SERVER&HTTP DAY01:

* 学习原因
  * WEB 前端招聘要求 - 熟悉或掌握一门服务器端技术者优先
    * 对于 WEB 前端,熟悉服务器端机制或流程,更好提高前端开发
* SERVER 安排
  * MySQL 一天 | PHP 语言 一天 | Http 协议 一天
  * 对服务器端技术的掌握 - 入门
* 基本内容
  * 服务器概念
    * 简单来说就是一个 PC 机
    * 商业中使用小型机|中型机|大型机|超级电脑
  * 对于开发人员
    * 硬件服务器 - PC 机
    * 软件服务器 - 中间件
  * WEB 架构
    * C/S 架构 - Client(客户端)/Server(服务器端)
      * 举例
        * QQ 聊天
      * 大型网游
      * 优点
        * 运行稳定
      * 对带宽要求相对低
      * 用户体验更好
      * 缺点
        * 更新过于复杂(客户端+服务器端)
      * 占硬盘空间
    * B/S 架构 - Browser(浏览器)/Server(服务器)
      * 举例
        * 轻应用 - PC 端
      * 流应用 - 移动端
      * 网页游戏
      * 优点
        * 更新简单(服务器端)
      * 实现更容易
      * 缺点
        * 体验相对差一点(越来越好)
      * 对带宽要求高
* XAMPP 软件
  * Apache - 用于运行 PHP 的服务
    * 出现错误
      * 日志 - Error: Apache shutdown unexpectedly.
      * 原因 - Apache 服务使用端口号被占用()
      * 解决
        * 打开"开始菜单"的"搜索程序和文件"
        * 输入"cmd" - 打开命令行
      * 在命令行中,输入命令
        * netstat -ano - 查看当前使用的端口号
      * 查看 80 端口号被占用的 PID 是多少
      * 在状态栏鼠标右键,选择"任务管理器"
      * 打开"任务管理器",切换到"进程"
      * 点击工具栏"查看"中的"选择列"
      * 将 PID 的选项勾选,点击确定
    * 如果 PID 也是为 4,说明 Windows 操作系统占用 80
      * 使用端口号
        * 默认端口号为 80(默认不写)
      * 修改 Apache 的端口号
        * 打开 Xampp Controle Panel 界面
        * 选择"Apache"的"config"按钮
      * 选择"[Browse]Apache",打开 Apache 的安装目录
        * 在该目录中,打开 conf 目录
        * 在该目录中,打开 httpd.conf 文件
          * 修改 listen 80 为 listen 8888
      * 进行保存
        * 重新启动 Apache 服务
      * 访问 Apache 服务
        * 打开浏览器,在地址栏中输入以下内容
          * http://127.0.0.1:8888
        * http://localhost:8888
      * 搭建本地 WEB 应用
        * 打开 Xampp Controle Panel 界面
        * 选择"Apache"的"config"按钮
      * 选择"[Browse]Apache",打开 Apache 的安装目录
        * 向上返回一级,Xampp 软件的安装目录
        * 打开 htdocs 目录,将该目录的内容删除
        * 新建名为"index.html"的 HTML 页面
        * 重新访问 Apache 的服务,主页信息
      * 配置顶级域名
        * 找到 C:/Windows/System32/drivers/etc 目录
        * 打开 hosts 文件

          127.0.0.1 www.jd.com
        * 另存为...,保存在桌面上
        * 将桌面的新文件,替换旧文件即可
        * 在浏览器地址栏输入配置的域名进行访问
  * Tomcat - 用于运行 Java 的服务
  * MySQL - 数据库产品
    * 基本内容
      * MySQL - 默认使用端口号 3306
      * 不要修改 MySQL 的默认端口号
    * 访问数据库
      * 图形化界面
        * 启动图形化界面的要求
      * Apache 和 MySQL 同时启动服务
      * 访问地址

        http://localhost:8888/phpmyadmin
      * 使用 Apache 的服务
      * php 提供的一种服务,访问 MySQL 数据库
      * 命令行方式

* 出现错误
* 报错 - "mysql"不是内部命令
* 原因 - mysql 的环境变量没有配置
* 解决
* 第一种方式
* 打开 Xampp Controle Panel 界面
* 选择"MySQL"中的"config"按钮,选择"[Browse]"
* 打开 MySQL 的安装目录,打开 bin 目录
* 鼠标双击"mysql.exe"文件
* 第二种方式
* 在命令行中输入以下命令
cd c:\xampp\mysql\bin
* 输入"mysql"命令即可

* 数据库
* 关系型数据库 - 以表格(行和列)为主
* Oracle - Oracle(甲骨文)
* 主要应用于企业级开发市场
* MySQL - Oracle(甲骨文)
* 主要应用于互联网开发市场
* SQL Server - 微软
* DB2 - 基本弃用
* Access - Office(弃用)
* 非关系型数据库
* 是一种运动 - 反关系型数据库
* 主流产品
* mangoDB - 以 JSON 格式为主
* ...

* MySQL 产品
* 最初是由瑞典公司 MySQL AB 推出的
* 原因 - 免费 开源 好用
* 网站架构 - LAMP(Linux+Apache+MySQL+PHP)
* 企业级架构 - Linux(AIX)+JavaEE+Oracle+Weblogic
* MySQL AB 公司后期被 SUN 公司收购
* SUN 公司明星产品 - Java
* SUN 公司被 Oracle 公司收购
* Oracle
* MySQL - 社区版(免费)和商业版(收费)
* Java
* 谷歌公司为了规避 Java 的版权问题 - Go 语言

* SQL 语言(语句) - 所有数据库的通用操作语言
* DDL - Data Define Language(数据定义语言)
* DCL - Data Controle Language(数据控制语言)
* DML - Data Manipulate Language(数据操作语言)
* DQL - Data Query Language(数据查询语言)
* 注意
* SQL 官方建议 - 所有关键字全部大写
* 每句 SQL 语句结束时,增加结束符";"

* 登录和退出 MySQL 数据库(命令行方式)
* 登录命令 - mysql -u 用户名 -p
* 退出命令 - exit

* DDL - 数据定义语言 - CREATE|ALTER|DROP
* 数据库 - 数据仓库
* 创建数据库
CREATE DATABASE 数据库名称;
* 设置编码
CREATE DATABASE 数据库名称 CHARACTER SET utf8;
* 增加判断 - 判断当前创建的数据库是否存在
CREATE DATABASE IF NOT EXISTS 数据库名称 CHARACTER SET utf8;
* 查看数据库
SHOW DATABASES;
* 切换数据库
USE 数据库名称;
* 修改数据库(编码规则)
ALTER DATABASE 数据库名称 CHARATER SET utf8;
* 删除数据库
DROP DATABASE 数据库名称;
DROP DATABASE IF EXISTS 数据库名称;
* 数据表 - 具有行(记录)和列(字段)的表格
* 数据类型
* 数值(Number)数据类型
* INT - 整型(整数)
* FLOAT/DOUBLE - 浮点型(小数)
* DECIMAL - 精确值(金额) - 整数
* 日期时间数据类型
* DATE - YYYY-MM-DD 标准日期格式
* DATETIME - YYYY-MM-DD HH:MM:SS
* TIMESTAMP - 时间戳(标识:唯一不可重复)
* 字符串数据类型
* CHAR - 长度固定的字符串
* VARCHAR - 长度可变的字符串
* 创建数据表
CREATE TABLE(
字段名称 1 数据类型,
字段名称 2 数据类型,
...
);
* 案例
```
CREATE TABLE jd_user(
user_id INT,
user_name VARCHAR(30),
user_gender VARCHAR(30),
user_email VARCHAR(30)
);
CREATE TABLE IF NOT EXISTS jd_order(
order_id INT PRIMARY KEY AUTO_INCREMENT,
order_number VARCHAR(30),
order_price DECIMAL,
```

order_time DATETIME
);
* 约束
* 主键约束 - PRIMARY KEY
* 主键自增 - AUTO_INCREMENT
* 删除数据表
DROP TABLE IF EXISTS 数据表名;
* 修改数据表
ALTER TABLE 数据表名 条件;
* 查看数据表
DESC 数据表名;
* DML - 数据操作语言 - INSERT|UPDATE|DELETE
* 新增
* INSERT INTO 数据表名 VALUES(字段值 1,字段值 2,...);
* VALUES()中的内容为该表的所有字段值
* INSERT INTO 数据表名(字段名 1,字段名 2,...) VALUES(字段值 1,字段值 2,...)
* VALUES()中的内容与表名()的内容一一对应
* 案例
* INSERT INTO jd_user VALUES(0,'张无忌','男','zwj@qq.com');
* INSERT INTO jd_order VALUES (NULL,'1234567890',6288.09,'2015-12-03 15:31:15');
* INSERT INTO jd_order VALUES (NULL,'2346546234',4288,2015-12-03 15:31:15);
* 练习
* 向 jd_user 表新增记录
(2,赵敏,女,zm@qq.com)
INSERT INTO jd_user VALUES
(2,'赵敏','女','zm@qq.com');
* 修改(更新)
* UPDATE 数据表名 SET 字段名=字段值;
* 注意 - 默认修改所有记录的当前字段
* UPDATE 数据表名 SET 字段名=字段值 WHERE 字段名=字段值;
* UPDATE 数据表名 SET 字段名 1=字段值 1,...,字段名 n=字段值 n WHERE 条件;
* 案例
* UPDATE jd_user SET user_name='金毛狮王';
* UPDATE jd_user SET user_name='张无忌' WHERE user_id=0;
* 练习
* 修改 user_id 为 2 记录,将 user_name 改为"赵敏"
UPDATE jd_user SET user_name="赵敏" WHERE user_id=2;
* 修改 user_id 为 1 记录,将 user_name 改为"周芷若",user_gender 改为'女'
UPDATE jd_user SET user_name='周芷若',user_gender='女' WHERE user_id=1;
* 删除
* DELETE FROM 数据表名;
* 注意 - 默认删除当前表中所有数据
* DELETE FROM 数据表名 WHERE 条件;
* 删除符合条件的所有数据内容
* 案例
* 删除 jd_user 表中 user_id 为 2 这条记录
DELETE FROM jd_user WHERE user_id=2;
* 练习
* 删除 jd_order 表中所有的记录
DELETE FROM jd_order;
* 问题
* 一旦执行 DELETE 语句的话,将数据内容删除(无法恢复)
* 注意
* 物理删除 - 真正执行 DELETE 语句
* 逻辑删除 - 从逻辑上讲删除,并没有真正删除
* 并不执行 DELETE 语句
* 解决方案
* 为数据表增加一个字段(state)
* 该字段专门用于表示当前这条记录的状态
* 删除 - 0 - false
* 正常 - 1 - true
* DQL - 数据查询语言 - QUERY
* SELECT * FROM 数据表名;
* 特点 - 默认查询表中所有记录(字段)
* SELECT 字段名 1,字段名 2,... FROM 数据表名;
* 自定义结果中的字段名称
* SELECT 字段名 1,字段名 2,... FROM 数据表名 WHERE 条件;
* 案例
* select * from jd_user;
* select user_name,user_gender,user_email from jd_user;
* select user_id,user_name,user_gender,
user_email from jd_user;
* select * from jd_user where user_id=0;
* 练习
* 查询 jd_user 表 user_gender 为'女'记录的
user_name,user_gender,user_email
SELECT user_name,user_gender,
user_email FROM jd_user WHERE user_gender='女';
* SQL 错误
* 数据库存在 - ERROR 1007:Can't create database 'day1203'; database exists
* 数据表存在 - ERROR 1050 (42S01):
Table 'jd_order' already exists
* 数据库不存在 - ERROR 1008 (HY000):
Can't drop database 'day1203'; database doesn't exists

SERVER&HTTP DAY02:

* PHP 语言
  * 基本内容
    * LAMP - Linux+Apache+MySQL+PHP
    * PHP 文件的扩展名为".php"
    * 如何运行 PHP 页面
      * 将创建好的 PHP 页面拷贝到 Xampp 安装目录中的 htdocs 目录中
      * 启动 Apache 服务,在浏览器中进行访问
    * PHP 允许编写 HTML 代码+PHP 代码
    * PHP 与 JavaScript 的区别
      * PHP - 运行在服务器端的脚本语言
      * JavaScript - 运行在客户端页面的脚本语言
  * 第一个 PHP 页面
    * 是以"<?php"开始,是以"?>"结束的
  * 变量和常量
    * 变量 - 值允许改变的
      * $变量名 = 值;
      * PHP 中的变量与 JavaScript 的变量类似,都是弱类型
    * 常量 - 值不允许变的
      * const 常量名 = 值;
      * define(常量名,值);
      * PHP 中的常量一旦定义,不能重新赋值
        * 如果重新赋值,在浏览器中的页面报错(Parse error)
  * 数据类型
    * 四种标量类型
      * Integer - 整型类型
      * Float/Double - 浮点型类型
      * String - 字符串类型
        * 单引号 - 直接定义字符串
        * 并不识别变量
        * 效率高
      * 双引号 - 如果包含变量名,自动替换为变量值
        * 可以识别变量
        * 效率低
      * Boolean - 布尔类型
    * 两种复合类型
      * Array - 数组
        * 直接量方式 - [ele1,ele2,ele3,..]
      * 内置对象方式

```
array(
    key => value,
    key => value
)
```

      * Object - 对象
        * 定义类(Class)

```
class 类名{
  $属性名 = 值;
  function 方法名(){}
}
```

        * 基于类创建对象

```
$对象名 = new 类名;
```

        * 调用对象的属性或方法

```
$对象名->属性名;
$对象名->方法名();
```

    * 两种特殊类型
      * Resource - 资源
      * Null - 空
        * 释放资源
    * Null 和""的区别
      * null - 不存在
      * "" - 存在,但值为空
  * 运算符
    * 字符串的连接符为"."
  * 循环结构
    * while - 先判断后执行
    * do..while - 先执行后判断
    * for
    * foreach - 类似于 JavaScript 中的 forin
  * 分支结构
    * if...else if...else
    * switch...case
  * 关键字
    * break - 停止循环
    * continue - 停止本次循环
  * PHP 预定义
    * 变量
      * $_GET - 客户端的 GET 请求方式
        * 定义 HTML 页面中的表单的 method 属性值为 get
      * 定义 HTML 页面中的表单元素时,定义 name 属性值
      * 在 PHP 页面中使用$_GET[name 属性值]
      * $_POST - 客户端的 POST 请求方式
      * 定义 HTML 页面中的表单的 method 属性值为 post
      * 定义 HTML 页面中的表单元素时,定义 name 属性值
      * 在 PHP 页面中使用$_POST[name 属性值]
      * $_REQUEST - 客户端的 GET|POST 请求方式
      * $_COOKIE - 客户端的 COOKIE 请求
      * $_FILES - 专门处理文件上传
    * 函数 - MySQL 数据库的扩展
      * mysql - 原生 MySQL API
      * mysqli - MySQL 增强版扩展
* PHP 连接 MySQL
  * 面向过程风格
    * DML(增删改)
      * 与 MySQL 数据库建立连接

```
mysqli_connect(host,username,passwd,dbname,port)
```

      * host - MySQL 数据库所在的计算机地址
        * 本机地址 - 127.0.0.1 或 localhost

* username - 登录 MySQL 数据库的用户名
* passwd - 登录 MySQL 数据库的用户密码
* 如果为空的话,是字符串空''
* dbname - 数据库名称
* port - MySQL 数据库的端口号
* 默认值为 3306
* 定义 SQL 语句
* 使用字符串类型定义即可
* 注意(测试)
* 定义 SQL 语句
* 输出打印
* 将其复制到命令行进行执行(一定成功)
* 向 MySQL 数据库发送 SQL 语句
mysqli_query(link,query)
* link - 与 MySQL 数据库建立的连接对象
* query - 执行的 SQL 语句
该方法的返回值,表示 SQL 语句是否执行成功
* true - 表示 SQL 语句执行成功
* false - 表示 SQL 语句执行失败
* 关闭与 MySQL 数据库的连接
mysqli_close(link)
* link - 与 MySQL 数据库建立的连接对象
* DQL(查询)
* 与 MySQL 数据库建立连接
mysqli_connect(host,username,passwd,dbname,port)
* host - MySQL 数据库所在的计算机地址
* 本机地址 - 127.0.0.1 或 localhost
* username - 登录 MySQL 数据库的用户名
* passwd - 登录 MySQL 数据库的用户密码
* 如果为空的话,是字符串空''
* dbname - 数据库名称
* port - MySQL 数据库的端口号
* 默认值为 3306
* 定义 SQL 语句
* 使用字符串类型定义即可
* 注意(测试)
* 定义 SQL 语句
* 输出打印
* 将其复制到命令行进行执行(一定成功)
* 向 MySQL 数据库发送 SQL 语句
mysqli_query(link,query)
* link - 与 MySQL 数据库建立的连接对象
* query - 执行的 SQL 语句
该方法的返回值,表示 SQL 语句是否执行成功
* mysqli_result 对象 - 表示 SQL 语句执行成功
* false - 表示 SQL 语句执行失败
* 关闭与 MySQL 数据库的连接
mysqli_close(link)
* link - 与 MySQL 数据库建立的连接对象 2015/12/4
* 解析结果集(mysqli_result)对象
* 属性
* field_count - 返回字段数量
* num_rows - 返回记录数量
* 方法
* mysqli_fetch_array(result,resulttype) - 将结果集对象转换为数组
* result - 结果集对象
* resulttype - 转换的数组类型
* MYSQLI_BOTH - 既关联数组还索引数组
* MYSQLI_NUM - 索引数组
* MYSQLI_ASSOC - 关联数组
* mysqli_fetch_assoc() - 将结果集对象转换为关联数组
* 解析结果集对象的所有记录
while($arr = mysqli_fetch_array($result)){
得到查询后的所有记录
}
* 解决中文乱码问题
* 在向 MySQL 数据库发送 SQL 语句之前
mysqli_query($conn,'SET NAMES utf8');
* 面向对象风格
* 创建 mysqli 对象(与 MySQL 数据库建立连接)
$mysqli =
new mysqli(host,username,passwd,dbname,port);
* 定义 SQL 语句
* 向 MySQL 数据库发送 SQL 语句
$result = $mysqli->query(sql)
* 返回结果集对象
* 解析结果集对象
$arr = $result->fetch_array();
* 关闭连接
$mysqli->close();
* 练习
* 使用 PHP 向 MySQL 数据库(jd)的表(jd_user),user_id 为 2 的记录进行删除
* 查询 jd_user 表,条件 user_id 为 1 的记录,输出 user_name 值
* 扩展内容
* 面向对象与面向过程
* 面向对象
* 基于原型的面向对象 - JavaScript
* 基于类的面向对象 - PHP/Java
* 万物且对象 - 属性或方法
* 一个人的一生
* 将这个人当做对象
* 属性
* 年龄 * 身高 * 长相
* 方法
* 睡觉 * 吃饭
* 面向过程
* 出生 * 成长 * 结婚 * 工作
* 去世

案例

Null 与空的区别

```
var n = null;       // 在内存没有空间
console.log(n);
var k = "";
console.log(k);   // 在内存具有空间
```

```
<?php
    // 定义一个常量 str = zhangwuji
    const str = "zhangwuji";
    // 进行测试
    var_dump(str);
    // 修改常量 str = zhouzhiruo
    //str = "zhouzhiruo";
    // 进行测试
    //var_dump(str);
    // 定义一个常量 username = zhouzhiruo
    define("username","zhouzhiruo");
    // 进行测试
    var_dump(username);
?>
<?php
    // 1. 直接量方式定义数组
    $arr1 = [1,2,3,4,5];
    // 进行测试
    var_dump($arr1);
    // 2. 定义索引数组 - 索引值 0,1,2,3,4,...
    $arr2 = array(
        '0' => 'zhangwuji',
        '1' => 'zhouzhiruo'
    );
    // 进行测试
    var_dump($arr2);
    // 3. 定义关联数组
    $arr3 = array(
        'p1' => 'zhangwuji',
        'p2' => 'zhouzhiruo'
    );
    // 进行测试
    var_dump($arr3);
?>
<?php
    /*JavaScript 定义对象的三种方式
        * 直接量方式
        {
            属性名:属性值,
            方法名:function(){}
        }
      * Object 方式 - new Object();
      * 构造器方式
        function 构造器(){}
        var 对象 = new 构造器();
      JavaScript 的面向对象是基于原型(Prototype)的
      PHP 的面向对象是基于类(Class)的*/
    // 1. 定义类
    class Hero{
        function sayMe(){
            echo "this is zhangwuji.";
        }
    }
    // 2. 创建对象
    $hero = new Hero;
    // 对象如何调用方法?
    $hero->sayMe();
?>
<?php
    /* 1. while 循环
    $i = 0;
    while($i < 0){
        echo $i;
        $i++;
    }*/
    /* 2. do...while 循环
    $i = 0;
    do{
        echo $i;
        $i++;
    }while($i < 0);
    */
    /* 3. for 循环
    for($i = 0; $i < 10; $i++){
        echo $i;
    }*/
    /*
      4. foreach 循环
        foreach(array as value){
            * 遍历后的 value,其实是 PHP 的变量
            * 得到的 value,遍历后的结果(value)
        }*/
    $arr = [1,2,3,4,5];
    /*foreach($arr as $value){
        echo $value;
    }*/
    foreach($arr as $key => $value){
        echo "[$key] => $value";
    }
?>
<?php
    /* 1. PHP 与 MySQL 之间建立连接
       PHP 的数据库扩展 mysqli 提供相关方法
```

```
        mysqli_connect(host,username,passwd,dbname,port)
        * host - MySQL 数据库所在计算机的 IP 地址
          * 本地的 MySQL - 127.0.0.1 或 localhost
        * username - 登录 MySQL 数据库的用户名
          * 默认用户名为"root"
        * passwd - 登录 MySQL 数据库的用户密码
          * 默认为空,但是必须占位(空的字符串)
        * dbname - 操作的 MySQL 数据库的数据库名称
        * port - MySQL 数据库使用的端口号
          * 默认为 3306
        该方法具有返回值,返回一个连接对象*/
    $conn = mysqli_connect('127.0.0.1','root','','jd','3306');
    // 进行测试
    //var_dump($conn);
    /*2. 定义 SQL 语句
        如何测试 SQL 语句编写是正确的?
        * 在 PHP 页面中定义 SQL 语句
        * 将其打印出来 var_dump()函数
        * 将 SQL 语句放在命令行中进行运行*/
    $sql = "INSERT INTO jd_user VALUES
                    (2,'zhaomin','female','zm@qq.com')";
    // 进行输出
    //var_dump($sql);
    /*3. PHP 向 MySQL 数据库发送 SQL 语句
        mysqli_query(link,query)方法
        * link - 与 MySQL 数据库建立的连接对象
        * query - 向 MySQL 数据库发送的 SQL 语句
        该方法具有返回值
        * DML - 返回 Boolean 值
          * true - 表示 SQL 语句执行成功
          * false - 表示 SQL 语句执行失败
        * DQL*/
    $result = mysqli_query($conn,$sql);
    // 进行测试
    var_dump($result);

    /*4. 关闭 PHP 与 MySQL 之间的连接
        mysqli_close(link)
        * link - 与 MySQL 数据库建立的连接对象*/
    mysqli_close($conn);
?>
<?php
    // 1. 与 MySQL 建立连接
    $conn = mysqli_connect('127.0.0.1','root','','jd','3306');
    // 2. 定义 SQL 语句
    $sql = 'SELECT * FROM jd_user';
    /*3. 向 MySQL 发送 SQL 语句
        mysqli_query()方法的返回值
        * false - 表示 SQL 语句执行失败
        * SQL 语句执行成功,返回结果集(mysqli_result)对象
          * 查询回来的结果(二维表格)被封装在结果集对象中
          * mysqli_result 结果集对象
            * field_count - 得到当前数据表的字段数量
            * num_rows - 得到当前数据表的记录(行)数*/
    mysqli_query($conn,'SET NAMES utf8');
    $result = mysqli_query($conn,$sql);
    // 进行解析
    // 面向对象方式
    //var_dump($result->$num_rows);
    // 面向过程方式
    //var_dump(mysqli_num_rows($result));
    /*mysqli_fetch_array(result,resulttype)
        * 作用 - 将结果集转换为数组
        * 参数
          * result - 结果集对象
          * resulttype - 设置当前转换后的数组类型
            * MYSQLI_BOTH - 默认值
            * MYSQLI_ASSOC - 表示转换为关联数组
            * MYSQLI_NUM - 表示转换为索引数组*/
    //$arr = mysqli_fetch_array($result,MYSQLI_ASSOC);
    //var_dump($arr);
    /*foreach($arr as $key => $value){
        echo "$key : $value ";
    }*/
    //$arr = mysqli_fetch_assoc($result);
    while($arr = mysqli_fetch_assoc($result)){
    echo $arr['user_id'].$arr['user_name'].
                $arr['user_gender'].$arr['user_email'];
    }
    //var_dump($arr);
    // 4. 关闭连接
    mysqli_close($conn);
?>
<?php
    /*1. 创建 mysqli 对象(通过构造器)
        $mysqli =
      new mysqli(host,username,passwd,dbname,port);*/
    $mysqli = new mysqli('127.0.0.1','root','','jd','3306');
    // 2. 定义 SQL 语句
    $sql = 'SELECT * FROM jd_user WHERE user_id=0';
    /* 3. 向 MySQL 数据库发送 SQL 语句
        通过 mysqli 对象调用 query(query)方法
        * query - 发送的 SQL 语句*/
    $mysqli->query('SET NAMES utf8');
    $result = $mysqli->query($sql);
    // 4. 解析结果集
    $arr = $result->fetch_array();
    echo $arr['user_name'];// 进行输出
    $mysqli->close();// 5. 关闭连接
?>
```

SERVER&HTTP DAY03:

* HTTP 协议
  * 简单来讲,就是一个网络协议(客户端与服务器端)
  * 分类
    * 请求协议　　* 响应协议
  * HTTP 与 HTTPS
    * HTTP - 普通协议　　* HTTPS - 加密协议
* URL - 访问地址
  * URL - 统一资源定位符　　* URI - 统一资源标识符
  * 注意
    * 是 URL,一定是 URI;是 URI,不一定是 URL.
  * URL
    * 案例
* https://www.baidu.com:80/　　* http://127.0.0.1:8888
    * 完整 URL

      网络协议:IP 地址:端口号/路径;参数?数据#锚点
* 请求(Request)协议
  * GET 请求方式
    * 请求行
      * http 协议的版本信息 1.1
      * 请求地址 - URL?key=value&key=value
      * 请求方式 - GET
      * 状态码 - 200
    * 请求头
      * 格式　　* key : value

      * key : value,value,value,...
      * 选项
        * Accept - 服务器端允许接收的 MIME 类型

      * Accept-Encoding - 是否压缩数据
        * 使用 gzip 压缩格式(Linux 系统的压缩方式)

      * Accept-Language - 表示接受的语言
        * 中文: zh_CN,zh

      * Connection - 表示是否保持连接
        * keep-alive - 表示保持连接

      * Host - 服务器端的地址
      * Referer - 表示当前请求是来源于哪里
        * 防盗链功能

      * User-Agent - 用户浏览器的相关信息

Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/42.0.2311.90 Safari/537.36

Mozilla/5.0 (Windows NT 6.1; WOW64; rv:42.0) Gecko/20100101 Firefox/42.0

      * Cookie - 服务器端向客户端的缓存数据(自动携带)

      * Content-Length - 请求体的字符个数
    * 请求体 - 空
    * 注意
      * GET 方式的请求,请求数据增加在 URL?key=value
  * POST 请求方式
    * 请求行
      * http 协议的版本 1.1
      * 请求地址 - URL
        * URL 后没有请求数据
      * 请求方式 - POST
      * 状态码 - 200 OK
    * 请求头
      * Cache-Control - 表示是否包含缓存内容
        * max-age=0 - 设置最大缓存周期为 0
      * Content-Length - 表示请求数据的字符个数
      * Content-Type - 表示表单的 enctype 属性值
        * application/x-www-form-urlencoded

      * POST 方式是无法向服务器端发送数据的
    * 请求体
      * 请求数据
        * 格式

        key : value
  * GET 与 POST 方式的区别
    * GET
      * 请求行中请求类型 - GET　　* 请求体为空
      * 请求数据增加在 URL(请求地址)后
        * 不安全

      * 大小限制
      * 可能出现中文乱码
    * POST - 实际开发使用 POST 的原因
      * 请求行中请求类型 - POST
      * 请求体 - 请求数据
      * 请求数据并没有增加在 URL 后
        * 更安全

      * 没有限制
      * 可能出现中文乱码
* 响应(Response)协议
  * 响应行
    * http 协议的版本 1.1
    * 状态码 - 200 OK
  * 响应头
    * 格式
      * key : value　　* key : value,value,...
    * 选项
      * Content-Length - 响应数据的字符个数

      * Content-Type - 设置响应数据的 MIME 类型和编码格式
        * html - text/html;charset=utf-8

      * Date - 返回当前响应的时间
        * 标准英文格式 - Mon, 07 Dec 2015 07:40:49 GMT

      * Keep-Alive - 设置保持连接的时间
        * timeout=5 - 设置超时时间

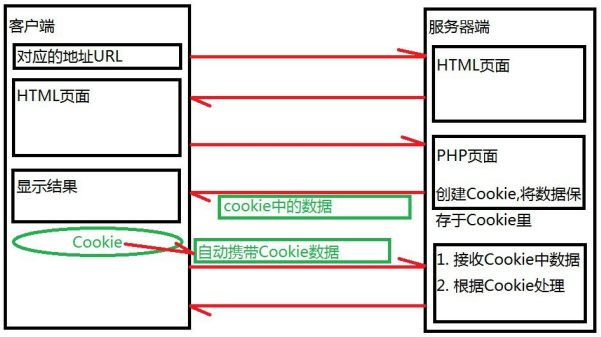
      * max=100　　- 设置最大存活时间
      * Server - 返回服务器端的相关信息
      * X-Powered-By - 服务器端所使用的语言
      * Accept-Ranges - 设置当前响应数据的单位
        * bytes - 表示字节

* Last-Modified - 返回服务器端最后一次修改的时间
* 响应体
* 所有响应数据
* 设置响应头
* 在服务器端(PHP)
* 使用 header()函数
header('key:value');
* HTML 页面设置响应头信息
* 使用 HTML 页面的<meta>元元素
<meta http-equiv="" content="">
* http-equiv - 设置响应头的 key
* content - 设置响应头的 value
* 特殊响应头
* 设置客户端是否允许缓存
* Cache-Control - no-cache(不缓存)
* Pragma - no-cache(不缓存)
* Expires - 0(不缓存)
* 注意 * 兼容各种浏览器的缓存机制
* 设置重定向
* Status - 302(重定向)
* Location - 设置重定向到的地址
* cookie 缓存控制
* 性能优化
* 尽量减少对外部资源的引用
* 页面中的图片
* 页面中的 JavaScript 脚本
* 页面中的 CSS 样式
* 页面中的视频或音频
* 尽量减少创建连接的次数
* 原则 - 客户端能完成的功能,就不麻烦服务器端
* 优化服务器端和数据库
* 尽量减小请求和响应的数据内容
* 扩展名称
* COOKIE 与 SESSION
* Cookie - 客户端浏览器的缓存(存储在硬盘)
* Cookie 中的数据是以明文(未加密)存储的
* Cookie 有限制
* 单个 Cookie 的大小有限制(4KB)
* 每个网站最多只能存储 200 多 Cookie 文件
* 安全性并不高(Flash Cookie)
* Session - 服务器端的缓存(存储在硬盘)
* 状态码
* 1xx - 获取信息 * 2xx - 请求成功
* 200 - OK
* 3xx - 重定向
* 302 - 重定向,配合 location
* 304 - (服务器端)没有修改 - 访问缓存
* 305 - 使用代理
* 4xx - 客户端错误
* 400 - 请求失败 * 403 - 被拒绝
* 404 - 网页未找到 * 405 - 请求类型不被允许
* 5xx - 服务器端错误
* 500 - 服务器端内部错误 * 502 - 路径错误
* 504 - 请求超时 * 505 - http 版本不支持
* 请求类型(方式)
* GET * POST * HEAD * PUT * TRACE
* OPTIONS * DELETE
* 面试题
* GET 和 POST 两种请求类型是最常用的
* 请求类型至少具有 7 种
* 标准请求 API
* 新增 - PUT * 修改 - POST
* 删除 - DELETE * 查询 - GET
* MIME 类型
* 用于表示当前文件的格式
* 举例
* html - text/html
* xhtml - application/xhtml+xml
* javascript - text/javascript,application/javascript
* css - text/css
* text - text/plain
* xml - text/xml,application/xml
* jpg - image/jpeg
* png - image/png
* mp4 - video/mp4
* ogv - video/ogg
* mp3 - audio/mpeg
* 浏览器内核
* IE : IE6/7/8 IE/9/10/11 JScript
* 其他浏览器 : Webkit
* chrome : V8 引擎 - Node.js
* firefox : Gecko
* safari : Webkit
* 众多国内浏览器 - 不建议使用
* 360 浏览器 - IE 内核|Chrome
* 猎豹浏览器 - Chrome
* 搜狗浏览器 - IE|Webkit
* 百度浏览器 - 号称自主内核 V5
* QQ 浏览器 - 号称自主内核 V5|X5
* 遨游浏览器 - 号称自主内核
* 前端陷阱
* 看到的效果,逻辑不一定是这样
* 语法正确、逻辑正确,结果不一定正确(浏览器解析问题)
* 中文乱码
* 客户端与服务器端之间的交互
* 客户端页面中文乱码问题
* 定义<meta charset="utf-8" />元元素
* 设置 HTML 页面文件的编码也是 UTF-8
* 设置浏览器的编码格式也是 UTF-8
* 客户端向服务器端发送请求数据时

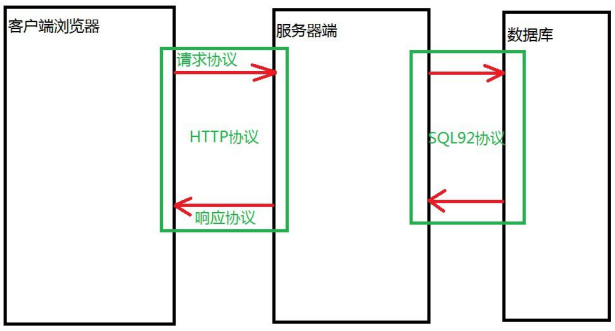
* 先将中文转换为 Unicode 码
* 在将 Unicode 码转换为中文
* 服务器端向客户端响应数据时
* 设置响应头 "Content-Type"为"charset=utf-8"
* 服务器端与数据库之间的交互
* mysqli_query($conn,"SET NAMES utf8");
* 中文编码
* UTF-8 编码 - Unicode
* UTF-16|UTF-32
* GBK * GB2312
* 普通方式加密
* 定义一个规则 - 只有我们自己知道
* 静态资源与动态资源
* 静态资源
* HTML CSS JAVASCRIPT
* 动态资源
* PHP - PHP + HTML
* JSP(Java Server Page) - Java + HTML
* ASP(.net) * Node.js

```
<title>GET 方式的请求协议</title>
    <form action="01.php" method="get">
        用户名:<input type="text" id="user" name="user" value="
请输入你的用户名"><br>
        密码:<input type="text" name="pwd"><br>
        <input type="submit" value="登录">
    </form>
    <script>
        var user = document.getElementById("user");
        console.log(user.value);
    </script>
<?php
        // 1. 接收客户端的请求数据
        $user = $_GET['user'];
        $pwd = $_GET['pwd'];
        // 2. 向客户端响应数据
        echo "$user : $pwd";
?>
```

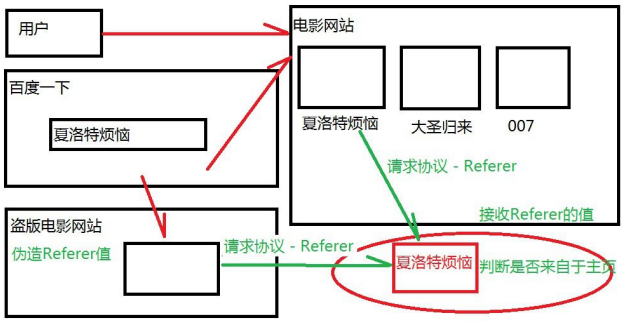
Cookie 在 Web 应用的作用



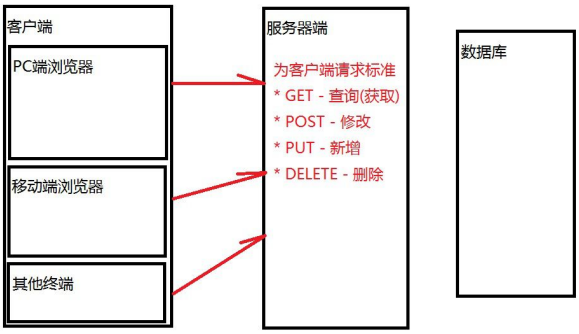
HTTP 协议的含义及作用



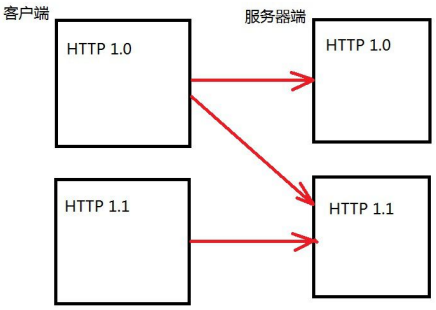
Referer 防盗链流程图



标准式 API



客户端与服务器端的 HTTP 版本对应图



客户端与服务器端交互的中文乱码问题