《PHP和MySQL Web开发》

chen

# 第一章 PHP快速入门

## 使用php标记

以<?php 开始，以?>为结束

## PHP中的空格

所有的间隔字符（换行、回车、空格、tab），都被认为是空格，多个空格放在一起，只会认为是一个空格。

## PHP中的注释

两种注释：

1. c语言风格的多行注释  
 ／\* ……  
 ……  
 \*/   
2. c++风格的单行注释  
 //……

## PHP的数据类型

* integer
* float
* string
* Boolean
* Array
* Object
* NULL(空)
* resource（资源）
* callable 可以传递给其他函数的函数

## 类型强度

PHP是弱类型语言（动态类型语言），变量的类型由赋给变量的值决定。

## date函数

* 用法：date('H:i, jS F');
* 说明：
  + H 是24小时格式的小时；
  + i 是分钟（两位数）；
  + j 是该月的日期，不需要补0；
  + s 是顺序后缀（比如th）；
  + F 是月份的全称；

## 理解标示符

标示符是变量的名称。 （函数和类的名称也是标示符） 。

* 标示符可以是任何长度， 可以由任何字母、数字、下划线组成。
* 标示符不能以数字开头。
* 标示符区分大小写
* 变量名名可以与函数名相同， 这一点容易引起混淆， 虽然是允许的， 但应当尽量避免。。函数名不能同名。

## 声明和使用常量

* 常量值一旦被设定后，就不能再更改。使用define函数定义常量。

define ('TIREPRICE' , 100);

* 常量的引用

引用常量不需要$, 直接使用常量的名称。

## PHP中的比较操作符

|  |  |
| --- | --- |
| 操作符 | 名称 |
| == | 等于 |
| = | 恒等于 |
| ！= | 不等于 |
| ！== | 不恒等（比较操作） |
| <> | 不等 |
| < | 小于 |
| <= | 小于等于 |
| > | 大于 |
| >= | 大于等于 |

## PHP中的逻辑操作符

|  |  |  |  |
| --- | --- | --- | --- |
| 操作符 | 名称 | 使用方法 | 结果 |
| ! | 非 | ！$b | 如果$b是false，则返回true |
| && | 与 | $a && $b |  |
| and | 与 |  | 与&&相同，但优先级较低 |
| or | 或 |  | 与两竖相同，优先级低 |

## 三元操作符 （ ？ ： ）

（$grade >= 50 ? 'Passed' : 'Failed'）

## 数字格式函数numberformat() 【来自php的math库】

* numberformat($totalamout , 3) —— 小数点后保留3位

## htmlspecialchars()函数

* htmlspecialchars() 函数把预定义的字符转换为 HTML 实体。
* 预定义的字符是：
  + & （和号）成为 &
  + " （双引号）成为 "
  + ' （单引号）成为 '
  + < （小于）成为 <
  + > （大于）成为 >

## 错误抑制操作符@

用法：

$a = @(57/0) //抑制了除数为0的错误。

如果在php.ini中启用了PHP的trackerrors特性，错误信息将会保存在全局变量$phperrormsg中。

## 执行操作符

是一对反向单引号(` `)，将反向单引号之间的命令当做服务器命令来执行。表达式的值就是命令的执行结果。具体实例如下：

$out = `ls -la`;  
` echo '<pre>.$out.</pre>';

## PHP的数据类型

* integer
* float
* string
* boolean
* array
* object
* NULL
* resource
* callable

## 常量的声明define

### 常量的声明

define ('TIREPRICE', 100);

### 常量的使用

不需要使用$符号，直接用变量名称。

### switch …… case语句 （同C语言）

switch ($find){  
 case "a" :  
 echo " ";  
 case "b" :  
 echo " ";  
 default :  
 echo " ";  
}

## 输出格式控制numberformat($val, 位数)

## declare结构

php的另一种控制结构是declare结构， 它并没有像其它结构一样在日常编程中经常使用。 常见的控制结构的常见形式如下：

declare (directive){  
 //block  
}

这种结构用来设置代码块的执行指令， 也就是， 关于后续代码如何运行的规则。 目前，PHP提供了两个执行指令， ticks和encoding

* ticks 插入指令ticks=n ， 就可以使用ticks。 它允许在代码内部每执行n行代码后运行特定函数， 这对于性能调优和调试来说是非常有用的。
* encoding 用来设置脚本的编码
* declare(encoding = 'UTF-8');

# 第2章 数据的存储与检索

## 本章主要内容

* 保存内容以便后期使用
* 打开文件
* 创建文件并写入文件
* 关闭文件
* 读文件
* 给文件加锁
* 其它有用的文件操作函数
* 更好的方式：数据管理系统

## 保存数据以便后期使用

存储数据有两种基本方法：保存到普通文件（flat file）和保存到数据库中

## 存储和检索订单

## 文件处理

### 写入文件

* 打开文件
* 将数据写入文件
* 关闭文件

### 读出文件

* 打开文件
* 从文件中读出数据
* 关闭文件

## 打开文件

使用fopen（）函数，指定文件模式。

### 选择文件模式

打开文件的三种模式：

* 打开文件为只读、只写或者读和写
* 如果是写入文件，可以选择覆盖原文件，或者在原文件末尾追加新文件，还可 以选择在该文件存在的前提下，终止程序的执行
* 如果希望在一个区分来二进制方式和纯文本方式的系统写上一个文件，还必须 指定采用的方式。

fopen（）支持以上三种的组合。

### 使用fopen（）打开文件

$fp=fopen("$DOCUMENT\_ROOT/../orders/orders.txt", 'w');

说明：

* 第一个参数代表文件路径（相对路径）| r | 只读 |

|  |  |
| --- | --- |
| r+ | 只读 |
| w | 只写 |
| w+ | 只写 |
| x | 谨慎写 |
| x+ |  |
| a | 追加 |
| a+ | 追加 |
| b | 二进制 |

* 解决打开文件时可能遇到的问题
  + 打开文件时可能会遇到文件权限问题， 需要对文件赋予权限
  + 使用@符号抑制错误，使用@抑制错误后，需要自己编写错误处理代码，应对

错。如果已经在php.ini中启用了php的trackerrors特性，错误相信将被保存在 全变量$phperrormsg中。

### 写文件

1. fwrite（）函数

使用fwrite（）或者fputs（），fputs（）是fwrite（）的别名函数。fwrite（）的调用方式如下：

fwrite($fp, $outputstring);

将保存在$outputstring中的字符串写入到$fp指向的文件中。

1. fileputcontents()

这个函数是fwrite（）的替换函数， 可以不需要调用fopen（）函数打开文件，就将字符串写入， 调用方式如下：

int fileputcontents( string filename, string data, [ int flags, resource context ] )

1. fwrite()的参数

fwrite()的原型：

int fwrite ( resource handle, string string [, int length ]);

第三个参数length是写入的最大字符数。 如果给出这个参数，fwrite（）将向handle指向的文件写入字符串，写入长度为length。

1. strlen（）函数

获取字符串的长度

### fgets() fgetss() fgetcsv()

* fgets()每次读取一行数据
* fgetss()可以过滤字符串中的php和html标记
* fgetcsv() 读取一行，将结果返回数组

### 通过FTP或HTTP打开文件

### 解决打开文件时可能遇到的问题

* 文件权限问题

chmod 777 创建最高的读写权限

### 文件的锁定flock（）

* 避免两个客户同时打开同一个文件产生问题，使用文件锁定锁定文件。
* flock（文件，锁定模式，可选）
* 锁定模式

|  |  |
| --- | --- |
| 操作值 | 意义 |
| LOCKSH | 读操作锁定，文件可以共享，其它人可以读该文件 |
| LOCKEX | 写操作锁定，此时文件不被共享，其它人无法读取 |
| LOCKUN | 释放已有的锁定 |

# 第3章 使用数组

## 本章内容

* 数字索引数组
* 非数字索引数组
* 数组操作符
* 多维数组
* 数组排序
* 数组函数

## 数字索引数组

### 数字索引数组的初始化

1. 索引默认从0开始
2. 初始化方式1:

* $products = array('tires', 'oil' 'spark plugs');

1. 初始化方式2：（php5.4开始支持）

* $products = ['tires', 'oil' 'spark plugs'];

1. array（）结构简介

* array()是一个语言结构，不是函数。 可以简单将数组复制给另一个数组。

1. 使用rang（）函数自动创建数字数组

* $number = range(1,10); //创建一个1-10的数字数组  
  $odds = range(1,10,2); //创建一个1-10之间的偶数数组  
  $letters = range('a','z'); //创建一个a-z的字符数组

### 访问数组内容

使用$products[0],$products[1]……来访问数组$products的内容。php的数组不需要预先初始化或创建，在第一次使用的时候会自动创建。使用$products[n]可以直接扩充数组。

### 使用循环访问数组

1. 使用for循环访问数组

* for($i=0;$i<3;$i++){  
   echo "$products[$i]";  
  }

1. 使用foreach访问数组

* foreach ($products as $current){  
   echo $current." ";  
  }  
  // 说明：每次循环都取出数组中的值，并保存到变量$current中，一直到数组的最后一个元素，结束循环

## 使用不同索引的数组（相关数组）

### 初始化相关数组1

$price = array('tires'=>100, 'oil'=>10, 'spark plugs'=>4);

说明：相关数组由键和相对应的值组成。

### 初始化相关数组2

$price[‘tire’] = 20;

### 访问相关数组

$price['tire'];

### 使用循环语句访问数组元素（foreach、list、each）

1. foreach循环

* foreach($price as $key => $value)  
   echo $key.'=>' .$value.<br>';

1. each（）结构

* while($element = each($price))  
  {  
   echo $element['key'];  
   echo '-';  
   echo $element['value'];  
   echo '<br>';  
  }

1. 说明：
   * each()一次返回一个包含两个元素的数组，一个元素是被执行数组的索引，一个

* 元素是该索引对应的值，利用each（）的这个特性，可以配合list（）函数来访 问数组： list($product,$price) = each($prices);
* 再利用循环，可以遍历所有的数组元素： while(list($product,$price) = each($prices)) { echo "$product - $price<br>"; }
  + each()将改变数组的指针位置，若要再次使用该数组，需要使用reset（）函 数，如reset($prices),将数组$prices的指针回到初始位置。

### 数组操作符

### 多维数组

数字索引的多维数组，可以直接使用嵌套的for循环来访问

### 数组的排序

1. 使用sort函数——按升序排列（大写字母排在小写字母的前面）

* $products = array('tires', 'oil', 'spark plugs');  
  sort($products);
* sort()将对数组内的三个元素按照字母升序排列

1. 关于sort（）的说明
2. 使用asort（）函数和ksort（）函数对相关数组排序
   * asort（）按值排序
   * ksort（）按索引排序
3. 反向排序
   * rsort（）
   * arsort（）
   * krsort（）

### 多维数组的排序

1. 用户定义排序
2. 反向用户排序

### 对数组进行重新排序（随机）

1. 使用shuffle（）函数

* 实现随机功能

### 从文件载入数组

* 用file（）载入文件，返回一个数组，文件的每一行为一个数组元素；
* count（）用来计算数组中的元素个数;
* explode()分割字符串，返回一个数组；

# 第4章 字符串操作和正则表达式

## 本章要点

* 字符串的格式化
* 字符串的连接和分割
* 字符串的比较
* 使用字符串函数匹配和替换自字符串
* 使用正则表达式

## 创建一个智能表单邮件

### mail（）函数

* 几个参数：目的地邮件地址，邮件主题，邮件内容，第四个参数可选，可以用来发送有

效的额外的邮件头

* 示例：

mail($toaddress, $subject, $mailcontent, $fromaddress);

## 字符串的格式化

### 字符串的整理：chop（）、ltrim（）和trim（）

* trim（）——去除字符串开始位置和结束位置的空格，并将结果返回。默认情况 下除去的字符是换行符和回车符（），水平和垂直制表符（0b）。
* ltrim（）——去除字符串开始位置的空格
* rtrim（）——去除字符串结束位置的空格
* chop()是rtrim（）的别名

### 格式化字符串以便显示

* 使用html格式化：nl2br（）函数：用字符串作为输入参数，用xhtml中的<br />标记代替字符串中的换行符
* 为打印输出而格式化字符串
* 改变字符串中字母的大小写

strtoupper（）——将字符串转变为大写 strtolower（）——将字符串转变为小写 ucfirst（）——如果字符串的第一个字符是字母，将该字母转化为大写 ucwords（）——将字符串中的每个单词的第一个字母转化为大写。

### 格式化字符串以便存储：addslashes（）和stripslashes（）

重新格式化字符串，以便存入数据库。对于字符串来说，某些字符是有效的，当 时将它们插入数据库时可能会引起一些问题，比如引号（单引号、双引号）、反 斜杠和null字符，数据库会将这些字符解释成控制符。为了将这些字符进行转义， 需要在它们前面加一个反斜杠（这个规则对所有特殊字符通用）。addslashes（） 和stripslashes（）就是专门用来进行这样的格式化处理

addslashes（）——添加反斜杠标记 stripslashes（）——去除反斜杠标记

### 字符串连接和字符串的分割

1. explode（）、implode（）和join（）
   * explode（）——根据指定的分隔符分割字符串
   * implode（）和join（）实现与explode（）相反的效果
2. 使用strtok（）函数
3. substr（）函数

* 截取字符串 **用法：** string substr（string string， int start，【int length】）；
* 第二个参数代表起始位置，若为0，代表从头部开始。
* 第三个参数可选，代表截取的长度，若省略，将从起始处截取到结尾，若是负数， 从起始处往头部截取。

### 格式化字符串以便存储：addslashes（）和stripslashes（）

重新格式化字符串，以便存入数据库（主要是自动处理引号、斜杠、NULL字符等）

magicquotesgpc这个配置将自动开启格式化。

## 字符串的比较

### 字符串的排序：strcmp（）、strcasecmp（）和strnatcmp（）

* strcmp()——比较两个字符串

int strcmp（string str1，string str2）；

str1和str2相等，返回0 str1排在str2的后面（大于），返回一个正数 小于，返回一个负数

### 使用strlen（）计算字符串长度

## 字符串函数匹配和替换字符串

### 在字符串中查找字符串

* strstr（）
* strchr（）
* strrchr（）
* stristr（）

### 查找子字符串的位置

* strpos（）
* strrpos（）

### 替换字符串

* strreplace()

srtreplace("","",$name); //防止header注入。

* substrreplace()

## 正则表达式介绍

PHP支持两种风格的正则表达式语法：POSIS和Perl（POSIX正则表达式更容易掌 握，但是不是二进制安全的）

### 基础知识

1. 分隔符
   * 最常用的分隔符是 / , /shop/这个正则表达式的作用是匹配shop，正则表达式中使用/，需要使用。如/<http:\/\//>
   * 分隔符后可以添加模式修饰符；如/shop/!，表示不区分大小写的方式匹配shop

### 字符集7和类

# 第5章 代码重用与函数编写

## 本章内容

* 代码重用的好处
* 使用require（）和include（）函数
* 函数介绍
* 定义函数
* 使用参数
* 理解作用域
* 返回值
* 参数的饮用调用和值调用
* 实现递归
* 使用命名空间

## 代码重用的好处

* 成本
* 可靠性
* 一致性

## 使用require（）和include（）函数

### 两者的差异

唯一的区别在于函数失败后require（）函数将给出一个致命的错误。而include 只是给出一个警告。

### requireonce()和includeonce()

这两个是变体函数，确保文件只被引用一次, 防止函数重复定义

## 理解作用域

关键字global可以把局部变量转化为全局变量， 全局可见

## 参数的引用传递和值传递

才参数前加上&符号，表示参数的引用传递。

## 使用return关键字

## 关于本章实例 --home.html-- 的笔记

### 在html文件内部编写css

<style type="text/css">  
 h1 {  
 color:white; //文字颜色  
 text-align:center;  
  
 }  
  
 .menu {  
 color:white;  
 font-size:24pt;  
 text-align:center;  
 font-wight:bold;  
 }  
 td {  
 background:black; //背景颜色  
 }  
 p.foot {  
 color:white;  
 text-align:center;  
 }

### 表格table的处理

<table width="100%" cellpadding="12" cellspacing="0" border="0">  
 <tr bgcolor="black">  
 <td align="left"><img src="./img/s-logo.gif" alt="TLA logo"  
 height="70" width="70"></td>  
 <td>  
 <h1 ="white">TLA Consulting</h1>  
 </td>  
 <td align="right"><img src="./img/s-logo.gif" alt=TLA logo" height="70" width="70" ></td>  
 </tr>  
 </table>

# 第6章 面向对象的php

## 本章主要内容

* 面向对象的概念
* 类、属性和操作
* 类属性
* 类常量
* 类方法的调用
* 继承
* 访问修饰符
* 静态方法
* 类型提示
* 延迟静态绑定
* 对象克隆
* 抽象类
* 类设计
* 设计的实现
* 高级的面向对象功能

## 理解面向对象的概念

### 类和对象

### 多态性

### 继承

### 在PHP中创建类、属性和操作

1. 类的结构

* **类的定义：** class classname { }

1. 构造函数

* 构造函数的名称必须是\_construct( );
* **示例** class classname { function \_\_construct（） }

1. 析构函数

* 名称必须是\_destruct()，不能带任何参数

1. 类的实例化

* 在声明一个类后，需要创建一个对象（一个特定的个体，即类的一个成员）并使 用这个对象。这个过程就是创建一个类的实例或实例化一个类。
* 使用关键词new来实例化一个类。

1. 使用类的属性
   * 在类自身中，使用$this指针来访问属性
   * 类的外部访问接口
     + get()函数返回类中的属性的值；
     + set()函数设置类中属性的值；

* \*应用实例：\*
* class classname  
  {  
   public $attribute;  
   function \_\_\_get($name)  
   {  
   return $this->$name;  
   }  
    
   function \_\_set($name, $value)  
   {  
   $this->$name = $value;  
   }  
  }  
  $a = new classname();  
  $a->$attribute = 5 ; // 该语句间接调用\_set()函数  
  $a->$attribute //该语句间接调用\_get()函数

1. 使用private和public关键字控制访问
   * 默认是public（公有），公有的属性或方法可以在类的内部和外部进行访问，可

* 以被继承。
  + private（私有），只能在类的内部进行访问，不会被继承。
  + protected（保护），只能在类的内部进行访问，可以被继承

1. 类的操作和调用
2. 在php中实现继承

* 使用extends实现继承 class b extends a { …… }

1. 通过继承使用private和proteted访问修饰符控制可见性
2. 重载
3. 理解多重继承

* php不支持多重继承，每个子类只能有一个父类。

1. 实现接口

* php通过接口来实现多重继承

1. PHP面向对象的高级功能
   1. 使用Per-class常量(用::来调用）
   * 可以在不初始化类的情况下，调用类中的常量 **实例** class Math() { const pi = 3.14159; } $pi = Math::pi;
   1. 实现静态方法
   * static关键字，实现static方法，等价与Per-Class常量的思想。 **实例** class Math { static function squared($input) { return $input\*$input; } $result = Math::squared(8); }

# 第7章 错误和异常处理

## 本章主要内容

* 异常处理的概念
* 异常控制结构：try ... throw... catch
* Exception类
* 用户自定义异常
* 异常和PHP的其他错误处理机制

## 异常处理的概念

异常处理的基本思想是代码在try代码块被调用执行。

例如： try { // code goes here }

如果try代码块出现某些错误，我们可以执行一个抛出异常的错误。PHP中，异常 必须手动抛出。

例如： throw new Exception('message' ,code);

throw 关键字将触发异常处理机制。它是一个语言结构而不是函数，必须给它传 递一个参数。它要求接收一个对象。在最简单的情况下，可以实例化一个内置的 Exception类。

# 第18章 MySQL数据库概述（兄弟连内容插入）

MySQL是小型关系型数据库管理系统。

## 数据库的应用

数据库是计算机应用系统中一种专门管理数据资源的系统。管理这个数据库的软件就称为数据库管理系统. 一个数据系统（database system） 可以分为数据（database）和数据库管理系统（database management system，DBMS）

## 结构化查询语言SQL

数据库操作指令称为SQL（structured Query Language）语言，即结构和查询语言，是一种专门用于查询和修改数据库中的数据，以及对数据库进行管理和维护的标准化语言。

SQL语言的4个部分：

* 数据定义语言DDL
* 数据操作语言DML
* 数据查询语言DQL
* 数据控制语言DCL

## MySQL的常见操作

### 数据库的连接和关闭

* 连接数据库
* mysql -h 服务器地址 -u 用户名 -p 密码

### 创建新用户并授权

GRANT 权限 ON 数据库.数据表 TO 用户名@登陆主机 IDENTIFIED BY “密码”

说明：

* @“%” 表示全网所有的主机，这存在安全隐患。
* @localhost 表示只允许本机登陆。

### 创建、删除数据库

* 创建

CREATE DATABASE 数据库名；

* 删除
* DROP DATABASE 数据库名；

### 查看数据库

查看所有数据库

SHOW DATABASES;

### 使用数据库

USE 数据库名；

## 数据表内容的简单管理

### 向数据表插入行记录（INSERT）

INSERT INTO book(bookname,publisher,author,price) VALUES("细说PHP”，‘电子工业出版社’，‘骆高峰’,89.00);

### 从MySQL数据表中查询数据记录（SELECT）

SELECT id，bookname, publisher, author, price FROM book;

### 更改数据表中的记录（UPDATE）

UPDATE book SET price=79 WHERE id=2;

说明： 若没有WHERE条件限制，则更改所有行。

### 删除数据表中的记录（DELETE）

DELETE FROM book WHERE ic=‘1’;

说明：若没有WHERE条件限制， 则删除所有行。

# 第8章 设计web数据库

## 本章主要内容

* 关系数据库的概念和术语
* web数据库的设计
* web数据库的架构

## 关系数据库的概念和术语

关系数据库是最常用的数据库类型

### 表

关系数据库由关系组成，这些关系通常称为表。一个表格就是一个数据的表。

### 列

表中的每一列都有唯一的名称，包含不同的数据。每一列都是一个相关的数据类型。 列也叫做字段或属性。

### 行

每一行具有相同的格式，具有相同的属性。行也称为记录或元组（Tuple）

### 值

行和列的交点组成一个值。该值必须与该列定义的数据类型相同。

### 键（主键和外键）

用来标识数据的列称为键或主键，一个键可能由几列组成。经常使用键作为多个表之间的引用。

### 模式

数据库整套表的完整设计称为数据库的模式（Schema），模式应该显示表格及表格的列、每个表格的主键和外键。模式不包含任何数据。

### 关系

* 用外键表示两个表中数据之间的关系。
* 三种基本的关系类型：
  + 一对一
  + 一对多
  + 多对多

## 考虑真实的建模对象

## 避免保存冗余数据

冗余造成的三种异常

* 修改异常
* 插入异常
* 删除异常

## 表格类型的总结

通常，数据库有两种类型的表组成：

* 描述现实世界对象的简单表。 这些表也可能包含其他简单对象的键，他们之间有一对一或一对多的关系。
* 描述两个世界对象的多对多关系的关联表。

# 第9章 创建Web数据库

## 本章概要

* 创建一个数据库
* 设置用户权限
* 权限系统的介绍
* 创建索引
* 选择MySQL中的列类型

## 使用MySQL监视程序

* 命令用；分开
* SQL语句不区分大小写
* 数据库和数据表名称区分大小写

## 登录到MySQL

在mac中 ，进入/usr/local/mysql/bin, 执行: ./mysql -u root -p 登陆 mysql 命令：

mysql -h hostname -u username -p

说明：

* h 命令选项用于指定连接的主机。 如果服务器是本机，可忽略该参数。
* u 数据库用户名。
* p 告诉服务器需要一个密码来连接

## 创建数据库和用户

### 创建数据库

create databse dbname;

## 设置用于与权限

一个MySQL系统可能有许多用户。 为安全起见，root用户通常只用作管理目的。 对于每个需要使用该系统的用户，应该为他们创建一个账号和密码。

## MySQL权限系统介绍

### 最少权限原则

最少权限原则可以用来提高任何计算机系统的安全性。 它是一个基本的、 但又非常重要的原则。该原则内容如下：

一个用户（或者一个进程）应该拥有能够执行分配给他的任务的最低级别的权限。

### 创建用户：GRANT命令

GRANT和REVOKE命令分别用来授予和取消MySQL用户的权限， 这些权限分4个级别。 它们分别是：

* 全局
* 数据库
* 表
* 列

1. GRANT命令

* 用来创建用户并赋予他们权限。GRANT命令的语法：
* GRANT privileges [columns] ON item TO username [IDENTIFIED BY 'password'] [REQUIRE ssloptions] [WITH [GRANT OPTION | limitoptions]]
* 说明：
  + []中的是可选的
  + 占位符
    - privileges 由逗号分开的一组权限
    - columns 是可选的， 可以用它对没一个列指定权限。 也可以使用单列的名称或者用逗号分开的一组列的名称。

### 权限的类型和级别

1. 用户权限

|  |  |  |
| --- | --- | --- |
| * 权限 | * 应用于 | * 描述 |
| * select | * 表和列 | * 允许用户从表中选择行（记录） |
| * insert | * 表和列 | * 允许用户在表中插入新行 |
| * update | * 表和列 | * 允许用户修改现存表里行的值 |
| * delete | * 表 | * 允许用户删除现存表的行 |
| * index | * 表 | * 允许用户创建和拖动特定表索引 |
| * alter | * 表 | * 允许用户改变现存表的结构，例如，可添加列、重命名列或表、修改列的数据类型 |
| * create | * 数据库、表、索引 | * 允许用户创建新数据库、表或索引。如果在grant中指定了一个特定的数据库、表或索引，用户就只能够创建他们，即用户必须首先删除它 |
| * drop | * 数据库、表、视图 | * 允许用户删除数据库、表或视图 |
| * event | * 数据库 | * 允许用户查看、创建、修改以及删除事件调度器中的事件（目前不讨论） |
| * trigger | * 表 | * 允许用户对已授权表执行创建、执行、或删除触发器 |
| * create view | * 视图 | * 允许用户查看创建视图的查询 |
| * proxy | * 所有对象 | * 允许用户切换到其它用户，类似于UNIX的su命令 |
| * create routine | * 存储过程 | * 允许用户创建存储过程和函数 |
| * execute | * 存储过程 | * 允许用户运行存储过程和函数 |
| * alter routine | * 存储过程 | * 允许用户修改存储过程和函数的定义 |

* 常规用户的权限大多数都是相对无害的。 通过重命名表，alert权限可以用来绕过权限系统设置， 但是大多数用户需要它。 安全常常是可用性与安全性的折中。

1. 特殊权限

|  |  |
| --- | --- |
| * 权限 | * 描述 |
| * all | * 授予所有的权限，也可以写成all privileges |
| * usage | * 不授予权限。 这将创建一个用户并允许他登陆， 但是不允许进行任何操作。 通常在以后会授予该用户更多的权限。 使用grant和usage语句创建用户并授予权限等同于create user 语句 |

### revoke命令 -- 收回权限

和grant命令相反，从一个用户收回权限。 语法上于grant相似。

revoke privileges [(columns)]  
ON item  
FROM user\_name;

如果已经给出了with grant option 字句， 可以按如下方式撤销权限（以及所有其它权限）

revoke all privileges ， grant option  
FROM user\_name;

### 使用GRANT和REVOKE示例

* 创建一个管理员
* grant all  
  on \*.\*  
  to 'fred' identified by 'mnb123'  
  with grant option;
* 以上命令授予了用户名为fred、密码为mnb123的用户使用所有数据库的所有权限、并允许他向其他人授予这些权限。
* 如果不希望用户在系统中存在，可以用用如下方式撤销授权
* revoke all privileges, grant option  
  from 'fred';
* 创建一个没有任何权限的常规用户
* grant usage  
  on books.\*  
  to 'sally'@'localhost';
* 授予权限
* grant select,insert,update,delete,index,alter,create,drop  
  on books.\*  
  to 'sally'@'localhost'
* 当不再使用数据库时，可以按如下方式撤销所有权限
* revoke all  
  on books.\*  
  from 'sally'@'localhost';

## 设置Web用户

要通过PHP连接到MySQL，需要为PHP脚本创建一个用户。 这里，同样使用最少权限原则。 在大多数情况下，PHP脚本只需要能执行选择（select）、插入（insert）、删除（delete）和更新（update）操作， 因此可以按如下方式设定这些权限：

grant select,insert,delete,update  
on books.\*  
to 'bookorama' identified by 'bookorama123';

如果使用了主机托管服务，通常可以获得基于用户类型的数据库权限。 典型的，可以提供相同用户名和密码以用于命令行（建立表等）操作和Web脚本连接（查询数据库）。 对命令行和Web连接使用相同的用户名和密码是不够安全的。 可以建立其它具有相同权限级别的用户，操作如下：

grant select,insert,update,delete,index,alter,create,drop  
on books.\*  
to 'bookorama'@'localhost'  
identified by 'bookorama123';

## mysql 8.0 建立用户和设置权限的特别的地方

8.0版本建立用户和设置权限不能放在一起，要分开，否则会报错

-- 创建一个数据库 CREATE DATABASE usdemodev DEFAULT CHARACTER SET utf8mb4 ;

-- 创建一个user create user 'frank'@'%' IDENTIFIED BY '123456';

-- 授权 GRANT SELECT,INSERT ,UPDATE ,DELETE ,DROP,ALTER ,INDEX ON usdemodev.\* TO 'frank'@'%'; -- 刷新权限 flush privileges;

-- 查看权限 SHOW GANTS FOR 'frank'@'%';

-- 回收权限 REVOKE SELECT, INSERT, UPDATE, DELETE, DROP, INDEX, ALTER ON `usdemodev`.\* from `frank`@`%` ;

## 创建数据库表

### Book-O-Rama 数据库模式

1. 模式

* Customers(CustomerID, Name, Address, City) Orders(OrderID, CustomerID, Amount, Date) Books(ISBN, Author, Title, Price) OrderItems(OrderID, ISBN, Quantity) BookReviews(ISBN, Review)

1. 脚本代码

* create table Customers(  
   CustomerID int unsigned not null auto\_increment primary key,  
   Name char(50) not null,  
   Address char(100) not null,  
   City char(30) not null  
  );  
  create table Orders(  
   OrderID int unsigned not null auto\_increment primary key,  
   CustomerID int unsigned not null,  
   Amount float(6,2),  
   Date date not null,  
   foreign key(CustomerID) references Customers(CustomerID)  
  );  
  create table Books(  
   ISBN char(13) not null primary key,  
   Author char(50),  
   Title char(100),  
   Price float(4,2)  
  );  
  create table Order\_Items(  
   OrderID int unsigned not null,  
   ISBN char(13) not null,  
   Quantity tinyint unsigned,  
    
   primary key(orderid,ISBN),  
   foreign key (OrderID) references Orders(OrderID),  
   foreign key (ISBN) references Boosk(ISBN)  
  );  
  create table Book\_reviews(  
   ISBN char(13) not null primary key;  
   Review text,  
    
   foreign key (ISBN) references Books(ISBN)  
  );

1. 关键字解释
   * NOT NULL 表示必须有一个值， 如果没有指定， 则表示可以为空NULL。
   * Autoincrement 是一个特殊的MySQL特性，可以在整数列上使用，意义为在表中插入行时，如果该字段为空，那么MySQL将自动产生一个唯一的标识符（自动增加1）
   * primary key 表示主键， 只适用于单列主键

### 插入数据库

sql脚本：

use bookorama;  
  
insert into Customers values  
 (1,'Julie Smith','25 Oak Street','Airport West'),  
 (2,'Alian Wong','1/47 Haines Avenue','Box Hill'),  
 (3,'Michelle Arthur','357 North Road','Yarraville');  
insert into Books values  
 ('0-672-31697-8','Michael Morgan','Java 2 for Professional developers',34.99),  
 ('0-672-31745-1','Thomas Down','Installing Debian GNU/Linux',24.99),  
 ('0-672-31509-2','Pruitt, et al.','Teach Yourself GIMP in 24 Hours',24.99),  
 ('0-672-31769-9','Thomas Schenk','Caldera OpenLinux System Administration Unleashed',49.99);  
insert into Orders values  
 (null,3,69.98,'2007-04-02'),  
 (null,1,49.99,'2007-04-15'),  
 (null,2,74.98,'2007-04-19'),  
 (null,3,24.99,'2007-05-01');  
INSERT INTO Order\_Items VALUES  
 (1, '0-672-31697-8', 2),  
 (2, '0-672-31769-9', 1),  
 (3, '0-672-31769-9', 1),  
 (3, '0-672-31509-2', 1),  
 (4, '0-672-31745-1', 3);  
INSERT INTO Book\_reviews VALUES  
 ('0-672-31697-8', 'The Morgan book is clearly written and goes well beyond  
 most of the basic Java books out there.');

# 第11章 使用PHP从Web访问MySQL

## 本章主要内容

* Web数据库架构及工作原理
* 从Web查询数据库的基本步骤
* 设置数据库连接
* 获取可用数据库信息
* 选择要使用的数据库
* 查询数据库
* 断开数据库连接
* 写入新信息
* 使用prepared statement
* 使用PHP与数据库交互的其它接口
* 使用通用数据库接口：PDO

## web数据库架构及工作原理

1. 用户Web浏览器发出针对特定页面的HTTP请求。例如，用户可能在Book-O-Rama站点发起搜索Michael Morgan编写的所有图书的请求，搜索结果页面是results.php
2. Web服务器接收到针对results.php的请求，读取该文件，并将文件传给PHP引擎处理
3. PHP引擎开始接卸该脚本。 脚本中有一个连接数据库的命令，执行查询命令（执行图书搜索操作）。 PHP打开MySQL服务器连接，并且发送查询命令
4. MySQL数据库接收到数据库查询指令，执行该指令并将结果返回给PHP引擎。
5. PHP执行脚本结束，将查询结果格式化为HTML，发送HTML至Web服务器。
6. Web服务器将HTML返回给用户浏览器

## 从Web查询数据库

### 步骤

任何可以从Web访问数据库的脚本都会执行如下的基本步骤：

1. 检查和过滤来自用户的输入数据
2. 创建和设置数据库连接
3. 查询数据库
4. 读取查询结果
5. 向用户展示搜索结果

### 检查并过滤输入数据

过滤掉关键字前后可能存在的空格。使用trim（）函数去除输入数据前后的空格。

检查是否输入了搜索条件

if(!$searchtype || !$searchterm){  
 echo '<p>You have not entered search details. <br />/  
 Please go back and try again. </p>' ;  
 exit;  
}

### 设置数据库连接

连接MySQL数据库的PHP基础函数库是mysqli。 i表示优化版本，未优化版本为mysql

1. 面向对象的语法：

* @$db = new mysqli('localhost', '用户名', '密码');
* 备注：在MAC中本机地址用127.0.0.1， 使用localhost可能会无法连接， 调用$db对象访问数据库。

1. 面向过程的语法：

* @$db = mysqli\_connect('localhost', '用户名', '密码');

1. 两者的差别

* 面向过程的版本以mysqli开始，并且要求将从mysqliconnect()函数返回的资源句柄作为参数在面向过程版本函数中传递。 对于数据库连接，这是一个例外，因为可以由mysqli对象的构造函数实现。

1. 数据库连接有效性检查（面向对象和面向过程相同）

* if(mysqli\_connect\_errno()){  
   echo '<p>Error: Could not connect to database <br />  
   Please try again later. </p>';  
   exit;  
  }
* 备注：该函数返回一个资源，而不是对象。 该资源代表到数据库的链接，如果用面向过程的方式，需要将该资源传递给所有mysqli函数。

# 第12章 MySQL高级管理

## 本章概要

* 深入理解权限系统
* 提升MySQL数据库安全
* 获得数据库更多信息
* 使用索引加速查询
* 优化数据库
* 备份和恢复
* 实现复制

## 深入理解权限系统

当执行一个GRANT语句时， 它将影响存在于特殊数据库的表， 该数据库名为mysql。 权限信息保存在该数据库的7张表中。 基于此设计，在授权时必须关注对mysql数据库访问权限的控制。

以管理员身份登录数据库，使用如下语句查看mysql数据库信息：

use mysql；

# 第19章 与文件系统和服务器交互

## 17.1 上传文件

### php.ini中关于文件上传的配置设置

|  |  |  |
| --- | --- | --- |
| 指令 | 描述 | 默认值 |
| fileuploads | 是否支持HTTP文件上传 | on |
| uploadtmpdir | 文件上传的临时目录。如果没有设置该值，将使用系统默认值。 | NULL |
| uploadmaxfilesize | 控制文件上传的最大容量，日过文件大于该值，php将写入0字节大小的占位符文件。 | 默认值为2m |
| postmaxsize | php可以接受的post数据大小的最大值，必须大于上传最大容量 | 8m |

1. mac 中的php.ini

* Mac OS X中没有默认的php.ini文件，但是有对应的模版文件php.ini.default， 位于/etc/php.ini.default ，可以自行拷贝一份进行修改。

### 文件上传的html表单

<!doctype html>  
<html>  
 <head>  
 <title>Upload a File</title>  
 <meta charset="utf-8" />  
 </head>  
 <body>  
 <div align="center">  
 <h1>Upload a File</h1>  
 <form action="upload.php" method="post" enctype="multipart/form-data"> /\* 文件上传表单 \*/  
 <input type="hidden" name="MAX\_FILE\_SIZE" value="1000000" /> /\*用隐藏域设置上传文件最大大小\*/  
 <label for="The\_file">Upload a file:</label>  
 <input type="file" name="the\_file" id="the\_file" />  
 <input type="submit" value="Upload File" />  
 </form>  
 </div>  
 </body>  
</html>

文件上传表单必须使用POST， 不能使用GET， 该表单的其它特性包括：

* <form>标记中，必须设置属性enctype = "multipart/form-data" ，告诉服务器有常规信息的文件要上传；
* 如果没有服务器端的配置控制文件上传的大小，必须有个表单域用来设置上传文件的大小。可以使用如下的隐藏域来指定：

<input type="hidden" name="MAX\_FILE\_SIZE" value="1000000" >//单位为字节

* type属性值设置为file，用于上传文件，并设置name属性（在php处理中代表上传文件名称）
* 指定的大小是可以上传文件的最大大小（以字节为单位）。 上述代码指定为1000000字节（1MB）。根据应用需要，可以调整该值设置。 如果使用MAXFILESIZE作为隐藏的表单域， 它将覆盖服务，器端的最大大小设置（如果其值小于uploadmaxfilesize和postmaxsize设置）。
* 需要类型为file的输入框
* <input type="file" name="the\_file" id="the\_file" />
* 可以选择任何文件名称， 但是必须记住， 将在处理上传文件的PHP脚本中使用该名称访问文件。

### 处理文件的php脚本

文件上传后，被保存在php.ini文件的uploadtmpdir指定的临时目录中。如果在php脚本结束前没有移动、复制、或重命名该文件，该文件会在脚本结束后自动删除。

需要在PHP脚本中处理的数据保存在超级全局数组$FILES中。 $FILES数组项将通过HTML表单的<file> 标记名称来保存。 假设表单元素名称为thefile：

1. 关于$FILE全局数组的说明：

* html代码：
* <input type="file" name="the\_file" …… />
  + name属性的值“thefile",用于$FILE数组，代表上传的文件名
* -关于$FILE[]数组：
  + $FILE['thefile']['tmpname'] --- 上传文件的临时文件的名称和位置
  + $FILE['thefile']['name'] --- 上传文件的初始名称
  + $FILE['thefile']['size'] --- 上传文件的大小
  + $FILE['thefile']['type'] --- 文件的类型，比如text/plain或image/png
  + $FILE['thefile']['error'] --- 与上传相关的错误代码
    - 0 —— 没有错误
    - 1 —— 上传的文件太大，超出php.ini的规定值
    - 2 —— 超出表单元素规定的最大值
    - 3 —— 文件只是部分上传
    - 4 —— 文件没有上传
    - 6 —— 没有指定临时目录
    - 7 —— 文件写入失败
    - 8 —— php扩展停止了文件上传过程

1. 文件上传处理
   * $uploadfile = './upload/' . $FILE['thefile']['name'];前一半是上传的路径，后一半是上传的文件名称
   * isuploadfile($FILE['thefile']['tmpname']) —— 判断文件是否上传
   * moveuploadedfile($FILE['thefile']['tmpname'],$uploadfile) —— 移动临时文件到指定的目录和文件名。

# 第11章 使用PHP从Web访问MySQL数据库

## 从Web查询数据库的基本步骤

1. 检查并过滤来自用户的数据。
2. 建立一个到适当数据库的连接。
3. 查询数据库。
4. 获取查询的结果。
5. 将结果显示给用户。

## 检查与过滤用户输入数据

### 过滤开始和结尾处的空格（trim（）函数）

使用trim函数过滤，trim（）还可以过滤掉指定的字符。 例如： trim（$sdinng,'dkji')——过滤掉dkji

### 格式化字符串，以便输入到数据库（addslashes（））

用addslashes（）自动转义 空格、引号、斜杠、NULL等。 防止 数据库把这些字符当做控制字符处理。

## 和数据库建立连接

### 面向对象的方法

@ $db = new mysqli ( ' 主机名 ' , ' 数据库用户名 ' , ' 数据库密码 ' , ' 数据库名称 ' );

### 面向过程的方法

@ $db = mysqliconnect ( ' 主机名 ' , ' 数据库用户名 ' , ' 数据库密码 ' , ' 数据库名称 ' );

## 查询数据库

### 建立查询语句；

$query = " select \* from books where $searchtype like ' %$searchterm% ' ";

——这里用了模糊查询 like ，和通配符 %

### 执行查询；

1. **面向过程：**

$result = mysqliquery($query);

1. **面向对象：**

$result = $db->query($query) ——$db是连接数据库是建立的数据库对象，返 回的是一个对象

### 检索查询结果

1. 返回查询结果的行数（即满足条件的数据有几条）
   1. **面向过程：**

* $numresults = mysqlinumrows($result);
  1. **面向对象：**
* $numresults = $result->numrows;

1. 返回查询到的每一行的数据
   1. **面向过程：**

* $row = $result->fetchassoc();——以数组的形式返回每一行的数据，要放在循 环内
  1. **面向对象：**
* $row = mysqlifetchassoc($result);

1. 显示查询到的数据的字段

* stripslashes($row['isbn']);——数据库字段名就是 返回的每行数据数组$row的 下标，stripslashes()函数的作用是去除数据库格式。

## 断开数据库连接

### 释放数据库的连接

1. **面向对象：**

$result ->free();

1. **面向过程：**

mysqlifreeresult($result);

### 断开数据库连接

1. **面向对象：**

$db->close();

1. **面向过程：**

mysqliclose($db);

## 将新信息写入数据库

# 第23章 在PHP中使用会话控制

## 本章主要内容

* 什么是会话控制
* cookie
* 创建一个会话控制
* 会话变量
* 会话和身份验证

## 什么是会话控制

http是无状态的协议，没有一个内建机制来维护两个事物之间的状态。当一个用 户在请求一个页面后再请求另外一个页面时，http将无无法告诉我们这两个请求 是来自同一个用户。

会话控制的思想是指能够在网站中根据一个会话跟踪用户。 如果我们可以做到这 点，就可以很容易的做到对用户登录的支持，并根据其授权级别和个人喜好显示 相应的内容。我们可以根据会话控制记录该用户的行为。 还可以实现购物车。

在php4及其以后版本中，php自身包含了会话控制函数。 自从超级全局变量概 念的引入， 会话控制方法就发生来一些变化。 如今可以使用$SEEION超级全 局变量。