1、 PHP概述
2、 PHP基础
3、 面向对象思想
4、 面向对象的PHP
5、 字符串和正则表达式
6、 PHP专题相关
7、 PHP常用API分类汇总
8、 MVC框架及模板
9、 PHP网站部署及安全
10、 PHP网站常见模块（非框架版）
11、 PHP网站常见模块（TP框架版）
12、 用PHP+Mysql构造网站系统案例（详细设计文档）

1、 PHP概述

PHP（“PHP: Hypertext Preprocessor”，超文本预处理器的字母缩写）是一种被广泛应用的开放源代码的多用途脚本语言，它可嵌入到 HTML中，尤其适合 web 开发。和客户端的 JavaScript 不同的是，PHP 代码是运行在服务端的。
使用 PHP 的一大好处是它对于初学者来说极其简单，同时也给专业的程序员提供了各种高级的特性。PHP 主要是用于服务端的脚本程序，因此可以用 PHP 来完成任何其它的 CGI 程序能够完成的工作，例如收集表单数据，生成动态网页，或者发送 / 接收 Cookies。但 PHP 的功能远不局限于此。

PHP 脚本主要用于以下三个领域：

- 服务端脚本。这是 PHP 最传统，也是最主要的目标领域。开展这项工作需要具备以下三点：PHP 解析器（CGI 或者服务器模块）、web 服务器和 web 浏览器。需要在运行 web 服务器时，安装并配置 PHP，然后，可以用 web 浏览器来访问 PHP 程序的输出，即浏览服务端的 PHP 页面。如果只是实验 PHP 编程，所有的这些都可以运行在自己家里的电脑中。请查阅安装一章以获取更多信息。

- 命令行脚本。可以编写一段 PHP 脚本，并且不需要任何服务器或者浏览器来运行它。通过这种方式，仅仅只需要 PHP 解析器来执行。这种用法对于依赖 cron（Unix 或者 Linux 环境）或者 Task Scheduler（Windows 环境）的日常运行的脚本来说是理想的选择。这些脚本也可以用来处理简单的文本。请参阅 PHP 的命令行模式以获取更多信息。

- 编写桌面应用程序。对于有着图形界面的桌面应用程序来说，PHP 或许不是一种最好的语言，但是如果用户非常精通 PHP，并且希望在客户端应用程序中使用 PHP 的一些高级特性，可以利用 PHP-GTK 来编写这些程序。用这种方法，还可以编写跨平台的应用程序。PHP-GTK 是 PHP 的一个扩展，在通常发布的 PHP 包中并不包含它。如果对 PHP-GTK 感兴趣，请访问其» 网站以获取更多信息。

PHP 能够用在所有的主流操作系统上，已经支持了大多数的 web 服务器，使用 PHP，可以自由地选择操作系统和 web 服务器。同时，还可以在开发时选择使用面对过程和面对对象，或者两者混和的方式来开发。尽管 PHP 4 不支持 OOP 所有的标准，但很多代码仓库和大型的应用程序（包括 PEAR 库）仅使用 OOP 代码来开发。PHP 5 弥补了 PHP 4 的这一弱点，引入了完全的对象模型。

使用 PHP，并不局限于输出 HTML。PHP 还能被用来动态输出图像、PDF 文件甚至 Flash 动画（使用 libswf 和 Ming）。还能够非常简便的输出文本，例如 XHTML 以及任何其它形式的 XML 文件。PHP 能够自动生成这些文件，在服务端开辟出一块动态内容的缓存，可以直接把它们打印出来，或者将它们存储到文件系统中。

PHP 最强大最显著的特性之一，是它支持很大范围的数据库。同时还有一个叫做 PDO 的数据库抽象扩展库使得可以自由地使用该扩展库支持的任何数据库。另外，PHP 还支持 ODBC，即 Open Database Connection Standard（开放数据库连接标准），因此可以连接任何其它支持该世界标准的数据库。

PHP 还支持利用诸如
LDAP、IMAP、SNMP、NNTP、POP3、HTTP、COM（Windows 环境）等不计其数的协议的服务。还可以开放原始网络端口，使得任何其它的协议能够协同工作。PHP 支持和所有 web 开发语言之间的 WDDX 复杂数据交换。关于相互连

接，PHP 已经支持了对 Java 对象的即时连接，并且可以将他们自由的用作 PHP 对象。甚至可以用我们的 CORBA 扩展库来访问远程对象。

PHP 具有极其有效的文本处理特性，支持从 POSIX 扩展或者 Perl 正则表达式到 XML 文档解析。为了解析和访问 XML 文档，PHP 4 支持 SAX 和 DOM 标准，也可以使用 XSLT 扩展库来转换 XML 文档。PHP 5 基于强健的 libxm2 标准化了所有的 XML 扩展，并添加了 SimpleXML 和 XMLReader 支持，扩展了其在 XML 方面的功能。

另外，还有很多其它有趣的扩展库。例如 mnoGoSearch 搜索引擎函数、IRC 网关函数、多种压缩工具（gzip、bz2、zip）、日历转换、翻译……

如果需要自己配置服务器和 PHP，有两个方法将 PHP 连接到服务器上。对于很多服务器，PHP 均有一个直接的模块接口（也叫做 SAPI）。这些服务器包括 Apache、Microsoft Internet Information Server、Netscape 和 iPlanet 等服务器。其它很多服务器支持 ISAPI，即微软的模块接口（OmniHTTPd 就是个例子）。如果 PHP 不能作为模块支持 web 服务器，总是可以将其作为 CGI 或 FastCGI 处理器来使用。这意味着可以使用 PHP 的 CGI 可执行程序来处理所有服务器上的 PHP 文件请求。

对于PHP的安装和配置需要说明的是，不同版本的PHP和服务器、数据库组件之间的装配方法并不统一，需要根据不同版本安装，有关各模块最新版本之间的搭配组合安装参见 一般PHP在不同环境下的安装配置

配置文件（PHP 3 中是 php3.ini，自 PHP 4 起是 php.ini）在 PHP 启动时被读取。对于服务器模块版本的 PHP，仅在 web 服务器启动时读取一次。对于 CGI 和 CLI 版本，每次调用都会读取。 更多关于PHP配置文件的说明请参阅手册或网络

2、 PHP基础

2.1基本语法

开始和结束标记：共有4种不同的界定形式，其中两种，<?php ?> 和 <script language="php"> </script> 总是可用的，另两种是短标记<? ?>和 ASP风格标记<% %>，可以在 *php.ini* 配置文件中打开或关闭。为了代码的移植及发行，确保不要使用短标记。和asp标记。

指令分隔符：PHP 需要在每个语句后用分号；结束指令，在一个 PHP 代码段中的最后一行可以不用分号结束。

注释： 单行#或// 多行/* */

数据类型：PHP 支持8种基本的数据类型。

四种标量类型： boolean （布尔型） integer （整型） float （符点型, 也称作 double) string （字符串）

两种复合类型： array （组） object （对象）

两种特殊类型： resource （资源） NULL （NULL）

为了确保代码的易读性，手册还介绍了一些伪类型，

mixed：说明一个参数可以接受多种不同的（但并不必须是所有的）类型。

Number：说明一个参数可以是 integer 或者 float。

Callback：有些诸如 call_user_function() 或 usort() 的函数接受用户自定义的函数作为一个参数。Callback 函数不仅可以是一个简单的函数，它还可以是一个对象的方法，包括静态类的方法。

PHP 在变量定义中不需要（或不支持）明确的类型定义；变量类型是根据使用该变量的上下文所决定的。关于类型之间的转换规则及注意事项参阅PHP手册，对于数组类型下面会详细讲到。

变量：PHP 中的变量用一个美元符号后面跟变量名来表示。变量名是区分大小写的。一个有效的变量名由字母或者下划线开头，后面跟上任意数量的字母，数字，或者下划线。按照正常的正则表达式，它将被表述为：'[a-zA-Z_\x7f-\xff][a-zA-Z0-9_\x7f-\xff]*'。变量默认总是传值赋值。

虽然在 PHP 中并不需要初始化变量，但对变量进行初始化是个好习惯。未初始化的变量具有其类型的默认值。依赖未初始化变量的默认值在某些情况下会有问题，例如把一个文件包含到另一个之中时碰上相同的变量名。

使用未初始化的变量会发出E_NOTICE错误，但是在向一个未初始化的数组附加单元时不会。isset() 语言结构可以用来检测一个变量是否已被初始化。

PHP 将会自动将变量名中的点替换成下划线。PHP 包括几个函数可以判断变量的类型，例如：gettype()，is_array()，is_float()，is_int()，is_object() 和 is_string()。

关于PHP系统的预定义变量等更多变量内容参见PHP手册

常量：

定义常量define()函数通过给一个变量名赋值来定义一个常量，其形式如下：boolean define(string name, mixed value[, bool case_insensitive])，如果使用可选参数case insensitive，并且这个参数值为下RUE，那么后面对此常量的引用将不区分大小写。常量的范围是全局的。

魔术常量：有七个魔术常量它们的值随着它们在代码中的位置改变而改变。这些特殊的常量不区分大小写

__LINE__	文件中的当前行号。
__FILE__	文件的完整路径和文件名。如果用在被包含文件中，则返回被包含的文件名。自 PHP 4.0.2 起，__FILE__ 总是包含一个绝对路径（如果是符号连接，则是解析后的绝对路径），而在此之前的版本有时会包含一个相对路径。
__DIR__	文件所在的目录。如果用在被包括文件中，则返回被包括的文件所在的目录。它等价于 *dirname(__FILE__)*。除非是根目录，否则目录中名不包括末尾的斜杠。（PHP 5.3.0中新增） =
__FUNCTION__	函数名称（PHP 4.3.0 新加）。自 PHP 5 起本常量返回该函数被定义时的名字（区分大小写）。在 PHP 4 中该值总是小写字母的。
__CLASS__	类的名称（PHP 4.3.0 新加）。自 PHP 5 起本常量返回该类被定义时的名字（区分大小写）。在 PHP 4 中该值总是小写字母的。
__METHOD__	类的方法名（PHP 5.0.0 新加）。返回该方法被定义时的名字（区分大小写）。
__NAMESPACE__	当前命名空间的名称（大小写敏感）。这个常量是在编译时定义的（PHP 5.3.0 新增）

关于系统的预定义常量参见PHP手册

表达式&运算符：最精确的定义一个表达式的方式就是“任何有值的东西”。

运算符是可以通过给出的一或多个值（用编程行话来说，表达式）来产生另一个值（因而整个结构成为一个表达式）的东西。有三种类型的运算符。第一种是一元运算符，只运算一个值，例如 !（取反运算符）或 ++（加一运算符）。第二种是有限二元运算符，PHP 支持的大多数运算符都是这种。第三种是三元运算符：?:。它应该被用来根据一个表达式在另两个表达式中选择一个，而不是用来在两个语句或者程序路线中选择。把整个三元表达式放在扩号里是个很好的主意。

有关具体运算符和其优先级参见PHP手册

控制结构:基本的控制结构同c语言，其他像foreach等将在后面具体说明

函数：function functionName(parameters)

{ function一body

}

函数名是大小写无关的，不过在调用函数的时候，通常使用其在定义时相同的形式。一般函数必须在其调用之前定义。

PHP 支持按值传递参数（默认），通过引用传递参数以及默认参数。也支持可变数量的参数。

返回语句会return（）立即中止函数的运行，并且将控制权交回调用该函数的代码行。

函数不能返回多个值，但可以通过返回一个数组来得到类似的效果。从函数返回一个引用，必须在函数声明和指派返回值给一个变量时都使用引用操作符 &

PHP 支持可变函数的概念。这意味着如果一个变量名后有圆括号，PHP 将寻找与变量的值同名的函数，并且尝试执行它。可变函数可以用来实现包括回调函数，函数表在内的一些用途。

关于系统函数API参见PHP手册。

3、 面向对象思想

面向对象编程（**Object Oriented Programming**，OOP，面向对象程序设计）是一种计算机编程架构。OOP 的一条基本原则是计算机程序是由单个能够起到子程序作用的单元或对象组合而成。为了实现整体运算，每个对象都能够接收信息、处理数据和向其它对象发送信息。

简而言之，面向对象可以用下面公式概括：

OO=Objects+Classes+Inheritance+Communication With messagees

面向对象程序设计，首先学会面向对象的思维解题方法，各种面向对象的语言无非是实现面向对象解题方法的工具。

面向对象的优点是，接近人类的思维习惯、稳定性好、可重用性和可维护性好。

对象内部的数据只能通过对象的公有方法来访问或处理，对象是具有相同状态的一组操作的集合。形式化定义为：

对象＝{标识或名字，操作集合，数据结构，受理的消息名集合}

OOP的几个基本概念：

封装：通过众所周知的接口将用户与实际应用程序的内部工作原理分离，使用一个对象（组件）的时候，只需知道他向外界提供的接口形式，无需知道他的数据结构细节和实现操作的算法。

继承：特殊类的对象拥有其一般类的全部属性与方法。

多态：子类对象可以和父类的对象使用相同名字的方法，而不同子类对象对方法的具体实现由本身决定。

重载：分　　函数和运算符重载，即根据不同的数据类型或参数个数实现不同的操作。

面向对象分析过程：

OOM：为了理解事物而对事物作出的一种抽象，是对事物的一种无歧义的书面描述。通常，模型由一组图示符号和组织这些符号的规则组成。

用面向对象方法开发软件，需要建立三种形式的模型：

- ü　对象模型（Object Model）：描述系统数据结构，是最重要、最基本、最核心的
- ü　动态模型（Dynamic Model）：描述系统控制结构
- ü　功能模型（Function Model）：描述系统功能

对象模型：表示静态的、结构化的系统的“数据”性质。模拟客观世界实体的对象以及对象彼此间的关系的映射，描述了系统的静态结构的“数据”性质。

建立对象模型的表示方法，包括下列符号：

表示类的符号（应该即能表示属性又能表示服务）；

表示对象（类实例）的符号；

表示继承关系的符号；

表示类或对象间其它关系的符号；

类-&-对象的图形符号：

类名
属性
服务

类-&-对象的符号

类名
属性
服务

类的符号

类符号是代表一个没有实例的抽象类；

类－&－对象符合外的虚框代表属于该类的对象；

表示结构的图形符号：

一般类

具体类**1**

具体类**2**

整体

部分**1**

部分**2**

1,m
1,n

1. 归纳关系（**Conclusion relationship**）
2. 组合关系**(Combination relationship)**

当组合关系有多个层次时，可以用一棵聚集树表示：

教材

前言

1,m
封面

目录

章

节

习题

1,n

关联关系：

教师

书

1＋

关联关系

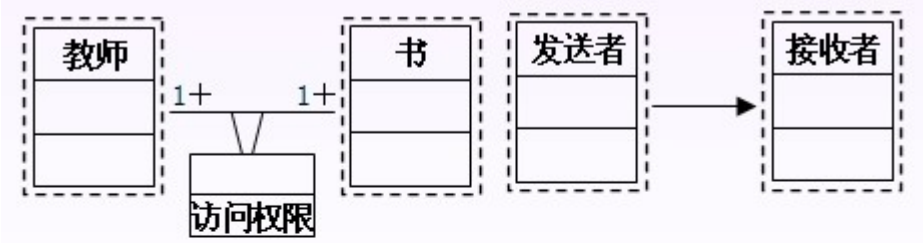
两个对象间的二元关联分为(1:1)、(1:M1)、(M:N)；

尽量避免使用多元关联。

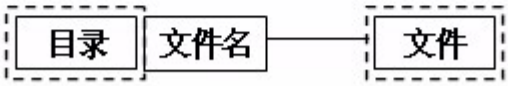
(3)链属性：关联链的性质

(4)限定：特殊的链属性

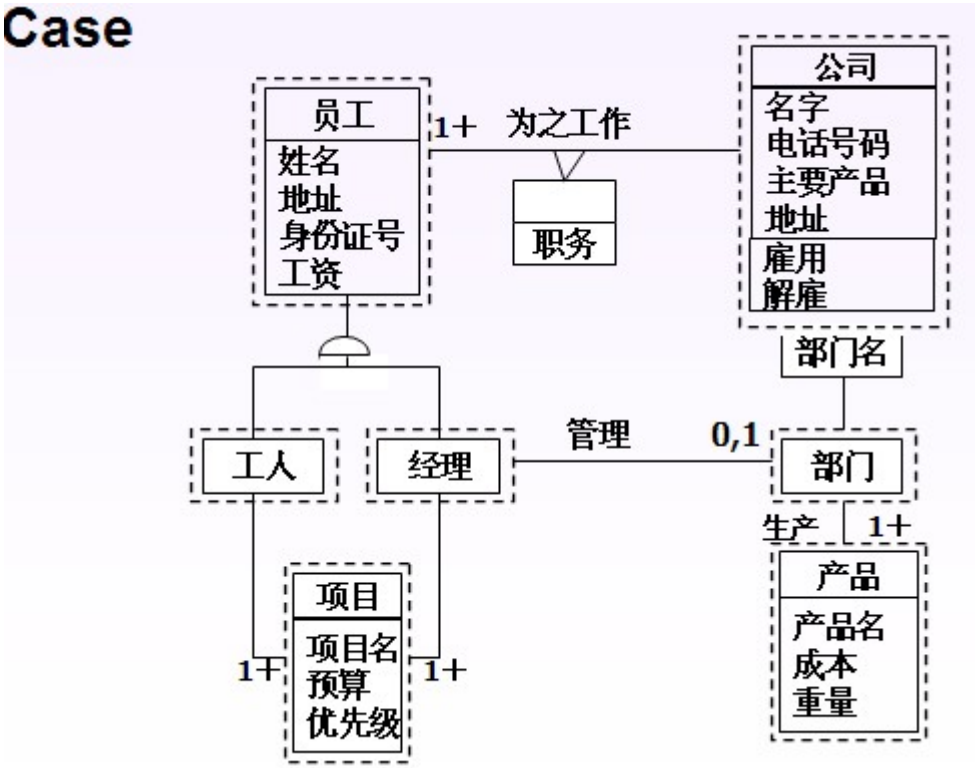
(5)消息连接：一个对象对另一个对象的依赖性



链属性之例　　表示消息连接的图形符号



受限关联例子



OOA：

就是抽取和整理用户需求并建立问题域精确模型的过程。分析工作包括理解（Comprehension）、表达（Expression）和验证（Verification）三项内容。

建立的模型应包括

对象模型（静态结构,Static Structure）

动态模型（交互次序,Interaction Sequence）

功能模型（数据变换,Data Transformation）

大型系统的对象模型由五个层次组成

主题层（范畴层）(Subject Layer)

类-&-对象层　(Class-&-Object Layer)

结构层　(Structure Layer)

属性层　(Attribute Layer)

服务层　　(Service Layer)

面向对象分析工作大体上按下列顺序进行：

1、寻找类&对象

1.找出候选的类-&-对象：一般按语法分析，寻找主语名词或名词短语

2.筛选出正确的类-&-对象：筛选标准:1、冗余2、无关、3、笼统4、属性5、操作6、实现

2、识别结构（确定关联）

3、定义属性：既对象所具有的性质，一般是宾语名词

4、识别继承关系：确定了类中属性之后，利用继承机制共享的公共性质，对系统中的类加以组织。一般有两种方法建立继承关系：

（1）自底向上：抽象出现有类的共同性质泛化出父类

（2）自顶向下：把现有类细化成更具体的子类

5、定义方法

在确定类中应该有的的方法时，既要考虑该类实体的常规行为，又要考虑完成本系统功能所需要提供的服务。把笼统的功能细化为具体的功能，再把所有该完成 的功能分配给适当的对象去承担。

OOD：从分析到设计是一个逐渐扩充模型反复迭代　的过程，或者说，面向对象设计是用面向对象观点建立求解域模型的过程

一个实用的软件系统在逻辑上由下述四部分组成

- ü　问题域子系统　（Problem Domain Sub-system）
- ü　人机交互子系统　(Human-Machine Interaction Sub-system)
- ü　任务管理子系统　(Task Management Sub-system)
- ü　数据管理子系统　(Data Management Sub-system

尽管分析和设计是有区别的，但实际开发中二者界限模糊

面向对象设计准则：

1、　模块化2、抽象3、信息隐藏

4、弱耦合：对象之间的耦合一般分1、交互耦合2继承耦合　应该使前者尽量松散而提高继承耦合程度

5、强内聚6、可重用

OOP：即把面向对象设计的的结果翻译成用某种面向对象程序语言书写的程序

OOT：

4、　面向对象的PHP

4、1基础

PHP中一般类的创建语法

```
Class name{
        #数据成员var $name
        #方法声明scope function functionName()
        {
                //Function body goes here
        }
        }
```

对象使用new关键字创建，如下

```
$employee=new Employee()
```

创建对象之后，这个刚实例化的对象就具有了类中定义的所有性质和行为。

作用域：PHP支持5种字段作用域:

public,：任何地方，默认为该项

private,：本类内部
protected,：本类及继承类；如果希望扩展类，就应当使用保护字段而不是私有字段。
final：子类将无法重写final定义的属性或方法，final类：阻止该类被继承
static ：
但在类中一般不使用Public定义属性，那样违背了封装性的原则，而是采取{通过一些称为公共方法(public method)的接口来访问。类似下面：

```
class Employee
{
  private $name;
  public function setName($name){
     $this->name=$name;
    }
}
```

类常量：用const
使用字段或方法
$object->field
$object->method_ name();
在本类中引用使用$this，$this表示要引用当前类(要访问或操作的字段所在的类)中的字段。不能在类中使用$this来引用声明为static的字段
动态创建新的变量来扩展类：即当对象试图访问类中不存在的成员时，会自动触发。
1、 设置属性
bool __set([string property name],[mixed value to assign])
它接受一个属性名和相应的值作为输入，如果方法成功执行就返回TRUE，否则返回false.例如：class{

```
                    Employee
                    var $name
                    function
                    {
                        _set($propName, $propValue)
                        $this->$propName=$propValue;
                    }
                    }

                    $employee = new Employee();
$employee->name = "Mario";
$employee->title=”Executive Chef";
echo "Name: ".$employee->name;
     echo "<br/>”;
echo "Title:”.$employee->title;
```

输出：Name: Mario
Title: Executive Chef
2.获取属性
boolean _get((string property name])
它接受一个属性名作为输入参数，即要获取该属性的值。它在成功执行时返回TRUE，否则返回FALSE。
例：class Employee

```
{
   var $name;
   var $city;
   protected $wage;
   function  __get($propName)
   {
      echo "__get called!<br/>";
    $vars=array("name","city");
      if (in_array($propName, $vars))
      {
       return $this一>$propName;
      }else{
         return "No such variable!";
      }
   }
}
   $employee=new Employee();
   $employee ->name='Mario';
   echo $employee->name.'<br />';
   echo $employee->age;
?>
```

输出：Mario
__get called!
No such variable!
注意到：当要访问的变量确实是类的属性的时候，set方法和get方法不会被调用。
函数的作用域除了字段作用域外，多一个abstract.
抽象(abstract)方法很特殊，只在父类中声明，但在子类中实现。只有声明为abstract的类可以声明抽象方法。如果想定义一个应用编程接口(API　)，以后作为实现的一个模型，就可以声明一个抽象方法。开发人员会知道，如果能满足抽象方法定义的所有需求，那么他为该方法开发的特定实现应该能正常工作。抽象方法如下声明:
abstract function methodName();
继承：一个扩充类总是依赖一个单独的基类，也就是说，不支持多继承。使用关键字“extends”来扩展一个类。

作用域识别操作符：用于在没有声明任何实例的情况下访问类中的函数或者基类中的函数和变量，用：：实现
与其常见的用法：parent：：基类成员或方法
基类名：：基类成员或方法
Self：：变量名或方法（子类中引用本身成员或方法）
静态类成员：有时，可能有必要创建供所有类实例共享的字段和方法，这些字段和方法与所有的类实例有关，静态字段和方法应使用self关键字和类名来引用，而不是通过
this和箭头操作符。
Instanceof关键字：　　PHP 5的另一个新成员是ins七anceof关键字。使用这个关键字可以确定一个对象是类的实例、类的
子类，还是实现了某个特定接口，并进行相应的操作。例如，假设希望了解名为manager的对象是否为
类Employee的实例:

$manager=new Employee();
if ($manager instanceof Employee) echo "Yes";

有两点值得注意。首先，类名没有任何定界符(引号)。使用定界符将导致语法错误。其次，如果比较失败，脚本将退出执行。instanceof关键字在同时处理多个对象时特别有用。例如，你可能要重复地调用某个函数，但希望根据对象类型调整函数的行为。可以使用case语句和instanceof关键字来实现这个目标。

自动加载对象：通过定义特殊的_autoload函数，当引用未在脚本中定义的类时会自动调用这个函数。回到图书馆的例子，通过定义如下函数，就不必再手工包含各个类文件
了:

```
function __ autoload($class){
        require_ once("classes/$class.class.php");
}
```

定义这个函数后，将不再需要那些require once()语句，因为当第一次调用一个类时，就会调用__autoload()，并根据__autoload()中定义的命令加载类。这个函数可以放在某个全局应用程序配置。

4、2高级OOP特性

PHP不支持的高级特性：运算符重载和多重继承

接口：　　接口(interface)定义了实现某种服务的一般规范，声明了所需的函数和常量，但不指定如何实现。之所以不给出实现的细节，是因为不同的实体可能需要用不同的方式来实现公共的方法定义。关键是要建立必须实现的一组一般原则，只有满足了这些原则才能说实现了这个接口。接口中定义的所有方法都必须是public，这是接口的特性。

接口中不定义类成员!类成员的定义完全交给实现类来完成。在PHP中，要这样创建接口：

```
interface InterfaceName
{
        CONST 1;
                CONST N;
    function methodNamel();
    function methodNameN();
}
```

接口中的所有方法都必须实现，倘若实现类没有实现所有方法，则必须声明为抽象类。

可以看到，接口特别有用。因为，虽然它们定义了发生某一行为需要多少个方法，以及各个方法的名字，但接口允许不同的类，以不同的方式来实现这些方法。

如果要实现多个接口，可以用逗号来分隔多个接口的名称。

抽象类：抽象类是不能实例化的类，只能作为由其他类继承的基类。用于描述各种公共的共同性质。声明为抽象的类必须在定义前面加上关键字abstract

```
abstract class Class Name
{
    //insert attribute definitions here
     //insert method definitions here
}
```

任何派生类都必须实现从该抽象类继承的所有抽象方法。

什么时候使用接口或抽象类：

1、 如果要创建一个模型，这个模型将由一些紧密相关的对象采用就可以使用抽象类。如果要创建将由、些不相关对象采用的功能，就使用接口。

2、 如果必须从多个来源继承行为，就使用接口，PHP类可以继承多个接口，但不能扩展多个抽象类

3、 如果知道所有类都会共享一个公共的行为实现，就使用抽象类、并在其中实现该行为。在接口中无法实现行为

命名空间： 随着类库的不断扩展，最后很可能遭遇这样一种情况:两个类库使用了相同的类名。由于要求每个类有唯一的名，所以不可能在同一个PHP脚本中使用两个同名的类。 在你自己的应用程序中实现PHP 6的命名空间特性之前，一定要仔细研究PHP手册。

克隆：在将一个类变量赋值给另一个变量时，执行的是引用赋值，即引用类的同一实例，对任何一个变量的值的改变都将导致另一变量的改变。克隆生成的实例具有自己独立变化的属性，这样对克隆对象的修改就不会影响到原对象。

5、 字符串函数和正则表达式

5、1正则表达式

在正则表达式(regular expression)的基础上，可以根据预定义的语法规则描述或匹配数据。正则表达式本身只是字符串模式，用于匹配某些文本。PHP提供了支持POSIX风格和Perl风格的两组正则表达式实现的函数库。每种正则表达式都有自己独特的语法风格。前者更容易掌握，但不是二进制安全的，后者则有更快的速度。

正则表达式中包括的元素

(1)、原子（普通字符：a-z A-Z 0-9 、原子表、 转义字符）

(2)、元字符 （有特殊功能的字符）

(3)、模式修正符 （系统内置部分字符 i 、m、S、U…）模式修正符是为正则表达式增强和补充的一个功能，使用在正则之外

/ 正则 / U

构造正则表达式的方法和创建数学表达式的方法一样。也就是用多种元字符与操作符将小的表达式结合在一起来创建更大的表达式。正则表达式的组件可以是单个的字符、字符集合、字符范围、字符间的选择或者所有这些组件的任意组合。

PHP的正则表达式函数(Perl兼容)：

PHP为使用Perl兼容的正则表达式搜索字符串提供了7个函数，包括

preg _grep()：函数搜索数组中的所有元素，返回由与某个模式匹配的所有元素组成的数组。

preg_ match()：函数在字符串中搜索模式，如果存在则返回TRUE，否则返回FALSE

preg_ match_ all()：函数在字符串中匹配模式的所有出现，以便通过可选的输入参数order所指定的顺序，将每次出现放在某个数组中。

preg_quotet()：在每个对于正则表达式语法而言有特殊含义的字符前插入一个反斜线。

正则表达式的特殊字符包括：. \ + * ? [^] $ ( ) { } = ! < > | :

preg_replace()：

preg__replace _callback()

preg_ split()：

有关正则表达式的构建参见正则表达式手册。

常用的正则表达式整理

5、2字符串函数

除了前半部分讨论的基于正则表达式的函数，PHP还提供了100多个函数，能够处理字符串中我们所能想到的每一个方面。一般而言，对于同样的功能 ，正则表达式的函数效率要低于字符串函数，如果程序足够简单，应选择字符串函数，但是，对于可以通过单个正则表达式执行的任务来说，如果使用多个字符串函数是不对的。

详细的函数使用规则参见PHP手册

6、 PHP专题相关

6、1数组

PHP 中的 数组 实际上是一个有序映射。映射是一种把 *values* 关联到 *keys* 的类型。用 array() 语言结构来新建一个 array。它接受任意数量用逗号分隔的 *键(key)* => *值(value)* 对。*key* 可以是 integer 或者 string型，values的值任意。如果对给出的值没有指定键名，则取当前最大的整数索引值，而新的键名将是该值加一。如果指定的键名已经有了值，则该值会被覆盖。

通过在方括号内指定键名来给数组赋值：$arr[key] = value;

用list()提取数组

list()函数与array()类似，只是它可以在一次操作中从一个数组内提取多个值，同时为多个变量赋值。从数据库或文件中提取信息时，这种构造尤其有用。使用list()同时为多个变量赋值:例：

```
//Open the users.txt file
fusers=fopen("users.txt", "r"
//While the EOF hasn't been reached, get next line
while ($line=fgets{fusers, 4096)){
//use explode() to separate each piece of data.
list($name, $occupation, $color)=explode("|"，$line);
//format and output the data
printf("Name: %s <br/>"，$name);
printf("Occupation: %s <br/>"，$occupation);
printf("Favorite color: %s <br/>"，$color);
}
fclose($users);
```

list()依靠函数explode()将每一行分解为三个元素，这些元素分别赋给了$name, $occupation和$color。至此，余下的只是如何格式化以及显示到浏览器。

用预定义的值范围填充数组:

range()函数是一个快速创建数组的简单方法，并会使用low到high范围的整数值填充数组。这个函数将返回一个包含此范围内所有整数的数组。其形式如下:

array range(int low, int high[，int step])

range()函数还可以用于字符序列。例如，假设希望创建一个包含字母A到F的数组:

```
$letters=range("A","F");
//$letters=array("A,"，.".B".,"C","D",.".E"..".F.")
```

打印数组用于测试：print_r（）函数boolean print _r(mixed'variable[，boolean return])

可选参数return用来修改函数的行为，使函数将输出返回给调用者，而不是发送到标准输出。因此，如果希望返回以上$states数组的内容，只需将return设置为TRUE:

测试数组：内置函数is _array()可用于测试某变量是否为数组格式。

在编程中数组的作用不可或缺，有很多数组的操作函数，可以产生强大的功能，参见PHP手册-数组函数。

foreach循环：*foreach* 仅能用于数组，当试图将其用于其它数据类型或者一个未初始化的变量时会产生错误。有两种语法，第二种比较次要但却是第一种的有用的扩展。

```
foreach (array_expression as $value)
        statement
foreach (array_expression as $key => $value)
        statement
```

foreach 所操作的是指定数组的一个拷贝，而不是该数组本身。

可以很容易地通过在 *$value* 之前加上 & 来修改数组的元素。此方法将以引用赋值而不是拷贝一个值。

```
<?php
$arr = array(1, 2, 3, 4);
foreach ($arr as &$value) {
    $value = $value * 2;
}
// $arr is now array(2, 4, 6, 8)
?>
```

foreach 不支持用“@”来抑制错误信息的能力。

6、2函数

PHP 支持按值传递参数（默认），通过引用传递参数以及默认参数。也支持可变数量的参数。

按值传递：意味着，函数范围内对这些值（非静态）的任何改变在函数外部都会被忽略。如果希望在函数范围外也能反映出这些修改，则可以按引用传递参数。

不要求非得在调用之前定义函数，因为PHP在执行前会把整个脚本读到引擎中。但并不推荐这种随便的做法。

按引用传递参数：可能希望在函数内对参数的修改在函数范围外也能反映，如果想要函数的一个参数总是通过引用传递，可以在函数定义中该参数的前面预先加上符号 &，（调用时不必）

默认参数值：可以为输入参数指宁默认值，在没有提供其他值的情况下，就会把这个默认值自动赋给该参数。默认参数值必须位于参数列表末尾或为常数表达式，而不能指定函数调用或变量等非常量值。

可以指定某个参数为可选(optional)参数，为此，这些参数需要放在参数列表末尾，而且要指定其默认值为空，如下:

```
function calcSalesTax($price, $tax="")
{

$total=$price+($price*$tax);
echo "Total cost:$total";
}
```

返回值：return()语句可以向函数调用者返回任意确定的值，将程序控制权返回到调用者的作用域。return()在全局作用域内调用，将终止脚本的执行。

从函数中返回多个值通常很方便。例如，假设要创建一个函数，从数据库中获取用户数据，比如用户的姓名、电子邮件地址和电话号码，然后返回给调用者。通常的做法是用数组做返回值，用list（）构造承接。

函数库：常将函数组织到函数库中，可以在以后类似的应用程序中重复使用。在一个文件中简单地聚集函数定义就可以创建PHP库。保存这个库，最好使用一个能清楚地说明其用途的命名约定来命名，如taxes.library.php

6、3错误和异常处理

配置指令：许多配置指令确定了PHP的错误报告行为。本节将介绍其中一些指令。

error_reporting指令确定报告的敏感级别。共有14个不同的级别，这些级别的任何组合都是有效的。每个级别都包括位于其下面的所有级别。

E_ALL	所有错误和警告
E_COMPILE_ERROR	致命的编译时错误
E_COMPILE_WARNING	编译时警告
E_CORE_ERROR	PHP开始启动时发生的致命错误
E_CORE_WARNING	PHP开始启动时发生的警告
E_ERROR	致命的运行时错误
E_NOTICE	运行时注意消息
E_PARSE	编译时解析错误
E_RECOVERABLE_ERROR	
E_STRICT	PHP版本可移植性建议（PHP 5.0中引入）
E_USER_ERROR	用户导致的错误
E_USER_NOTICE	用户导致的注意消息
E_USER_WARNING	用户导致的警告
E_WARNING	运行时警告

在PHP6中，E _STRIC被集成到了E ALL中，因此，当运行PHP 6时，需要将error_reporting指令设置为E_ ALL来查看这些可移植性建议。

error_reporting指令使用一字符表示逻辑操作符NOT., 例

```
error_reporting E_ALL& ~(E_ERROR | E_WARNING | E_NOTICE);
```

启用display errors时，将显示满足error_ reporting所定义规则的所有错误。应当只在测试期间启用此指令，网站投入使用时要将其禁用。显示这些消息不仅可能会让终端用户更迷惑，还可能会过多地泄露有关应用用及务器的信息。

错误日志：　　如果要决定在单独的文本文件中记录错误日志，那么Web服务器进程所有者必须有足够的权限来写这个文件。此外，要确保这个文件存放在文档根之外，以减少遭到攻击的可能性。

异常处理：　　对于初学者来说，异常处理对于标识和报告应用程序错误提供了通用策略，另外，对于指定程序在遇到错误时如何进行处理也有一般性的策略，通过使用这些策略，异常处理要优于错误管理过程。此外，异常处理的语法促进了将错误处理器与一般的应用程序逻辑相分离，因而可以得到更有组织、更具可读性的代码。大多数实现异常处理的语言将此过程抽象为4个步骤:

(1)应用程序尝试做一些操作;

(2)如果尝试失败，则异常处理特性抛出一个异常;

(3)指定的处理器捕获该异常，完成必要的任务;

(4)异常处理特性清除在尝试期间占用的资源。

PHP 5 添加了类似于其它语言的异常处理模块。在 PHP 代码中所产生的异常可被 *throw* 语句抛出并被 *catch* 语句捕获。需要进行异常处理的代码都必须放入 *try* 代码块内，以便捕获可能存在的异常。每一个 *try* 至少要有一个与之对应的 *catch*。使用多个 *catch* 可以捕获不同的类所产生的异常。当 *try* 代码块不再抛出异常或者找不到 *catch* 能匹配所抛出的异常时，PHP 代码就会在跳转到最后一个 catch 的后面继续执行。当然，PHP 允许在 *catch* 代码块内再次抛出（*throw*）异常。
当一个异常被抛出时，其后（译者注：指抛出异常时所在的代码块）的代码将不会继续执行，而 PHP 就会尝试查找第一个能与之匹配的 *catch*。如果一个异常没有被捕获，而且又没用使用 set_exception_handler() 作相应的处理的话，那么 PHP 将会产生一个严重的错误，并且输出 *Uncaught Exception ...*（未捕获异常）的提示信息。所以，应设置处理未捕获的异常函数，set_exception_handler()。
用户可以用自定义的异常处理类来扩展 PHP 内置的异常处理类。
如果使用自定义的类来扩展内置异常处理类，并且要重新定义构造函数的话，建议同时调用 parent::__construct() 来检查所有的变量是否已被赋值。当对象要输出字符串的时候，可以重载 __toString() 并自定义输出的样式。
虽然可以扩展异常基类，但不能覆盖任何一个方法，因为它们都声明为final。
内置异常类及异常类的扩展参见PHP手册。
6、4处理文件和操作系统
解析目录路径：
basemane()函数返回路径的文件名部分。
dirname()函数恰好与basename()相反，它提供路径的目录部分，**dirname()** 的操作从 PHP 5.0.0 版开始是二进制安全的。常用`dirname(__FILE__)`，返回当前文件的绝对路径。
pathinfo()函数创建一个关联数组，其中包括路径中的三个部分:目录名、基本名和扩展名。
realpath()函数将path中的所有符号链接和相对路径引用转换为相应的硬链接和绝对路径。
filesize()函数返回指定文件的大小，以字节为单位。
disk_ free space()函数返回指定的目录disk=total_ space()函数返回指定的目录所在磁盘分区的总容量(以字节为单位)。所在磁盘分区的可用空间(以字节为单位)。
确定访问和修改时间：
fileatime()函数返回文件的最后访问时间，采用UNIX时间戳格式，有错误时返回FALSE.
filectime()函数返回文件的最后改变时间，采用UNIX时间戳格式，有错误时返回FALSE
filemtime()函数返回文件的最后修改时间，采用UNIX时间戳格式，否则返回FALSE.
“最后改变时间”不同于“最后修改时间”，最后改变时间指的是对文件inode'致据的任何改变，包括改变权限、所有者、组或其他inode特定的信息，而最后修改时间指对文件内容的修改(特别是以字节为单位的文件大小)。
文件处理：
识别文件末尾字符int feof{string resource)
打开和关闭文件：在处理文件内容之前，通常需要创建所谓的句柄。同样，结束该资源的操作之后，应当销毁该句柄。
fopen()函数将文件绑定到一个句柄。绑定之后，脚本就可以通过句柄与此文件交互。fopen()还能通过一些协议(包括HTTP, HTTPS和FTP)打开资源，这些概念将在第16章讨论。打开资源时，如果指定了模式，就可以确定该资源的访问级别。

好的编程实践指出，一旦完成资源的处理，就要撤销其指针。fcloset)函数就会做此处理。
读取文件：　　file()函数能够将文件读取到数组中，各元素由换行符分隔，同时换行符仍附加在每个元素的末尾。它不像其他读/写函数，你不必建立文件句柄来读取该函数。
file _get _contents()函数将文件中的内容读到字符串中。
fgetcsv()函数使用起来很方便，将解析标记为CSV的文件中的每一行。
　　fgtes()函数返回通过打开的资源句柄读入的若干个字符，或者返回遇到换行或EOF之前读取的所有内容。
fgetss()函数与fgets()相似，只是它将从输入中清除所有HTML和PHP标记。
如果希望从用户通过表单提交的输入中删除HTML标记，请使用strip tags()函数，
　　fread()函数从handle指定的资源中读取leng七h个字符。当到达EOF或读取到length个字符时，读取将停止。使用fread()时不考虑换行符。
readfile()函数读取由filename指定的整个文件，立即输出到输出缓冲区，并返回读取的字节数。
将字符串写入文件：fwrite()函数将字符串的内容输出到指定的资源中。
读取目录内容：
opendir()会打开路径指定的目录流。
closedir()函数关闭目录流。
readir()函数返回目录中的各个元素。readdir()还返回在一般的UNIX目录列表中常见的“.”和“二”项。可以利用if语句过滤
scandir()函数是PHPS的新函数，返回一个由directory中文件和目录组成的数组，
执行Shell命令：
rmdir()函数删除指定的目录，成功时返回下RUE，否则为FALSE
如果用户对目录没有写权限，rmdir将失败。此外，目录必须为空。要删除一个非空的目录，编写一个递归函数，在删除目录前先删除其中的所有文件内容。
renamet)函数重命名文件，成功时返回丁RUE，否则返回FALSE
touch()函数设置文件filename的最后修改时间和最后访问时间，成功时返回TRUE，否则为FALSE。
系统级程序执行：
清理输入：如果没有清理可能顺次传递给系统级函数的用户输入，可能会让攻击者有机可趁，用户的输入传递给任何PHP程序执行函数之前，先对输入进行清理。有两个标准函数可以很方便地完成此工作:escapeshellargt)和escapeshellcmd()。
escapeshellargt)函数用单引号界定给定的参数，并为输入的参数中的单引号加上前缀(转义)。
　escapeshellcmd{)函数的工作前提与escapeshellarg{)相同，通过对shell元孚符转义米清理可能危险的输入。其形式为:
string escapeshellcmd{string command)
这些字符包括:#&;，1*?，一<>‘()〔〕{}$¥¥。
PHP的程序执行函数：
1.执行系统级命令
exec()函数最适合执行在服务器后台连续执行的操作系统级应用程序。
2.获取系统命令的结果
如果希望输出执行命令的结果，system()函数很有用。
passthru()函数与exec()函数相似，只不过它应当用于向调用者返回二进制输出。假设希望在浏览器显示GIF图片前，先将GIF图片转换为PNG格式图片。
4.用反引号执行shell命令

使用反引号(backtick)界定字符串时，就是在告诉PHP:该字符串应当作为shell命令来执行，反引号不是单引号，而是一个倾斜的引号，在大多数美国键盘上一般与波浪线(~)在同一个按键上。5.可代替反引号的函数
shell exec()函数提供了与反引号相同的语法形式，会执行一个shell命令，返回输出。

虽然只使用PHP来构建有意义而且强大的Web应用程序肯定是可行的，但是，如果能与底层平台和其他技术相集成，这些功能可以得到极大的扩展。
逻辑上已经实现了很多常见任务(不论应用程序是何种类型，都肯定需要反复执行这样一些任务)，所以，利用这些以社区为后盾的服务，会节省很多编程时间。
6.5日期和时间

7、 PHP基础模块代码
8、 PHP常用API分类汇总
9、 MVC框架及模板

关于MVC的知识见MVC及框架学习笔记。
要努力以可靠的开发实践作为开发的基础，其中一些实践经过多年的发展已经越来越明确。许多开发人员已经联手打造了多种Web框架，这些Web框架可以帮助其他开发人员以一种高效、快速并体现可靠开发原则的方式开发Web应用程序。
常见框架：CakePHP框架、Sola框架、symfony框架
Zend框架：
ThinkPHP：

ThinkPHP是一个快速、兼容而且‘简单的轻量级国产PHP开发框架，诞生十2006年初，原名FCS,，2007年元旦正式更名为ThinkPHP，遵循Apache2开源协议发布，从Struts结构移植过来并做了改进和完善，同时也借鉴了国外很多优秀的框架和模式，使用面向对象的开发结构和MVC模式，融合了Struts的思想和TagLib(标签库)、ROR的ORM映射和 ActiveRecord模式，封装了 CURD和一些常用操作，单一入口模式等，在模版引擎、 数据库抽象层、缓存机制、认证机制和扩展性方面均有独特的表现，从诞生到现在经历了不少的变化和发展，得到了很多朋友的关注和支持，同时也已经
越来越多地受到国内PHP开发人员的认可。

10、 PEAR包

PEAR，即PHP扩展与应用库C PHP Extension and Application Repository)的缩写，是可重用PHP组件的一个框架和分发系统，
如果你还没有开始使用PEAR，那你肯定曾花费大量精力和时间闭门造车，重复地实现了PEAR包中已有的一些特性。
在使用前先安装pear。

使用已安装的PEAR包很简单。只需利用include或require(更可取)。记住，需要将PEAR的基目录添加到include_ path指令中

11、 PHP网站部署及安全
12、 杂项
13、 PHP网站常见模块（非框架版）
14、 PHP网站常见模块（TP框架版）
15、 用PHP+Mysql构造网站系统案例（详细设计文档）