## 控制台

打开控制台：空白处shift+右键 启动后显示当前所在的目录

dir 查看目录下所有内容

cd 路径 修改当前目录 注：这个不能切换盘符

盘符：回车 切换盘符

使用反斜线表示路径分隔符号

## 安装服务器

我们采用手动安装的方式, 可以更好的控制, 推荐大家在 C 盘根目录下

1> 解压

2> 安装 C++ 运行环境( 默认安装即可 )

3> 在 c 盘根目录下建一个文件夹( 命名为 WAMP )

W, Windows

A, Apache

M, MySQL

P, PHP

4> 解压 apache 的压缩包, 并将其放到 wamp 文件夹中

5> 找到 Apache24 文件夹

bin, binary, 二进制, 里面放置的就是程序

conf, configuration, 配置, 里面放置的就是配置文件

htdocs, html documents, html 的文件

6> 在 conf 文件夹中找到 httpd.conf 文件

这个文件是 文本文件. 打开文件, 找到 Define SRVROOT "/Apache24"

大约在 38 行, 将 解压的 apahce 所在的目录, 复制过去.

再去修改 反斜线 为 正斜线

7> 在 C 盘下, 找到 windows, 在找到 system32, 找到 cmd 文件

在上面右键, 以管理员身份运行

使用 cd 命令进入到 apache 所在的 bin 目录下

使用命令:

> httpd -k install

安装 apache 服务( services ), 如果看到了 successfully installed

表示安装成功.

如果安装不成功, 表示已经安装过 apache,

如果是启动不成功, 表示已经安装过其他服务器软件( iis, tomcat 等 )

如果是启动不成功, 找到配置文件中的 Listen 80 修改为 Listen 8080

然后再启动

8> 启动的时候, 可以使用 bin 目录下的 ApacheMonitor.exe 程序

该程序启动后会运行在 系统托盘中

右键打开工具, 也可以左键按照操作提示执行

## 环境配置

apache 是服务器, 应该让 apache 知道 什么程序应该有 php 来处理

1> 下载解压 php

2> 找到 apache 的 配置, 打开, 将文件拖到最底部

3> 加入:

LoadModule php5\_module "C:/WAMP/php/php5apache2\_4.dll"

使得 apache 可以找到 php 程序

4> 加入:

AddType application/x-httpd-php .php

这一句话是说 什么后缀的文件让 php 引擎处理

## 基本概念

IP, IP 地址是在网络中唯一定位计算机的办法. 也就是说要找到一台计算机就需要知道它的 IP 地址就可以了.

端口, 端口可以分为物理端口和逻辑端口. 我们现在讨论的端口是指逻辑端口. 在一台计算机中有很多应用程序.

我们利用 IP 可以找到计算机, 我们利用端口可以找到计算机上面的某一个程序.

默认端口, 在访问我们的网站的时候, 如果仅仅提供 ip 而提供端口, 那么默认就是在访问 80 端口

协议, 所谓的协议就是一种访问的规则.

协议约定了我们的计算机是如何通信的. 我们建立网站使用的 是 http 协议, 即 超文本 传输 协议

如果我们的计算机中搭建的是 ftp 服务器( 文件传输协议 )

如果要与计算机进行通信, 就需要三个基本的东西:

1> 是 IP 先找到计算机

2> 确定端口, 找到计算机中的程序

3> 确定协议, 保证采用制定的方式传输

域名( domain ), 早期 ip 不容易记忆. 因此发明了一个特殊的键值对( 键是一个有意义的字符串, 值就是 ip 地址 ) 就是ip的别名

在浏览器中输入 http://www.itcast.cn 然后回车

我们的浏览器会发送请求到最近的 DNS 服务器( 域名系统服务器, 就是存储 键值对的 计算机 )

我们可以使用 一个 命令 ping 来嗅探我们的域名对应的 ip 是多少

> ping www.baidu.com

回环地址与 hosts 文件

我们在做网站的时候, 最终的目标是可以在公网上访问.

回环地址是指当前计算机, 无论网线是否连接到计算机中, 都可以使用这个回环地址, 即 127.0.0.1, 表示的就是本机

host 文件就是用来在本机模拟 dns 服务器使用的文件

这个文件就在 C:/windows/system32/drivers/etc/

三次握手 补充

目的: 确保数据发送时我们处于连接状态

发送者与接收者必须得到通知, 同时做好准备

三次握手只是在保证数据连接上了, 但是与数据的传输没有关系

你发送数据, 好比你寄快递( 国际快递 )

他接收数据, 好比他接收快递( 国际快递 )

## php执行原理

在哪运行

在服务器上运行，不在浏览器中，用户无法看到php逻辑

运行结果的作用

将php结果发回浏览器，由浏览器来解析显示（html字符串）

执行原则

在php引擎中有一个缓存（cache）

php在执行时 会 ’输出’ 很多东西 放在缓存中 执行结束后 才会将缓存里的数据一次性返回到浏览器

和js 预解析类似 先解析 后执行

从上往下执行代码 不是php代码则缓存 是php代码则执行

## 路径

url 地址, 统一资源定位符 通俗的说就是一个在网络上的文件的地址.

格式: 协议://主机名或IP地址:端口/路径?键=值&键=值#锚点

相对路径：

./ 当前目录

../ 上一级目录

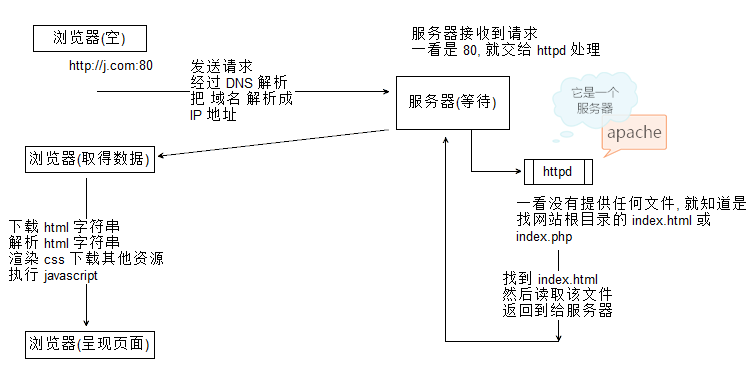
这个相对路径是以 url 地址为参考的

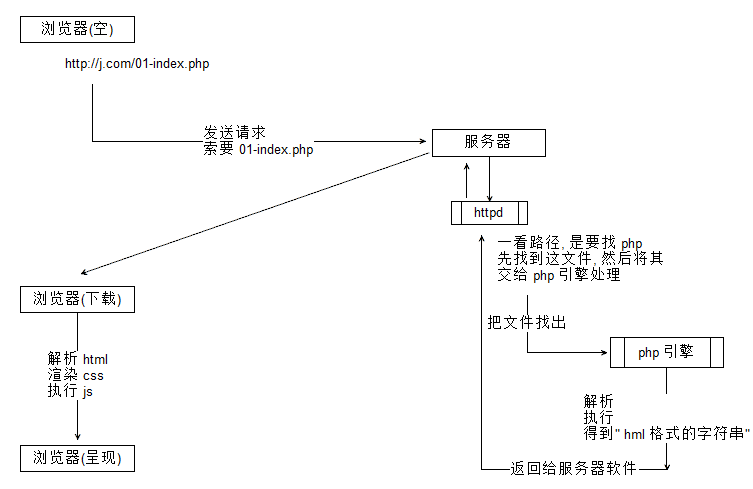
处理文件的路径的时候, 推荐使用绝对路径

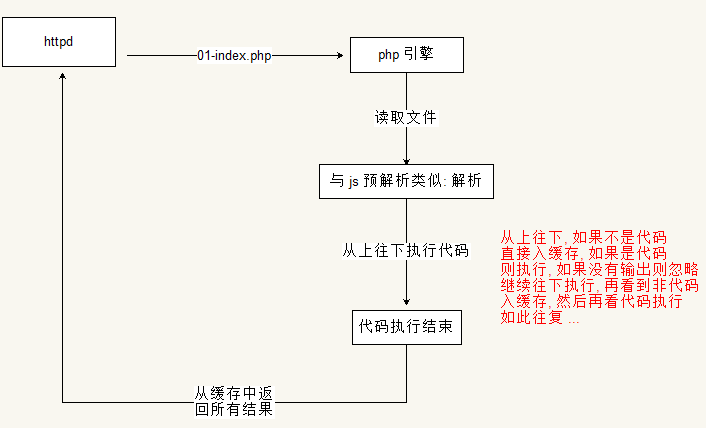
html里都用绝对路径（指的是由浏览器读取的代码，无论是不是写在php里）

跟php有关的都用相对路径（由服务器读取的代码）

## 网络交互模型（图）







## php基础

❀ php的代码结构

<?php

Php代码

?>

❀ 输出

echo "html 格式的字符串"

❀ php 与 js 一样是结构松散的语言( 可以随意的加空格 ) 弱类型语言

❀ 注释

// 行注释

/\* 块级注释 \*/

# shell 风格的 行注释

❀ php 代码语句以分号结束

❀ 如果 php 文件中只有一段 php 的代码, 其后 的 ?> 可以省略

### 数据类型与表达式：

运算符( operator, js 一样 )

算术运算符: +, -, \*, /, %( 取模 )

关系运算符: >, <, ..., ==, ===, !=, !==

逻辑运算符: &&, ||, !

自增自减运算符( 自操作运算符 ): ++, --

其他运算符: (), [], ?:

赋值运算符与复合赋值运算符: =, +=, -=, ...

特殊的: 字符串连接运算符, 在 php 中使用点号 "."

例如:

echo "你好" . "张三";

### AND, OR 关系运算符

表示的也是 并且与或者的含义, 但是其优先级比 赋值运算符低.

例如:

$num = 1 && 2; 其结合顺序是 先计算 1 && 2, 然后将其结果赋值给 变量 $num.

$num = 1 AND 2; 其结合顺序是 先进行赋值运算操作 $num = 1, 然后赋值表达式的值是 等号右边的值 再利用赋值表达式的值 与 右边的 2 作 并且运算.

因此 and 或 or 主要用于执行命令操作时 进行逻辑中断处理.

例如: 我要打开一个文件( 打开成功, 打开失败 )

$file = 打开文件;

if ( $file ) {

继续操作文件

} else {

结束代码

}

其等价于

$file = 打开文件 OR 结束操作;

等价

$file = 打开文件;

$file OR 结束操作;

( $file = 打开文件 ) OR 结束操作

$file = ( 打开文件 || 结束操作 )

### 表达式( expression )

将数据与运算符连接起来就构成表达式

所谓的表达式, 就是最后可以得到结果的 语法结构.

常见的表达式

1> 常量表达式: 123, false, "abc" ...

2> 算术表达式: 1 + 2, 1 \* 3, 10 % 2, ...

3> 逻辑表达式: 1 && 2, 2 || 3, 3 AND 4, 4 OR 5, ...

在 js 中 判断的时候会进行真值转换, 但是取值的时候 不会进行真值转换

在 js 中 10 && 2, 那么 10 为真, 那么与判断取右边的值, 右边的值为 2 则整个表达式的值为 2

在 php 中真值判断与取值都会进行真值转换, 如果 10 && 2, 左边转为真, 即( 1 ), 与判断取右边的值

在取右边值的时候不是取原值而是取其 boolean 值, 右边为 2, 转换为 boolean 为 true 即 1

4> 关系表达式: 1 > 2, 3 == 4, 4 === 5, ...

5> 三元表达式: 1 == 2 ? 表达式1 : 表达式2

在 js 中 三元算符 的结合顺序是从左往右 的

在 php 中 三元运算符是 从右往左 的

例如:

1 > 2

? 'a'

: 2 > 1

? 'b'

: 'c'

总结： 括号里看成整体

1 > 2 ? 'a' : ( 2 > 1 ? 'b' : 'c' ) js

1>2为false 所以取 ( 2 > 1 ? 'b' : 'c' ) 然后再往后判断

最终的结果在 a 和....（通过括号内容判断）中选

( 1 > 2 ? 'a' : 2 > 1 ) ? 'b' : 'c' php

最终的结果在 b c 中选

### 流程控制

❀ 顺序结构

❀ 分支结构

if, if-else, if-else if ...

注意: 在 php 中

if ( ... ) {

...

} else if ( ... ) {

...

}

可以写成

if ( ... ) {

...

} elseif ( ... ) {

...

}

switch-case

❀ 循环结构

while 循环

do-while 循环

for 循环

在 js 中有 for-in 循环

for ( var k in obj ) {

... // 注意的事情是 k 为索引

}

在 php 中支持 foreach 循环

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

foreach ( $arr as $index=>$value ) {

}

foreach 还有一个简化的写法

foreach( $arr as $item ) {

// 此处的 $item 是每次遍历数组的项

}

在 php 中, 数组不是对象. 要对数组做操作不像 js 那样使用方法点, 而是统统使用函数.

1> count() 用来获得数组的长度

2) strlen() 用来获得字符串的长度

## 数据类型与运算符

php 与 js 一样是弱类型 编程语言 变量可以存储任意类型的数据

### 八种数据类型

整型: 整数类型, 使用 int 或 integer 表示

浮点型: 小数类型, 使用 float 或 double 来表示

字符串类型: 使用单引号或双引号括起来的字符序列

数组类型: 在 js 中是引用类型, 但是在 php 中是值类型

布尔类型: true, false

对象类型: object

资源类型: resource

空类型: null

### 变量的定义与使用

变量以 $ 开头, 用数字, 字母, 下划线组成, 不以数字开头.

变量不需要声明, 直接赋值使用, 只要赋值了就相当于定义了.

### 输出(放到缓存中)

echo

语法:

echo "字符串1", "字符串2", "字符串3", ...

将这些字符串 一个个 依次放到缓存中

代码:

echo "a", "b", "c";

等价于

echo "a";

echo "b";

echo "c";

print

与 echo 的功能类似, 但是只允许提供一个输出参数

printf

print\_r

语法

print\_r( 数组 )

会格式化显示数组的内容

var\_dump

语法与 print\_r 一样

但是会打印出数据的类型

## 交互模型

url结构

协议：//地址：端口/路径? 参数#锚点

http 协议到底是怎么来传输数据的

结论：

http 协议是建立在TCP/IP协议族基础之上

TCP/IP协议族不是一个协议, 是很多协议来支撑我们的通信.

TCP/IP 协议保证了数据的传输的完整性, 准确性以及顺序等问题.

例如: 文本 "123"

我们可以知道文本中存储的其实是 49, 50, 51 这样的数字

我们通过 tcp/ip 协议来传送数据, 我们可以将其切分成三个小的部分( 每一个部分一个字节 )

然后每一个部分进行发送. 我们按照 文件的顺序发,

1> 发送 49,

2> 发送 50,

3> 发送 51

...

在接收的时候什么顺序都有可能, 但是在全部数据接收到以后. 依旧会按照我们切割的顺序将文件组织起来

最后还原成 字符串 123

### http协议的概念、

HTTP（HyperText Transfer Protocol，超文本传输协议）最早就是计算机与计算机之间沟通的一种标准协议，这种协议限制了通讯 内容的格式 以及各项 内容的含义 。

http基本上是使用字符串来传输的

http由 请求与响应 构成 采用 请求报文与响应报文 的形式来进行数据交互

特殊格式的字符串

请求报文语法：

请求行

请求头

请求头

...

空行

请求体

host:请求的主机 （ip地址）

Referer:防盗链

响应报文语法：

响应行

响应头

响应头

...

空行

响应体

实例1:

生活中, 你去吃饭, 会提供一个点餐的小票.

这个小票写的东西就是你的需求

随后这个小票会送往后厨, 后厨的厨师根据小票的内容来制作餐点 ...

后厨在处理好餐点后, 会将餐点与小票一同交给前台 的 服务员

服务员根据小票上的信息 将餐点给你

讨论一下本质

你发送的是什么: 一个特殊格式的字符串, 这个格式约束了很多的信息, 而菜单的信息就是你请求时提供的数据

你接收到的是: 一个特殊字符串 + 数据

实例2:

浏览器输入 http://j.com/02-http/01-index.html

回车, 浏览器会将该地址与当前操作系统的信息进行 "包装". 使用的是 http 请求格式来包装

GET /02-http/01-index.html HTTP/1.1 请求行: 请求谓词 地址 协议版本

host: j.com 请求头

Accept: ... 请求头

... 请求头

空行

请求体( 不传参, 所以是空 )

从net work里找

其中常用的请求头

Host: 访问的主机

Connection: 长连接, 连接方式

User-Agent: 客户端代理字符串

Accept: 浏览器支持的格式 接收类型

Accept-Encoding: 浏览器支持的编码 请求接收的编码

Accept-Language: 浏览器的语言

Referer: 从哪里来 防盗链

浏览器就会将字符串发送到 服务器 上 的 服务器软件, 服务器软件接收到这个字符串后, 服务器软件解析完字符串后

就可以给出响应的操作

服务器软件进行处理, 该交给 php 引擎的 就交给 php 引擎, 该直接返回的就直接返回 ...

服务器在准备返回数据的时候, 也会生成 一个 http 的响应报文

HTTP/1.1 200 OK 协议版本 响应状态码 响应状态码短语

Content-Length: 5 响应头, 基本上就是在告诉浏览器怎么处理数据

☆ Content-Type: text/html; charset=utf-8

...

空行

hello 响应体

服务器就将 这个 字符串 + 数据 发回浏览器

浏览器在接收数据后, 就开始解析 html 渲染 css, 执行 js, 最终呈现页面

要求掌握的内容:

1> 复述整个请求响应的过程, 越详细越好

2> 常见的 请求与响应头

3> 响应状态码

4> MIME 类型

常见的响应状态码

1xx 正在处理

2xx 成功

3xx 重定向( 不在当前请求中取得数据 )

4xx 找不到（客户端的错误）

5xx 服务器内部错误

MIME 类型

作用就是在告诉浏览器怎么处理返回的数据

MIME 类型的 语法形式

大类/子类

需要记住的几个类别

text/plain

text/html

text/xml

application/json

如何去使用( php )

header( "Content-Type: MIME类型" );

补充函数:

readfile( 文件路径 )

将文件的数据读取出来放到 缓存 中

### 请求报文

网络交互是利用请求与响应报文来实现的, 在请求中含有需要请求的文件地址

还可以提供很多数据参数交给服务器.

我们交给服务器的参数有两种

1> 是配置信息

2> 是提交的数据

在 PHP 中提供一些 超全局变量( 数组 )

$\_SERVER 是我们的所有的请求报文内容 和 计算机服务器的 配置信息

$\_GET 获得利用 get 方法请求时 提交的数据

$\_POST 获得利用 post 方法提交的 数据

$\_REQUEST 获得 利用 get 或 post 方法提交时 提价的数据

$\_COOKIE

$\_SESSION

$\_FILES

$GLOBALS

使用：

$\_GET[ "wd" ] ) wd为键值对的键名name 获取对应的值value

如何发送 get 请求

表单标签

<form action="我们将数据提交到对应的地址" method="提交的方式GET或POST">

在标签中含有 name 属性的 input 标签, textarea 标签, select 标签的数据可以被提交

</form>

如何发送 post 请求

只需要将 method 属性修改为 post 即可编程 post 请求

### get与post的区别

1) GET 请求 参数会附在 url 地址中传递, get 请求会有缓存

2) POST 请求 参数会在报文中提供, post 请求没有缓存. post 会增加一个新的 MIME 类型

application/x-www-form-urlencoded

这个 mime 类型是为了告诉服务器如何解释参数

3) 异同

- GET 请求会将数据暴露在 url 地址中, 因此敏感信息应该使用 post 请求.

- 由于 GET 请求时将数据放在 url 地址里面 而不同的浏览器 url 地址的容量是有限的. 如果请求发送的数据过大不易使用 GET.

- 由于 GET 请求使用 url 地址, 用字符串来提交参数, 因此无法处理二进制数据, 因此要提交文件数据, 必须使用 POST

GET 与 POST 的原义

GET 谓词原意是说向服务器索要数据, 提供的参数多半是用于在服务器检索数据的依据.

POST 谓词原意是说将数据从 浏览器 发送给 服务器.

在请求中还有 HEAD, DELETE, ...

restfull

我们刚刚所描述的内容: 如何将数据从浏览器发送到 服务器

## 字符串

### 定义和使用

1. 使用双引号或单引号即可定义字符串

2) 使用双引号的字符串具有变量解析与转义解析的功能( 区别于 js )

例如:

$num = 123;

echo "num = $num"; // "num = 123";

### 输出

printf 格式化输出函数. 例如我们想要输出 $num1 + $num2 = ...

js: console.log( $num1 + ' + ' + $num2 + ' = ' + ( $num1 + $num2 ) );

php: echo "$num1 + $num2 = " . ( $num1 + $num2 );

格式化输出可以提供一个字符串格式, 然后在里面 "挖坑", 最后用数据填坑, 得到想要的结果

格式:

"%d + %d = %d"

printf( "%d + %d = %d", $num1, $num2, $num1 + $num2 );

%d 整数

%f 小数

%s 字符串

### 字符串增加, 删除, 修改, 查询的方法

增加

.

删除

删除字符串中的某一个小的字符串

str\_replace( 查找字符串, 替换字符串, 原始字符串 )

借助于该方法可以实现, 修改字符串, 删除字符串, 新增字符串

trim( 字符串 )

修改

explode, 其功能与 js 中的 split 一模一样

语法: explode( 分割字符串, 原始字符串 ) -> 数组

Explode(“,” , $filename)

查询

strpos 类似于 js 中的 indexOf

## 数组

### 与js的比较

在 js 中, 数组就是 有顺序的 用下标从0开始计数的数据的集合. 我将其称为 线性数组( 有序数组 )

在 js 中, 将键值对记为 { k: v }, 将其称为对象.（关联数组）

在 php 中将线性数组与键值对 统称为 数组( 数据的集合 )

将其中的 线性数组 称为 有序数组 对应于 js 的数组

将其中的 键值对 称为 关联数组 对应于 js 的对象

### 定义数组

有序数组

$arr = array( 数据, 数据, 数据, ... ); 定义数组, 并对数组初始化

$arr = array(); 等价于 js 中的 var arr = [];

数据类型随意

关联数组

$arr = array( 键=>值, 键=>值, 键=>值, ... ); // 定义了关联数组, 并初始化, 类似于 js 中的 对象

数据类型, 在 数组中, 如果是有序数组数据类型随意.

如果是关联数组, 键是 字符串, 而值也是随意的. 与 js 对象也是一样.

一个 键=>值 称为一个项 键是字符串，值随意

### for循环 foreach循环

$arr = array(

array( "name"=>"jim1", "age"=>19, "gender"=>"male" ),

array("name"=>"jim2","age"=>29,"gender"=>"female" ),

array( "name"=>"jim3", "age"=>39, "gender"=>"male" )

);

有序数组里套一个关联数组

foreach( $arr as $row ) {

// 遍历行

}

foreach( $head\_data as $key => $value ) {

echo " "; // 遍历每一项

}

## 数组的操作

区别：

在js中访问数组的一个不存在的成员（或访问对象不存在的属性），得到的是undefined 在php中会报错

因此在 php 中访问数组成员的时候 一定要判断数组是不是存在该成员

### isset 函数

isset( 变量, 数组, 或 数组项 ) 返回的是bool类型

判断这个变量是否存在, 这个数组是否存在, 这个数组项是否有数据

isset无法判断数组的项是存在还是空 （存在/null 都返回true）

用法：

Isset($num);

Isset($\_GET[‘wd’]);

### empty函数

Empty(变量，数组，或数组项) 返回的是bool类型

判断这个数据是不是空 只有为空时才返回false 不存在也返回true

Empty 只能判断数组的项是不是空，

### array\_key\_exists 函数

array\_key\_exists（键名或下标，数组） 返回bool值

判断每一项是否为空 若值为null则输出 true

array\_key\_exists(1，$arr)

无法判断数据是否为空字符串

## 表单操作

表单标签:

<input type="text" />

<input type="button" />

<input type="reset" />

<input type="submit" />

<input type="file" />

<input type="image" />

<input type="radio" />

<input type="checkbox" />

<textarea></textarea>

<select>

<option><option>

<option><option>

</select>

表单的主要作用就是 收集用户输入的 数据, 提交给 服务器

input 文本标签( text, password )中：

name 用于描述提交数据时采用的 键

value 用户描述提交数据时采用的 值

标签在提供数据的时候 一个需要提供数据的值, 另一个要提供这个值对应的名字.这个 name 属性就是起到取名字的作用.

自定义属性

使用 data- 开头. 例如 <input type="text" data-id="txt">

## 重定向

在响应头中, 放一个响应头选项

浏览器在接收到 location 响应头之后，不会进行下载 html 数据( 实际上响应头如果含有 location, 则 Content-Length (发回了多少数据)就是 0 )，我的浏览器就会直接根据 locaiton 描述的路径, 到对应的路径中请求资源

header( "Location: " . $\_SERVER[ "HTTP\_REFERER" ] );

die();

header响应头：

Content-Length: 发回了多少数据

Content-Type: 告诉浏览器返回的数据是什么, 以至于如何处理

Location: 告诉浏览器直接跳转到某一个页面

set-cookie: 告诉浏览器保存什么数据

· http\_referer 是\_SERVER 数组中的一个项 指的是上一个页面（从哪来）的地址

· 直接跳转到特定页面：

location.href = "http://www.itcast.cn";

## 常用函数

trim（） 去除字符串的空格

strlen（） 获取字符串长度

## Include 系列语法结构

Include“文件路径”;

将对应的文件导入到 当前位置

Include\_once”文件路径”;（推荐使用）

将对应的文件导入到 当前位置，但是如果之前已经导入过则忽略

Require”文件路径”;

将对应的文件导入到 当前位置，但是如果之前已经导入过则后面代码不执行了

Require\_once”文件路径”; (推荐使用)

将对应的文件导入到 当前位置，但是如果之前已经导入过则忽略

## 上传checkbox 与radio的数据

### 复选框checkbox

<input type="checkbox" name="" value="" >

复选框的选中用 checked 属性来描述, 而这个属性是 单属性. 在提交的时候 会将含有 checked属性的复选框中的 name与value合成键值对进行提交.

多选时name写成数组形式 name=“nm[]”

将数据存到数组中，否则会覆盖

· 补充：Is\_array函数 判断是否是数组

## 文件上传

需要实现上传文件功能时写： （写在form里）

Enctype=”multipart/form-data” 允许上传文件 （记住）

记得设置为post提交

通过php的$

· 补充：sleep(x) 睡眠x秒

move\_uploaded\_file()

涉及到时间问题要设置时区

## 数据库相关概念

SQL 即 结构化查询语言( structure query language )

我们书写 sql 语句

通过 TCP/IP 连接到数据库上

将 sql 语法发过去

数据库管理系统解析 sql 语句

执行 sql 语句

将结果返回给连接的对象

## mysql中常见的数据类型

数字类型: int 和 bigint

小数类型: float, double, decimal

布尔类型: bit 存储的只能是 0 和 1

字符串类型: char( 数字 ) 固定长度 , varchar( 数字 ) 给一个长度范围

时间类型: date 日期, time 时间, datetime 日期时间

## 数据库的基本操作

### 增加数据

insert into 表名( 列名, 列名, ... ) values ( 值, 值, ... );

insert into student(name,age,sex) values(‘牛牛’,13,’f’);

补充:

在 sql 语句中, 字符串 使用 单引号 括起来, '张三'

在 sql 语句中的注释 -- 表示行注释

在 sql 语句如果要输入的是 数字, bit 类型直接写成数字的形式即可

如果在 sql 语句中表示时间, 使用字符串的形式 'yyyy-mm-dd hh:mm:ss'

### 删除数据( 慎用 )

delete from 表名 where 条件;

delete from tbl\_stu where stu\_id = 100;

### 修改数据( 慎用 )

update 表名 set 列名=值, 列名=值 where 条件

### 查询数据

语法:

select

列表达式,

列名,

常量,

函数,

...

from

表值结果集

where

条件

group by

分组依据

having

条件

order by 表名 升序排列

order by 表名 desc 降序排列

limit x,y x是从第几行开始，y是查询多少行

执行原理

数据库管理系统首先会读取 sql 语句, 并解析, 将其转换成查询计划( 具体的查询步骤 )

首先从 from 子句开始, 得到查询的原始数据集

执行 where 子句, 从原始的数据集中对数据进行筛选, 得到一个新的数据集.

然后执行分组, 将上一步得到的数据集进行分组, 得到一个新的数据集.

再使用 having 子句对分组后的数据进行筛选, 又一次得到一个新的数据集.

再使用 order by 对数据进行排序, 得到一个有序数据集( 游标 )

最后利用 select 选择 这个数据集中的 对应列将数据显示出来.

查询语句的案例

1>

select stu\_id, stu\_name, stu\_age, stu\_gender from tbl\_stu;

在 数据表 tbl\_stu 中, 将所有的数据全部取出来显示

select \* from tbl\_stu;

2>

select \* from tbl\_stu where stu\_id = 3;

在 数据表 tbl\_stu 中, 将 stu\_id 为 3 的数据取出来

条件不仅仅可以使用 一个列名 = 一个值来判断

还可以使用多个条件, 在数据库中逻辑运算并且用 AND 表示

或者用 OR 表示. 否定用 IS NOT 或 NOT( ... )

3>

select \* from tbl\_stu where stu\_age in ( 19, 20 );

检索 stu\_age 为 19 或 20 的数据. 离散范围

还有一个区间范围( 连续范围 )

select \* from tbl\_stu where stu\_age between 19 and 25;

检索 19 到 25 这个连续范围的所有数据, 包括两端

4> 分组案例

分组一般与聚合函数一起使用: 所谓的聚合函数就是将多个数据计算成一个数据. 常用聚合函数有:

count( \* ) 或 count( 列名 ) 或 count( 1 )

avg( 列名 ) 计算平均值

max( 列名 ) 计算最大值

min( 列名 ) 计算最小值

sum( 列名 ) 求和

count 聚合函数是用于求数据的条数

Count 写在select里

count( \* ) 和 count( 1 ) 语义是一样的. 求所有数据的从条数

count( 列名 ) 求的是 所有数据中, 这一列不为 空 的数据的总条数

例子: 将数据按照 stu\_gender 分组, 将每一组人数统计出来.

select count( \* ), stu\_gender from tbl\_stu group by stu\_gender;

在分组操作的时候, select 子句中只允许出现分组用的列名以及聚合函数

5> having 子句( 了解 )

在分组以后对数据进行检索, 例如将 按照 xxx 分组后 数据大于 xxx 的数据取出来

select 籍贯, count( \* ) from users group by 籍贯 having count( \* ) > 100;

6> order by

语法:

order by 列名 [desc]

将得到的结果按照列名来排序, 默认升序, desc 表示降序

7> limit 子句

1)

Limit x,y

Limit x offset y 从第几条开始:base0, 显示几条

2)

limit x 只写一个数字的时候表示显示几条 从0开始显示

## php操作数据库

### 连接、关闭数据库

连接数据库

$conn = mysqli\_connect( "127.0.0.1", "root", "", "student" );

库名

关闭数据库连接

mysqli\_close( $conn );

### 思路：

1. 连接数据库

$conn = mysqli\_connect( "127.0.0.1", "root", "", "web40php" );

数据库库名

2. 设置客户端字符集

mysqli\_query( $conn, "set names utf8" );

3. 执行 sql 语句

$reader = mysqli\_query( $conn, "select \* from tbl\_stu" );

· 发送 新增的 sql 语句

$is\_success = mysqli\_query( $conn, "INSERT INTO student ( name, age, sex ) VALUES( 'jim', 19, 'm' )" );

不要漏掉引号

注：删除/修改/增加 操作返回bool值 添加成功为true

查询 返回的是数组

mysqli\_query在使用时，括号内的第二个变量为字符串，需要用引号包裹

4. （查询操作）通过循环获得所有的数据, 得到 二维数组

$list = array();

while( $item = mysqli\_fetch\_assoc( $reader ) ) {

array\_push( $list, $item );

}

5. 释放相关资源

mysqli\_free\_result( $reader ); 释放读取器对象（执行查询操作的时候）

mysqli\_close( $conn ); //关闭数据库

1. 返回主页面

header( "Location: ./main.php" );

### 数据库乱码问题

mysql 数据默认采用 拉丁文

只需要在执行 sql 语句之前, 执行下面的代码

mysqli\_query( $conn, "set names utf8" );

设置数据的客户端语言为 utf-8 字符集

### 查询数据获取数据：

相当于遍历select 获取的数组

mysqli\_fetch\_row

将一个行以有序数组的形式返回 采用聚合函数 获取单个信息时使用

mysqli\_fetch\_assoc

将一个行以关联数组的形式返回, 键就是列名 获取多个信息时使用

mysqli\_fetch\_array

将关联数组与有序数组合并