户端，0或1。

例：setcookie(“username”, “admin”, time() + 60 * 60);

使用cookie

PHP对cookie有很好的支持，和form表单一样，在接收的时候PHP会自动从web服务器接收HTTP头并且分析它。接收的时候和表单接收一样

例：echo $_COOKIE["username"];

删除cookie

要删除一个已经存在的cookie，有两个办法：

1、调用只带有name参数的setcookie，那么名为这个cookie将被从客户机上删除；

例：setcookie(“MyCookie”);

2、设置Cookie的失效时间为time()或time()-1

例： setcookie("username","admin",time()-1);

注意:

time()减多少没有关系，只要是过期时间就行，那么这个Cookie在这个页面的浏览完之后就被删除了 （其实是失效了）。 当一个Cookie被删除时，它的值在当前页仍然有效。如果要把cookie设置成在浏览器关闭后就失效。那么可以直接把expiretime设为0，或者不设置此值。例:setcookie("name","value",0)。

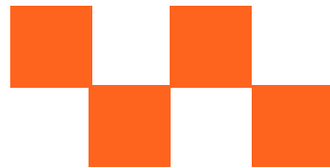
使用cookie注意事项

setcookie()之前不能有任何html输出，就是空格，空白行都不行。

setcookie()后，在当前页调用echo $_COOKIE[“name”]不会有输出。必须刷新或到下一个页面才可以看到cookie值。

使用cookie的限制。一个浏览器能创建的cookie数量最多为300个，并且每个不能超过4KB，每个WEB站点能设置的cookie总数不能超过20个。

cookie是保存在客户端的，用户禁用了cookie，你的cookie自然也就没作用啦！



一、Session

什么是session

Session从用户访问页面开始，到断开与网站连接为止，形成一个会话的生命周期。在会话期间，分配客户唯一的一个SessionID，用来标识当前用户，与其他用户进行区分。

Session会话时，SessionID会分别保存在客户端和服务器端两个位置，对于客户端使用临时的Cookie保存（Cookie名称为PHPSESSID）或者通过URL字符串传递，服务器端也以文本文件形式保存在指定的Session目录中。

Session通过ID接受每一个访问请求，从而识别当前用户、跟踪和保持用户具体资料，以及Session变量（在Session活动期间，可在Session中存储数字或文字资料），比如session_name等等，这些变量信息保存在服务器端。

SessionID可以作为会话信息保存到数据库中，进行Session持久化，这样可以跟踪每个用户的登陆次数、在线与否、在线时间等。

使用session的步骤

1.开始会话

2.注册会话变量

3.使用会话变量

4.注销变量并销毁会话

1.开始会话

session_start() 开始一个会话或者返回已经存在的会话。

说明：这个函数没有参数，且返回值均为true。如果你使用基于cookie的session，那么在使用session_start()之前浏览器不能有任何输出，否则会发生以下错误：

Warning: Cannot send session cache limiter - headers already sent (output started at /var/www/html/test.php:2)

也可以在php.ini里启动session.auto_start=1，这样就无需每次使用session之前都要调用session_start()。启用此指令的缺点是无法在会话中存储对象，因为为定义要在会话开始之前加载才能重新创建对象。

2.注册、使用会话变量

PHP5使用$_SESSION[‘xxx’]=xxx 注册SESSION全局变量。和GET，POST，COOKIE的使用方法相似。

例：
```
session_start();
$_SESSION[‘username’] = "david";
echo “Your username is “.$_SESSION[‘username’];
```

3.注销会话变量

例：session_start();

unset($_SESSION['username']);

4.销毁会话

session_unset()

session_unset()函数清除存储在当前会话中的所有变量，它能有效地将会话重置为创建时的状态。

session_destroy()

session_destroy()函数从存储机制中完全删除会话，使当前会话失效。

会话配置

php.ini配置文件中有一组会话配置选项，可以对其进行设置：

session.save_handler = files ; 如何存储session信息

session.save_path = /tmp; save_handler 设为文件时， session文件保存的路径

session.use_cookies = 1 ; 是否使用cookies

session.name = PHPSESSID； 用在cookie里的session的名字

session.auto_start = 0 ; 是否自动启动session

session.cookie_lifetime = 0 ; 设置会话cookie的有效期，以秒为单位，为0时表示直到浏览器被重启

session.cookie_path = / ; cookie的有效路径

session.cookie_domain = ; cookie的有效域

session.cache_expire = 180 ; 设置缓存中的会话文档在 n 分钟后过时

cookie和session的区别

cookie和session都可以暂时保存多个页面中使用的变量，但是它们有本质的差别：

cookie存放在客户端浏览器中;session保存在服务器上;

它们之间的联系是session ID 一般保存在cookie中，或者放在URL上。

当客户端禁用cookie时(点击IE中的“工具”—“Internet选项”，在弹出的对话框里点击“安全”—“自定义级别”项，将“允许每个对话COOKIE”设为禁用)，session_id将无法传递，此时session失效。不过php5在linux/unix平台可以自动检查cookie状态，如果客户端设置了禁用，则系统自动把session_id附加到url上传递。windows主机则无此功能。

三、面向对象

编程思维的分类

1.过程式编程

初学编程的方法通常由顺序结构开始。这是步骤式的过程性编程，过程式编程方法下的制成品，是一个“大胖子”，为什么呢？假设拆开这个制成品，里面是无数纠缠不清的程序和数据（变量等），数据是给各程序共享的。即任何程序都可以读取或修改它，一个程序接着另一个程序来执行。假设要修改这个制成品，就有一种触一发牵动全身的感觉，例如改了这个程序，可能会影响其他的程序。

2. OOP(Object-Oriented Programming）面向对象编程

面向对象程序设计的诞生为开发策略带来的极大的改变，使编程的注意力重新从应用程序的逻辑回到其数据上来。换句话说，OOP将焦点从过程式编程转向最终建模的真实实体。这使得应用程序更接近我们周围的现实世界。

OOP达到了软件工程的三个目标：重用性、灵活性和扩展性。

采用面向对象方法可以使系统各部分各司其职、各尽所能; 使其编程的代码更简洁、更易于维护，并且具有更强的可重用性。

类和对象简介

1、类 (class)

日常环境由无数实体组成:植物,人群,交通工具,食物...

每个实体都由一组性质和行为来定义。

例如： 男人 可以定义有：身高,体重,是否帅，肤色等性质，并且定义有能赚钱，能下厨，能开车等行为。在OOP术语中，实体的性质和行为的具体定义称为类

2、对象 (object)

通过类创建出来的实体称为对象。

对象是系统中用来描述客观事物的一个实体。

它是构成系统的一个基本单位,数据与代码都被捆绑在一个实体中。一个对象由一组属性和对这组属性进行操作的一组行为组成。

从更抽象的角度来说，对象是问题域或实现域中某些事物的一个抽象，它反映该事物在系统中需要保存的信息和发挥的作用；它是一组属性和有权对这些属性进行操作的一组行为的封装体。客观世界是由对象和对象之间的联系组成的。

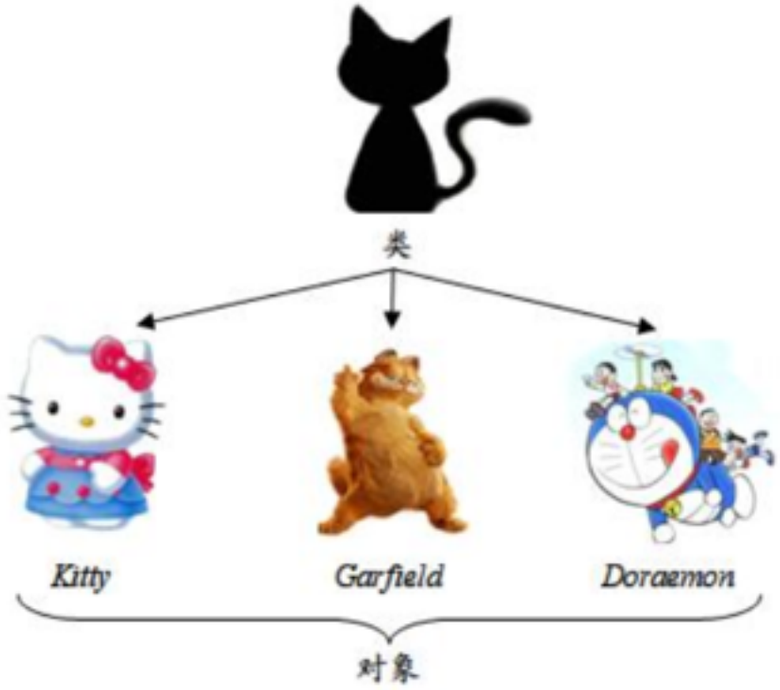
类和对象的关系

类与对象的关系就如模具和铸件的关系，**类的实例化结果就是对象，而对对象的抽象就是类**.类描述了一组有相同特性（属性）和相同行为（方法）的对象。

类

⇩

对象



php的面相对象

PHP4开始提供了面向对象功能，但存在许多不足。从PHP4到PHP5是一次全新革新，PHP5已经完全支持面向对象，对PHP面向对象功能大幅改进和提高。

1、定义类

格式：

```
class classname  [extends parent class] {
        var  property = value;                    //属性
        function functionname ( args ){    //方法
                //代码
        }
```

关于对象

1.创建对象（实例化）

创建对象使用 new 关键字

例： $e = new employee();

2.属性

属性用于描述类某个方面的性质,它与一般的PHP变量非常相似。因为PHP是弱类型的语类，属性甚至不需要声明;但不建议这么做。相反，常见的做法是在类开始处声明属性，可以为属性赋初值 。

例：class employee {

public $name;

public $age;

}

访问修饰符

访问修饰符允许开发人员对类成员的访问进行限制,这是PHP5的新特性。

2.1.public 公共修饰符

类的成员将没有访问限制，所有的外部成员都可以访问（读和写）这个类成员。

在属性或方法前面加上关键字public,或不加任何关键字，都可以声明一个公共属性或方法。

2.2.private 私有修饰符

被定义为private的成员，对于同一个类里的所有成员是可见的，即没有访问限制；但对于该类的外部代码是不允许改变甚至读操作，对于该类的子类，也不能访问。

2.3.protected 保护修饰符

被修饰为protected的成员不能被该类的外部代码访问。但是对于该类的直接子类有访问权限，可以进行属性、方法的读及写操作。

被子类继承的protected成员，在子类外部同样不能被访问

3.方法

方法与函数非常相似，只不过方法用来定义特定类的行为。与函数一样，方法可以接受输入参数，可以向调用者返回一个值。

格式：

```
scope function functionname() {
        //funciton body
}
```

例：

```
public function say() {
        echo ‘hello’;
}
```

访问对象中的成员

PHP 对象中的成员有两种:一种是成员属性，一种是成员方法;

怎么去使用对象的成员呢？要想访问对象中的成员就要使用一个特殊的操作符“->”来完成对象成员的访问。

例：$e = new employee();

$e->name = ‘david’;

echo $e -> name;//访问对象属性

$e->say();　　//访问对象方法

const常量

可以在类中定义常量，即不会在类中改变的值。对于从该类实例化的任何对象来说，常量值在这些对象的整个生命周期中都保持不变。

类常量如下创建：

const NAME = ‘value’;

例：假设定义一个与数学有关的类，其包括一些定义数学函数的方法以及常量

```
class math_function {
    const PI = 3.1415926;
}
访问常量
echo self::PI;                    //类内部访问
echo math_function::PI;                    //类外部访问
```

static方法及属性

使用static 关键字可以用来标识成员属性,也可以用来标识成员方法,

创建方式如下:

```
static $test_static = 'value'; //属性

static function static_method(){ //方法

}
```

❖声明类属性或方法为静态，就可以不实例化类而直接访问

❖静态方法不需要通过对象即可调用，所以伪变量 $this 在静态方法中不可用。

❖静态属性不可以由对象通过 -> 操作符来访问。

```
class Test{
        static $a='static';
        static function m(){
                return self::$a; //类内部访问
        }
}
    echo Test::$a;  //外部访问属性
    echo Test::m(); //外部访问方法
```

静态的成员属于类所有，所以我们在静态方法里，不能使用$this 来引用 静态成员,建议使用 self 关键字来调用。

构造函数和析构函数

1.构造函数

通常我们希望在对象实例化时可以初始化某些属性，或执行某些方法。当然，可以在对象实例化之后再这么做，不过，如果能在实例化的时候自动完成这些操作会更方便，oop就有这样一种机制，称为构造函数;

声明构造函数格式:

```
function __construct([arg1, arg2,…]) {
        // class init code
}
```

例：

```
public function __construct() {
        $this->say();
    }
```

2.析构函数

在PHP4中没有析构函数，php5中引入了析构函数。析构函数允许在销毁一个类之前执行的一些操作或完成一些功能，这些操作或功能通常在所有对该类的引用都被重置或超出作用域时自动发生。

例：function __destruct() {

echo 'class instance destroyed';

}

面向对象的三大特征

1.封装

封装就是把对象的属性和行为结合成一个独立的相同单位，并尽可能隐蔽对象的内部细节。

信息隐蔽，即尽可能隐蔽对象的内部细节，对外形成一个边界〔或者说形成一道屏障〕，只保留有限的对外接口使之与外部发生联系。

封装的原则在软件上的反映是：要求使对象以外的部分不能随意存取对象的内部数据（属性），从而有效的避免了外部错误对它的“交叉感染”，使软件错误能够局部化，大大减少查错和排错的难度。

类的封装性带来的优点：隐藏类的实现细节，让使用者只能通过事先定义好的方法来访问数据，可以方便的加入逻辑控制，进行数据检查,限制对属性的不合理操作。便于修改增强代码的可维护性。

例：Class employee {

```
    private $name;
    private $age;
    function get_age() {
        return $this->age;
    }
    function set_age($age) {
        if($age < 0 || $age > 130){
            return;
        }
         $this->age = $age;
    }
}
```

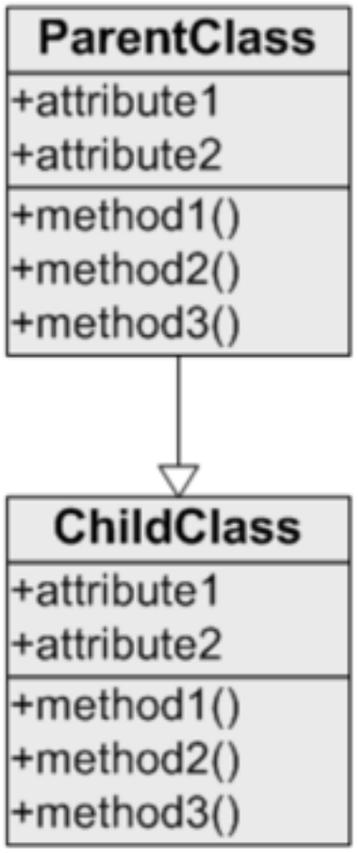
2.继承

面向对象开发方法建立在继承概念的基础上，这种策略提高了代码的重用性。

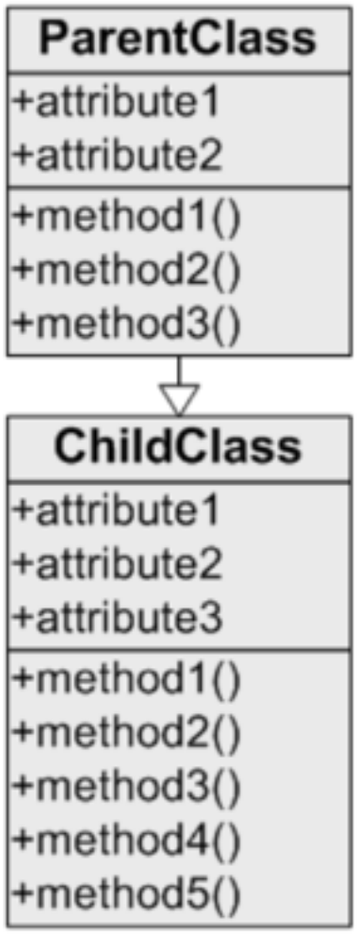
继承是指建立一个新的派生类，从先前定义的类中继承属性和方法，而且可以重新定义或加进新的属性和方法，从而建立类的层次或等级。说的简单点就是，继承是子类自动共享父类的数据结构和方法的机制，这是类之间的一种关系。在定义和实现一个类的时候，可以在一个已经存在的类的基础之上来进行，把这个已经存在的类所定义的内容作为自己的内容，并加入若干新的内容。

在PHP中，类继承通过 extends 关键字实现;
继承自其他类的类称为子类（child class 或 subclass)
子类所继承的类称为父类(parent class) 或 基类 (base class)

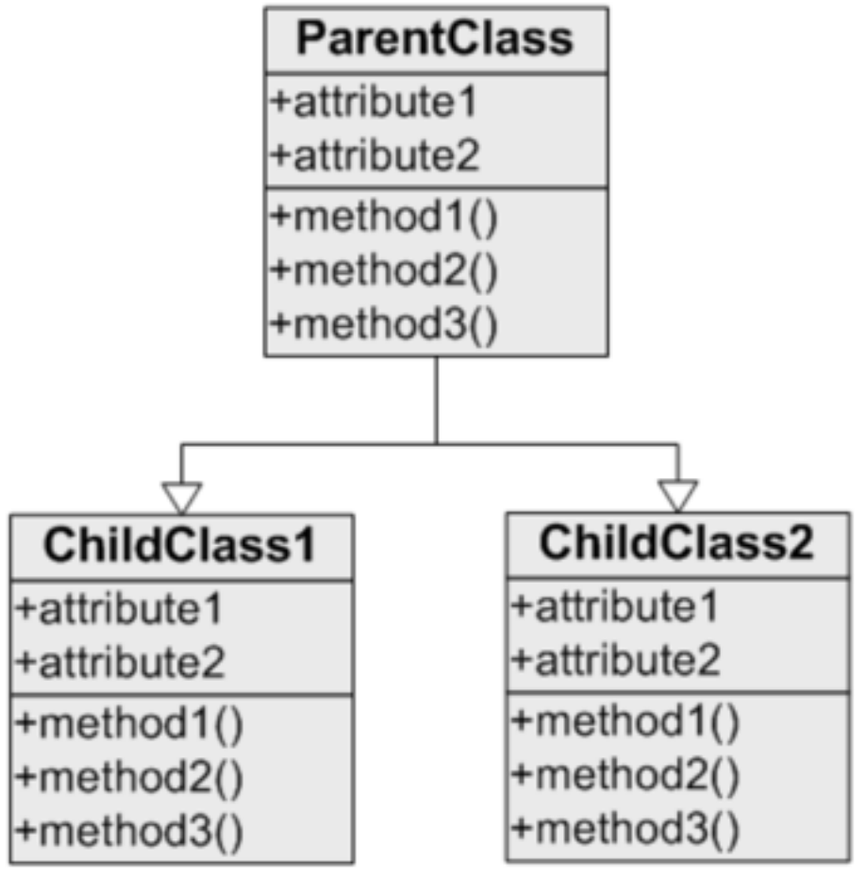
父类与子类的关系



子类继承父类的
全部属性和方法



子类还可以添加自己的成员
从而已父类有所区别



一个父为可以有无限多的后代

例：定义 programer 类，并继承自 employee 类

```
Class programer extends employee {
        public $language;
        function code(){
                echo ‘codeing…’;
        }
}
$p = new programer();
$p->name = ‘david’;
$p->age= 25;
$p->language = ‘PHP’;
$p->code();
```

3.多态

对象的多态性是指在父类中定义的属性或行为被子类继承之后，可以具有不同的数据类型或表现出不同的行为。这使得同一个属性或行为在父类及其各个子类中具有不同的语义。

例：

```
class programer extends employee {
        function colck_in(){
                echo ‘网络签到’;
        }
}
$p = new programer();
$p->name = 'david';
$p->colck_in();
```

自动加载类

在编写面向对象程序时，常规做法是将每一个类保存为一个PHP源文件。当在一个PHP文件中需要调用一个类时很容易就可以找到，然后通过include把这个文件引入就可以了。不过有的时候，在项目中文件众多，要一一将所需类的文件include进来，是一个很让人头疼的事，PHP5提供了一个__autoload()来解决这个问题。

在组织定义类的文件名时，需要按照一定的规则，最好以类名为中心，加上统一的前缀或后缀形成文件名，如：

class_db.php

employee_class.php

例：

```
//自动加载文件class_name中的类
function __autoload($class_name)  {
    include("$class_name"."_class.php");
}
//实例化对象时，如果类不存在，则调用__autoload()函数，其参数为类名
$db = new employee ();
```

四,数据传输

主流传输数据样式分类

网络传输数据的分类：

1.XML数据格式

2.JSON数据格式

XML概述

XML是eXtensible Markup Language的缩写，中文含义为“可扩展标记语言”。顾名思义，XML首先是一种标记语言，其次，它是一种可扩展的标记语言。

标记语言（Markup Language）是指在普通文本中加入一些具有特定含义的标记（Tag），以对文本的内容进行标识和说明的一种文件表示方法。

XML的应用

1. 不同系统平台间的信息互通

网络上存在着大量不同的系统平台，大到大型计算机，小到现在的掌上电脑。这些系统上的软件和数据库系统互不兼容，数据格式也各不相同，如果要相互交换信息的话，必须先进行转换。

XML文件是纯文本文件，可以被几乎所有计算机识别；而所要交换的信息，如文本文件、图像文件、数据库查询出的数据等，都可以被很好地处理成XML文件格式。所以，只要装有XML解析器的系统，就能以XML格式交换数据，而XML解析器是很容易取得的，在网上就有许多的优秀的解析器软件可免费下载。

2. 整合多种不同数据源的数据

如果用户要同时操作多个数据库的数据，必须编写大量的程序，分别从不同数据库取得数据并转换成统一的格式，处理后再一一转换成不同的格式，送回不同的数据库。如果在数据库和用户之间加一个**XML中间层**，将从数据库取得的数据都转换成XML格式，则所有数据库呈现给用户的数据都是统一的格式。由于XML的自定义性及可扩展性，使得它足以表达各种类型的数据。客户收到数据后可以进行处理，也可以在不同数据库间进行传递。总之，在这类应用中，**XML解决了数据的统一接口问题**。

3. 平衡客户端和服务器端的处理负荷

在当前情况下，许多运算负荷集中在服务器端，这对服务器的配置提出了严格的要求。而且如果用户的处理要求多变而复杂，则服务器端的软件开发任务将极其繁重。

使用XML方式，即服务器只向用户发送以XML格式组织的纯数据，用户收到数据后自行进行处理。因为XML文档和数据具有自解释性，使得客户端在收到数据的同时，也能够理解数据的逻辑结构与含义，显示也好，转化也好，都可以在客户端进行。这样既减轻了服务器的运算压力，也减少了在网络上来回传递的数据量，降低了网络的负荷。

4. 以灵活多变的方式显示数据

5. 更精确的数据检索

因为XML可以自定义能够表达特定意义的标记，而标记中内容的分割可以任意小，所以在XML文档中检索特定的内容将更精确，也更简单、更有效率。

6. 更长的生命力

XML文档简单易读，数据与格式分离，同时极易转化为其它格式，因而具有保值的特征，可以作为保存重要数据的一种手段。

XML编写规范

文档必须有一个顶层元素（文档元素或根元素）。所有其它元素必须嵌入其中。

元素必须被正确地嵌套。也就是说，如果一个元素在另一个元素中开始，那么它必须在同一个元素中结束。

每个元素必须同时拥有起始标签和结束标签。

起始标签中的名称必须与结束标签名称完全匹配。

元素名称是区分大小写的。

元素可以在开始标签处自定义属性给元素提供信息

PHP中的生成XML 可以有很多种方式,例如:

1、使用纯粹的PHP代码生成字符串。

2、使用DomDocument生成XML文件

3、使用XMLWriter类创建XML文件

4、使用SimpleXML创建XML文档

XML写法

```
<?xml version="1.0" encoding="utf-8"?>
<books>
    <book>
        <name>从0到1</name>
        <price>45.00</price>
        <count>35</count>
    </book>
    <book>
        <name>node.js入门基础</name>
        <price>59.00</price>
        <count>17</count>
    </book>
    <book>
        <name>iOS编程经典</name>
        <price>109.00</price>
        <count>42</count>
    </book>
    <boo>
        <name><nnn>iOS编程经典</nnn></name>
        <price>109.00</price>
        <count>42</count>
    </boo>
</books>
```

SimpleXMLElement生成xml

```php
$xmlStr = <<<XML
<?xml version="1.0" encoding="utf-8"?>
<books>
</books>
XML;
$xml = simplexml_load_string($xmlStr);
$goods = $xml->addChild("goods");
$goods->addChild("bbb","ppp");

var_dump($xml);
echo $xml->asXML();
$xml->saveXML("new.xml");
```

SimpleXMLElement解析xml

```php
$xml = simplexml_load_file("new.xml");
$book = $xml->xpath("book");
foreach ($xml as $key => $value) {
    echo "$key<hr>";
    foreach ($value as $key => $val) {
        echo $key."=".$val."<hr>";
    }
}
```

JSON简介

什么是JSON

JSON(JavaScript Object Notation) 是一种轻量级的数据交换格式。 易于人阅读和编写。同时也易于机器解析和生成。 它基于 JavaScript Programming Language, Standard ECMA-262 3rd Edition - December 1999的一个子集。 JSON采用完全独立于语言的文本格式，但是也使用了类似于C语言家族的习惯（包括C, C++, C#, Java, JavaScript, Perl, Python等）

基础结构

JSON构建有两种结构：

1. “名称/值”对的集合（A collection of name/value pairs）。不同的语言中，它被理解为对象（object），记录（record），结构（struct），字典（dictionary），哈希表（hash table），有键列表（keyed list），或者关联数组 （associative array）。

2.值的有序列表（An ordered list of values）。在大部分语言中，它被理解为数组（array）。

基础示例

名称 / 值对

按照最简单的形式，可以用下面这样的 JSON 表示"名称 / 值对"：

```
{ "firstName": "Brett" }
```

这个示例非常基本，而且实际上比等效的纯文本"名称 / 值对"占用更多的空间：

```
firstName=Brett
```

但是，当将多个"名称 / 值对"串在一起时，JSON 就会体现出它的价值了。首先，可以创建包含多个"名称 / 值对"的 记录，比如：

```
{ "firstName" : "Brett", "lastName":"McLaughlin", "email": "aaaa" }
```

从语法方面来看，这与"名称 / 值对"相比并没有很大的优势，但是在这种情况下 JSON 更容易使用，而且可读性更好。例如，它明确地表示以上三个值都是同一记录的一部分；花括号使这些值有了某种联系。

接受传JSON

PHP中接收和传递JSON

json_decode — 对 JSON 格式的字符串进行编码

json_encode — 对变量进行 JSON 编码

json_decode

语法 : mixed json_decode (string $json [, bool $assoc])

参数 : json　待解码的 json string格式的字符串

assoc 当该参数为 TRUE 时，将返回 array 而非 object

示例 :

```
$json = '{"a":1,"b":2,"c":3,"d":4,"e":5}';
var_dump(json_decode($json));
var_dump(json_decode($json, true));
```

json_encode

语法 : string json_encode (mixed $value)

参数： 待编码的 value

注意:该函数只能接受 UTF-8 编码的数据

返回值：

编码成功则返回一个以 JSON 形式表示的 string 或者在失败时返回 FALSE 。

示例 :

```
$arr = array (‘a’=>1,‘b’=>2,‘c’=>3,‘d’=>4,‘e’=>5);
echo json_encode($arr);
// {"a":1,"b":2,"c":3,"d":4,"e":5}
```

谢　谢