[HTML5 1](#_Toc5545)

[文档结构 1](#_Toc9591)

[编码格式 1](#_Toc11775)

[html实体 1](#_Toc19459)

[文本标记 1](#_Toc27855)

[图像img 2](#_Toc29901)

[链接a 2](#_Toc19679)

[列表list 2](#_Toc9980)

[表格table 3](#_Toc6749)

[表单form 3](#_Toc256)

[框架集iframe 5](#_Toc30966)

[CSS 6](#_Toc743)

[css的使用方式 6](#_Toc28486)

[css使用方式优先级 6](#_Toc11392)

[选择器优先级 6](#_Toc19202)

[选择器 6](#_Toc5469)

[文本样式 8](#_Toc2903)

[盒子模型 9](#_Toc5691)

[列表list 11](#_Toc11081)

[转换transform 11](#_Toc5723)

[动画 12](#_Toc13348)

[透明度 13](#_Toc25793)

[表格 13](#_Toc46)

[老版本浏览器支持 13](#_Toc1162)

[布局 14](#_Toc6210)

[JAVASCRIPT 15](#_Toc19622)

[概述 16](#_Toc11010)

[调试工具 16](#_Toc32451)

[变量 16](#_Toc21174)

[字符串对象 17](#_Toc22056)

[日期对象 17](#_Toc18204)

[数组对象 17](#_Toc22226)

[数学对象Math 18](#_Toc17070)

[自定义函数 18](#_Toc9746)

[DOM 18](#_Toc24364)

[事件 20](#_Toc16762)

[BOM 20](#_Toc2927)

[Form表单操作 21](#_Toc19937)

[JQUERY 21](#_Toc3276)

[概述 21](#_Toc18362)

[安装 21](#_Toc20623)

[语法 22](#_Toc26404)

[选择器 22](#_Toc13912)

[DOM 22](#_Toc23887)

[事件 24](#_Toc612)

[特效 24](#_Toc32297)

[工具函数 25](#_Toc24320)

[AJAX 26](#_Toc16506)

[Json 29](#_Toc20156)

[Form表单操作 29](#_Toc4053)

[插件 29](#_Toc15643)

[MYSQL 30](#_Toc1795)

[概念 31](#_Toc2510)

[配置文件设置 31](#_Toc7892)

[windows 31](#_Toc2750)

[书写规范 31](#_Toc31870)

[数据类型 31](#_Toc9073)

[命令行操作 33](#_Toc9203)

[Mysql帮助 33](#_Toc17550)

[登录/退出 33](#_Toc13773)

[命令行窗口中文编码 33](#_Toc31884)

[数据库操作 33](#_Toc24009)

[表操作 34](#_Toc19154)

[表字段操作 34](#_Toc32230)

[表数据操作 35](#_Toc12494)

[联合查询 36](#_Toc17309)

[嵌套查询 36](#_Toc6349)

[数据库表的约束 37](#_Toc32727)

[视图view 38](#_Toc10399)

[事务transaction 38](#_Toc8068)

[索引index 38](#_Toc1610)

[存储过程procedure 39](#_Toc20999)

[触发器trigger 42](#_Toc1878)

[用户管理 43](#_Toc5066)

[权限管理 43](#_Toc18296)

[数据库备份和恢复 45](#_Toc18110)

[其他操作 45](#_Toc18860)

[Mysql可视化工具 46](#_Toc3987)

[Client/server模式 46](#_Toc7928)

[Browser/server模式 46](#_Toc9703)

[PHP 46](#_Toc10352)

[PHP简介 46](#_Toc30402)

[安装 46](#_Toc30261)

[Web服务器安装 47](#_Toc13333)

[PHP安装 47](#_Toc4989)

[PHP语法结构 48](#_Toc282)

[PHP数据类型 48](#_Toc20468)

[主要数据类型（8个） 48](#_Toc10949)

[伪类型（4个） 51](#_Toc22762)

[数据类型转换 51](#_Toc5283)

[变量作用域 53](#_Toc18732)

[局部变量 53](#_Toc1507)

[全局变量 53](#_Toc19584)

[超全局变量 54](#_Toc8016)

[预定义变量 54](#_Toc7191)

[常量 54](#_Toc19087)

[自定义常量 54](#_Toc881)

[系统常量 54](#_Toc13278)

[魔术常量 55](#_Toc22921)

[运算符 55](#_Toc11040)

[函数库 56](#_Toc30790)

[类型判别函数库 56](#_Toc28159)

[时间日期函数库 56](#_Toc19585)

[数学函数库 57](#_Toc11369)

[字符串函数库 58](#_Toc11034)

[数组函数库 61](#_Toc25016)

[函数函数库 64](#_Toc15357)

[mysql函数库 64](#_Toc12348)

[PDO函数库 65](#_Toc4614)

[文件上传 66](#_Toc6664)

[GD 68](#_Toc32531)

[概述 68](#_Toc2349)

[用途 68](#_Toc28124)

[安装 69](#_Toc27338)

[应用 69](#_Toc5950)

[HTTP 75](#_Toc31640)

[Session和cookie 75](#_Toc22296)

[Cookie 75](#_Toc6620)

[Session 76](#_Toc17654)

[REG 77](#_Toc20454)

[概述 77](#_Toc5907)

[语法 77](#_Toc9089)

[应用 79](#_Toc19075)

[贪婪模式 82](#_Toc23051)

[提高 82](#_Toc8570)

[FILE 83](#_Toc7684)

[概述 83](#_Toc20754)

[应用 83](#_Toc6739)

[OOP 95](#_Toc16895)

[概述 95](#_Toc159)

[语法 96](#_Toc9225)

[封装 97](#_Toc23391)

[继承 97](#_Toc21070)

[多态 97](#_Toc20890)

[生产模式 99](#_Toc13976)

[魔术方法 99](#_Toc29034)

[相关关键词 103](#_Toc3139)

[MVC 106](#_Toc32695)

[单一入口 106](#_Toc10128)

[MVC 106](#_Toc32466)

[PATHINFO模式 106](#_Toc19546)

[Apache URL Rewrite 106](#_Toc27342)

[ZendStudio安装Emmet 107](#_Toc14201)

[乱码问题 107](#_Toc25043)

# HTML5

## 文档结构

<!DOCTYPE html>

<html>

<head>

<title>我的第一个html页面</title>

<meta charset="utf-8"/>

</head>

<body>

正文

</body>

</html>

## 编码格式

utf-8 : 国际中文编码

gb2312：中文简体

gbk：简体,繁体 （国内中文编码）

## html实体

w3c规定在浏览器中特殊符号用html实体代替

> ==== &gt; 不间断空格 ==== &nbsp;

< ==== &lt; & ==== &amp;

" ==== &quot; 版权 ===== &copy;

' ==== &#39; 商标 ===== &reg;

## 文本标记

行元素：

加粗：<b>...</b> <strong>...</strong>

倾斜:<i>...</i> <em>...</em>

下划线：<u>...</u>

无实际意义的行元素：<span>...</span>

块元素：

标题：<h1~6 align=”center|left|right”>...</h1~6> align默认为left

段落：<p align=”center|left|right”>...</p> align默认为left

无实际意义的块元素：<div>...</div>

## 图像img

<img src=”路径” alt=”描述” width=””height=””border=””title=”鼠标悬停提示”/>

## 链接a

<a href=”#” target=”窗口打开方式”>...</a>

Target值：

\_self:默认值，目标文档在当前窗口打开

\_blank:目标文档在新窗口打开

\_parent:父级窗口,目标文档以父级窗口形式打开

\_top:顶级窗口,目标文档以顶级窗口形式打开

框架集name名称:目标文档在指定框架集名称窗口打开

## 列表list

无序列表

<ul type="disc|square|circle">

<li>...</li>

<li>...</li>

<li>...</li>

......

</ul>

有序列表

<ol type="1|a|A|i|I" start="起始序号">

<li>...</li>

<li>...</li>

<li>...</li>

...

</ol>

## 表格table

<table border="边框" width="像素值|百分比"height=""

cellspacing="外边距: 默认2像素" cellpadding="内边距: 默认值2像素">

<tr>

<th>...</th>

<th>...</th>

......

</tr>

<tr rowspan=”行合并”>

<td colspan=”列合并”>...</td>

<td>...</td>

......

</tr>

......

</table>

## 表单form

输出框output:

<form oninput ="res.value=parseInt(a.value) + parseInt(b.value)">

<label>输入值:</label>

<input type="number" min="0"

max="100" value="5" name="a"/>

+

<input type="range" min="0"

max="100" value="5" name="b"/>

=<output name="res">10</output>

</form>

<form action=”url”method=”get/post”name=”form”enctype=”multipart/form-data”

novalidate=”true”autocomplete=”off”>

单行文本框:<input type=”text”name=”” value=””maxlength=”文本最大长度”

autocomplete="on"/>

密码框:<input type=”password”name=””value=””maxlength=””pattern="[A-z]{3,5}"/>

单选框:<input type=”radio”name=”name”value=””checked=”checked”/>

<input type=”radio”name=”name”value=””/>

多选框:<input type="checkbox" name="name[]" value=""checked=”checked”/>

<input type="checkbox" name="name[]" value=""checked=”checked”/>

<input type="checkbox" name="name[]" value=""checked=”checked”/>

下拉列表框:<select name=”name[]”multiple=”multiple”>

<optgroup label=”分组名称”>

<option value=””selected=”selected”>...</option>

<option value=””>...</option>

<option value=””>...</option>

</optgroup>

</select>

类似下拉列表框:<input type=”text”name=””list=”list”/>

<datalist id=”list”>

<option label=”百度”value=”http://www.baidu.com”/>

<option label=”新浪”value=”http://www.sina.com.cn”/>

<option label=”搜狐”value=”http://www.sohu.com”/>

</datalist>

浏览框:单文件上传:<input type=”file”name=”name”value=””/>

多文件上传:<input type=”file”name=”name[]”value=””multiple=”multiple”/>

传输方式是post;多文件上传表单添加enctype=”multipart/form-data”

多行文本框:<textarea name=””value=””cols=”宽”rows=”高”>...</textarea>

可以用第三方html在线编辑器代替

邮箱:<input type=”email”name=””value=””required=”required”/>

网址:<input type=”url”name=””value=”” placeholder=”请输入http://”/>

数值:<input type=”number”name=””value=””max=””min=””step=””/>

<input type=”range”name=””value=””/>滑动条效果

日期:<input type=”date”name=””value=”” disabled=”disabled"/>

搜索:<input type=”search”name=””value=””autofocus=”autofocus”/>

拾色器:<input type=”color”name=””value=””/>

按钮: 提交:<input type=”submit”name=””value=””/>

重置:<input type=”reset”name=””value=””/>

普通:<input type=”button”name=””value=””/>

隐藏域:<input type=”hidden”name=””value=””/>

</form>

input中form属性的使用:<input type=”text”name=””value=””form=”form”/>

## 框架集iframe

格式:

<iframe src=””width=””height=””scrolling=”yes|no|auto”frameborder=”0”></iframe>

实例:

<body>

<iframe name=”iframe1”src="a.html"width="400"height="250"scrolling="no"

frameborder="0"></iframe>

<iframe name=”iframe2”src="b.html"width="400"height="250"scrolling=”auto”

frameborder=”1”></iframe>

<a href="c.html" target="iframe1">在iframe中的iframe1窗口打开c.html</a>

</body>

注:html5不支持frameset框架集

# CSS

## css的使用方式

内联样式:<标记名称 style="css样式代码"/>

<标记名称 style="css样式代码"></标记名称>

说明：

HTML公共属性(除html,head,body,link,meta外都可使用的属性):Style|title|class|id

内部样式:<style type="text/css">css样式代码</style>

外部样式:<link type="text/css" rel="stylesheet"href="" media="all"/>

说明:

media:不同媒介可以使用不同样式，移动端排版使用

media值:all所有设备|screen屏幕|print打印机| tv电视机

**内部样式和外部样式通过选择器将样式作用在相应的地方**

**外部样式文件的编码格式设置：@CHARSET "UTF-8";**

## css使用方式优先级

内联样式优先级最高

内部样式和外部样式优先级相同

优先级相同后面的样式覆盖前面的样式

## 选择器优先级

标记选择器 权值 0001

class选择器 权值 0010

id选择器 权值 0100

内联样式 权值 1000

说明: 权值越大优先级越高

权值相同后面的样式覆盖前面的样式

继承的样式优先级最低

!important 优先级最高（color:red !important;）

## 选择器

**设置超链接a的四个状态的选择器**

a:link 超链接未被访问过

a:visited 超链接被访问过

a:hover 鼠标悬停在超链接上

a:active 鼠标激活超链接(鼠标点住未放开）

**设置元素获得鼠标焦点后的状态的选择器**

:focus

实例:

input:focus{

outline:#00FF00 dotted thick;

background:red;

}

**设置元素状态的选择器**

:enabled 设置元素开启状态的选择器

:disabled 设置元素关闭状态的选择器

:checked 设置被选中状态的选择器

实例:

input:enabled{

background:yellow;

}

**设置子元素状态的选择器**

:first-child 如果父元素的第一个子元素是该子元素，则该子元素接受样式设置；

如果父元素的第一个子元素不是该子元素，则该子元素不接受样式设置

:last-child:

:nth-child(n)如果父元素的第n个子元素是该子元素的话，该子元素接受样式设置

n值是从1开始的

:only-child 父元素只有一个子元素且子元素是该子元素，该子元素接受样式设置

**在元素内容前后添加内容的选择器**

:before 在该元素原有内容的前面添加内容

:after 在该元素原有内容的后面添加内容

添加的内容可以是文本、图像、属性值

实例：

p:before{

content:”我爱中国”;

}

p:before{

content:url(“images/1.jpg”);

}

p:before{

content:attr(class); //这里的class是p标签的属性class，添加的内容是

//class的值

}

**锚点元素选择器**

:target 与超链接a一起用，找到超链接a指定的锚点元素

实例:

//创建锚点

<div id=”a”>内容</div>

//应用锚点

<a href=”#a”>aaaa</a>

//锚点样式

:target{

background:red;

}

## 文本样式

**文本:font**

实例:

font-style:normal|italic; //正常|倾斜

font-variant:normal|small-caps; //正常|小型大写

font-weight:normal|bold|100; //正常（400）|加粗（700）|值（100-900）

font-size:24px; //字体大小，单位:px|em|百分比

line-height:40px; //行高，当行高和盒子的高度一样时，

//文本在盒子里处于垂直居中的位置

font-size/line-height:24px/40px;

font-family:微软雅黑; //字体值

**font:italic small-caps bold 24px/40px 微软雅黑;**

**文本颜色:color:value;**

实例:

color:red;

color:#f00;

color:#ff0000;

color:rgb(255,0,0);

color:rgba(255,0,0,0.5);

color:rgba(100%,0%,0%,0.5);

**文本水平对齐:text-align:left|center|right;**

**文本首行缩进:text-index:2em;**

em是父元素默认文本大小的倍数

**文本缩进:text-indent:value;**

只能对块元素实现文本缩进，行元素不可以

**文本划线:text-decoration:underline|overline|line-through;**

**文本阴影:text-shadow:h-shadow v-shadow [blur color];**

实例:

text-shadow:2px -2px 5px #f00; //水平阴影,垂直阴影,模糊距离,阴影颜色

**文本折行:word-wrap:normal|break-word;**  //浏览器默认折行|长单词和url折行

**文本不折行:white-space:nowrap;**

当文本宽度大于盒子宽度时不换行，超出部分显示在盒子外面

**文本溢出隐藏:overflow:hidden;**

使显示在盒子外面的文本隐藏起来，不在浏览其中显示，

但是在F12里可以看到文本全部内容

**文本溢出替代:text-overflow:clip|ellipsis;** //剪切代替|省略号(...)代替

实例:

text-overflow:ellipsis;

**文本使用服务器上的字体:@font-face**

实例:

@font-face{

font-family:myfont; //定义服务器上字体的名称

src:url(font/RobotoSlab-Bold.ttf); //服务器上的字体

}

p{

font-family:myfont;

}

## 盒子模型

**div: 无实际意义的块元素,折行宽度是父级块元素的100%**

**span: 无实际意义的行元素,不折行,宽度是文本宽度**

**width/height:**

块元素和input、img可以设置宽度和高度

行元素不能设置宽度和高度

**padding/margin:**

padding:

内边距，无负值

margin:

外边距，有负值

**边框border:width style color;**

实例:

border:1px solid #f00;

border-top:1px solid #f00;

border-width:1px;

border-style:solid;

border-color:#f00;

style值:solid实线|dotted点线|dashed虚线|double双线

**边框圆角border-radius:value**

border-radius:value;

border-radius:value value; //左上角和右下角|左下角和右上角

border-radius:value value value value;

实例:

div{

width:40px;

height:60px;

border-radius:20px/30px; //实现椭圆

}

**边框图像border-image:**url(图像url) 图像边框宽度 repeat|round|stretch

实例:

div{

Width:100px;

Height:100px;

Border:10px solid #f00; //10px指div盒子的边框宽度

Border-image:url(images/1.jpg) 20 repeat; //20指图像的边框宽度

}

**盒子阴影box-shadow:**

box-shadow: h-shadow v-shadow [blur spread color inset|outset];

实例:

box-shadow:-5px 5px 8px 2px #f0f inset, //水平阴影位置,垂直阴影位置

-5px 5px 8px 2px #f0f outset; //模糊距离,阴影尺寸,阴影颜色

//内部阴影

**盒子背景background:**

实例:

background-color:#f00;

background-image:url(images/1.png);

background-repeat:repeat|no-repeat|repeat-x|repeat-y; //平铺

background-attachment:scroll|fixed;

background-position:left center;

background-position:20px 20px;

background-position:20px; //20px指水平位置,垂直位置默认为垂直居中

**bacdground:#f00 url(images/1.png) repeat scroll left cente**

**盒子背景图像大小background-size:**

实例:

background-size:100px 100px;

background-size:50% 50%;

background-size:cover|contain; //等比例填满,超出的部分裁切|

//等比例填满一边容器

## 列表list

实例:

list-style-type:disc; //默认值

square; //方块

circle; //空心圆

decimal; //阿拉伯数字

upper-alpha; //大写的字母

lower-alpha; //小写的字母

upper-roman; //大写的罗马数字

lower-roman; //小写的罗马数字

list-style-image:url(images/1.jpg);

list-style-position:outside|inside;

**list-style:square url(images/1.jpg) inside;**

list-style:none; //列的前面没有符号

## 转换transform

**移动:**

实例:

transform:translate(x,y); transform:translate(100px,100px);

transform:translateX(n); transform:translateX(20%); //元素宽度的比

transform:translateY(n); transform:translateY(40%); //元素高度的比

**缩放:**

实例:

transform:scale(x,y); transform:scale(0.4,0.5);

transform:scale(0.4);

transform:scaleX(n); transform:scaleX(0.3);

transform:scaleY(n); transform:scaleY(0.8);

**旋转:**

实例:

transform:rotate(49deg);

transform-origin:30% 40%; //调整中心轴位置

**倾斜:**

实例:

Transform:skew(x,y); transform:skew(30deg,20deg);

Transform:skewX(n); transform:skewX(30deg);

Transform:skewY(n); transform:skewY(39deg);

## 动画

**过渡动画transition:**

只有首尾两帧，通过事件触发

实例:

p{

transition:all 1s ease 2s; //发生动画的属性，动画过渡时长

//动画曲线，动画延迟时长

}

p:hover{

color:red;

font-size:24px;

}

动画曲线值:

linear 匀速

ease:慢进慢出（默认值）

ease-in:慢进快出

ease-out:快进慢出

ease-in-out:慢进慢出

动画延迟值:

默认值为0

**帧动画animation:**

多帧，页面加载后执行

定义:

@keyframes 动画名称 {}

实例:

@keyframes myanimation{

0%{

color:red;

}

50%{

color:blue;

}

100%{

color:gold;

}

}

p{

animation:myanimation 2s 3s ease infinite alternate; //动画名称，

//动画时长，

//动画延迟，

//动画曲线，

//动画执行次数,默认1次

Infinite无数次

//动画方向 alternate更替

}

## 透明度

**可以调节所有对象的透明度**

opacity: 0~1; //w3c 支持的写法,IE6,IE7不支持

filter:alpha(opacity=0~100); //IE支持的写法

实例:

opacity:0.5;

filter:alpha(opacity=50);

## 表格

1. border-collapse: seperate 分离|collapse 合并

2. border-spacing:value 单元格和单元格的距离

说明:

一般将下面的属性放到全局样式中。

border-collapse:collapse;

border-spacing:0;

## 老版本浏览器支持

-moz- 火狐浏览器

-webkit- 谷歌,safari

-o- 欧朋

-ms- IE

说明: 一般属性前面写前缀,目标对老版本浏览器的支持  
 实例:

p{

color:red;

-moz-color:red;

-webkit-color:red;

-o-color:red;

-ms-color:red;

}

## 布局

**文档流布局**

默认布局

文档流布局中行元素不能设置宽度和高度

**浮动布局float**

特点:

脱离文档流，不占位

浮动的块元素如果未设置宽度，则宽度是文本宽度

浮动的行元素可以设置宽度和高度

清除对象两边的浮动元素:

clear:right; //对象右边不出现浮动元素

clear:left; //对象左边不出现浮动元素

clear:both; //对象两边不出现浮动元素

万能清除:

.clearfix:after{

clear:both;

content:”.”;

height:0;

display:block;

visibility:hidden;

}

.clearfix{

zoom:1;

}

元素显示和隐藏:

display:

display:none; //元素不占位隐藏

display:block; //此时元素是块元素

display:inline; //此时元素是行元素

display:inline-block; //此时元素是块元素，但不折行

visibility:

visibility:visble; //不隐藏

visibility:hidden; //占位隐藏

**定位布局position**

固定定位fixed:

position:fixed; //默认原点在浏览器左上角

//坐标起始位置通过top,bottom,left,right调节

//脱离文档流，不占位

//不随滚动条滚动

实例:

p{

position:fixed;

left:50px;

bottom:50px;

Z-index:1;

}

绝对定位absolute:

position:asolute; //默认原点在浏览器（相对定位父元素）左上角

//坐标起始位置通过top,bottom,left,right调节

//脱离文档流，不占位

实例:

p{

position:absolute;

left:50px;

bottom:50px;

}

相对定位relative:

position:relative; //默认原点在自身元素左上角

//坐标起始位置通过top,bottom,left,right调节

//自身文档流，占位

实例:

div{

position:relative;

left:50px;

bottom:50px; }

# JAVASCRIPT

## 概述

Javascript是运行在客户端的解释型脚本语言

以.js作为文件后缀

严格区分大小写

命令行以；结尾，换行可以代替；

JavaScript应用形式:

<script type=”text/javascript” src=”js文件地址”></script>

<script type=”text/javascript”>

js代码

</script>

## 调试工具

IE浏览器 httpwatch 需要安装

FIREFOX firebug 需要安装

CHROME 自带 不需要安装

Firefox调试工具安装:

菜单->附加组件->扩展->搜索firebug->找到以后点击安装

## 变量

Js有八种变量类型:

string number boolean Object Array Function undefined null

复合类型:

Object 对象类型

var obj = new Object();

var obj = {"属性名称":"属性值"[,"属性名称":"属性值"...]};

Array 数组类型

var arr = new Array();

var arr = [1,2,"hello"];

Function 函数数据类型

var f = new Function();

当变量只定义未赋初值时，变量为undefined类型

var a; //a为undefined类型

typeof(变量) //判断变量数据类型

实例:

var str=”abc”;

document.write(typeof(str));

## 字符串对象

Var str=”hello World”;

str.length; //得到字符串长度

str.indexOf(“e”); //得到子字符在字符串中首次出现的位置

没有时返回-1

str.toUpperCase(); //将字符串转换为大写

str.toLowerCase(); //将字符串转换为小写

str.substr(0,2); //字符串截取 he

str.slice(0,2); //字符串截取 he

## 日期对象

var d=new Date();

d.getFullYear(); //得到四位数的年

d.getMonth(); //得到月份（0-11）

d.getDate(); //得到当前的日

d.getHours(); //得到当前的时

d.getMinutes(); //得到当前的分

d.getSeconds(); //得到当前的秒

d.getDay(); //得到当前的星期（0-6）

## 数组对象

var arr=new Array(

‘a’,’b’,’c’,’d’

);

var arr1=[‘e’,’f’];

arr.length; //得到数组的元素个数

arr.concat(arr1); //将两个数组合并得到一个新的数组

arr.pop(); //去掉数组中最后一个元素并返回这个元素

arr.shift(); //去掉数组中第一个元素并返回这个元素

arr.push(‘e’); //在数组最后面添加新元素并返回添加元素后数组的长度

arr.unshift(‘1’); //在数组最前面添加新元素并返回添加元素后数组的长度

arr.sort(); //给数组排序并返回排序后的数组

遍历数组:for(key in arr)

实例:

for(key in arr){

document.write("key="+key+" 值="+arr[key]+"<br/>");

}

## 数学对象Math

Var n;

Math.PI; //得到圆周率PI

Math.abs(n); //得到变量的绝对值

Math.round(n,2); //得到变量的四舍五入值

Math.ceil(n); //得到变量的向上取整值

Math.floor(n); //得到变量的向下取整值

Math.random(); //得到0-1的随机值

Math.min(a,b,c); //得到最小的变量

Math.max(a,b,c); //得到最大的变量

## 自定义函数

声明：

function shou(){

alert(‘hello’);

}

var fun=new Function(‘a’,’b’,’return a+b’);

Var fun=function(){

alert(‘hello’);

}

调用：

<script> show();</script>

<a href=”javascript:shou()”>shou()</a>

<p onclick=”shou()”>shou()</p>

var p=document.getElementsByTagName(‘p’);

p[0].onclick=function(){}

## DOM

**将html文档转换为js对象:**

实例:

<a href=”http://www.baidu.com”name=”a1”id=”a2”>百度</a>

var aobj=document.getElementsByTagName(“a”);

var aobj=document.getElementsByName(“a1”);

var aobj=document.getElementById(“a2”);

**Html文档文本设置:**

实例:

obj.innerHTML=”新浪”; //解析html标签

obj.innerText=”新浪”; //不解析html标签

**Html文档样式设置:**

实例:

Obj[0].style.backgroundColor=”red”;

**Html文档样式批量设置:**

实例:

<style>

.a{

font-size:14px;

color:red;

}

</style>

<a href=”http://www.baidu.com”name=”a1”id=”a2” class=”a”>百度</a>

<script>

var aobj=doucment.getElementsByTagName(“a);

aobj[0].className=”a”;

</script>

**节点node:**

概述:

Html文档结构中的文档、标签、注释、文本均是节点

节点有节点名称nodeName、节点类型nodeType、节点值nodeValue

节点间关系:

父级节点:obj.parentNode;

子级节点:obj.childNode; //所有子级节点

obj.firstChild; //第一个子级节点

obj.lastChild; //最后一个子级节点

同胞节点:obj.previousSibling; //上一同胞节点

obj.nextSibling; //下一同胞节点

创建新节点:

var newobj=doucment.createElement(“a”);

添加子节点:

divobj.appendChild(newobj);

删除子节点:

divobj.removeChild(newobj);

## 事件

**事件源:**

所有html标签都可视为事件源

**事件:**

鼠标事件:click、dblclick、mouseup、mousedown、mouseover、mouseout

键盘事件:keyup、keydown、keypress

文档事件:load、unload

表单事件:focus、blur、change、submit

**事件应用:**

A:

<p onclick=”test()”>百度</p>

B:

obj.onclick=function(){}

## BOM

**概述:**

浏览器对象模型(window可以省略不写）

**属性:**

window.location.href; //跳转到指定文档

window.innerWidth; //浏览器可视范围的宽度值

window.innerHeight; //浏览器可视范围的高度值

**方法:**

window.alert(“hello”);

window.confirm(“确定删除吗?”);

setInterval和clearInterval //每间隔一段时间执行一次代码和停止setInterval

实例:

var id=setInterval(function(){},1000); clearInterval(id);

var id=setInterval(“test()”,1000); clearInterval(id);

setTimeout和clearTimeout //一段时间后执行代码和停止setTimeout

实例:

var id=setTimeout(function(){},1000); clearTimeout(id);

var id=setTimeout(“test()”,1000); clearTimeout(id);

open和close

实例:

父窗口:产生sonobj对象

var sonobj=window.open(“http://www.baidu.com”,”\_blank”,”width=200,height=200”);

sonobj.close(); //只能关闭由open打开的子窗口

子窗口:产生opener对象

opener.alert(“hello father”);

## Form表单操作

**表单对象:**

实例:

<form name=”form”method=””action=””></form>

var formobj=document.form; //name值

**表单属性:**

实例:

formobj.method=”post”;

formobj.action=”index.php”;

formobj.enctype=”multipart/form-data”;

**表单方法:**

实例:

formobj.submit();

formobj.reset();

**表单元素值的获取:**

实例:

var usernameobj=document.getElementsByName(“username”);

# JQUERY

## 概述

jQuery是封装好的JavaScript库

## 安装

下载jQuery库文件:

网址:http://www.jquery.com

引入jQuery库文件:

<script type="text/javascript" src="jQuery库文件"></script>

## 语法

$(document).ready(function(){jquery代码});

$(function(){jquery代码});

## 选择器

实例:

<ul>

<li><a href=”http://www.baidu.com”name=”a1”class=”a2”id=”a3”>百度</a></li>

</ul>

<p>新浪</p>

$(“\*”) //通配符选择器

$(“.a2”) //class选择器

$(“#a3”) //id选择器

$(“a”) //标签选择器

$(“a,p”) //群组选择器

$(“li>a”) //子选择器

$(“ul a”) //后代选择器

$(“#p1+p”) //兄弟选择器，p1的下一个同胞标签

$(“#p1~p”) //范兄弟选择器,p1之后的所有同胞标签

<ul>

<li>新闻标题0</li> $(“li:first”)

<li>新闻标题1</li> $(“li:last”)

<li>新闻标题2</li> $(“li:eq(2)”)

<li class=”li3”id=”li3> $(“li:not(.li3|#li3)”) //不能为数字

新闻标题3</li>

<li>新闻标题4</li> $(“li:odd”) //奇数选择器

<li>新闻标题5</li> $(“li:even”) //偶数选择器

<li>新闻标题6</li>

</ul>

$(“div”).click(function(){$(this); //特殊的选择器

}); //this指div选择器

## DOM

**属性:**

$(“\*”).attr(“href”); //属性读取

$(“\*”).attr(“href”,”http://www.baidu.com”); //属性设置

$(“\*”).attr({

“href”:”http://www.baidu.com”,

“target”:”\_blank”

});

$(“\*”).removeAttr(“href”); //属性删除

**文本:**

$(“p”).html(); //文本读取,解析html标签

$(“p”).text(); //文本读取,不解析html标签

$(“p”).html(“<a href=’’>百度</a>”); //文本设置,解析html标签

$(“p”).text(“<a href=’’>百度</a>”); //文本设置,不解析html标签

**样式:**

$(“\*”).css(“background-color”); //样式读取

$(“\*”).css(“background-color”,”red”); //样式设置

$(“\*”).css({

“background-color”:”red”,

“font-size”:”24px”,

});

$(“\*).addClass(“test”); //批量添加样式

$(“\*”).removeClass(“test”); //批量删除样式

**节点:**

父级节点:

$(“\*”).parent(); //本级节点的直接父级节点

$(“\*”).parents(); //本级节点的所有父级节点

$(“\*”).parentsUntil(“div”); //到指定节点间的所有父级节点

//并且不包含该指定

子级节点:

$(“\*”).children(); //本级节点的直接子级节点

$(“\*”).find(“\*”); //本级节点的指定子级节点

同胞节点:

$(“\*”).siblings(); //当前节点的所有同胞节点

//但不包含该节点

$(“\*”).next(); //当前节点之后的第一个同胞节点

$(“\*”).nextAll(); //当前节点之后的所有同胞节点

$(“\*”).nextUntil(“div”); //当前节点到指定同胞节点的同胞

//节点且不包含该指定同胞节点

$(“\*”).prev(); //当前节点之前的第一个同胞节点

$(“\*”).prevAll(); //当前节点之前的所有同胞节点

$(“\*”).prevUntil(“div”); //当前节点到指定同胞节点的同胞

//节点且不包含该指定同胞节点

节点的添加和删除:

$(“\*”).append(“新内容”); //在选择器原内容的后面添加

$(“\*”).append(“<p>新内容</p>”); //新内容

$(“\*”).prepend(“新内容”); //在选择器原内容的前面添加

$(“\*”).prepend(“<p>新内容</p>”); //新内容

$(“\*”).after(“新内容”); //在选择器外面的后面添加

$(“\*”).after(“<p>新内容</p>”); //新内容

$(“\*”).before(“新内容”); //在选择器外面的前面添加

$(“\*”).before(“<p>新内容</p>”); //新内容

$(“div”).remove(); //删除指定节点

$(“div”).empty(); //清空指定节点

## 事件

**bind事件:**

$(“\*”).bind(“click”,function(){});

$(“\*”).bind({

“click”:test, //自定义函数名称不带括号

//带括号的话函数自动执行

“dblclick mouseover”:function(){},

});

**简写事件:**

$(“\*”).click(test); //自定义函数名称不带括号

//带括号的话函数自动执行

$(“\*”).click(function(){});

**hover事件:**

$(“\*”).hover(test1,test2); //鼠标滑入和滑出

$(“\*”).hover(

function(){},

function(){}

);

## 特效

**隐藏显示:**

$(“\*”).hide(“slow|fast”); //隐藏

$(“\*”).show(1000); //显示

**滑上滑下:**

$(“\*”).slideUp(“slow|fast”); //滑上

$(“\*”).slideDown(1000); //滑下

**淡入淡出:**

$(“\*”).fadeIn(“slow|fast”); //淡入

$(“\*”).fadeOut(1000); //淡出

$(“\*”).fadeTo(1000,0.4); //淡出到指定透明度,透明度值为0~1

**动画:**

$(“\*”).animate({ //样式类型是数值类型的样式

“width”:”200px”,

“height”:”200px”,

“left”:”200px”,

});

**特效停止操作:**

$(“div”).stop(); //停止指定选择器的特效

## 工具函数

**$.trim:**

去掉字符串两端的空格

实例:

var str=” hello “;

$.trim(str);

**$.type:**

返回变量的数据类型

实例:

var n=29;

$.type(n);

**$.merge:**

将两个数组合并,并返回这个数组

实例:

var arr1=[1,2,3];

var arr2=[4,5];

var arr3=$.merge(arr1,arr2);

**$.isArray:**

判断变量是否是数组类型，返回值是布尔值

实例:

var arr=new Array();

var boolean=$.isArray(arr);

**$.isNumeric:**

判断变量是否是数组类型，返回值是布尔值

实例:

var num=3;

var boolean=$.isNumeric(num);

**$.isFunction:**

判断变量是否是函数类型，返回值是布尔值

实例:

var fun=new Function();

var boolean=$.isFunction(fun);

**$.each:**

遍历数组

实例:

var arr=[“hello”,”world”];

$.each(arr,function(key,value){

document.write(“key:”+key+”&nbsp;”+”value:”+value+”</br>”);

});

**$.grep:**

过滤数组，并返回过滤后的数组

实例:

var arr=[“hello”,”world”,”how”,”are”,”you”,”11”];

var newarr=$.grep(arr,function(value,key){

If($.isNumeric(value)){

return value;

}

});

**$.map:**

过滤和计算数组，并返回过滤和计算后的数组

实例:

var arr=[“hello”,”world”,”how”,”are”,”you”,”11”];

var newarr=$.grep(arr,function(value,key){

If($.isNumeric(value)){

return value\*2;

}

});

## AJAX

**概述:**

AJAX（Asynchronous JavaScript And XML）是一个综合js、xml、json、XMLHTTPRequest的对象

AJAX 优点：

1、提升用户体验

2、减轻服务器负担

3、不需要插件

AJAX 缺点：

1、破坏浏览器的前进和后退按钮

2、对搜索引擎支持度不高。

AJAX数据异步提交和更新

**AJAX方法:**

**$(“\*”).load:**

将目标文档加载到指定的选择器中

实例:

$(“div”).load(“index.html”); //将目标文档加载到指定选择器中

$(“div”).load(“index.html #sp|.sp|span”); //将将目标文档指定的选择器

//内容加载到指定的选择器中

$("div").load( //以GET形式传递参数

"ajax.php",

"username=tom&password=123"

);

$("div").load( //以POST形式传递参数

"ajax.php",

{"username":"tom","password":"123"}

);

$("div").load(

"ajax.php",

{"username":"tom","password":"123"},

function(response,status,xhr){

$("p:eq(0)").html("response:"+response);

$("p:eq(1)").html("status:"+status);

$("p:eq(2)").html("xhr.responseText:"+xhr.responseText);

$("p:eq(3)").html("xhr.statusText:"+xhr.statusText);

}

);

**$.get:**

以get方式发送异步数据请求

实例:

$.get(

"get.php",

{"username":"tom","password":"123"},

function(response,status,xhr){

$("p:eq(0)").html("response:"+response);

$("p:eq(1)").html("status:"+status);

$("p:eq(2)").html("xhr.responseText:"+xhr.responseText);

$("p:eq(3)").html("xhr.statusText:"+xhr.statusText);

}

);

**$.post:**

以post方式发送异步数据请求

实例:

$.post(

"post.php",

{"username":"tom","password":"123"},

function(response,status,xhr){

$("p:eq(0)").html("response:"+response);

$("p:eq(1)").html("status:"+status);

$("p:eq(2)").html("xhr.responseText:"+xhr.responseText);

$("p:eq(3)").html("xhr.statusText:"+xhr.statusText);

}

);

**$.ajax:**

异步数据请求

实例:

$.ajax({

"url":"regist.php",

"type":"post|get",

/\*"data":{"username":$("#username").val(),

"password":$("#password").val(),

"email":$("#email").val(),

"birthday":$("#birthday").val()},\*/

"data":$("form").serialize(),

"success":function(response,status,xhr){

$("p:eq(0)").html("response:"+response);

$("p:eq(1)").html("status:"+status);

$("p:eq(2)").html("xhr.responseText:"+xhr.responseText);

$("p:eq(3)").html("xhr.statusText:"+xhr.statusText);

}

});

$(document).ajaxStart(function(){

//发送ajax请求以后，接到ajax响应之前

}).ajaxStop(function(){

//在接收到响应前，调用ajaxStop

});

## Json

**函数:**

json\_encode($arr):

将PHP数组转换为对象

实例:

$array = array("author"=>"tom","publisher"=>"机械工业出版社");

echo json\_encode($array);

json\_decode($obj,true):

将对象转换为php数组

实例:

$obj={“author”:”tom”,”publisher”:”机械工业出版社”};

var\_dump(json\_decode($obj,true));

## Form表单操作

**操作:**

实例:

$(“form”).serialize(); //将form表单中的name值与value值组成键值对形式。

$(“#username”).val(); //获取form表单中元素的值

## 插件

**插件应用:**

- 引入jQuery库文件

- 引入jQuery 插件

- 引入jQuery插件的周边产品。例如：汉化、css样式。

实例:

<!-- 引入jQuery库文件 -->

<script type="text/javascript" src="jquery-1.8.0.js"></script>

<!-- 引入jQuery 表单验证插件 -->

<script src="./validate/jquery.validate.js"></script>

<!-- 引入汉化插件 -->

<script src="./validate/jquery.validate.messages\_zh.js"></script>

**自定义插件:**

插件文件名称格式:

jquery.插件名称.js

插件语法定义:

$.extend([$.fn],{

"插件名称":function(){

插件实现的功能

},

"插件名称":function(){

插件实现的功能

},

}

);

**$.fn可选参数,$.fn存在,则表示插件必须绑定在选择器上,否则插件可以不用选择器**

实例:

(function($){

//定义插件

$.extend({

//修改nav的样式

"nav":function(){

//插件实现的功能

$(".nav").css({

"margin":0,

"padding":0,

"display":"none"

});

//插件的动态效果

$(".nav").parent().hover( function(){ //鼠标划入li ul滑下

$(this).find(".nav").slideDown();

},

function(){

//鼠标划出li 子节点ul滑上

$(this).find(".nav").slideUp();

}

);

},

});

}(jQuery));

**常见插件:**

//对话框(dialog)、日期（datepicker）、自动补全（autocomplete）

jquery-ui-1.10.3.custom.js (UI插件)

jquery-ui-1.10.3.custom.css （UI插件的样式文件）

//表单验证(validate)

jquery.validate.js(表单验证插件)

jquery.validate.messages\_zh.js（表单验证汉化插件）

//ajax form表单提交（ajaxSubmit）

jquery.form.js(ajaxform表单提交插件)

# MYSQL

## 概念

按特定形态存储数据的仓库

## 配置文件设置

### windows

my.ini文件配置:

端口:

port:3306

mysql安装位置:

basedir=””

mysql数据库位置:

datadir=””

字符集编码:

character\_set\_server=utf8

mysql引擎:

default\_storge\_engine=INNODB

引擎:

InnoDB:支持事务，外键

MyISAM:不支持事务，外键

## 书写规范

##SQL系统命令大写,名称小写

##SQL语句支持折行,但不能在函数或引号中折行

##SQL语句结束分隔符用 分号或\g

##名称不能用关键字和保留字，如果使用必须用反引号括起来.例如:`userName`

## 数据类型

**##整型**

tinyint 1字节 范围 0~255

smallint 2字节 范围0~65535

mediumint 3字节 范围0~1677725

int 4字节 范围0~2^32-1

bigint 8字节 范围0~2^64-1

##整型属性:

unsigned:无符号位

zerofill: 填补0,有这个属性的字段自动添加unsigned

实例:

CREATE TABLE test(

num1 smallint unsigned,

num2 smallint(3) zerofill

);

**##浮点型**

float([M,D]) 单精度浮点 4字节 小数点7位 unsigned|zerofill

double([M,D]) 双精度浮点 8字节 小数点15位 unsigned|zerofill

decimal([M,D]) 定长浮点 unsigned|zerofill

##decimal:

M 默认总长度10 ,D 默认0

M浮点数的总长度,D小数位数

实例:

float(7,2) unsigned

最大值为99999.99

##float,double 存近似值,decimal 存精确值

**##字符型**

char 定长字符 至多255字节

varchar 变长字符 至多65535字节

tinytext 至多255字节

text 至多65535字节

mediumtext 至多16777215字节

longtext 至多42亿多字节

**char 效率高不省空间，varchar 效率低省空间**

**##布尔型**

boolean boolean==tinyint(1)

**##枚举型**

enum 至多65535字节

实例:

create table test(sex enum('0','1','2'));

**##时间日期型**

datetime: 8个字节 YYYY-MM-DD HH:MM:SS

date： 4个字节 YYYY-MM-DD

time： 3个字节 HH:MM:SS

year： 1个字节 YYYY

timestamp 时间戳 : 4个字节 current\_timestamp

实例:

create table test(

aa date,

bb time,

cc year,

dd datetime,

ee timestamp

);

insert test(aa,bb,cc,dd,ee)

value

('2016-12-23','17:52:30','2016',now(),current\_timestamp);

## 命令行操作

### Mysql帮助

help;

?;

\h;

### 登录/退出

##登录mysql

mysql -uroot -proot -P3306 -hlocalhost -Dcms

##退出mysql

quit;

exit;

\q;

### 命令行窗口中文编码

set names gbk;

### 数据库操作

##查看数据库

show databases;

##创建数据库

create database test

character set utf8;

##删除数据库

drop database test;

##查看创建数据库的命令

show create database test;

##修改数据库字符集编码

alter database test

character set gbk;

##打开数据库

use test;

##查看当前打开的数据库

select database();

### 表操作

##查看表

show tables;

##创建表

create table test(

id int unsigned primary key auto\_increment,

username varchar(30) not null unique,

password char(32) not null,

sex tinyint unsigned default 0,

address varchar(100)

);

##查看创建表的命令

show create table test;

##查看表结构

desc test;

##删除表

drop table test;

##修改表编码格式

alter table test

default charset=utf8;

##修改表引擎类型

alter table test

engine=InnoDB;

### 表字段操作

##添加字段

alter table test

add tel char(11) not null first,

add phone char(11) not null after sex;

##删除字段

alter table test

drop tel,

drop phone;

##修改字段

alter table test

modify sex char(5) not null aftr address;

##修改字段名称

alter table test

change address addr char(5) not null first;

### 表数据操作

**##添加数据insert**

insert test set name=”tom”,password=”tom123”; //一次添加一条记录

insert into test(username,password,sex,address)

value

(“tom1”,”tom123”,0,”北京”),

(“tom2”,”tom123”,0,”北京”);

insert test value(1,”tom”,”tom123”,0,”北京”); //必须填写所有字段

**##删除数据delete**

delete from test where id=1;

truncate test; //auto\_increment重置

**##修改数据update**

update test set username=”jack” where id=1;

**##查询数据select**

select username,password from test;

select t.username as u,t.password p from test.test as t;

**##查询约束**

##where约束

select \* from test where id<>5;

select \* from test where id!=5;

select \* from test where id<=5;

select \* from test where id (not) between 5 and 10;

select \* from test where id (not) in(5,10,13);

select \* from test where id=4 and name=”tom” and address=”北京”;

select \* from test where id=5 or name=”tom” or address=”北京”;

## “\_”代表任意一个字符 “%”代表任意多个字符

select \* from test where name (not) like “\_o%”;

##group by约束

##按照age分组,包含null值

select count(\*) from test group by age;

##按照age分组,不包含null值

select count(age) from test group by age;

select count(age) as count,avg(age),max(age),min(age),sum(age)

from test group by address;

##having约束,与group by 一起用

select count(age) as count from test group by age

having count>10;

##order by 约束

select \* from test order by id asc|desc ##升序|降序

##limit约束

select \* from test limit 0,5;

##约束顺序

select count(age) from test where id>5

group by address

having count>10

order by id desclimit 20,20;

### 联合查询

##连接类型

内连接:inner join

外连接:

左外连接left join

右外连接right join

##内连接

select n.id as id,n.name as name,a.address as address,t.type as type from news as n

inner join address as a

on n.address=a.id

inner join type as t

on n.type=t.id;

##左外连接//from后连接的是左表，join后连接的是右表

select n.id as id,n.name as name,a.address as address from news as n

left join address as a

on n.address=a.id

##右外连接

select n.id as id,n.name as name,t.type as type from news as n

right join type as t

on n.type=t.id;

### 嵌套查询

**##where后**

select \* from user where address in(select id from prov where pname=”北京”);

##大于最小值

select \* from user where id>some(select id from user where id in(10,20,30));

##小于最大值

select \* from user where id<some(select id from user where id in(10,20,30));

##等同于in

select \* from user where id=some(select id from user where id in(10,20,30));

##大于最大值

select \* from user where id>all(select id from user where id in(10,20,30));

##小于最小值

select \* from user where id<all(select id from user where id in(10,20,30));

**##from后**

select \* from (select name,sex from user) as newuser;

### 数据库表的约束

##列约束

create table test(

id int key auto\_increment,

username varchar(30) not null unique,

password char(32) not null,

sex tinyint unsigned default 0,

address varchar(100)

);

##表约束

create table test( //常规写法

id int auto\_increment,

username varchar(30) not null,

password char(32) not null,

sex tinyint unsigned default 0,

address varchar(100),

key(id),

unique(username)

);

create table test( //简易写法

id1 int,

id2 int,

username varchar(30) not null,

password char(32) not null,

key(id1,id2),

unique(username,password)

);

create table cms\_news( //外键约束

id int unsigned PRIMARY KEY AUTO\_INCREMENT, //这时表引擎必须是InnoDB

title varchar(200) NOT NULL UNIQUE,

content text NOT NULL,

aid int unsigned NOT NULL,

tid int unsigned NOT NULL,

addtime int unsigned NOT NULL,

FOREIGN KEY(aid) REFERENCES cms\_admin(id),

FOREIGN KEY(tid) REFERENCES cms\_type(id)

);

### 视图view

##创建视图

create view carview

as

select n.id id,n.title title,n.content content,t.tname type,a.aname admin,

n.addtime addtime from cms\_news as n

join cms\_type as t

on t.id=n.tid

join cms\_admin as a

on a.id=n.aid;

### 事务transaction

##事务特点

原子性、隔离性、持久性、一致性

##引擎支持

InnoDB

##引擎区别

InnoDB 支持事务，支持行级别锁

MyISAM不支持事务,支持表级别锁

##查看自动提交状态

show variables like '%autocommit%';

##关闭自动提交

set autocommit=0;

##开启事务

start transaction;

##执行sql语句

##视执行情况 提交/回滚

提交：commit

回滚：rollback

##开启自动提交

set autocommit=1;

### 索引index

##查看某个数据表的索引

show index from test;

##创建索引

##alter方式

alter table test add index i\_address (address); ##普通索引

alter table test add unique u\_title (title); ##唯一索引

alter table test add key k\_id (id); ##主键索引

##create方式

create index i\_address on test(address); ##普通索引

create unique index u\_title on test(title); ##唯一索引

##联合索引

alter table test add index i\_aid\_tid (aid,tid);

create index i\_aid\_tid on test(aid,tid);

##删除索引

alter table test drop index i\_address; ##删除索引

alter table test drop key [k\_id]; ##删除主键索引

##重建索引

repair table test quick; ##有的版本不支持

##查看某个查询语句的性能

explain select \* from test;

##hash表

##hash:一个值，根据一定算法，生成一个16进制的数。

##hash表：根据hash,分段存储。

##作用：对表进行了拆分。

##格式:

create table 表名(

Id

title

content

clicknum

addtime

userid

)partirion by hash(字段名) partitions 10000;

### 存储过程procedure

##改变命令行的结束符

delimiter /

##查看存储过程

show procedure status;

show procedure status where db=”cms”;

##删除存储过程

drop procedure procname;

##查看创建存储过程和函数的代码

show create procedure procname;

show create function funcname;

##调用存储过程

call procname;

##语法格式：

create procedure 过程名称(in|out|inout 名称 类型,......)

begin

代码块;

end/

##实例:

##根据班级id，获取班级的人数。

create procedure getStudentNum(in cid int)

begin

select count(\*) as num from student where classid=cid;

end//

##根据班级id，获取班级的人数。要求人数值，使用参数。

create procedure getStudentNum1(in cid int,out num int)

begin

select count(\*) into num from student where classid=cid;

end//

##根据班级id，获取班级的人数。要求有一个参数。

create procedure getStudentNum2(inout num int)

begin

select count(\*) into num from student where classid=num;

end//

##在存储过程体外变量赋值

set @num=1;

##在存储过程体内变量赋值

set num=1;

##条件语句

##实例:

create procedure showSex(in s tinyint)

begin

if s=1 then

select '男' as sex;

end if;

end//

create procedure showSex1(in s tinyint)

begin

if s=1 then

select '男' as sex;

elseif s=2 then

select '女' as sex;

else

select '保密' as sex;

end if;

end//

##循环语句

##实例:

create procedure shownum()

begin

declare i int(2) default 1;

while i<=10 do

select i;

set i=i+1;

end while;

end//

##游标-----类似于结果集资源

cursor

##游标声明

declare 变量名称 cursor for sql语句;

##游标操作

-- 打开

open 游标的名称;

-- 获取记录

fetch 游标 into 变量1,变量2,......；

-- 关闭

close 游标;

##实例:

##编写一个存储过程，读表class中每一条记录，添加到class1中。查询class表中的所有记录，赋给一个变量。循环获取变量中每一条记录，写一次class1.

create procedure copclasstoclass1\_2()

begin

-- 班级表class的总条数

declare num int default 0;

-- 循环变量i

declare i int default 0;

-- 班级id

declare cid int;

-- 班级名称

declare cname char(20);

-- 产生结果集（班级信息）

declare classCur cursor for select id,name from class;

-- 给num赋值

select count(\*) into num from class;

-- 循环操作游标

open classCur;

while i<num do

fetch classCur into cid,cname;

insert into class1(id,name) value(cid,cname);

set i=i+1;

end while;

close classCur;

end//

##读一条，写一条。获取记录的总条数，读一条，写一条

create procedure copyclasstoclass1\_1()

begin

declare num int default 0;

declare i int default 0;

declare cid int;

declare cname char(20);

select count(\*) into num from class;

while i<num do

select name into cname from class limit i,1;

insert into class1(name) value(cname);

set i=i+1;

end while;

end//

### 触发器trigger

##一个事件，使得一段代码执行了。

##事件：

insert update delete

##作用：

一个表或多个表的记录同步另一张表中去。

##语法格式：

create trigger 名称 时机before|after 事件 on 表名 for each row

begin

-- 同步数据需要执行的sql语句

insert into zong ........

end//

##实例:

有一个搜索功能，能够根据文章的标题及产品的标题进行模糊查询

select \* from news where title like '%f%'

union

select \* from product where title like '%f%';

##实例:

create table news(

id int key auto\_increment,

title varchar(100)

)//

create table product(

id int key auto\_increment,

title varchar(100)

)//

create table zong(

id int,

title varchar(100),

type tinyint(1)

)//

type-----------1 文章

type-----------2 产品

##当文章表写入一条记录，zong中也写入一条记录

create trigger insert\_news after insert on news for each row

begin

insert into zong(id,title,type) value(new.id,new.title,1);

end//

##当文章表修改一条记录，zong中也修改这一条记录

create trigger update\_news after update on news for each row

begin

update zong set title=new.title where id=old.id and type=1;

end//

##当文章表中删除一条记录，总中也删除这一条记录

create trigger delete\_news after delete on news for each row

begin

delete from zong where id=old.id and type=1;

end//

### 用户管理

##更新命令

flush privileges;

##查看所有用户

use mysql;

select host,user from user;

##添加新用户

create user lvbo@localhost identified by ‘123’;

create user lvbo@localhost;

##修改用户名

rename user lvbo@localhost to lb@localhost;

##修改用户密码

set password for lb@localhost=password(‘234’);

##删除用户

drop user lb@localhost;

### 权限管理

**##查看用户权限**

show grants for lvbo@localhost;

**##权限分配**

**##权限作用层次**

grant all on \*.\* to lvbo@’%’;

##分配所有数据库的所有权限给用户

grant all on \*.\* to lvbo@localhost;

##分配某个数据库的所有权限给用户

grant all on cms.\* to lvbo@localhost;

##分配某个数据库的某张表的所有权限给用户

grant all on cms.cms\_news to lvbo@localhost;

##分配某个数据库的某张表的某些字段的所有权限给用户

grant select(id,title,content) on cms.cms\_news to lvbo@localhost;

##分配某个数据库的procedure,function的execute权限给用户

grant execute on procedure cms.pro\_test to lvbo@localhost;

grant execute on function cms.fun\_test to lvbo@localhost;

**##权限分配**

##分配某个数据库的grant,revoke权限给用户

grant grant,revoke on cms.\* to lvbo@localhost;

grant grant option on cms.\* to lvbo@localhost;

##分配某个数据库的insert,delete,update,select权限给用户

grant insert,delete,update,select on cms.\* to lvbo@localhost;

##分配某个数据库的create,alter,drop权限给用户

grant create,alter,drop on cms.\* to lvbo@localhost;

##分配某个数据库的create view,show view权限给用户(view)

grant create view,show view on cms.\* to lvbo@localhost;

##分配某个数据库的index权限给用户(index)

grant index on cms.\* to lvbo@localhost;

##分配某个数据库的create routine,alter routine,execute权限给用户(procedure)

grant create routine,alter routine,execute on cms.\* to lvbo@localhost;

##分配某个数据库的create user权限给用户

grant create user on cms.\* to lvbo@localhost;

##分配某个数据库的references权限给用户(外键)

grant references on cms.\* to lvbo@localhost;

##分配某个数据库的temporary tables权限给用户(临时表)

grant temporary tables on cms.\* to lvbo@localhost;

**##权限回收**

##回收用户所有数据库的所有权限

revoke all on \*.\* from lvbo@localhost;

**##权值表**

**权限 说明**

**all**

**alter**

**alter routine 使用alter procedure 和drop procedure**

**create**

**create routine 使用create procedure**

**create temporary tables 使用create temporary table**

**create user**

**create view**

**delete**

**drop**

**execute 使用call和存储过程**

**file 使用select into outfile 和load data infile**

**grant option 可以使用grant和revoke**

**index 可以使用create index 和drop index**

**insert**

**lock tables 锁表**

**process 使用show full processlist**

**Reload 使用flush**

**replication client 服务器位置访问**

**replocation slave 由复制从属使用**

**select**

**show databases**

**show view**

**shutdown 使用mysqladmin shutdown 来关**

**Mysql**

**super**

**update**

**usage 无访问权限**

### 数据库备份和恢复

##备份（登录时）

mysqldump -uroot -proot test > d:/test.sql;

##恢复（登录后）

use test

source d:/test.sql;

### 其他操作

##表达式

select 1+1;

##查看当前mysql用户名

select user();

##查看mysql版本号

select version();

##查看当前系统时间

select now();

##查看warning

show warinings;

##查看character变量类字符编码

show variables like ‘character%’;

## Mysql可视化工具

### Client/server模式

##Navicat

##Mysql\_Front

### Browser/server模式

##phpMyAdmin (php语言开始的)

说明:

php.ini 配置文件

## extension\_dir='D:\php5.4\ext'

## 加载函数库

extension=php\_mysql.dll

extension=php\_mysqli.dll

extension=php\_mbstring.dll

# PHP

## PHP简介

php是进行web程序开发的交互语言，是运行在服务器端的脚本语言

PHP的好处

(1)PHP代码开源

(2)PHP支持跨平台(windows linux)

(3)PHP支持多种web服务器(apache，IIS,Tomcat,Nginx)

(4)PHP支持多种数据库（MySQl,SQLserver,Oracle,DB2等）

## 安装

### Web服务器安装

(1) apache,IIS,Tomact,Nginx等

(2)安装 apache

(3)端口:计算机跟外界沟通的媒介

web服务器: 默认80

FTP服务器:默认 21

MySQL服务器: 默认 3306

1. web服务器访问

(1)远程访问

http://域名(一个域名可以绑定多个IP地址)

http://IP

(2)本地访问

http://localhost

http:127.0.0.1

(5)默认主目录

windows:htdocs文件夹

(6)默认首页(自动执行的页面)

index.html index.htm index.php

(7)apache 配置文件

windows: conf/httpd.conf

a. Listen 80 端口

b.DocumentRoot "D:/apache/htdocs" 默认主目录位置

c.<Directory "D:/apache/htdocs"> 浏览目录

d.DirectoryIndex index.html 设置默认自动执行的首页

(8)集成软件(phpstudy)

wamp = windows +apache +MySQL +php

Lamp = linux +apache +MySQL +php

(9)远程服务器的申请

1. 买空间和 域名 (PHP环境 MySQL 50M 150/年)

2. 虚拟主机服务器,服务器 (200元/月)

提供商:万网(阿里云),新网

### PHP安装

(1)安装apache:只解析静态页面

(2)PHP引擎安装

a.php5.4 解压缩将文件移动到安装目标位置

b.将文件 php.ini-development改为 php.ini作为PHP配置文件

c.将 apache 和 php5.4 绑定

配置Apache服务器的配置文件httpd.conf这个文件

#加载PHP5这个模块，加载路径php5apache2\_2.dll所在位置及全称

LoadModule php5\_module "d:\php5.4\php5apache2\_2.dll"

#加载PHP配置文件所在位置

PHPIniDir "d:\php5.4"

#告诉Apache什么样类型的文件交给PHP引擎处理

AddType application/x-httpd-php .php

说明: 重启apache 才能生效

1. Php运行原理:

客户端浏览器请求 .php页面，apache服务器只能解析静态页面，所以apache 服务器要请PHP引擎帮忙，PHP引擎执行PHP页面将静态结果交给apache服务器, 最后apache服务器响应给客户端浏览器，页面显示出来了

## PHP语法结构

(1)XML风格 (标准风格) xml自定义标记

<?php

echo “hello world”;

?>

(2)短风格

<?

Echo “hello world”;

?>

说明:将配置 php.ini 文件中的short\_open\_tag = Off改为 On并重启apache服务器

(3)asp风格

<%

Echo “hello world”;

%>

说明:将配置 php.ini 文件中的asp\_tags = Off改为 On并重启apache服务器

(4)脚本风格

<script language="php">

Echo “hello world”;

</script>

## PHP数据类型

### 主要数据类型（8个）

1.8个主要数据类型

(1)标量类型(scaler,存储单一值)

整型(Integer Int)

浮点型(Float Double)

布尔型(Boolean Bool)

字符类型(String)

(2)复合类型(存储多个值)

数组(Array)

对象(Object)

(3)特殊类型

资源(Resource)

NULL

2.字符类型(String):必须用定界符括起来

(1)定界符

a.单引号(PHP推荐)

b.双引号

c.HereDoc(自定义定界符)

d.NowDoc(自定义定界符)

(2)单引号和双引号区别

a.双引号能解析变量 而单引号不能解析变量

b.双引号能解析所有的转义字符，而单引号只能解析\'和\\两个

(3)转义字符:在源代码中的特殊符号

\' 单引号

\" 双引号

\r 回车

\n 换行

\t 水平制表符

\v 垂直制表符

\\ 反斜杠

(4) 单引号和双引号在HTML实体和转义字符中使用

a. w3c 规定在浏览器中看到的特殊符号必须用HTML实体,PHP代码中也必须用 HTML实体

b.在源代码中的特殊符号用转义字符,PHP中单引号和双引号冲突时必须用转义 字符

(5)**{}的使用**

a.用{}将变量名称作为一个整体括起来

说明:当用双引号输出变量，PHP引擎能解析双引号中的变量，PHP引擎将尽 可能多的合法字符串作为变量名称如果告诉PHP引擎到此为止，那要用

{}将变量名称作为一个整体括起来

b.**用{}对变量中的某一个值实现增删改查.**

说明:(a)可以用[]代替{},编号从0开始

(b)当编号大于长度，用一个空格填补

实例:$var=”abcd”;

增:$var{4}=”e”;

删:$var{3}=””;

改:$var{3}=”a”;

查:echo $var{3};

(6)**HereDoc(自定义定界符)**

说明:

(a)大段代码可以用HereDoc实现

(b)结束定界符必须在行首

(c)HereDoc能解析所有的变量和转义字符

实例:

$var ='hello';

$link =**<<<UL**

<ul>

<li>one</li>

<li>two</li>

<li>three</li>

<li>$var</li>

<li>fi\\ve</li>

</ul>

UL;

echo $link;

(7)NowDoc(自定义定界符)

说明:

(a)大段代码可以用NowDoc实现

(b)结束定界符必须在行首

(c)NowDoc不能解析所有的变量和转义字符

(d)PHP5.2以上支持

实例:

$var ='hello';

$link =**<<<'UL'**

<ul>

<li>one</li>

<li>two</li>

<li>three</li>

<li>$var</li>

<li>fi\\ve</li>

</ul>

UL;

echo $link;

3.数组(Array)

4.对象(Object)

5.资源(Resource)

操作磁盘中或网络中的文件,不能声明资源,用函数实现

例如:文件函数,文件夹函,数据库函数,GD库函数等.

实例:

$handle = fopen('01\_string.php','r'); //返回资源类型

6.NULL类型

说明:

产生NULL类型的方法

a.声明变量并赋NULL值

实例:

$null=NULL;

b.声明变量但未赋值

实例:

$null;

c.unset函数删除变量中的值

实例:

$var=”abc”;

unset($var);

php配置文件php.ini中错误报告设置

a.display\_errors = On

b.error\_reporting =E\_ALL //显示所有错误,推荐

error\_reporting = E\_ALL&~E\_NOTICE //显示所有错误除了通知错误

### 伪类型（4个）

1.mixed :混合类型，参数可以接受多种不同的（但不一定是所有的）类型

2.number: 数值类型,包含整型和浮点型

例如: pow:求次幂

number pow ( number $base , number $exp )

3.void :空类型, 如果void 在返回值上代表没返回值，如果void 在参数上代表没参数

例如:void echo ( string $arg1 [, string $... ] )

1. callback:回调函数,一个函数的名称作为另一个函数的参数使用

### 数据类型转换

PHP是弱类型语言，会根据程序的运行环境自动转换

1.自动转换(隐式转换)

(1)其它类型转为整型或浮点型

a.合法的数字字符串如果没有 . e E并且没有超出整型最大值那转为整型，否 则转为浮点

实例:

echo 8 + '13'; //21

echo 8 + '3.76'; //11.76

echo 8 + '1e2'; // 8+100=108

b.不合法的数字字符串,如果以数值开头那转为该数值,如果以字母开头转为0.

实例:

echo 8 + '3ok'; //11

echo 8 + '2e2abc'; //208

echo 8 + 'e3ok'; // 8+0

echo 8 + '3.5.6.7.8.9.2'; //11.5

c.布尔值 true 转为 1,false 转为 0

d. null 转为 0

e. 空字符串转为 0

实例:

echo 12.5 + '' + null + true + '8true'; // 12.5 +0 +0 +1 +8 = 21.5

(2)其它类型转为字符类型

a.整型和浮点转为字符类型就是该数值本身

实例:

echo 17; // '17'

echo 19.34; // '19.34'

b.布尔类型 true转为 '1',false转为 空字符串

实例:

$var = true;

echo $var; // '1'

$var = false;

echo $var; // 空字符串

c.null 转为 空字符串

实例:

echo '#',true,null,false,'#';

d.数组转为 字符类型 'Array'

实例:

$arr = array('a','b','c','d');

echo $arr; // 'Array' 错误的

echo $arr[2]; // 输出数组中的默认一个值 c

e.资源转为 字符类型 'Resource id #数字'

实例:

$handle = fopen('01\_string.php','r');

echo $handle;

(3)其它类型转为布尔类型

下面情况转为 false:

a.布尔值 false

b.空字符串 false

c.null 转为false

d.整数0,浮点数的0.0,字符串'0' "0"转为false

e.空数组 array() 转为false

其它情况都为true.

2.强制转换(显式转换)

(1)临时转换 (变量本身没变化)

1.1

1. 整型 (integer)$var|(int)$var

实例:

$var=4.7;

(int)$var; //向下取整

b.浮点型 (float)$var |(double)$var|(real)$var

c.布尔型 (boolean)$var| (bool)$var

d.字符类型 (string)$var

e.数组 (array)$var

f.对象 (object)$var

g.null类型 (unset)$var

1.2 函数

整数 intval($var)

浮点型 floatval($var)|doubleval($var)

实例:

$var=3;

floatval($var);

布尔类型 boolval($var)--PHP5.5以上支持

字符类型 strval($var)

(2)永久转换 (变量本身发生变化)

a. settype:设置变量类型

格式: bool settype ( mixed &$var , string $type )

成功返回true 失败返回 false

实例:

$float = 14.78;

settype($float,'int');

b. gettype:获得变量类型

格式: string gettype ( mixed $var )

实例:

$float = 14.78;

echo gettype($float);

## 变量作用域

### 局部变量

### 全局变量

function isTest3(){

global $n;

$n = 12;//全局变量

}

function isTest4(){

$GLOBALS['n'] = 12;//全局变量

}

function isTest5(&$n){//形参:引用传递全局变量

$n +=9;

//return $n;

}

### 超全局变量

任意范围内都有效,并且不用在函数中声明全局变量

### 预定义变量

1. $\_POST：表单传输方式'post',以名/值对接受数据

2. $\_GET：表单传输方式'get',以名/值对接受数据

3. $GLOBALS：声明全局变量

4. $\_FILES:上传接受的数据信息

5. $\_COOKIE：会话机制 跨页面传输数据的

6. $\_SESSION：会话机制 跨页面传输数据的

7. $\_REQUEST：等同与 $\_POST,$\_GET,$\_COOKIE接受数据

8. $\_SERVER:服务器信息(ip,主机名，根相对路径等...)

9. $\_ENV：环境变量

说明:a.所有的预定义变量都是超全局变量

b.所有的预定义变量都是以数组形式存储

## 常量

### 自定义常量

(1)定义常量

a. define('常量名称','值');(不能在类中定义常量)

b. const 常量名称='值';(可以在类中定义常量)

(2)输出常量

a. echo 常量名称;

b. echo constant('常量名称');

(3) 常量名称规定

a. 常量名称尽量大写字母

b. 常量名称没有 $

c. 常量值都是标量类型(整型,浮点,字符型,布尔型)

d. 常量是超全局变量，但除了const定义的常量外

### 系统常量

a. PHP\_INT\_MAX 获得整数最大值

b. PHP\_OS 服务器操作系统

c. PHP\_VERSION :PHP版本

### 魔术常量

\_\_LINE\_\_:获得当前文件的行号

\_\_FILE\_\_:获得当前文件所在绝对路径

\_\_DIR\_\_: 获得当前文件所在目录的绝对路径

\_\_FUNCTION\_\_:获得当前的函数名称

\_\_CLASS\_\_:获得当前的类名称

\_\_METHOD\_\_:获得当前的方法名称

\_\_NAMESPACE\_\_:获得命名空间的名称

命名空间:解决函数或类名称冲突的问题

必须写在第一行

实例:

<?php

/\* php5.3以上

\* \_\_NAMESPACE\_\_:获得命名空间的名称

命名空间:解决函数或类名称冲突的问题 \*/

namespace a;

function var\_dump(){

echo 'hello<br/>';

}

var\_dump(); // 命名空间下默认访问自定函数

echo '<br/>';

\a\var\_dump();// 访问命名空间a下的自定义函数

echo '<br/>';

\var\_dump(true); // 系统函数

echo '<br/>';

// \_\_NAMESPACE\_\_:获得命名空间的名称

$fn = \_\_NAMESPACE\_\_.'\var\_dump'; //变量函数

$fn(); //调用函数

echo '<br/>';

// use 命名空间下的类

## 运算符

按功能分

(1)算术运算符

%求余数

(a)求余数,求的是整数余数,操作数向下取整

实例:

echo 8.6 % 3.5; // 8%3 =2

(b)求余数符号与被除数一致

实例:

echo -8.6%3.5; // -2

echo -8.6%-3.5; //-2

echo 8.6%-3.5; //2

(c)求浮点数余数 fmod函数

实例:

echo fmod(8.6,3.5);//1.6

(2)递增,递减运算符

(3)字符(连接)运算符

(4)赋值运算符

(5)比较运算符

(6)逻辑运算符

(7)错误抑制符: @ 忽略错误,没错误不忽略

说明: 错误抑制符用在函数或表达式前面

实例:

$str ='12ab';

@settype($str,'int');

(8)执行运算符：`` 反引号

说明:

a.PHP命令行模式

b.php.ini safe\_mode=Off关闭

实例:

echo `ipconfig`; // 查看IP地址

## 函数库

### 类型判别函数库

is\_int |is\_integer|is\_long检测变量是否是整数

格式:bool is\_int(string $var)

实例:

$int=12;

echo is\_int($int);

is\_float |is\_double|is\_real检测变量是否是浮点

is\_bool检测变量是否是布尔

is\_string检测变量是否是字符

is\_array 检测变量是否是数组

is\_object 检测变量是否是对象

is\_resource 检测变量是否是资源

is\_null 检测变量是否是null

is\_numeric 检测变量是否数值(整数,浮点,合法的数字字符串)

is\_scalar 检测变量是否是标量(整型,浮点型,布尔,字符型)

### 时间日期函数库

(1)设置时区

date\_default\_timezone\_set(string $timezone)

说明:

$timezone值 默认UTC欧洲时区

a. PRC

b.

Asia/Shanghai

Asia/Chongqing

c. php.ini 配置文件

date.timezone = PRC

重启apache服务器

(2)获得时区

date\_default\_timezone\_get()

(3)格式化时间日期函数

string date(string $format [,int $timestamp])

说明:

a. $format

Y: 4位的年(year)

m: 2位的月(month)

d: 2位的日(day)

H: 2位的小时(hours)

i: 2位的分钟(minutes)

s: 2位的秒(seconds)

b. $timestamp

默认值 time() 时间戳:从1970-1-1 00:00:00 到现在的秒数

1. time：返回当前的 Unix 时间戳

date('Y-m-d H:i:s',time() + 60\*60\*24\*3);

1. strtotime：将任何英文文本的日期时间描述解析为 Unix 时间戳

实例:

date('Y-m-d H:i:s',strtotime('+3years +7months +15days +35minutes'));

date('Y-m-d H:i:s',strtotime('+3days'));

date('Y-m-d H:i:s',strtotime('2017-1-37'));

1. mktime() 返回时间日期时间戳(秒)

参数 小时 分钟 秒 月 日 年

实例:

echo date('Y-m-d H:i:s',mktime(0,0,0,1,37,2017));

(7)microtime() 返回时间戳(秒)和微妙数

说明:

microtime(true) 返回浮点数的微妙数

### 数学函数库

1. floor:舍去法取整,向下取整

实例:

echo floor(5.9);//5

1. ceil:进一法取整,向上取整

实例:

echo floor(5.9);//5

3. round:四舍五入

实例:

echo round(4.78); //5

echo round(4.789,2);//4.79

1. max:最大值

实例:

echo max(6,1,8,18,22,5);

1. min:最小值

实例:

echo min(6,1,8,18,22,5);

1. abs:绝对值

实例:

echo abs(-28);

1. sqrt:开平方

实例:

echo sqrt(4); //2

8. pow:求次

实例:

echo pow(2,3); //8

echo pow(2.5,3);// 15.625

1. rand:获得随机整数

实例:

echo rand();

echo rand(1,10);

10 mt\_rand:获得更好的随机整数

实例:

echo mt\_rand();

echo mt\_rand(1,10);

### 字符串函数库

1. substr:字符串截取函数

格式:string substr(string $var,int $start [,int $length]);

说明:

a.$start:起始编号,编号从0开始

b.$length:截取长度,如果省略$length则从起始编号开始一直到字符串结束

c.如果$start,$length 是负值,位置倒数

d.如果$start 大于字符串长度返回 false

实例:

$var = 'abcdefg';

echo substr($var,3,2);//de

echo substr($var,3);// defg

echo substr($var,-5,2);//cd

echo substr($var,0,-3);//abcd

echo substr($var,-5,-3);//cd

echo substr($var,9,3);//false

2. strlen :获得字符的长度

实例:

$var ='abcdefg';

echo strlen($var);

3.

strtolower:将字符串转为小写

strtoupper:将字符串转为大写

ucfirst:将字符串首字母大写

ucwords:将字符串中每个单词首字母大写

实例:

$var ='helLO worLD';

echo strtolower($var);

echo strtoupper($var);

echo ucfirst($var);

echo ucwords($var);

4.

trim: 清除两边空格，也可以清除指定字符

ltrim:清除左边空格，也可以清除指定字符

rtrim:清除右边空格，也可以清除指定字符

实例:

$var = ' AB ';

echo '#',$var,'#';

echo '#',trim($var),'#';

$var ='.abc.';

echo trim($var,'.');

5.

strpos:查找字符串首次出现的编号位置,编号从0开始,区分大小写

格式:mixed strpos ( string $haystack , mixed $needle [, int $offset = 0 ] )

成功返回编号位置，失败返回false

实例:

$str = 'aBbBbBbBcde';

echo strpos($str,'b');//2 区分大小写

stripos:不区分大小写

实例:

echo stripos($str,'b');//1 不区分大小写

strrpos:查找字符串最后出现的编号位置,编号从0开始,区分大小写

实例:

echo strrpos($str,'b');//6 最后，区分大小写

strripos:查找字符串最后出现的编号位置,编号从0开始,不区分大小写

实例:

echo strripos($str,'b');//7 最后，不区分大小写

6.

strstr: 查找字符串的首次出现位置,区分大小写

string strstr ( string $haystack ,mixed $needle [, bool $before\_needle = false ] )

说明:

a.返回 haystack 字符串从 needle 第一次出现的位置开始到 haystack 结尾

的字符串。

b.$before\_needle若为 TRUE，strstr() 将返回 needle 在 haystack 中的位

置之前的部分。

实例:

$str = 'aBbBbBbBcde';

echo strstr($str,'b');//bBbBbBcde 区分大小写

echo strstr($str,'b',true);//aB 区分大小写

stristr:不区分大小写

实例:

echo stristr($str,'b');//BbBbBbBcde 不区分大小写

strchr:是 strstr 的别名

实例:

echo strchr($str,'b');//bBbBbBcde 区分大小写

strrchr:查找字符串的最后出现位置,区分大小写

实例:

echo strrchr($str,'b');//bBcde

7.

strrev:字符串反转

实例:

$var = 'abc';

echo strrev($var);

8.str\_replace:字符串替换

格式:

mixed str\_replace ( mixed $search ,mixed $replace , mixed $subject [, int &$count ] )

实例:

$str ='abcbdbebfbg';

echo str\_replace('b','<span style="color:red">\*</span>',$str);

$arr = array('b','c','d');// 将字母b c d 替换成 null

var\_dump(str\_replace($arr,null,$str,$count));

echo $count; // 获得替换次数

9.

htmlspecialchars:自动将特殊符号转为HTML实体

格式:

string htmlspecialchars ( string $string [, int $flags = ENT\_COMPAT]);

说明:

a. htmlspecialchars只能转换 > < & '和"特殊符号

b. $flag默认值 ENT\_COMPAT 默认不处理单引号

c. $flag:

ENT\_COMPAT 默认不处理单引号

ENT\_QUOTES 处理单引号和双引号

ENT\_NOQUOTES 不处理单引号和双引号

实例:

$str =<<<AA

D>A B<D

Tom & Jerry

he said" I'm fine"

AA;

echo htmlspecialchars($str);// 自动将特殊符号转为HTML实体

echo htmlspecialchars($str,ENT\_QUOTES);

10 加密函数

md5:计算字符串的 MD5 散列值,返回32位字符串

sha1:计算字符串的sha1 散列值,返回40位字符串

实例:

$pwd ='123456';

echo md5($pwd);

echo sha1($pwd);

echo md5(sha1(md5($pwd)));

11. strip\_tags:从字符串中去除 HTML 和 PHP 标记

格式:

string strip\_tags ( string $str [, string $allowable\_tags ] )

说明:

$allowable\_tags值,去除HTML 和 PHP 标记外,可以保留的标记

实例:

$str='<spanstyle="color:rgb(89,179,42)">q</span><span style="color:rgb(164,121,77)">v</span><span style="color:rgb(189,80,135)">k</span><span

style="color:rgb(45,238,243)">P</span>';

echo strlen($str);

echo strlen(strip\_tags($str));

$str ='<div><a href="#">链接</a></div>'; // 保留 a标记

echo strip\_tags($str,'<a>');

### 数组函数库

(1) count/sizeof:获得数组或对象的长度个数

实例:

$arr = array('a',20,'c','d');

$count = count($arr);

$count=sizeof($arr);

(2) join/implode:将一维数组转为字符串

实例:

$arr = array('a','b','c','d');

echo join('\*',$arr);

echo implode('\*',$arr);

(3) explode:将字符串转为一维数组

array explode ( string $delimiter , string $string [, int $limit ] )

说明:

a.如果 limit 参数是负数，则返回除了最后的 -limit 个元素外的所有元素。

实例:

$str = 'a|b|c|d';

$newarr = explode('|',$str,-2);

b.如果 limit 是 0，则会被当做 1。

实例:

$str = 'a|b|c|d';

$newarr = explode('|',$str);

$newarr = explode('|',$str,2);

(4) 数组指针函数

current:将数组指针指向当前的成员值,并返回该值

next:将数组指针指向下一个的成员值,并返回该值

prev:将数组指针指向上一个的成员值,并返回该值

reset:将数组指针指向第一个成员值,并返回该值

end:将数组指针指向最后一个成员值,并返回该值

实例:

$arr = array('a','b','c','d','e','f');

echo current($arr); //a

echo next($arr);//b

echo next($arr);//c

echo current($arr);//c

echo end($arr);//f

echo prev($arr);//e

echo reset($arr);//a

each: 返回数组中当前的键／值对并将数组指针向前移动一步

array each ( array &$array )

实例:

$arr = array('a','b','c','d','e','f');

echo next($arr);//b

echo next($arr);//c

echo current(each($arr));// c

echo current(each($arr));// d

list:把数组中的值赋给一些变量

实例:

$arr = array('one','two','three');

list($a,$b,$c) = $arr;

(5)

in\_array:判断变量是否是数组的值

bool in\_array(mixed $needle,array $arr [,bool $strict=false]);

说明:

a. 区别大小写

b. 如果$strict 默认值 false 松散比较 ==

如果$strict true 严格比较 ===

实例:

$var = 'a';

$arr = array('a','b','c','d');

var\_dump(in\_array($var,$arr));

var\_dump(in\_array($var,$arr,true));

array\_key\_exsits:判断变量是否是数组的键名称

实例:

$var ='one';

$arr = array(

'one'=>'apple',

'two'=>'orange',

'three'=>'pear'

);

var\_dump(array\_key\_exists($var, $arr));

(6)

array\_push:在当前数组的尾部添加一个或多个成员值,返回新长度

array\_unshift:在当前数组的头部添加一个或多个成员值,返回新长度

array\_pop:在当前数组的尾部删除一个成员值，并返回该值

array\_shift:在当前数组的头部删除一个成员值，并返回该值

实例:

$arr = array('a','b','c','d');

echo array\_push($arr,'e','f','g');

echo array\_unshift($arr,'h','i');

echo array\_pop($arr);

echo array\_shift($arr);

(7) array\_map:将回调函数作用到给定数组的单元上

格式:array array\_map ( callable $callback ,array $arr1 [, array $... ] )

实例:

$arr = array('hELLo','gooD','haPPy');

$newarr = array\_map('strtoupper',$arr);

### 函数函数库

func\_num\_args:获得函数的参数个数

func\_get\_args:获得函数的参数的值所组成的数组

func\_get\_arg:获得函数的参数的值所组成的数组的某一个值

实例:

function getAvg2(){

$count = func\_num\_args();//获得参数个数

$arrArgs = func\_get\_args();//参数值所组成的数组

//echo func\_get\_arg(0); //获得数组中的某一个值

$sum =0; // 参数和

foreach($arrArgs as $v){

$sum += $v;

}

return round($sum/$count,2);//参数和/参数个数

}

function\_exists:判断变量是否是函数//method\_exists($class,$method)

实例:

function arrayMap($fn,$arr){

foreach($arr as $v){

//$newarr[] =$fn($v); // 变量函数

$newarr[] = function\_exists($fn)?$fn($v):$v;

}

return $newarr;

}

$arr = array('3dfRe','2dfEd','4YUab');

$newarr = arrayMap('strtolower',$arr);

### mysql函数库

##连接数据库

$link=mysql\_connect("localhost:3306","root","root");

##选择数据库

mysql\_select\_db("cms");

##执行sql语句

$result = mysql\_query($query); ##没有返回结果集的sql语句返回是布尔值。

##有返回结果集的sql语句，返回是资源。

##返回增删改影响的行数

$rows= mysql\_affected\_rows(); ##成功返回 >=1整数,没有影响行数返回 0

##失败返回 -1

##返回上一条成功insert产生的自增id

$id=mysql\_insert\_id(); ##成功insert操作产生的自增id

##若id不是自增的则返回 0

##将结果集转为数组，一次一条

##索引数组

mysql\_fetch\_row($result);

##关联数组

mysql\_fetch\_assoc($result);

##索引加关联数组

mysql\_fetch\_array($result[,MYSQL\_BOTH|MYSQL\_ASSOC|MYSQL\_NUM]);

##获得有返回结果集的查询行数

$rows=mysql\_num\_rows($result);

##释放结果集

mysql\_free\_result($result);

##关闭数据库

mysql\_close($link);

### PDO函数库

##实例化PDO类，获取类对象

$pdo=new PDO(“mysql:host=localhost;dbname=cms”,”root”,”root”);

##执行没有返回结果集的sql语句

$result = $pdo->exec($query); ##返回被影响的行数,有可能受影响行数是:0

##获取最近一次成功insert操作产生的自增id(必须)

$id=$pdo->lastInsertId();

##执行有返回结果集的sql语句

$statm = $pdo->query($query);

foreach($statm as $value){var\_dump($value);}

##sql语句编译不执行

$statm = $pdo->prepare($query);

##编译后的sql语句绑定参数

##方式一

$query = "update cms\_admin set pwd='123' where id=? or id=?";

$statm = $pdo->prepare($query);

$id1 = 29;$id2 = 30;

$statm->bindParam(1,$id1); ##必须使用变量

$statm->bindParam(2,$id2);

##方式二

$query = "update cms\_admin set pwd='123' where id=:id1 or id=:id2";

$statm = $pdo->prepare($query);

$id1 = 29;$id2 = 30;

$statm->bindParam(:id1,$id1);

$statm->bindParam(:id2,$id2);

##执行编译好的sql语句

$result = $statm->execute(); ##没有返回结果集的sql语句,返回是布尔值

##返回没有返回结果集的sql语句被影响行数或者有返回结果集的sql语句的查询行数

$rows=$statm->rowCount();

##获取查询的结果,一次一行,获取不到返回false

$row=$statm->fetch(); ##关联和索引数组

$row=$statm->fetch(PDO::FETCH\_BOTH); ##关联和索引数组

$row=$statm->fetch(PDO::FETCH\_ASSOC); ##关联数组

$row=$statm->fetch(PDO::FETCH\_ROW); ##索引数组

##事务实例

$pdo=new PDO(“mysql:host=localhost;dbname=cms”,”root”,”root”);

//1、关闭自动提交

$pdo->setAttribute(PDO::ATTR\_AUTOCOMMIT,0);

//2、开启事务

$pdo->beginTransaction();

//3、执行

//jerry的钱减少50

$query = "update demo set money=money-50 where id=2";

$result\_1 = $pdo->exec($query);

//tom的钱加入50

$query = "update demo set money=money+50 where id=1";

$result\_2 = $pdo->exec($query);

//4、视两条sql语句执行的结果 绝对提交还是回滚

if($result\_1&&$result\_2){

//都执行成功

echo "交易成功";

$pdo->commit();

}else{

//若有其中之一不成功,回滚

echo "交易失败";

$pdo->rollBack();

}

//5、开启自动提交

$pdo->setAttribute(PDO::ATTR\_AUTOCOMMIT,1);

## 文件上传

1.表单中具有浏览框的条件

a.传输方式 'post'

b.form 标记中添加 enctype="multipart/form-data"

2. 接受上传信息 $\_FILES

(1)上传文件名称

$\_FILES['浏览框名']['name']

(2)上传文件类型

$\_FILES['浏览框名']['type']

(3)上传文件在服务器上的临时目录及文件位置

$\_FILES['浏览框名']['tmp\_name']

(4)上传文件报错信息

$\_FILES['浏览框名']['error'] 0代码成功

(5)上传文件大小

$\_FILES['浏览框名']['size']

说明:

a. 单位字节(Byte), 1B =2^8 =8bit

b.

1024B = 1KB

1024KB = 1MB

1024MB = 1GB

1024GB = 1TB

3 上传函数:将上传文件的临时位置移动到目标位置

move\_uploaded\_file($tmpName,$filename)

4上传文件报错信息

a.UPLOAD\_ERR\_OK

其值为 0，没有错误发生，文件上传成功。

b.UPLOAD\_ERR\_INI\_SIZE

其值为 1，上传的文件超过了 php.ini 中

upload\_max\_filesize 选项限制的值,默认值2M。

c.UPLOAD\_ERR\_FORM\_SIZE

其值为 2，上传文件的大小超过了 HTML 表单中 MAX\_FILE\_SIZE 选项指定的值。

d.UPLOAD\_ERR\_PARTIAL

其值为 3，文件只有部分被上传。

e.UPLOAD\_ERR\_NO\_FILE

其值为 4，没有文件被上传。

f.UPLOAD\_ERR\_NO\_TMP\_DIR

其值为 6，找不到临时文件夹。PHP 4.3.10 和 PHP 5.0.3 引进。

说明:php.ini upload\_tmp\_dir =临时文件位置

g.UPLOAD\_ERR\_CANT\_WRITE

其值为 7，文件写入失败。PHP 5.1.0 引进。

5. 包含文件函数

(1) include:当加载文件失败时，报警告(Warning)错误但后面

代码会继续执行

(2) require:当加载文件失败时，报致命(Fatal)错误并且后面

代码停止执行

(3) include\_once:相同文件只加载一次

(4) require\_once:相同文件只加载一次

6.多文件上传

(1)表单元素名称以数组形式命名:

上传数据三维数组，所以要转为索引+关联的

二维数组

(2)表单元素名称单独命名：上传数据关联+关联

的二维数组,所以转为索引+关联的二维数组,

和单文件上传的数据区别，单独命名多文件上传

是多个成员值(多个文件)

7.上传步骤

(1) 表单中具有浏览的条件

a.传输方式 'post'

b.form 标记中添加 enctype="multipart/form-data"

(2)封装代码

a.单文件和多文件(以数组命名和单独命名)

上传的信息转为索引+关联的新二维数组

b.循环遍历新二维数组，获得每个文件信息

c.判断每个文件报错等于 0

d.为了避免文件冲突，对每个文件重命名

e.对每个文件实现上传

f.返回重命名后的文件名称的数组

(3)要求

a.限制文件大小 php.ini upload\_max\_filesize=2M

\* b.限制文件类型(jpg,gif,png)

c.为了避免文件冲突，对每个文件重命名

d.判断上传的文件是否是图像 getimagesize

e.判断上传传输类型是否是post is\_uploaded\_file函数

\* f.判断是否存在文件夹，如果不存在创建文件夹

## GD

### 概述

GD库用来制作动态图片

GD库支持三种互联网中流行的图片格式：jpeg、gif、png。

### 用途

-- 验证码

-- 水印

-- 图片的缩放和剪切

-- 动态图表

### 安装

1、确认是否安装

phpinfo()

2、编辑PHP的配置文件 php.ini

extension\_dir = "D:\php5.4\ext"

extension=php\_gd2.dll

去掉extension前面的分号

3、重启网络服务器（apache）

4、再次确认是否安装

### 应用

imagecreatetruecolor(int $width,int $height)

功能：创建画布资源

参数：

width 画布的宽

height 画布的高

返回：资源

实例:

$width = 200;

$height = 200;

$gd = imagecreatetruecolor($width,$height);

imagecolorallocate(resource $img,int $red,int $green,int $blue)

功能：创建颜色

参数：

img 创建画布时产生的资源

red 红色，取值范围0~255

green 绿色，取值范围0~255

blue 蓝色，取值范围0~255

返回：颜色值

实例:

$red = imagecolorallocate($gd,255,0,0);

$green = imagecolorallocate($gd,0,255,0);

imagesetpixel(resouece $img,int $x,int $y,int $color)

功能：画点

参数：

img 创建画布时产生的资源

x,y 点的坐标

color 点颜色

实例:

imagesetpixel($gd,100,100,$red);

imagefill(resouece $img,int $x,int $y,$color)

功能：填充画布

参数：

img 创建画布时产生的资源

x，y 填充画布背景色时的坐标（0,0）

color 颜色

实例:

imagefill($gd,0,0,$green);

header("Content-Type:image/图片格式")

功能:通知浏览器输出图片,图片格式 png、gif、 jpeg

实例:

header("Content-Type:image/png");

image图片格式(resource $img)

功能: GD库函数输出图片

实例:

imagepng($gd);

image图片格式(resource $img,string $filename)

功能：画布资源的收藏

参数：

img 创建画布时产生的资源

filename 保存图片的文件地址.

注意：

保存文件的图片格式要跟函数中指定的图片格式保持一致

实例:

imagejpeg($img,"pixel.jpg");

imagedestroy(resource $img)

功能：释放内存

参数：

img 创建画布时产生的资源

实例:

imagedestroy($gd);

imageline(resource $img,int $x\_1,int $y\_1,int $x\_2,int $y\_2,int $color)

功能：画线

参数：

img 创建画布时产生的资源

x\_1,y\_1 线的起点坐标

x\_2,y\_2 线的终点坐标

color 线的颜色

实例:

imageline($img,0,0,200,200,$red);

imagerectangle(resource $img,int $x\_1,int $y\_1,int $x\_2,int $y\_2,int $color)

功能：空心的矩形

参数：

img 创建画布时产生的资源

x\_1,y\_1 矩形的起点坐标

x\_2,y\_2 矩形的终点坐标

color 空心矩形边框的颜色

实例:

imagerectangle($img,50,50,100,100,$red);

imagefilledrectangle(resource $img,int $x\_1,int $y\_1,int $x\_2,int $y\_2,int $color)

功能：实心的矩形

参数：

img 创建画布时产生的资源

x\_1,y\_1 矩形的起点坐标

x\_2,y\_2 矩形的终点坐标

color 实心矩形的颜色

实例:

imagefilledrectangle($img,100,100,150,150,$red);

imageellipse(resource $img,int $x,int $y,int $width,int $height,int $color)

功能：画圆，空心圆

参数：

img 创建画布时产生的资源

x,y 圆心点坐标

width 圆的宽

height 圆的高

color 颜色

实例:

imageellipse($img,100,100,100,100,$red);

imagefilledellipse(resource $img,int $x,int $y,int $width,int $height,int $color)

功能：画圆，实心圆

参数：

img 创建画布时产生的资源

x,y 圆心点坐标

width 圆的宽

height 圆的高

color 颜色

实例:

imagefilledellipse($img,70,70,50,50,$blue);

imagestring(resource $img,int $font,int $x,int $y,string $text,int $color)

功能：在图片上输出文字

参数：

img 创建画布时产生的资源

font 文字的字体大小，取值范围（1~5）

x，y 文字的起始点坐标

text 文字

color 字体颜色

实例:

imagestring($img,5,100,100,"hello world",$blue);

imagettftext(resource $img,int $size,float $angle,int $x,int $y,int $color,string $ttffilename,

string $text)

功能：输出文字

参数：

img 创建画布时产生的资源

size 字体的大小

angle 文字出现的角度

x，y 文字出现的位置

color 字体的颜色

ttffilename 字体文件的位置,C:\WINDOWS\Fonts

text 文字内容

实例:

imagettftext($img,20,0,100,100,$blue,"simfang.ttf","你好");

imagecreatefrom图片格式(string $filename)

功能：从现有图片上获取GD资源

参数：

filename 现有图片位置

返回：资源

注意：现有图片的图片格式要跟函数中指定的图片格式保持一致

实例:

$filename = "dog.jpg";

$img = imagecreatefromjpeg($filename);

getimagesize(string $filename)

功能：返回图片的信息

参数：

filename 要获取图片信息的图片文件地址

返回：图片的信息，格式索引数组。

下标 0 图片的宽

1 图片的高

2 图片的格式，格式是以数值来表示1 gif 2 jpeg 3 png

实例:

$filename = "qq.png";

$fileinfo = getimagesize($filename);

var\_dump($fileinfo);

imagefontwidth(int $font)

功能：获取font字体的宽度，仅支持font（1~5）

参数：

font 字体大小，取值范围（1~5）

返回：字体的宽度

实例:

$font=5;

imagefontwidth($font);

imagefontheight(int $font)

功能：获取font字体的高度，仅支持font（1~5）

参数：

font 字体大小，取值范围（1~5）

返回：字体的高度

实例:

$font=5;

imagefontheight($font);

imagecopy(resource $dst\_img,resource $src\_img,int $dst\_x,int $dst\_y,int $src\_x,int $src\_y,

int $src\_w,int $src\_h)

功能：将两张图片合并为一张图片

参数：

dst\_img 目标图片的资源（大图）

src\_img 源图片的资源（水印图片）

dst\_x，dst\_y 目标图片上的坐标

src\_x,src\_y 小图片的坐标

src\_w,src\_h 小图片的宽和高

实例:

$filename = "dog.jpg"; //$dst 目标图片

$wetermark = "qq.png"; //$src 水印图片 源图片 小图片

$dst\_img = imagecreatefromjpeg($filename);

$src\_img = imagecreatefrompng($wetermark);

imagecopy($dst\_img,$src\_img,100,200,12,0,10,25);

imagecopyresampled(resource $dst\_img,resource $src\_img,

int $dst\_x,int $dst\_y,int $src\_x,int $src\_y,

int $dst\_w,int $dst\_h,int $src\_w,int $src\_h)

功能：图片的缩放和剪切

参数：

dst\_img 目标图片的资源

src\_img 源图片的资源

dst\_x,dst\_y 目标图片起始点坐标

src\_x,src\_y 源图片的起始点坐标

dst\_w,dst\_h 目标图片的宽和高

src\_w,src\_h 源图片的宽和高

实例1: 缩放

$dst\_img = imagecreatetruecolor(500,500);

$white=imagecolorallocate($dst\_img,255,255,255);

imagefill($dst\_img,0,0,$white);

$filename = "qq.png";

$src\_img = imagecreatefrompng($filename);

imagecopyresampled($dst\_img,$src\_img,100,100,0,0,300,300,22,25);

实例2: 剪切

$dst\_img = imagecreatetruecolor(100,100);

$filename = "dog.jpg";

$src\_img = imagecreatefromjpeg($filename);

imagecopyresampled($dst\_img,$src\_img,0,0,0,0,100,100,150,150);

imagerotate(resource $src\_img,float $angle,int $bg\_color)

功能：图片的旋转

参数：

src\_img 源图片的资源，要被旋转的图片资源

angle 旋转的角度

bg\_color 旋转后的背景色

返回：资源 被旋转后的图片资源

实例1:旋转

$filename = "dog.jpg";

$src\_img = imagecreatefromjpeg($filename);

$angle = 45;

$bg\_color = imagecolorallocate($src\_img,0,255,0);

$dst\_img = imagerotate($src\_img,$angle,$bg\_color);

实例2:沿Y轴翻转

$filename = "dog.jpg";

fz($filename);

function fz($filename){

list($width,$height,$type) = getimagesize($filename);

$dst\_img = imagecreatetruecolor($width,$height);

$type\_array = array(1=>"gif",2=>"jpeg",3=>"png");

$type\_str = $type\_array[$type];

$fun = "imagecreatefrom".$type\_str;

$src\_img = $fun($filename);

for($i=0;$i<$width;$i++){

$dst\_x = $width-$i-1;

$dst\_y = 0;

$src\_x = $i;

$src\_y = 0;

imagecopy($dst\_img,$src\_img,$dst\_x,$dst\_y,$src\_x,$src\_y,1,$height);

}

header("Content-Type:image/gif");

imagegif($dst\_img);

imagedestroy($dst\_img);

imagedestroy($src\_img);

}

## HTTP

http协议:

用来规定客户端与服务器进行数据交互的方式和格式。

方式：

请求（request）和应答(response)的方式

特点：

无状态：无法记录用户的行为

## Session和cookie

### Cookie

1、cookie的生成

setcookie(string $name,string $value,int $expire)

功能：生成cookie

参数：

name cookie的名称

value cookie的值

expire cookie的存活时间。存活时间是从现在向后顺延的秒数。

例如cookie的存活时间是一个小时

time()+3600

若expire存活时间没有给出，则关闭浏览器，cookie就失效。

2、读取cookie

$\_COOKIE['cookiename']

3、删除COOKIE

setcookie(sting $name,string $value,int $expire)

功能：删除cookie

参数：

name 要删除的cookie的名称

value 空字符串“”

expire 当前时间戳剪去任意时间。

实例:

//设置cookie

$expire = time()+24\*3600;

setcookie("uname","tim",$expire);

//读取cookie

echo $\_COOKIE["uname"];

//删除cookie

$expire = time()-100;

setcookie("uname","",$expire);

### Session

1、生成SESSION

session\_start()

- 分配sessionid给客户端。

kpdq9lu2gfm9p6e54cuvi9ni35

- 在服务器端创建session文件。

sess\_kpdq9lu2gfm9p6e54cuvi9ni35

$\_SESSION['sessionname'] = sessionvalue

2、读取SESSION

session\_start()

将浏览器上的sessionid与服务器端的session文件建立连接

$\_SESSION['sessionname']

3、SESSION的删除

session\_start()

- 将sessionid与session文件建立连接

- unset($\_SESSION['sessionname'])

删除一个session元素

- $\_SESSION = array();

将整个session文件清空

- session\_destroy();

删除session文件

实例:

//生成session

session\_start();

$\_SESSION['uname']="jerry";

$\_SESSION['age']=10;

//读取session

session\_start();

echo $\_SESSION['uname'];

//删除session

session\_start();

unset($\_SESSION['uname']);

$\_SESSION = array();

session\_destroy();

SESSION的垃圾回收

session.gc\_maxlifetime = 1440（24分钟）

记录session文件的最大存活时间。超过1440秒，session文件就被视为垃圾文件。

删除是由session\_start()

session\_start()并不是每次启动都执行垃圾删除。

session\_start()执行垃圾删除的时机：

session.gc\_probability = 1/session.gc\_divisor = 1000

session\_start()每启动1000次，有1次删除垃圾文件的机会。

## REG

### 概述

正则表达式是有语法结构的字符串

正则表达式主要用于表单验证和网络爬虫

函数库:

PCRE正则表达式库的语法和java、c 的PERL正则表达式兼容，效率高，相对安全

POSIX存在安全问题，不建议使用

### 语法

正则表达式由定界符、原子、元字符、模式修正符组成

**定界符:**

定界符可以是除了字母、数字、反斜线、空格以外的任何字符。定界符成对使用，

例如{}、||。行业中常用的定界符是一对正斜线//

实例:

$pattern=”/hello/”;

**原子:**

原子是正则表达式最小的组成单位，有意义的正则表达式至少有一个原子。

原子种类:

所有打印字符:

非打印字符:

\n 换行

实例:

$pattern=”/\n/”;

特殊字符:

? \* + 需要用反斜线\进行转义

实例:

$pattern=”/\?/”;

特殊原子:

\d 匹配所有的数字

\D 匹配所有的非数字

实例:

$pattern = "/\d/";

\w 匹配所有的字母、数字、下划线

\W 匹配所有的非字母、数字、下划线

实例:

$pattern = "/\w/";

\s 匹配所有的空白

\S 匹配所有的非空白

实例:

$pattern = "/\S/";

自定义原子表:

[]

^ 在自定义原子表中代表：非

用法:

[abc] 匹配要匹配的字符串中的a、b和c三个字母

[a-z] 匹配要匹配的字符串中的26个小写的英文字母

[^a-z] 匹配要匹配的字符串中的除了26个小写的英文字母的字符

[a-zA-Z] 匹配要匹配的字符串中的52个小写、大写的英文字母

实例:

$pattern = "/[bcd]/";

**元字符:**

元字符是用来修饰原子的

元字符介绍:

? 用?修饰的原子可以出现0次或者1次

\* 用\*修饰的原子可以出现0次或者多次

+ 用+修饰的原子可以出现1次或者多选

实例:

$pattern = "/go?gle/";//字母o连续出现0次或者1次

$pattern = "/go\*gle/";//字母o连续出现0次或者多次

$pattern = "/go+gle/";//字母o连续出现1次或者多次

{n} 用{n}修饰的原子连续出现n次

{n,m} 用{n,m}修饰的原子连续出现至少n次,至多m次,包含n次和m次

{n,} 用{n,}修饰的原子连续出现至少n次，至多不限制,包含n次

实例:

$pattern = "/go{2}gle/"; //o连续出现2次

$pattern = "/go{1,3}gle/";//o连续出现至少1次，至多3次

$pattern = "/go{1,}gle/";

| 或

实例:

$pattern = "/mysql|oracle/";

/b 匹配字符串的边界

/B 匹配字符串的非边界

实例:

$pattern = "/\Band/";

$pattern = "/and\B/";

^或\A 表示以 ^和\A 来修饰的原子作为字符串的起始

$或\Z 表示以$和\Z 来修饰的原子作为字符串的结尾

实例:

$pattern = "/\Aabc/";//得到的字符串以abc为起始

$pattern = "/abc\Z/"; //得到的字符串以abc为结尾

() 作用:

- 子模式

实例:

$pattern = "/(s)(q)/";

$pattern = "/mysq(l|o)racle/";

- 改变匹配的范围

- 反向引用

**模式修正符:**

i 写在定界符的外面,使正则表达式忽略字母大小写

实例:

$pattern = "/TEST/i";

### 应用

匹配、查找、分隔、替换

普通的字符串处理函数

- 简单 效率高

- 功能有限

正则表达式字符串处理函数

- 复杂 效率低

- 功能强大

**匹配、查找:**

substr(string $haystack,int $start[,int $length])

功能：字符串的截取

参数：

haystack 要截取的字符串

start 截取的起始位置

length 截取的长度，length没有给出，截取到字符串的末尾

返回：被截取后的字符串

实例:

$haystack = "tom@qq.com";

echo substr($haystack,0);

strpos(string $haystack,string $needle)

功能：查找needle在haystack中首次出现的位置

参数：

haystack 要进行查找的字符串

needle 子字符串

返回：位置

实例:

$haystack = "this is a test";

echo strrpos($haystack,"s");

preg\_match\_all(string $pattern,string $subject[,$match])

功能：正则匹配

参数：

pattern 正则表达式

subject 要进行正则匹配的字符串

match 正则匹配的结果

返回：正则匹配到的次数

实例:

$subject = "this is a test";

$pattern = "/t/";

var\_dump(preg\_match\_all($pattern,$subject,$match));//3次

preg\_match(string $pattern,string $subject[,array $match])

功能：正则匹配

参数：

pattern 正则表达式

subject 要进行正则匹配的字符串

match 正则匹配到的结果

返回： 正则匹配到的次数。0或者1

实例:

$subject = "this is a test";

$pattern = "/t/";

var\_dump(preg\_match($pattern,$subject,$match));//1次

preg\_grep(string $pattern,array $subject)

功能：正则匹配

参数：

pattern 正则表达式

subject 要进行正则匹配的数组，数组的元素都是字符串。

返回：匹配到的结果

实例:

$subject = array("team","term","time");

$pattern = "/t/";

$result = preg\_grep($pattern,$subject);

var\_dump($result);

**分隔:**

explode(string $delimiter,string $subject[,int $limit])

功能：字符串的分隔

参数：

delimiter 进行分隔标识字符串,delimiter不能为空字符串

subject 要进行分隔的字符串

limit 限制最大分隔次数

返回：被分隔好的字符串,数组形式

实例:

$subject = "this is a test";

$delimiter = " ";

$result = explode($delimiter,$subject,3);

var\_dump($result);

implode(string $glub,array $subject) 别名：join

功能：将数组转化为字符串

参数：

glub 联合数组中元素的字符串

subject 要联合数组

返回：字符串

实例:

$subject = array("this","is","a","test");

echo implode("",$subject);

preg\_split(string $pattern,string $subject)

功能：正则匹配拆分

参数：

pattern 正则表达式

subject 要进行拆分的字符串

返回：正则拆分的结果

实例:

$subject = "t3h6is i8s a t0est";

$pattern = "/[\d ]/"; //数字或空格

$result = preg\_split($pattern,$subject);

var\_dump($result);

$subject = "t,h.i;s i:s a t!es?t";

$pattern = "/[,\.;:!\?]/";

$result = preg\_split($pattern,$subject);

var\_dump($result);

**替换:**

str\_replace(mixed $search,mixed $replace,mixed $subject)

功能：字符串的替换

参数：

search 要查找的字符串

replace 替换的字符串

subject 源字符串

返回：被替换后的字符串

实例:

$subject = "疯子说下半年是雾霾天";

$search = array("疯子","雾霾");

$replace = "\*\*";

echo str\_replace($search,$replace,$subject);

preg\_replace(mixed $pattern,mixed $replace,mixed $subject)

功能：正则替换

参数：

pattern 正则表达式

replace 进行替换的字符串

subject 源字符串

返回： 被替换后的字符串

实例:

$subject ="含有<span >HTML</span>标签的<b>字符</b>串";

$pattern = "/<\/?[a-z0-9]+[\sa-z0-9=':;\"]\*>/i";

$replace = "A";

echo preg\_replace($pattern,$replace,$subject);

### 贪婪模式

.\* 贪婪模式

.\*? 非贪婪模式

### 提高

实例1:

$subject = "2017/01/10";

$pattern = "/\d{4}([\/\-])\d{2}\\1\d{2}/"; //\\1代表 第一个小括号中匹配到的结果

preg\_match($pattern,$subject,$match);

var\_dump($match);

实例2:

$subject="abchttps://www.baidu.comabchttp://www.sohu.comabcftp://image.biadu.c

omabcftps://www.sohu.comabc";

$pattern = "/(https?|ftps?):\/\/([a-z0-9]+)\.([a-z0-9]+)\.(com|cn|org|net|com.cn)/";

$replace = "<a href='\\1://\\2.\\3.\\4'>\\1://\\2.\\3.\\4</a>";

echo preg\_replace($pattern,$replace,$subject);

实例3:

$subject="abchttps://www.baidu.comabchttp://www.sohu.comabcftp://image.biadu.c

omabcftps://www.sohu.comabc";

$pattern = "/(https?|ftps?):\/\/.\*?\.(com|cn|org|net|com.cn)/i";

preg\_match\_all($pattern,$subject,$match);

var\_dump($match);

## FILE

### 概述

**文件系统的用途:**

网站静态化

日志的收集

**文件系统的分类:**

file 文件

dir 目录

block 块文件

char 字符类型文件。

fifo 管道文件

link 链接文件

unknow 未知

**目录:**

**相对目录和绝对目录**

绝对目录：从根目录计算起得目录就成为绝对目录

windows：盘符

linux： /

相对目录：从当前文件计算起得目录成为相对目录

**目录分隔符**

反向目录分隔符：\

正向目录分隔符：/ 建议使用正向目录分隔符

### 应用

**PHP文件系统函数:**

filetype(string $filename)

功能：获取文件的类型

参数：

filename 要获取的文件类型的地址

返回：文件类型的字符串形式

实例:

$filename = "test.txt";

echo filetype($filename);//file

$filename=”test”;

echo filetype($filename);//dir

is\_dir(string $filename)

功能：判断文件是否是一个目录

参数：

filename 要进行判断的文件地址

返回：布尔值

is\_file(string $filename)

功能：判断是否是一个文件

参数：

filename 要进行判断的文件地址

返回：布尔值

实例:

$filename = "./test/b";

if(is\_file($filename)){

echo "是文件";

}

if(is\_dir($filename)){

echo "是目录";

} //是目录

is\_readable(string $filename)

功能：查看文件是否可读

参数：

filename 要查看的文件地址

返回：布尔值

is\_writable(string $filename)

功能：判断文件是否可写

参数：

filename 要判断的文件地址

返回：布尔值

is\_executable(string $filename)

功能：判断文件是否可执行

参数：

filename 要判断的文件地址

返回：布尔值

实例:

$filename = "php.exe";

$filename = "test.txt";

if(is\_readable($filename)){echo "可读";}

if(is\_writable($filename)){echo "<br/>可写";}

if(is\_executable($filename)){echo "<br/>可执行";} //可读、可写

file\_exists(string $filename)

功能：判断文件或者目录是否存在

参数：

filename 要进行判断的文件或者目录

返回：布尔值

实例:

$filename = "test";

if(file\_exists($filename)){

echo "文件存在";

}else{

echo "文件不存在";

}

filesize(string $filename)

功能：返回文件的大小

参数：

filename 要判断文件大小的文件地址

返回：文件的大小,单位 byte 字节

1024types(2^10bytes) = 1KB

1024KB(2^20 bytes) = 1MB

1024MB (2^30 bytes)= 1GB

1024GB(2^40 bytes) = 1TB

实例:

function getFileSize($filename){

$size = filesize($filename);

$unit = ""; //用来记录文件的单位

if($size>=pow(2,40)){

$unit = "TB";

$size = $size/pow(2,40);

}else if($size>=pow(2,30)){

$unit = "GB";

$size = $size/pow(2,30);

}else if($size>=pow(2,20)){

$unit = "MB";

$size = $size/pow(2,20);

}else if($size>=pow(2,10)){

$unit = "KB";

$size = $size/pow(2,10);

}else{

$unit = "bytes";

}

return $size.$unit;

}

echo getFileSize($filename);

filectime(string $filename)

功能：获取文件的创建时间

参数：

filename 要获取文件创建时间的文件地址

返回：文件的创建时间，返回格式 时间戳1970-01-01

filemtime(string $filename)

功能：获取文件的修改时间

参数：

filename 要获取文件修改时间的文件地址

返回：文件的修改时间，返回格式 时间戳1970-01-01

fileatime(string $filename)

功能：获取文件的访问时间

参数：

filename 要获取文件访问时间的文件地址

返回：文件的访问时间，返回格式 时间戳1970-01-01

实例:

date\_default\_timezone\_set("PRC");

header("Content-Type:text/html;charset=utf-8");

$filename = "export.php";

echo "文件的创建时间:".date("Y-m-d H:i:s",filectime($filename));

echo "<br/>文件的修改时间:".date("Y-m-d H:i:s",filemtime($filename));

echo "<br/>文件的访问时间:".date("Y-m-d H:i:s",fileatime($filename));

**PHP对目录的操作:**

glob(string $pattern)

功能：目录的遍历

参数：

pattern 遍历的目录的匹配模式

返回：数组。遍历的目录中的内容。

- 不支持solaris操作系统

- 不支持远程文件目录遍历

实例1:

$pattern = "E:/\*";

var\_dump(glob($pattern)); //e盘中所有的文件和文件夹

实例2:

$filename = "./\*";

$result = glob($filename);

foreach($result as $value){

if(is\_file($value)){

echo "文件：".$value."<br/>";

}

if(is\_dir($value)){

echo "目录：".$value."<br/>";

}

}

opendir(string $pathname)

功能：打开目录

参数：

pathname 要打开的目录地址

返回：资源

readdir(resource $dir)

功能：读取已经打开的目录资源

参数：

dir 已经打开的目录资源

返回：目录下的内容,执行一次获取一次,当获取不到时返回false

注意: 所有目录下都有.和.. . 表示当前目录 .. 表示上一级目录

closedir(resource $dir)

功能：关闭已经打开的目录资源。释放内存

参数：

dir 已经打开的目录资源

实例:

$pathname = "test";

$dir = opendir($pathname);

echo readdir($dir);

echo "<br/>";

echo readdir($dir);

echo "<br/>";

echo readdir($dir);

echo "<br/>";

echo readdir($dir);

echo "<br/>";

echo readdir($dir);

closedir($dir);

实例1:统计目录的大小

function sumDir($pathname){

$dir = opendir($pathname);

while($filename = readdir($dir)){

if($filename!="."&&$filename!=".."){

$filename = $pathname."/".$filename;

if(is\_dir($filename)){

$size+=sumDir($filename);

}else{

$size+=filesize($filename);

}

}

}

closedir($dir);

return $size;

}

$pathname = "test";

echo sumDir($pathname);

实例2:统计目录下的文件和目录数

function countDir($pathname){

global $file\_num,$dir\_num;

$dir = opendir($pathname);

while($filename = readdir($dir)){

if($filename!="."&&$filename!=".."){

$filename = $pathname."/".$filename;

if(is\_dir($filename)){

$dir\_num++;

countDir($filename);

}else{

$file\_num++;

}

}

}

closedir($dir);

}

$pathname = 'test';

$file\_num = 0; //用来计算目录下的文件数

$dir\_num = 0; //用来计算目录下的目录数

countDir($pathname);

echo "文件数:".$file\_num;

echo "<br/>目录数:".$dir\_num;

**PHP对文件和目录的操作:**

mkdir(string $pathname[,int $mode][,boolean $recursive])

功能：创建目录

参数：

pathname 创建的目录名称

mode 创建目录的权限。最大权限：777

recursive 是否可以递归创建目录。默认false。

返回：布尔值

实例:

$pathname = "test";

var\_dump(mkdir($pathname)); //true

$pathname = "test\_new/test";

var\_dump(mkdir($pathname,777,true));

rename(string $oldname,string $newname)

功能：对目录或者文件进行重命名

参数：

oldname 源文件名称

newname 目标文件名称

返回：布尔值

实例:

$oldname = "test";

$newname = "test\_new";

var\_dump(rename($oldname,$newname))

$oldname = "a.txt";

$newname = "c.txt";

var\_dump(rename($oldname,$newname));

copy(string $oldfilename,string $newfilename)

功能：文件内容的复制

参数：

oldfilename 源文件名称

newfilename 目标文件名称

返回：布尔值

实例:

$oldfilename = "c.txt";

$newfilename = "a.txt";

var\_dump(copy($oldfilename,$newfilename));

实例1:目录的复制

function copyDir($oldpath,$newpath){

if(file\_exists($newpath)){

if(is\_file($newpath)){

echo "目录路径是文件，无法拷贝";

exit;

}

}else{

mkdir($newpath);

}

$dir = opendir($oldpath);

while($filename = readdir($dir)){

if(!($filename=="."||$filename=="..")){

$oldfilename = $oldpath."/".$filename;

$newfilename = $newpath."/".$filename;

if(is\_dir($oldfilename)){

copyDir($oldfilename,$newfilename);

}else{

echo "oldfilename=".$oldfilename;

echo "<br/>";

echo "newfilename=".$newfilename;

echo "<hr/>";

copy($oldfilename,$newfilename);

}

}

}

closedir($dir);

}

//将 test目录复制到 test\_new

$oldpath = "test";

$newpath = "test\_new";

copyDir($oldpath,$newpath);

rmdir(string $pathname)

功能：删除一个空目录

参数：

pathname 要删除的目录地址

返回：布尔值

实例:

$pathname = "test\_new";

var\_dump(rmdir($pathname));

unlink(string $filename)

功能：删除文件

参数：

Filename 要删除的文件地址

返回：布尔值

注意：被unlink删除的文件不在回收站里

实例:

$filename=”b.txt”;

var\_dump(unlink($filename));

实例1:递归删除目录

function delDir($pathname){

if(file\_exists($pathname)){

if(is\_file($pathname)){

unlink($pathname);

exit;

}

}else{

echo "目标目录不存在";

exit;

}

$dir = opendir($pathname);

while($filename = readdir($dir)){

if($filename!="."&&$filename!=".."){

$filename = $pathname."/".$filename;

echo $filename;

echo "<br/>";

if(is\_dir($filename)){

delDir($filename);

}else{

unlink($filename);

}

}

}

closedir($dir);

rmdir($pathname);

}

$pathname = 'test';

delDir($pathname);

**PHP对文件的操作:**

touch(string $filename)

功能：创建一个空的新文件

参数：

filename 要创建的文件的地址

返回：布尔值

实例:

$filename = "d:/a.txt";

var\_dump(touch($filename));

unlink(string $filename)

功能：删除文件

参数：

filename 要删除的文件地址

返回：布尔值

copy(string $oldfilename,string $newfilename)

功能：文件的复制

参数：

oldfilename 源文件名称

newfilename 目标文件名称

返回：布尔值

rename(string $oldname,string $newname)

功能：对目录或者文件进行重命名

参数：

oldname 源文件名称

newname 目标文件名称

返回：布尔值

file\_get\_contents(string $filename)

功能：获取文件内容

参数：

filename 要获取文件内容的地址，

地址可以是远程文件地址（http，ftp协议）,也可以是本地文件地址

返回： 文件内容。格式：字符串

实例:

$filename = "http://www.baidu.com";

$contents = file\_get\_contents($filename);

var\_dump($contents);

$filename = "a.txt";

$contents = file\_get\_contents($filename);

var\_dump($contents);

file\_put\_contents(string $filename,string $contents)

功能：向filename中写入contents内容

参数：

filename 要写入内容的文件地址（ftp 有权限）

contents 要写入的内容

返回：成功：成功写入的字节数；失败：false

实例:

$filename = "a.txt"; $contents = "新的内容";

var\_dump(file\_put\_contents($filename,$contents));

实例1:

$filename = "index.php";

$contents = file\_get\_contents($filename);

file\_put\_contents("index1.php",$contents);

fopen(string $filename,string $mode)

功能：打开文件

参数：

filename 要打开的文件地址

mode 以何种形式打开文件

r 以只读方式打开文件，将文件指针指向文件首。

若文件不存在，不尝试创建。

r+ 以读写方式打开文件，将文件指针指向文件首。

若文件不存在，不尝试创建。

w 以只写方式打开文件，将源文件内容清空，

文件指针指向文件首。若文件不存在，则尝试创建。

w+ 以读写方式打开文件，将源文件内容清空，

文件指针指向文件首。若文件不存在，则尝试创建。

a 以追加的方式打开文件，以只写的方式打开，

文件指针在文件内容末尾，若文件不存在，尝试创建新文件。

a+ 以追加的方式打开文件，以读写方式打开文件，

若读，则文件指针在文件首，若写，则文件指针在文件末尾。

若文件不存在，尝试创建新文件

返回：文件资源

实例:

$filename = "d.txt";

$mode = "a+";

$fh = fopen($filename,$mode);

echo fgetc($fh);

fclose($fh);

fgetc(resource $fh)

功能：从打开的文件资源中获取一个字节的内容

参数：

fh 已经打开的文件资源

返回：获取一个字节的内容

实例1:

$filename = "a.txt";

$mode = "r";

$fh = fopen($filename,$mode);

echo fgetc($fh);

fclose($fh);

实例2:

$filename = "a.txt";

$fh = fopen($filename,"r");

while($char=fgetc($fh)){

echo $char;

echo "<br/>";

}

fclose($fh);

fgets(resource $fh)

功能：获取文件中的内容，一次获取一行

参数：

fh 已经打开的文件资源

返回：每执行获取到一行文件中的内容，获取不到false。

实例1:

$filename = "a.txt";

$fh = fopen($filename,"r");

echo fgets($fh);

fclose($fh);

实例2:

$filename = "a.txt";

$fh = fopen($filename,"r");

while($string=fgets($fh)){

echo $string;

echo "<br/>";

}

fclose($fh);

fwrite(resource $fh,string $contents)

功能：向已经打开的文件中写入内容

参数：

fh 已经打开的文件资源

contents 要写入的内容

返回： 成功写入的字节数

实例:

$fh = fopen("a.txt","a+");

fwrite($fh,"hello你好");

fclose($fh);

fclose(resource $fh)

功能：释放已经打开的文件资源

参数：

fh 已经打开的文件资源

实例:

$filename = "a.txt";

$mode = "r";

$fh = fopen($filename,$mode);

echo fgetc($fh);

fclose($fh);

fread(resource $fh,int $length)

功能：读出文件中的内容。

参数：

fh 已经打开的文件资源

length 读出的长度，单位：字节

返回：文件中的内容。

实例1:

$filename = "a.txt";

$fh = fopen($filename,"r");

echo fread($fh,10);

fclose($fh);

实例2:

$filename = "a.txt";

$fh = fopen($filename,"r");

echo fread($fh,filesize($filename));

fclose($fh);

feof(resource $fh)--file end of file

功能：判断文件指针是否到达文件尾

参数：

fh 已经打开的文件资源

返回：布尔值 true 到达文件尾 false 没到达文件尾。

实例:

$filename = "a.txt";

$fh = fopen($filename,"r");

while(!feof($fh)){

echo fgetc($fh);

}

fclose($fh);

## OOP

### 概述

面向对象编程（OOP Object Oriented Programming） 实现了软件功能的三大目标：

- 灵活性

- 重用性

- 扩展性

面向对象三大特征:

-封装

-继承

-多态

面向过程

- 以事件为解决问题的中心。

- 将解决问题的步骤封装为一个个独立的函数。

- 在解决的过程中逐一调用。

面向对象

- 以功能为解决问题的中心。

- 注重的是对象本身。

- 只有面向对象的编程才是符合人类思维逻辑的。

对象

-对象具有属性和行为。

### 语法

类：类是具有相同属性和行为的一组对象的集合。

类名称和类文件:

类名称:

- 类名称要起得有意义

- 数字、字母和下划线，数字不能作为类名称的开头

- 类名称建议使用大驼峰式命名规则

类文件:

- 一个类在一个独立的类文件中。

- 类文件命名建议采用 ： 类名称.class.php

类的声明:

class 类名称{

}

实例:

Mobile.class.php文件夹中

class Mobile{

}

类的实例化:

$变量名称 = new 类名称；

$变量名称 = new 类名称();

类对象访问类的成员属性和方法:

一般访问方式:(类外访问）

类对象->成员属性;

类对象->成员方法;

特殊访问方式:(类中访问)

$this->成员属性;

$this->成员方法;

//获取类名、变量、方法

$class = get\_class($this);

$attributes = get\_object\_vars($this);

$methods = get\_class\_methods($this);

method\_exists($class,$method);

### 封装

应用访问修饰符对成员属性和成员方法进行修饰，从而达到访问控制和隐藏。

访问修饰符:

public:

类里、类外、子类都能访问成员属性和方法

protected:

类里、子类能访问成员属性和方法，类外不能

private

类里能访问成员属性和方法，类外、子类不能

### 继承

定义一个新类，继承先前定义的类的成员属性和成员方法，并在新定义的类中写入新的成员属性和成员方法，从而达到类之间的层级关系。

PHP中的继承是单继承的。一个子类只能有一个父类，一个父类可以有多个子类。

在子类中定义父类中的成员方法，称为重写。

在子类中访问父类中被重写了的成员方法应用关键词parent。

PHP继承的语法：

class 子类名称 extends 父类{

}

实例:

include "Person.class.php";

class Student extends Person{

function walk(){

parent::say();

echo "<hr/>";

$this->say();

}

function say(){

echo "I am a student";

parent::say();

}

}

$s = new Student;

$s->say();

### 多态

让有继承关系的类，通过对相同的成员方法的调用，产生不同的结果。多态也称为：同名异式

抽象方法和抽象类:

a、定义抽象方法时必须加上关键词abstract.抽象方法没有方法体。

b、定义抽象类时要求类的前面加上 abstract

c、含有抽象成员方法的类必须被声明为抽象类；抽象类不一定含有抽象方法

d、抽象类可以含有普通的成员方法和成员属性。

e、抽象类不能够被实例化，抽象类是用来定义规范的。

实例:

abstract class Person{

public $name;

abstract function say();

function walk(){

}

}

接口:

a、接口中只能有常量和抽象方法。

b、接口中的抽象方法没有关键词 abstract。

c、接口不能够被实例化，接口是用来定义规范的。

实例:

interface Test{

const DNS = "www.baidu.com";

function say();

}

抽象类和接口的继承:

继承抽象类:

子类继承抽象类后,必须重写抽象类中的所有抽象方法

格式:

class Children extends Parent{

}

继承接口:

a、一个子类可以继承多个接口，多个接口之间以","进行间隔

b、子类继承接口后，必须重写接口中的所有抽象方法

格式:

interface TestA{

}

interface TestB{

}

class Children implements TestA,TestB{

}

继承抽象类和接口:

格式:

class Children extends Parent implements TestA,TestB{

}

接口继承接口:

格式:

interface TestA extends TestB{

}

知识点:

function useUsb(USB $u){//USB是一个抽象类，判断类对象$u是不是属于USB

$u->start();

$u->run();

$u->stop();

}

### 生产模式

**单例模式:**

保证类仅仅被实例化一次，内存空间仅仅被分配一次，节省内存空间。

应用：

数据库类(MySQL)实例化。

NoSQL-redis（键值对）（内存）类

实例:

class Single{

static $obj = null;

private function \_\_construct(){

}

private function \_\_clone(){

}

static function getObj(){

if(is\_null(self::$obj)){

self::$obj = new Single();

}

return self::$obj;

}

function \_\_destruct(){

echo "bye";

}

}

$s = Single::getObj();

$s2 = Single::getObj();

**工厂模式:**

### 魔术方法

**魔术方法的特点：**

- 在语法定义上名称都是固定的。通常以\_\_作为魔术方法的名称的起始。

- 魔术方法都有各自特殊的用途。

- 魔术方法都有自己的语法要求。

- 魔术方法会在特定的触发机制下调用。

**\_\_构造方法**

定义：

function 类名称([参数..]){

方法体

}

或

function \_\_construct([参数..]) {

方法体

}

- 构造方法在类被实例化时自动被访问。

- 用途：给成员属性赋初值。

实例:

class Mobile{

public $voltage;

function \_\_construct($voltage){

$this->voltage = $voltage;

}

}

$m = new Mobile(220);

echo $m->voltage;

**\_\_析构方法**

定义：

function \_\_destruct(){

方法体

}

- 在类对象被访问结束时，析构方法会被自动访问。

- 用途：释放内存

实例:

class Mobile{

function \_\_destruct(){

echo "destruct";

}

}

echo 1;

$m = new Mobile();

echo 2;

$mx = new Mobile();

echo 3;

**\_\_get()**

- 魔术方法\_\_get要求必须有一个参数

- 在类外访问私有化成员属性时，魔术方法\_\_get会被自动调用。

实例:

class WareHouse{

private $keeper="tom";

private $location="beijing";

function \_\_get($args){

echo $args;//参数就是在类外访问的私有化成员属性名称

}

}

$w = new WareHouse;

echo $w->location;

**\_\_set()**

- 魔术方法 \_\_set要求有两个必须的参数。

- 在类外设置私有化成员属性时，魔术方法\_\_set会被自动调用

实例:

class WareHouse{

private $keeper;

private $location;

private $test;

function \_\_set($method,$args){

if($method=="keeper"){

$can\_keeper = array("tom","jerry");

if(in\_array($args,$can\_keeper)){

$this->$method = $args;

}else{

$this->$method = "tom";

}

}else if($method=="location"){

$can\_location = array("beijing","shanghai");

if(in\_array($args,$can\_location)){

$this->$method = $args;

}else{

$this->$method = "beijing";

}

}else{

echo "这个私有化成员属性禁止被访问";

}

}

function \_\_get($args){

return $this->$args;

}

}

$w = new WareHouse;

$w->keeper = "jack";

echo $w->keeper;

echo "<br/>";

$w->location = "guangzhou";

echo $w->location;

echo "<br/>";

$w->test = "test";

**\_\_clone**

- 魔术方法\_\_clone在应用关键词clone时会被自动触发。

- 用途：给成员属性赋初值

class Person{

public $name;

function \_\_clone(){

$this->name = "john";

}

}

$p = new Person;

//将$p所指向的内存空间复制给$p2、$p3

$p2 = clone $p;

echo $p2->name;

$p3 = clone $p;

echo $p3->name;

**\_\_toString()**

- 要求在魔术方法体中return一个字符串

- 在以字符形式输出类对象时，会被自动触发

- 用途：介绍类

实例:

class Person{

function \_\_toString(){

return "this a Person class";

}

}

$p = new Person;

echo $p;

**\_\_call()**

- 要求有两个必须的参数

- 在访问类中不存在的成员方法时，会被自动调用。

- 用途：将多个功能相似的成员方法封装到魔术方法call中。

实例:

class Person{

function \_\_call($method,$args){

$method\_array = array("a","b","c");

if(in\_array($method,$method\_array)){

for($i=1;$i<=5;$i++){

echo $method.' ';

echo 'arguments';

echo '<br/>';

}

}else{

echo "没有这个成员方法:".$method;

}

}

}

$p = new Person;

$p->a();

echo "<br/>";

$p->b();

echo "<br/>";

$p->c();

echo "<br/>";

$p->d();

**\_\_autoload()**

**spl\_autoload\_register("autoload"); //spl\_autoload\_register里面使用一个自定义**

- 要求必须有一个参数

- 当实例化一个类时，魔术方法autoload会被自动触发

- 用途：实现自动加载

实例:

Autoload文件夹中的文件:

Grandparent.class.php

Parent.class.php

Student.class.php

index.php

index.php文件中内容:

function \_\_autoload($classname){

echo $classname;

echo "<br/>";

include $classname.".class.php";

}

$s = new Student();

浏览器显示内容:

Student

Person

grandparent

### 相关关键词

instanceof:

判断一个类对象是否属于某个类

实例:

include "Person.class.php";

class Student extends Person{

}

$s = new Student;

$p = new Person;

if($s instanceof Student){

echo 'yes'; //yes

}else{

echo 'no';

}

if($s instanceof Person){

echo "yes"; //yes

}else{

echo "no";

}

if($p instanceof Student){

echo "yes";//no

}else{

echo "no";

}

const:

定义:

在类中定义常量，也可以在类外定义常量。

const 常量名称=常量值

- 常量一经定义不能改变。

- 常量名称尽量大写。

//define("DNS","www.localhost.com",false);false表示对大小写敏感

//define不能用于在类中定义常量

访问:

- 在类外对常量的访问：类名称::常量名称

- 在类里面对常量的访问：self::常量名称

实例:

class Person{

const COUNTRY="中华人民共和国";

function say(){

echo "我是".self::COUNTRY."人";

}

}

echo Person::COUNTRY;

final:

- 用final修饰的类不能够被继承。

- 被final修饰的成员方法不能够被重写。

实例:

class Person{

final function say(){

echo "I can say";

}

}

final class Person{

final function say(){

echo "I can say";

}

}

static:

- static 关键词用于修饰成员属性和成员方法

- 被static修饰的成员属性和成员方法保存静态内存区。

- 在静态化的成员方法中不能访问非静态化的成员方法和成员属性。

在静态化的成员方法中不能出现$this

优点:

节省内存空间,访问方便

访问成员属性和方法的方式:

类里:

self::成员属性

self::成员方法()

类外:

类名称::成员属性

类名称::成员方法()

实例:

class Person{

static $country = "中国";

public $name;

static function say(){

echo self::$country;

echo 'I can say';

}

function walk(){

self::say();

}

}

echo Person::$country;

echo "<hr/>";

Person::say();

## MVC

### 单一入口

**什么是单一入口:**

单一入口是指程序中有一个(但不并唯一一个)入口文件,其实质就是通过URL参数

来动态访问某个文件内的函数/某个类文件的某个方法

**为什么要采用单一入口:**

A.便于URL记忆;

B.可以将频繁使用的业务(如数据库连接、SESSION初始化等)初始化完成;

**单一入口的实现原理**

A.URL地址栏可以传递任意合法参数;

B.PHP支持动态包含文件;

C.PHP支持变量函数;

### MVC

**什么是MVC**

MVC是Model(模型) -- View(视图) -- Controller(控制器)的简写;

Model表示应用程序核心(比如数据库记录列表);

View表示显示的HTML页面;

Controller表示应用程序中处理用户交互的部分;如从视图读取数据,并且模

型发送数据;

### PATHINFO模式

**什么是PATHINFO模式**

PATHINFO模式就是通过路径表现形式在URL中传递相关参数的方式;

**为什么要采用PATHINFO模式**

便于网络推广(SEO)

**如何获取地址栏中的PATHINFO参数**

string $\_SERVER['PATH\_INFO']

### Apache URL Rewrite

**什么是URL Rewrite**

URL Rewrite(URL重写),就是把一个客户端请求的地址转换服务器真实地址的过程

**为什么要采用URL Rewrite**

A.把复杂的URL地址转换在短洁的URL地址;

B.便于SEO;

**开启Apache URL重写模块**

A.编辑Apache的配置文件 -- httpd.conf

B.启用LoadModule rewrite\_module modules/mod\_rewrite.so选项

C.将AllowOverride None修改为AllowOverride All

D.保存设置后,重新启动Apache

**.htaccess文件**

作用:定义目录级别的重写规则

**重写规则**

RewriteEngine On/Off,打开/关闭重写引擎

RewriteRule 源地址正则表达式 目标地址

RewriteCond 重写条件

%{HTTP\_HOST}

%{REQUEST\_URL}

%{REQUEST\_FILENAME}

### ZendStudio安装Emmet

**安装**

A.Help --> Install New Software

B.在Work With中输入 http://emmet.io/eclipse/updates/

**Emmet快捷键**

父子:标记名称 > 标记名称

兄弟:标记名称 + 标记名称

向上:标记名称 ^ 标记名称

数量:标记名称 \* 量词

文本:{}

属性:[]

div id="container" : #container

标记名称 id="container" : 标记名称#id

div class="a b" : .a.b

标记名称 class="a b" : 标记名称.a.b

## 乱码问题

##zend工具 :

window-Perferences-General-workspace - utf8

##HTML5

<meta charset="utf-8"/>

##PHP

header('Content-Type:text/html;charset=utf-8');

##mysql

##mysql 服务器配置文件my.ini

default-character-set=utf8

character-set-server=utf8

##命令行

创建数据库 utf8

创建表结构 utf8

set names gbk；

show variables like 'character%'; utf8