

Phpmyadmin 配置，新建站点：vhost 文件

到 host 文件中去添加一条对应站点域名的解析信息：操作系统/system32/driver/etc/hosts：

```
127.0.0.1 www.myadmin39.com
```

并重启 apache。

3. 到 host 文件中去添加一条对应站点域名的解析信息：操作系统/system32/driver/etc/hosts：

```
127.0.0.1 www.myadmin39.com
```

4. 然后，就可以浏览该网站了，但结果是：



Fatal error: Call to undefined function mb_detect_encoding() in H:\

报错：该函数未定义。

原因：php 的某个模块没有打开。

去 php.ini 中打开该模块：

```
968 ;extension=php_ldap.dll
969 extension=php_mbstring.dll
970 ;extension=php_exif.dll ; Must be after mbstring as it c
971 extension=php_mysql.dll
```

当前同样要重启 apache

字段类型

mysql 中的字符串，应该使用“单引号”引起来。

Varchar: 可变长度字符串，使用时必须设定其长度，最大长度“理论值”65535.实际其最大只能是 65533.

如果是存储中文 gbk，最多为 65533/2 个；

如果是存储中文 utf8，最多为 65533/3 个。

Char: 定长字符串；使用时通常设定其长度，如果不设定，默认是 1，最大理论长度是 255 个。

定长字符串适用于存储的数据都是可预见的明确的固定长度字符串，比如手机号、中国邮政编码、实际存储的时候，如果少于设定长度也能存，但都会

补空格填满。

```
mysql> create table tab_char_varchar(  
  ->     postcode char(6), /*中国邮政编码*/  
  ->     name varchar(10) /*姓名*/  
  -> );  
Query OK, 0 rows affected (0.01 sec)
```

Enum: 单选项字符串数据类型。非常适合于存储表单界面中的“单选项值”。

Set: 多选项字符串数据类型。非常适合于存储表单界面中的“多选项值”。

enum 类型:

单选项字符串数据类型。它非常适合于存储表单界面中的“单选项值”；
它设定的时候，是需要给定“固定的几个选项”，然后存储的时候，就只存储其中一个值；
形式如下：

```
enum("选项 1", "选项 2", "选项 3", .....);
```

实际内部:

这些字符串选项值对应的是如下数字值: 1, 2, 3, 4, 5,最多 65535 个选项;

set 类型:

多选项字符串数据类型。它非常适合于存储表单界面中的“多选项值”；
它设定的时候，也需要给定“固定的几个选项”，然后存储的时候，就可以存储其中若干个值；
形式如下：

```
set("选项 1", "选项 2", "选项 3", .....);
```

实际内部:

这些字符串选项值对应的是如下数字值: 1, 2, 4, 8, 16,最多 65535 个选项;

单选项字符串数据类型。它非常适合于存储表单界面中的“单选项值”；
它设定的时候，是需要给定“固定的几个选项”，然后存储的时候，就只存储其中一个值；
形式如下：

```
enum("选项 1", "选项 2", "选项 3", .....);
```

实际内部:

这些字符串选项值对应的是如下数字值: 1, 2, 3, 4, 5,最多 65535 个选项;

写入数据形式:

可以用该选项字符串本身，也可以用对应的数字:

set 类型:

多选项字符串数据类型。它非常适合于存储表单界面中的“多选项值”；它设定的时候，也需要给定“固定的几个选项”，然后存储的时候，就可以存储其中若干个值；形式如下：

```
set("选项 1", "选项 2", "选项 3", ..... );
```

实际内部:

这些字符串选项值对应的是如下数字值: 1, 2, 4, 8, 16,最多 64 个选项;

写入数据形式:

可以用该选项字符串并用逗号分开本身，也可以用对应的数字的和;

Text:

text 类型:

它成为“长文本”字符类型。通常，其中存储的数据不占据表格中的数据容量限制。其本身最长可存储 65535 个字符。

其他同类字符类型: smalltext, tinytext, longtext。

其他(了解):

binary 类型: 定长二进制字符串类型，里面存储的是二进制值;

varbinary 类型: 变长二进制字符串类型，里面存储的是二进制值;

blob 类型: 二进制数据类型，存的仍然是二进制值，但其适用于存储“图片”，“其他文件”等，但极少用!

时间类型:

Datetime: 时间日期类型

Date: 日期类型

Time: 时间类型

Year: 年份类型

Timestamp: 时间戳类型，类似 js 中的 Gettime ()，或 js 是 Time ()，得到的是一个“整数数字”。在自己给定数据的情况下，需要使用单引号引起来，跟字符串一样!

```
#演示时间日期类型的字段使用:
create table tab_time(
    dt datetime,
    d2 date,
    t2 time,
    y year,
    ts timestamp/*这个字段通常不用插入数据*/
);
insert into tab_time(dt, d2, t2, y )values
('2015-7-8 10:12:30', '2015/7/8', '13:14:15', '2014' );
insert into tab_time(dt, d2, t2, y )values
(now(), now(), now(), '2015' );
```

查询出结果为:

```
mysql> select * from tab_time;
+-----+-----+-----+-----+-----+
| dt                | d2        | t2        | y      | ts                |
+-----+-----+-----+-----+-----+
| 2015-07-08 10:12:30 | 2015-07-08 | 13:14:15 | 2014   | 2015-07-23 14:49:56 |
| 2015-07-23 14:52:34 | 2015-07-23 | 14:52:34 | 2015   | 2015-07-23 14:52:34 |
+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)
```

表定义语句

基本语法形式:

- **create table** **[if not exists]** 表名 (字段列表 **[, 索引或约束列表]**) **[表选项列表]** ;
- 或这样来表达 :
- **create table** **[if not exists]** 表名 (字段 1 , 字段 2 , **[, 索引 1 , 索引 2 ,]**) **[表选项 1, 表选项 2 ,]**

字段设定形式:

字段名 类型 **[字段属性 1 字段属性 2]**

说明:

1. 字段名可以自己取;
2. 类型就是前面所学的数据类型: int, tinyint, float, double, char(6), varchar(25), text, datetime。。。。
3. 字段属性可以有多个(根据具体的需要), 相互之间直接空格隔开; 主要如下几个:

auto_increment: 只用于整数类型，让该字段的值自动获得一个增长值。通常用于做一个表的第一个字段的设定，并且通常还当做主键（primary key）；

primary key: 用于设定该字段为主键，此时该字段的值就可以“唯一确定”一行数据；

unique key: 设定该字段是“唯一的”，也就是不重复的。

not null: 用于设定该字段不能为空（null），如果没有设定，则默认是可为空的。

default XX 值: 用于设定该字段的默认值，此时如果 insert 没有给值的时候就使用该默认值
comment ‘字段说明文字’:

```
mysql> create table tab_shuxing(  
-> id int auto_increment primary key,  
-> user_name varchar(20) not null unique key,  
-> password varchar(48) not null,  
-> age tinyint default 18,  
-> email varchar(50) comment '电子邮箱'  
-> );  
Query OK, 0 rows affected (0.01 sec)
```

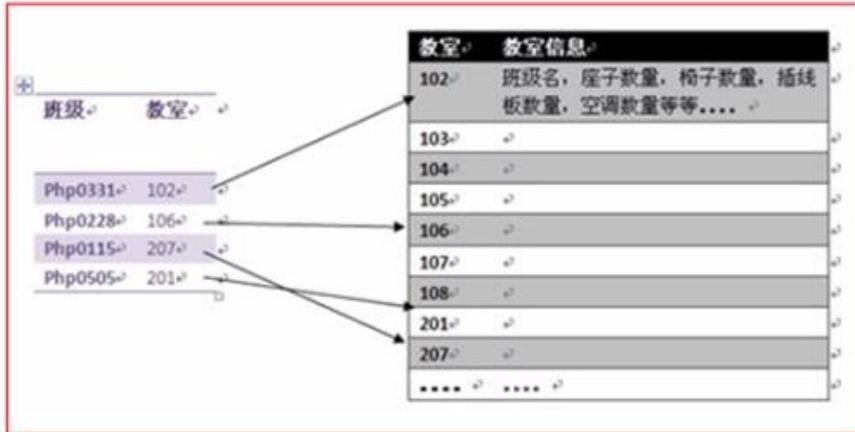
I

38 #演示字段属性的使用:

```
39 create table tab_shuxing(  
40 id int auto_increment primary key,  
41 user_name varchar(20) not null unique key,  
42 password varchar(48) not null comment ,  
43 age tinyint default 18,  
44 email varchar(50) comment '电子邮箱'  
45 );  
46  
47 insert into tab_shuxing (id, user_name, password, age, email)values  
48 (1, 'user1', '1234', 20, 'admin@qq.com');  
49 insert into tab_shuxing (id, user_name, password, age, email)values  
50 (null, 'user2', md5('1234'), null, 'ldh1@qq.com');
```

索引：

索引是系统内部自动维护的隐藏的“数据表”，它的作用是，可以极大地加快数据的查找速度！
这个隐藏的数据表，其中的数据是自动排好序的，其查找速度就是建立在这个基础上。



通常，所谓建立索引，其实是指定一个表的某个或某些字段作为“索引数据字段”就可以了，形式为：
索引类型(要建立索引的字段名)

索引类型有如下几个：

- 普通索引：形式：key (字段名)
含义：就是一个索引而已，没有其他作用，只能加快查找速度；
- 唯一索引：形式：unique key (字段名)
含义：是一个索引，而且还可以设定其字段的值不能重复（唯一性）；
- 主键索引：形式：primary key (字段名)
含义：是一个索引，而且，还具有区分该表中的任何一行数据的作用（其实也是唯一性）
它其实比唯一性索引多一点功能：唯一性可以为空 null，而主键不能为空；
- 全文索引：形式：fulltext (字段名)。
- 外键索引：形式：foreign key (字段名) references 其他表(对应其他表中的字段名)；

演示索引创建语法：

```
55 #演示索引创建语法：
56 create table tab_suoyin (
57     id int auto_increment ,
58     user_name varchar(20),
59     email varchar(50),
60     age int,                /*没有索引*/
61     key (email),           /*这是普通索引*/
62     primary key(id),      /*这就是主键索引*/
63     unique key(user_name) /*这就是唯一索引*/
64 );
```

此时，该表中如果以 id, user_name, 或 email 做条件进行查找，就会“很快”，而以 age 做条件就会“很慢”。

外键索引：

形式: foreign key (字段名) references 其他表(对应其他表中的字段名);

什么叫外键?

外键,就是指,设定的某个表 (tab1) 某个字段 (f1), 它的数据的值, 必须是在另一个表 (tab2) 中的某个字段 (f2) 中存在!

学生主信息表				班级表		
学生主键	学号	姓名	班级号	班级主键	班号	开班时间
1	Php130331003	郭靖	29	29	Php0228	2013-02-28
3	Php130331005	黄蓉	29	29	Php0228	2013-02-28
4	Php130425001	谢逊	30	30	Php0331	2013-03-31
5	Php130425003	张翠山	30	30	Php0331	2013-03-31
7	Php130425004	周芷若	30	30	Php0331	2013-03-31

注: 红色箭头指向 '班级号' 列, 标注为 '设置为外键: reference 班级表(班级主键)'; 红色箭头指向 '班级主键' 列, 标注为 '需设置为主键 (primary key) | 班级表'。

#演示外键索引:

```
create table banji(  
  id int auto_increment primary key,  
  banjihao varchar(10) unique key comment '班级号',  
  banzhuren varchar(10) comment '班主任',  
  open_date date comment '开班日期'  
);  
  
create table xuesheng (  
  stu_id int auto_increment primary key,  
  name varchar(10),  
  age tinyint,  
  banji_id int comment '班级id',  
  foreign key (banji_id) references banji(id)  
);
```

此时, 插入 xuesheng 表中的数据时, banji_id 字段的值, 就不可以随便插入了, 而是必须是 banji 表中的 id 字段所已有的数据值, 才可以插入。