视频点播项目

1. 视频点播系统认识

搭建视频共享点播服务器，可以让所有人通过浏览器访问服务器，实现视频的上传查看，以及管理并播放的功能。

2. 实现目标

主要是完成服务器端的程序业务功能的实现以及前端访问界面`html`的编写，能够支持客户端浏览器针对服务器上的所有视频进行操作。

3. 服务端程序负责功能

- 针对客户端上传的视频文件以及封面图片进行备份存储。
- 针对客户端上传的视频完成增删改查功能
- 支持客户端浏览器进行视频的观看功能

4. 服务端功能模块划分

- 数据管理模块：负责针对客户端上传的视频信息进行管理。
- 网络通信模块：搭建网络通信服务器，实现与客户端通信。
- 业务处理模块：针对客户端的各个请求进行对应业务处理并响应结果。
- 前端界面模块：完成前端浏览器上视频共享点播的各个`html`页面，在页面中支持增删改查以及观看功能。

5. 环境搭建-`Gcc`升级7.3版本

```
sudo yum install centos-release-scl-rh centos-release-scl
sudo yum install devtoolset-7-gcc  devtoolset-7-gcc-c++
source /opt/rh/devtoolset-7/enable
echo "source /opt/rh/devtoolset-7/enable" >> ~/.bashrc
```

```
[san@192 ~]$ g++ -v
Using built-in specs.
COLLECT_GCC=g++
COLLECT_LTO_WRAPPER=/opt/rh/devtoolset-7/root/usr/libexec/gcc/x86_64-redhat-linux/7/lto-
wrapper
Target: x86_64-redhat-linux
Configured with: ../configure --enable-bootstrap --enable-languages=c,c++,fortran,lto --
prefix=/opt/rh/devtoolset-7/root/usr --mandir=/opt/rh/devtoolset-7/root/usr/share/man --
infodir=/opt/rh/devtoolset-7/root/usr/share/info --with-
bugurl=http://bugzilla.redhat.com/bugzilla --enable-shared --enable-threads=posix --enable-
checking=release --enable-multilib --with-system-zlib --enable-__cxa_atexit --disable-
libunwind-exceptions --enable-gnu-unique-object --enable-linker-build-id --with-gcc-major-
version-only --enable-plugin --with-linker-hash-style=gnu --enable-initfini-array --with-
default-libstdcxx-abi=gcc4-compatible --with-isl=/builddir/build/BUILD/gcc-7.3.1-
20180303/obj-x86_64-redhat-linux/isl-install --enable-libmpx --enable-gnu-indirect-function
--with-tune=generic --with-arch_32=i686 --build=x86_64-redhat-linux
Thread model: posix
gcc version 7.3.1 20180303 (Red Hat 7.3.1-5) (GCC)
```

6. 环境搭建-安装Jsoncpp库

```
sudo yum install epel-release
sudo yum install jsoncpp-devel
[san@localhost ~]$ ls /usr/include/jsoncpp/json/
assertions.h  config.h    forwards.h  reader.h  version.h
autolink.h    features.h  json.h      value.h   writer.h
#注意，centos版本不同有可能安装的jsoncpp版本不同，安装的头文件位置也就可能不同了。
```

7. 环境搭建-下载httplib库

GitHub链接

```
git clone https://github.com/yhirose/cpp-httplib.git
```

8. 环境搭建-Mysql数据库及开发包安装

安装配置博客：https://zhuanlan.zhihu.com/p/49046496

```
sudo yum install -y mariadb
sudo yum install -y mariadb-server
sudo yum install -y mariadb-devel
sudo vim /etc/my.cnf.d/client.cnf
sudo vim /etc/my.cnf.d/mysql-clients.cnf
sudo vim /etc/my.cnf.d/server.cnf
```

```
# /etc/my.cnf.d/client.cnf
# These two groups are read by the client library
# Use it for options that affect all clients, but not the server
#
[client]
# 新增下边一行配置，设置客户端默认字符集为utf8
default-character-set = utf8

# This group is not read by mysql client library,
# If you use the same .cnf file for MySQL and MariaDB,
# use it for MariaDB-only client options
[client-mariadb]
```

```
# /etc/my.cnf.d/mysql-clients.cnf
# These groups are read by MariaDB command-line tools
# Use it for options that affect only one utility
#
[mysql]
# 新增配置
default-character-set = utf8

[mysql_upgrade]

[mysqladmin]

[mysqlbinlog]

[mysqlcheck]

[mysqldump]

[mysqlimport]

[mysqlshow]

[mysqlslap]
```

```
# /etc/my.cnf.d/server.cnf
# These groups are read by MariaDB server.
# Use it for options that only the server (but not clients) should see
#
# See the examples of server my.cnf files in /usr/share/mysql/
#

# this is read by the standalone daemon and embedded servers
[server]

# this is only for the mysqld standalone daemon
[mysqld]
# 新增以下配置
collation-server = utf8_general_ci
init-connect = 'SET NAMES utf8'
```

```
character-set-server = utf8
sql-mode = TRADITIONAL

# this is only for embedded server
[embedded]

# This group is only read by MariaDB-5.5 servers.
# If you use the same .cnf file for MariaDB of different versions,
# use this group for options that older servers don't understand
[mysqld-5.5]

# These two groups are only read by MariaDB servers, not by MySQL.
# If you use the same .cnf file for MySQL and MariaDB,
# you can put MariaDB-only options here
[mariadb]

[mariadb-5.5]
```

```
[san@localhost src]$ sudo systemctl start mariadb
[san@localhost src]$ mysql -uroot
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 2
Server version: 5.5.68-MariaDB MariaDB Server

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> show variables like "%character%";
+--------------------------+----------------------------+
| Variable_name            | Value                      |
+--------------------------+----------------------------+
| character_set_client     | utf8                       |
| character_set_connection | utf8                       |
| character_set_database   | utf8                       |
| character_set_filesystem | binary                     |
| character_set_results    | utf8                       |
| character_set_server     | utf8                       |
| character_set_system     | utf8                       |
| character_sets_dir       | /usr/share/mysql/charsets/ |
+--------------------------+----------------------------+
8 rows in set (0.00 sec)

MariaDB [(none)]>
```

9. 第三方库认识-`json`认识

`json`是一种数据交换格式，采用完全独立于编程语言的文本格式来存储和表示数据。

例如：小明同学的学生信息

```
char name = "小明";
int age = 18;
float score[3] = {88.5, 99, 58};
则json这种数据交换格式是将这多种数据对象组织成为一个字符串：
[
    {
        "姓名" : "小明",
        "年龄" : 18,
        "成绩" : [88.5, 99, 58]
    },
    {
        "姓名" : "小黑",
        "年龄" : 18,
        "成绩" : [88.5, 99, 58]
    }
]
```

`json`数据类型：对象，数组，字符串，数字

对象：使用花括号`{}`括起来的表示一个对象。

数组：使用中括号`[]`括起来的表示一个数组。

字符串：使用常规双引号`""`括起来的表示一个字符串

数字：包括整形和浮点型，直接使用。

10. 第三方库认识-`jsoncpp`认识

`jsoncpp`库用于实现`json`格式的序列化和反序列化，完成将多个数据对象组织成为`json`格式字符串，以及将`json`格式字符串解析得到多个数据对象的功能。

这其中主要借助三个类以及其对应的少量成员函数完成：

```
//Json数据对象类
class Json::Value{
    Value &operator=(const Value &other); //Value重载了[]和=，因此所有的赋值和获取数据都可以通过
    Value& operator[](const std::string& key);//简单的方式完成 val["姓名"] = "小明";
    Value& operator[](const char* key);
    Value removeMember(const char* key);//移除元素
    const Value& operator[](ArrayIndex index) const; //val["成绩"][0]
    Value& append(const Value& value);//添加数组元素val["成绩"].append(88);
    ArrayIndex size() const;//获取数组元素个数 val["成绩"].size();
    std::string asString() const;//转string      string name = val["name"].asString();
    const char* asCString() const;//转char*   char *name = val["name"].asCString();
    Int asInt() const;//转int                int age = val["age"].asInt();
    float asFloat() const;//转float
    bool asBool() const;//转 bool
};

//json序列化类，低版本用这个更简单
class JSON_API Writer {
```

```
  virtual std::string write(const Value& root) = 0;
}
class JSON_API FastWriter : public Writer {
  virtual std::string write(const Value& root);
}
class JSON_API StyledWriter : public Writer {
  virtual std::string write(const Value& root);
}
//json序列化类，高版本推荐，如果用低版本的接口可能会有警告
class JSON_API StreamWriter {
    virtual int write(Value const& root, std::ostream* sout) = 0;
}
class JSON_API StreamWriterBuilder : public StreamWriter::Factory {
    virtual StreamWriter* newStreamWriter() const;
}

//json反序列化类，低版本用起来更简单
class JSON_API Reader {
    bool parse(const std::string& document, Value& root, bool collectComments = true);
}
//json反序列化类，高版本更推荐
class JSON_API CharReader {
    virtual bool parse(char const* beginDoc, char const* endDoc,
                       Value* root, std::string* errs) = 0;
}
class JSON_API CharReaderBuilder : public CharReader::Factory {
    virtual CharReader* newCharReader() const;
}
```

11. 第三方库认识-`jsoncpp`实现序列化

```
#include <iostream>
#include <sstream>
#include <string>
#include <memory>
#include <jsoncpp/json/json.h>

int main()
{
    const char *name = "小明";
    int age = 19;
    float score[] = {77.5, 88, 99.5};

    Json::Value val;
    val["姓名"] = name;
    val["年龄"] = 19;
    val["成绩"].append(score[0]);
    val["成绩"].append(score[1]);
    val["成绩"].append(score[2]);

    Json::StreamWriterBuilder swb;
    std::unique_ptr<Json::StreamWriter> sw(swb.newStreamWriter());
```

```
    std::stringstream ss;
    int ret = sw->write(val, &ss);
    if (ret != 0) {
        std::cout << "write failed!\n";
        return -1;
    }
    std::cout << ss.str() << std::endl;
    return 0;
}
```

```
g++ json_example1.cpp -o json_example1 -ljsoncpp
```

12. 第三方库认识-jsoncpp实现反序列化

```
int main()
{
    std::string str = R"({"姓名":"小明", "年龄":18, "成绩":[76.5, 55, 88]})";
    Json::Value root;
    Json::CharReaderBuilder crb;
    std::unique_ptr<Json::CharReader> cr(crb.newCharReader());

    std::string err;
    cr->parse(str.c_str(), str.c_str() + str.size(), &root, &err);

    std::cout << root["姓名"].asString() << std::endl;
    std::cout << root["年龄"].asInt() << std::endl;
    int sz = root["成绩"].size();
    for (int i = 0; i < sz; i++) {
        std::cout << root["成绩"][i].asFloat() << std::endl;
    }
    for (auto it = root["成绩"].begin(); it != root["成绩"].end(); it++){
        std::cout << it->asFloat() << std::endl;
    }
    return 0;
}
```

```
g++ json_example2.cpp -o json_example2 -ljsoncpp
```

13. 第三方库认识-Mysql数据库API认识

咱们在课堂上已经学习了Mysql数据库的SQL学习，所以这里主要介绍Mysql的C语言API接口。

Mysql是C/S模式，其实咱们编写代码访问数据库就是实现了一个Mysql客户端，实现咱们的专有功能。

```
//Mysql操作句柄初始化
MYSQL *mysql_init(MYSQL *mysql);
//参数为空则动态申请句柄空间进行初始化
```

```
//失败返回NULL

//连接mysql服务器
MYSQL *mysql_real_connect(MYSQL *mysql,  const char *host,  const char *user,
                          const char *passwd,const char *db,   unsigned int port,
                          const char *unix_socket,   unsigned long client_flag);
//mysql--初始化完成的句柄
//host---连接的mysql服务器的地址
//user---连接的服务器的用户名
//passwd-连接的服务器的密码
//db ----默认选择的数据库名称
//port---连接的服务器的端口:  默认0是3306端口
//unix_socket---通信管道文件或者socket文件，通常置NULL
//client_flag---客户端标志位，通常置0
//返回值：成功返回句柄，失败返回NULL

//设置当前客户端的字符集
int mysql_set_character_set(MYSQL *mysql, const char *csname)
//mysql--初始化完成的句柄
//csname--字符集名称，通常："utf8"
//返回值：成功返回0，  失败返回非0;

//选择操作的数据库
int  mysql_select_db(MYSQL *mysql,  const char *db)
//mysql--初始化完成的句柄
//db-----要切换选择的数据库名称
//返回值：成功返回0，  失败返回非0;

//执行sql语句
int  mysql_query(MYSQL *mysql,  const char *stmt_str)
//mysql--初始化完成的句柄
//stmt_str--要执行的sql语句
//返回值：成功返回0，  失败返回非0;

//保存查询结果到本地
MYSQL_RES *mysql_store_result(MYSQL *mysql)
//mysql--初始化完成的句柄
//返回值：成功返回结果集的指针，  失败返回NULL;

//获取结果集中的行数与列数
uint64_t mysql_num_rows(MYSQL_RES *result);
//result--保存到本地的结果集地址
//返回值：结果集中数据的条数;
unsigned int mysql_num_fields(MYSQL_RES *result)
//result--保存到本地的结果集地址
//返回值：结果集中每一条数据的列数;

//遍历结果集
MYSQL_ROW mysql_fetch_row(MYSQL_RES *result)
//result--保存到本地的结果集地址
//返回值：实际上是一个char **的指针，将每一条数据做成了字符串指针数组  row[0]-第0列 row[1]-第1列
//并且这个接口会保存当前读取结果位置，每次获取的都是下一条数据
```

```
//释放结果集
void mysql_free_result(MYSQL_RES *result)
//result--保存到本地的结果集地址
//返回值：void

//关闭数据库客户端连接，销毁句柄：
void mysql_close(MYSQL *mysql)

//获取mysql接口执行错误原因
const char *mysql_error(MYSQL *mysql)
```

14. 第三方库认识-`Mysql-API`实现数据的增删改查操作

```
create database if not exists test_db;
use test_db;
create table if not exists test_tb(
    id int primary key auto_increment,
    age int,
    name varchar(32),
    score decimal(4, 2)
);
```

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <mysql/mysql.h>

#define HOST "127.0.0.1"
#define USER "root"
#define PASSWD ""
#define DBNAME "test_db"

void add(MYSQL *mysql) {
    char *sql = "insert into test_tb values(null, 18, '张三', 88.88), (null, 17, '李四',
77);";
    int ret = mysql_query(mysql, sql);
    if (ret != 0) {
        printf("mysql query error:%s\n", mysql_error(mysql));
        return ;
    }
    return ;
}
void mod(MYSQL *mysql) {
    char *sql = "update test_tb set age=34 where name='张三';";
    int ret = mysql_query(mysql, sql);
    if (ret != 0) {
        printf("mysql query error:%s\n", mysql_error(mysql));
        return ;
    }
```

```
    return ;
}
void del(MYSQL *mysql) {
    char *sql = "delete from test_tb where name='张三';";
    int ret = mysql_query(mysql, sql);
    if (ret != 0) {
        printf("mysql query error:%s\n", mysql_error(mysql));
        return ;
    }
    return ;
}
void get(MYSQL *mysql) {
    char *sql = "select * from test_tb;";
    int ret = mysql_query(mysql, sql);
    if (ret != 0) {
        printf("mysql query error:%s\n", mysql_error(mysql));
        return ;
    }
    MYSQL_RES *res =  mysql_store_result(mysql);
    if (res == NULL) {
        printf("mysql store result error:%s\n", mysql_error(mysql));
        return ;
    }
    int row = mysql_num_rows(res);
    int col = mysql_num_fields(res);
    printf("%10s%10s%10s%10s\n", "ID", "年龄", "姓名", "成绩");
    for (int i = 0; i < row; i++) {
        MYSQL_ROW row_data = mysql_fetch_row(res);
        for (int i = 0; i < col; i++) {
            printf("%10s", row_data[i]);
        }
        printf("\n");
    }
    mysql_free_result(res);
    return ;
}
int main()
{
    MYSQL *mysql = mysql_init(NULL);
    if (mysql == NULL) {
        printf("init mysql handle failed!\n");
        return -1;
    }
    if (mysql_real_connect(mysql, HOST, USER, PASSWD, DBNAME, 0, NULL, 0) == NULL) {
        printf("mysql connect error:%s\n", mysql_error(mysql));
        return -1;
    }
    mysql_set_character_set(mysql, "utf8");
    //mysql_select_db(mysql, DBNAME);

    add(mysql);
    get(mysql);
    mod(mysql);
```

```
    get(mysql);
    del(mysql);
    get(mysql);

    mysql_close(mysql);
    return 0;
}
```

```
[san@localhost example]$ make
gcc mysql.c -o mysql_test -L/usr/lib64/mysql -lmysqlclient
[san@localhost example]$ ./mysql_test
        ID        年龄    姓名     成绩
         5        18      张三      88.88
         6        17      李四      77.00
        ID        年龄     姓名     成绩
         5        34      张三      88.88
         6        17      李四      77.00
        ID        年龄     姓名     成绩
         6        17      李四      77.00
```

15. 第三方库认识-`httplib`库

`httplib`库，一个`C++11`单文件头的跨平台`HTTP/HTTPS`库。安装起来非常容易。只需包含`httplib.h`在你的代码中即可。

`httplib`库实际上是用于搭建一个简单的`http`服务器或者客户端的库，这种第三方网络库，可以让我们免去搭建服务器或客户端的时间，把更多的精力投入到具体的业务处理中，提高开发效率。

```
namespace httplib{
    struct MultipartFormData {
        std::string name;
        std::string content;
        std::string filename;
        std::string content_type;
    };
    using MultipartFormDataItems = std::vector<MultipartFormData>;
    struct Request {
        std::string method;//存放请求方法
        std::string path;//存放请求资源路径
        Headers headers;//存放头部字段的键值对map
        std::string body;//存放请求正文
        // for server
        std::string version;//存放协议版本
        Params params;//存放url中查询字符串 key=val&key=val的 键值对map
        MultipartFormDataMap files;//存放文件上传时，正文中的文件信息
        Ranges ranges;
        bool has_header(const char *key) const;//判断是否有某个头部字段
        std::string get_header_value(const char *key, size_t id = 0) const;//获取头部字段值
        void set_header(const char *key, const char *val);//设置头部字段
        bool has_file(const char *key) const;//文件上传中判断是否有某个文件的信息
        MultipartFormData get_file_value(const char *key) const;//获取指定的文件信息
    };
```

```
    struct Response {
        std::string version;//存放协议版本
        int status = -1;//存放响应状态码
        std::string reason;
        Headers headers;//存放响应头部字段键值对的map
        std::string body;//存放响应正文
        std::string location; // Redirect location重定向位置
        void set_header(const char *key, const char *val);//添加头部字段到headers中
        void set_content(const std::string &s, const char *content_type);//添加正文到body中
        void set_redirect(const std::string &url, int status = 302);//设置全套的重定向信息
    };
    class Server {
        using Handler = std::function<void(const Request &, Response &)>;//函数指针类型
        using Handlers = std::vector<std::pair<std::regex, Handler>>;//存放请求-处理函数映射
        std::function<TaskQueue *(void)> new_task_queue;//线程池
        Server &Get(const std::string &pattern, Handler handler);//添加指定GET方法的处理映射
        Server &Post(const std::string &pattern, Handler handler);
        Server &Put(const std::string &pattern, Handler handler);
        Server &Patch(const std::string &pattern, Handler handler);
        Server &Delete(const std::string &pattern, Handler handler);
        Server &Options(const std::string &pattern, Handler handler);
        bool listen(const char *host, int port, int socket_flags = 0);//开始服务器监听
        bool set_mount_point(const std::string &mount_point, const std::string &dir,
                             Headers headers = Headers());//设置http服务器静态资源根目录
    };
}
```

16. 第三方库认识-`httplib`库搭建简单服务器

```
/* ./www/index.html  */
<html>
    <head>
        <meta content="text/html; charset=utf-8" http-equiv="content-type" />
    </head>
    <body>
        <h1>Hello Bit</h1>
        <form action="/multipart" method="post" enctype="multipart/form-data">
            <input type="file"  name="file1">
            <input type="submit" value="上传">
        </form>
    </body>
</html>
```

```
#include "httplib.h"

int main(void)
{
    using namespace httplib;
    Server svr;
    auto ret = svr.set_mount_point("/", "./www");
    svr.Get("/hi", [](const Request& req, Response& res) {
        res.set_content("Hello World!", "text/plain");
```

```
    });
    svr.Get(R"(/numbers/(\d+))", [&](const Request& req, Response& res) {
        auto numbers = req.matches[1];
        res.set_content(numbers, "text/plain");
    });
    svr.Post("/multipart", [&](const auto& req, auto& res) {
        auto size = req.files.size();
        auto ret = req.has_file("file1");
        const auto& file = req.get_file_value("file1");
        std::cout << file.filename << std::endl;
        std::cout <<  file.content_type << std::endl;
        std::cout <<  file.content << std::endl;
    });
    svr.listen("0.0.0.0", 9090);
    return 0;
}
```

```
g++ -std=c++14 http_server.cpp -o http_server -lpthread
```

```
[san@localhost http]$ tree ./
./
├── httplib.h
├── http_server
├── http_server.cpp
└── www
    └── index.html
```

17. 服务端工具类实现-文件实用工具类设计

在视频点播系统中因为涉及到文件上传，需要对上传的文件进行备份存储，因此首先设计封装文件操作类，这个类封装完毕之后，则在任意模块中对文件进行操作时都将变的简单化

- 获取文件大小（属性）
- 判断文件是否存在
- 向文件写入数据
- 从文件读取数据
- 针对目录文件多一个创建目录

C++17中filesystem手册

```
/*util.hpp*/
#ifndef __MY_UTIL__
#define __MY_UTIL__
#include <iostream>
#include <fstream>
#include <sstream>
#include <string>
#include <unistd.h>
#include <sys/stat.h>
namespace aod{
```

```
class FileUtil{
    private:
        std::string _name;
    public:
        FileUtil(const std::string &name);
        bool Exists();
        size_t FileSize();
        bool GetContent(std::string *content);
        bool SetContent(const std::strint content);
        bool CreateDirectory();
};
}
#endif
```

18. 服务端工具类实现-Json实用工具类设计

- 实现序列化
- 实现反序列化

```
/*util.hpp*/
#ifndef __MY_UTIL__
#define __MY_UTIL__
#include <iostream>
#include <sstream>
#include <string>
#include <memory>
#include <jsoncpp/json/json.h>
namespace aod{
class JsonUtil{
    public:
        static bool Serialize(const Json::Value &root, std::string *str);
        static bool UnSerialize(const std::string &str, Json::Value *root);
};
}
```

19. 服务端数据管理模块-视频数据表的设计

在视频共享点播系统中，视频数据和图片数据都存储在文件中，而我们需要在数据库中管理用户上传的每个视频信息。

只是完成一个简单的视频信息表。

- 视频ID
- 视频名称
- 视频描述信息
- 视频文件的url路径（加上相对根目录实际上就是实际存储路径）
- 视频封面图片的URL路径（只是链接，加上相对根目录才是实际的存储路径）

```
drop database if exists aod_system;
create database if not exists aod_system;
use aod_system;
create table if not exists tb_video(
    id int primary key auto_increment comment '视频ID',
    name varchar(32) comment '视频名称',
    info text comment '视频描述',
    video varchar(256) comment '视频文件url，加上静态资源根目录就是实际存储路径',
    image varchar(256) comment '封面图片文件url，加上静态资源根目录就是实际存储路径'
);
```

20. 服务端数据管理模块-数据管理类设计

数据管理模块负责统一对于数据库中数据的增删改查管理，其他所有模块要进行数据的操作都通过数据管理模块完成。

然而，数据库中有可能存在很多张表，每张表中数据又有不同，要进行的数据操也各不相同，因此咱们将数据的操作分摊到每一张表上，为每一张表中的数据操作都设计一个类，通过类实例化的对象来访问这张数据库表中的数据，这样的话当我们要访问哪张表的时候，使用哪个类实例化的对象即可。

- 新增
- 修改
- 删除
- 查询所有
- 查询单个
- 模糊匹配

视频信息在接口之间的 传递因为字段数量可能很多，因此使用 `Json::Value` 对象进行传递

```
#ifndef __MY_DATA__
#define __MY_DATA__
#include "util.hpp"
#include <mutex>
#include <cstdlib>
#include <mysql/mysql.h>

namespace aod {
    static MYSQL *MysqlInit();
    static void MysqlDestroy(MYSQL *mysql);
    static bool MysqlQuery(MYSQL *mysql, const std::string &sql);

    class TableVideo {
        private:
            MYSQL *_mysql;//一个对象就是一个客户端，管理一张表
            std::mutex _mutex;//防备操作对象在多线程中使用存在的线程安全 问题
        public:
            TableVideo();//完成mysql句柄初始化
            ~TableVideo();//释放msyql操作句柄
            bool Insert(const Json::Value &video);//新增-传入视频信息
            bool Update(int video_id, const Json::Value &video);//修改-传入视频id，和信息
```

```
            bool Delete(const int video_id);//删除-传入视频ID
            bool SelectAll(Json::Value *videos);//查询所有--输出所有视频信息
            bool SelectOne(int video_id, Json::Value *video);//查询单个-输入视频id,输出信息
            bool SelectLike(const std::string &key, Json::Value *videos);//模糊匹配-输入名称关键
字，输出视频信息
    };
}
#endif
```

21. 服务端业务处理模块-网络通信接口设计

`restful` 认识：

- `REST` 是 `Representational State Transfer` 的缩写，一个架构符合 `REST` 原则，就称它为 `RESTful` 架构
- `RESTful` 架构可以充分的利用 `HTTP` 协议的各种功能，是 `HTTP` 协议的最佳实践，正文通常采用 `JSON` 格式
- `RESTful API` 是一种软件架构风格、设计风格，可以让软件更加清晰，更简洁，更有层次，可维护性更好.

`restful` 使用五种 `HTTP` 方法，对应 `CRUD(增删改查)` 操作

- `GET` 表示查询获取
- `POST` 对应新增
- `PUT` 对应修改
- `DELETE` 对应删除

获取所有视频信息

```
请求:
GET /video HTTP/1.1


响应:
HTTP/1.1 200 OK
[
    {
        "info": "好电影",
        "id": 1,
        "image": "/img/thumbs/mysql.png",
        "name": "Mysql注意事项",
        "video": "/video/movie.mp4",
    },
    {
        "info": "好电影",
        "id": 2,
        "image": "/img/thumbs/linux.png",
        "name": "Linux注意事项",
        "video": "/video/movie.mp4",
    }
]
```

搜索指定关键字名称视频信息

```
请求:
GET /video?search="Mysql" HTTP/1.1

响应:
HTTP/1.1 200 OK

[
    {
        "info": "好电影",
        "id": 1,
        "image": "/img/thumbs/mysql.png",
        "name": "Mysql注意事项",
        "video": "/video/movie.mp4",
    }
]
```

获取指定视频信息

```
请求:
GET /video/1 HTTP/1.1

响应:
HTTP/1.1 200 OK

[
    {
        "info": "好电影",
        "id": 1,
        "image": "/img/thumbs/mysql.png",
        "name": "Mysql注意事项",
        "video": "/video/movie.mp4",
    }
]
```

删除指定视频信息

```
请求:
DELETE /video/1 HTTP/1.1

响应:
HTTP/1.1 200 OK
```

修改指定视频信息

```
请求:
PUT /video/1 HTTP/1.1

{
    "info": "这是一个非常好的教学视频，深入浅出，引人深思",
    "id": 1,
    "image": "/img/thumbs/mysql.png",
    "name": "Mysql注意事项",
    "video": "/video/movie.mp4",
}
响应:
HTTP/1.1 200 OK
```

上传视频信息以及文件

因为上传视频信息的时候，会携带有视频文件和封面图片的文件上传，而这些文件数据都是二进制的，用`json`不好传输，因此在这里使用传统的`http`上传文件请求格式，而并没有使用`restful`风格。

```
请求:
POST /video HTTP/1.1
Content-Type: multipart/form-data; boundary=----WebKitFormBoundarydsrFiETIzKETHWkn

------WebKitFormBoundarydsrFiETIzKETHWkn
Content-Disposition: form-data; name="name"

Xhsell连接事项，也就是视频名称
------WebKitFormBoundarydsrFiETIzKETHWkn
Content-Disposition: form-data; name="info"

一部非常好看的视频的描述信息
------WebKitFormBoundarydsrFiETIzKETHWkn
Content-Disposition: form-data; name="image"; filename="image.jpg"
Content-Type: text/plain

image封面图片数据
------WebKitFormBoundarydsrFiETIzKETHWkn
Content-Disposition: form-data; name="video"; filename="video.mp4"
Content-Type: text/plain

video视频数据
------WebKitFormBoundarydsrFiETIzKETHWkn
Content-Disposition: form-data; name="submit"


------WebKitFormBoundarydsrFiETIzKETHWkn--

响应:
HTTP/1.1 303 See Other
Location: "/"
```

22. 服务端业务处理模块-业务处理模块类的设计

业务处理模块负责与客户端进行网络通信，接收客户端的请求，然后根据请求信息，明确客户端端用户的意图，进行业务处理，并进行对应的结果响应。

在视频共享点播系统中，业务处理主要包含两大功能：1、网络通信功能的实现；2、业务功能处理的实现

其中网络通信功能的实现咱们借助`httplib`库即可方便的搭建`http`服务器完成。这也是咱们将网络通信模块与业务处理模块合并在一起完成的原因。

而业务处理模块所要完成的业务功能主要有：

- 客户端的视频数据和信息上传
- 客户端的视频列表展示（视频信息查询）
- 客户端的视频观看请求（视频数据的获取）
- 客户端的视频其他管理（修改，删除）功能

```
#ifndef __MY_SERVERE__
#define __MY_SERVERE__
#include "httplib.h"
#include "data.hpp"

namespace aod {
    #define WWW_ROOT "./www"
    #define VIDEO_ROOT "/video/"
    #define IMAGE_ROOT "/image/"
    //因为httplib基于多线程，因此数据管理对象需要在多线程中访问，为了便于访问定义全局变量
    TableVideo *table_video = NULL;
    //这里为了更加功能模块划分清晰一些，不使用lamda表达式完成，否则所有的功能实现集中到一个函数中太过庞大
    class Server {
        private:
            int _port;//服务器的 监听端口
            httplib::Server _srv;//用于搭建http服务器
        private:
            //对应的业务处理接口
            static void Insert(const httplib::Request &req, httplib::Response &rsp);
            static void Update(const httplib::Request &req, httplib::Response &rsp);
            static void Delete(const httplib::Request &req, httplib::Response &rsp);
            static void GetOne(const httplib::Request &req, httplib::Response &rsp);
            static void GetAll(const httplib::Request &req, httplib::Response &rsp);
        public:
            Server(int port):_port(port);
            bool RunModule();//建立请求与处理函数的映射关系，设置静态资源根目录，启动服务器，
    };
}
```

23. 服务端业务处理模块-最终合并调试

```
#include "server.hpp"

int main()
{
    aod::Server server(9090);
    server.RunModule();
    return 0;
}
```

服务器的功能测试借助一个工具`postman`完成。

postman下载地址：https://www.postman.com/downloads/

24. 前端界面模块实现-`HTML`基础认识

`HTML`代码是由标签构成的。我们可以理解不同的标签代表不同的控件元素，前端浏览器拿到`html`代码之后，根据标签之间的关系进行解析，得到一棵`DOM (Document Object Mode - 文档对象模型的缩写)`树。

然后根据`DOM`树渲染出不同的控件元素，得到我们所看到的页面。

标签之间具有不同的关系：

- 父子关系
- 兄弟关系

```
<html>
    <head>
        <title>第一个页面</title>
    </head>
    <body id="myId">
        hello world
    </body>
</html>
```

- 标签名 (body) 放到 < > 中
- 大部分标签成对出现. `<body>` 为开始标签, `</body>` 为结束标签.
- 少数标签只有开始标签, 称为 "单标签".
- 开始标签和结束标签之间, 写的是标签的内容. (hello)
- 开始标签中可能会带有 "属性". id 属性相当于给这个标签设置了一个唯一的标识符(身份证号码).

25. 前端界面模块实现-`HTML`基础标签

标题标签: `h1-h6`

```
<h1>hello</h1>
<h2>hello</h2>
<h3>hello</h3>
<h4>hello</h4>
<h5>hello</h5>
<h6>hello</h6>
```

段落标签: `p`

```
<p>段落，
在html中一般的回车并不起作用，会被解释成为一个空格<br/>但是br不一样，br标签的作用就是换行。
</p>
```

把一段比较长的文本粘贴到 `html` 中, 会发现并没有分成段落. 在 `html` 中使用 `<p>` 标签括起一个段落进行换行。当然也可以在段落内使用 `<br/>` 标签进行换行操作。

图片标签: `img`

```
<img src="./rose.jpg" alt="显示失败时的信息" title="图片提示信息" width="150px" height="150px"
border="5px">
```

超链接标签: `a`

```
<a href="http://www.baidu.com" target="_blank">点击这里打开新标签访问百度</a>
```

表格标签: `table`

- table 标签: 表示整个表格
- tr: 表示表格的一行
- td: 表示一个单元格
- th: 表示表头单元格. 会居中加粗
- thead: 表格的头部区域(注意和 th 区分, 范围是比 th 要大的)
- tbody: 表格得到主体区域.

```
<table align="center" border="1" cellpadding="1" cellspacing="0" width="200" height="20">
    <tr>
        <th>菜名</th>
        <th>单价</th>
        <th>折扣</th>
    </tr>
    <tr>
        <td align="center">红烧茄子</td>
        <td align="center">￥18</td>
        <td align="center">8.8折</td>
    </tr>
    <tr>
        <!--colspan合并列,  rowspan合并行-->
        <td colspan="3" align="right">总价: ￥18</td>
    </tr>
</table>
```

列表标签: `ol & ul & dl`

主要使用来布局的. 整齐好看.

- 无序列表[重要] `ul` `li`, .
- 有序列表[用的不多] `ol` `li`
- 自定义列表[重要] `dl`(总标签) `dt`(小标题) `dd`(围绕标题来说明) 上面有个小标题, 下面有几个围绕着标题来展开的.

```
<ul>
    <li>ul/li是无序列表</li>
    <li>ul/li是无序列表</li>
</ul>
<ol>
    <li>ol/li是有序列表</li>
    <li>ol/li是有序列表</li>
</ol>
<dl>
    <dt>dl/dt是小标题</dt>
    <dd>dl/dd是围绕标题的描述</dd>
    <dd>dl/dd是围绕标题的描述</dd>
</dl>
```

表单标签：`form`

表单是让用户输入信息的重要途径.

分成两个部分:

- 表单域: 包含表单元素的区域. 重点是 form 标签.
- 表单控件: 输入框 , 提交按钮等. 重点是 input 标签.

form标签认识

被`form`标签括起来的部分称之为表单域，当点击表单提交按钮时，将会将表单域中所有表单控件数据提交给指定服务器。

- `action`：表单动作，或者说当点击表单提交时的请求链接
- `method`：请求方法
- `enctype`：编码类型，其中`multipart/form-data`常用于文件上传

```
<form action="/upload" method="post" enctype="multipart/form-data">
    <input type="text" placeholder="input标签默认是文本框"> <br/>
    <input type="file" value="file是文件选择按钮框"><br/>
    <input type="submit" value="submit是提交按钮">点击这里就会向服务器提交表单域中的表单数据<br/>
</form>
```

input标签的认识

```
<input type="text" placeholder="input标签默认是文本框"> <br/>
<input type="password" placeholder="type属性为password是密码框"> <br/>
<input type="radio" name="sex">type属性为radio是单选框,name属性相同则默认为同一组-男 <br/>
<input type="radio" name="sex" checked="checked">type属性为radio是单选框-女<br/>
<input type="checkbox"> checkbox是复选框-吃饭 <br/>
<input type="checkbox"> checkbox是复选框-睡觉 <br/>
<input type="checkbox"> checkbox是复选框-打游戏<br/>
<input type="checkbox" id="testid">
<label for="testid">label标签for属性与对应的输入框id对应起来，这时候点击文字也能选中</label><br/>
<input type="button" value="button是普通按钮" onclick="alert('alert是提示框调用函数')"><br/>
<input type="submit" value="submit是提交按钮">点击这里就会向服务器提交表单域中的表单数据<br/>
<input type="file" value="file是文件选择按钮框"><br/>
<input type="reset" value="reset是清空按钮，会清空表单域的所有数据"><br>
```

下拉菜单标签：`select`

- option 中定义 selected="selected" 表示默认选中.

```
<select>
    <option selected="selected">--请选择年份--</option>
    <option>1990</option>
    <option>1991</option>
    <option>1992</option>
</select>
```

文本域标签：`textarea`

```
<textarea name="文本域标签" id="" cols="30" rows="10" placeholder="textarea是文本域标签">
</textarea>
```

无语义标签: div & span

div 标签, division 的缩写, 含义是 分割

span 标签, 含义是跨度

说白了就是两个盒子. 常用于网页布局

- div 是独占一行的, 是一个大盒子.
- span 不独占一行, 是一个小盒子.

```
<div>div是个大盒子独占一行</div>
<span>span是个小盒子并不独占一行</span>
<span>span是个小盒子并不独占一行</span>
```

`html5` 语义化标签

div 没有语义. 对于搜索引擎来说没有意义. 为了让搜索引擎能够更好的识别和分析页面(SEO 优化), HTML 引入了更多的 "语义化" 标签. 但是这些标签其实本质上都和 div 一样(对于前端开发来说). 然而对于搜索引擎来说, 见到 header 和 article 这种标签就会重点进行解析, 而 footer 这种标签就可以适当忽略.

- header: 头部

- nav: 导航
- article: 内容
- section: 某个区域
- aside: 侧边栏
- footer: 尾部

```
<header>头部容器标签</header>
<nav>导航容器标签</nav>
<article>内容容器标签</article>
<section>某个区域的容器标签</section>
<aside>侧边栏容器标签</aside>
<footer>尾部容器标签</footer>
```

多媒体标签：`video & audio`

video标签：视频 audio标签：音频

```
<video src="https://www.runoob.com/try/demo_source/movie.mp4" controls="controls"
type="video/mp4">video是视频控件标签</video>
    <audio src="https://www.runoob.com/try/demo_source/horse.mp3" controls="controls"
type="audio/mp3">audio是音频控件标签</audio>
```

常见标签总体演示：

```
<!DOCTYPE html>
<!-- html是一个标签化语言，由标签构成，大多标签成对出现，但是也存在单标签 -->
<!-- html的根标签，html是标签名，使用<>括起来，标签中有时候也会有一些其他属性 -->
<!-- 标签之间也存在父子与兄弟等关系，例如body标签就属于html标签的子标签 -->
<html lang="en">
<head><!--head标签是头部标签，内部通常编写页面的属性-->
    <meta charset="UTF-8">
    <title>html扫盲</title>
</head>
<body id="app"> <!--页面上显示的内容-->
    <h1>标题1</h1>
    <h2>标题2</h2>
    <h3>标题3</h3>
    <h4>标题4</h4>
    <h5>标题5</h5>
    <h6>标题6</h6>
    <hr/><!--这是一个分割线-->

    <p>段落，
        在html中一般的回车并不起作用，会被解释成为一个空格<br/>但是br不一样，br标签的作用就是换行。
    </p>
    <hr/>

    在html中的文字也有各种不同的标签属性，这些属性可以用在任何有文字显示的位置使用标签将文字包含
    <p><b>比如b标签就是加粗</b></p>
    <p><i>比如i标签就是斜体</i></p>
    <p><s>比如s标签就是删除线</s></p>
    <p><u>比如u就是下划线</u></p>
```

```
    <hr/>

    图片标签img，可以在页面上渲染显示图片文件,不过要注意图片文件与当前html文件的相对路径关系<br/>
    <img src="./rose.jpg" alt="显示失败时的信息" title="图片提示信息" width="150px"
height="150px" border="5px">
    <hr/>

    a超链接标签，连接内容可以是文字也可以是图片
    <a href="http://www.baidu.com" target="_blank">点击这里就能打开一个新标签跳转访问百度</a>
    <hr/>

    table表格相关的标签（行列）
    <table align="center" border="1" cellpadding="1" cellspacing="0" width="200"
height="20">
        <tr>
            <th>菜名</th>
            <th>单价</th>
            <th>折扣</th>
        </tr>
        <tr>
            <td align="center">红烧茄子</td>
            <td align="center">￥18</td>
            <td align="center">8.8折</td>
        </tr>
        <tr>
            <td colspan="3" align="right">总价: ￥18</td>
        </tr>
    </table>
    <hr/>

    列表标签
    <ul>
        <li>ul/li是无序列表</li>
        <li>ul/li是无序列表</li>
    </ul>
    <ol>
        <li>ol/li是有序列表</li>
        <li>ol/li是有序列表</li>
    </ol>
    <dl>
        <dt>dl/dt是小标题</dt>
        <dd>dl/dd是围绕标题的描述</dd>
        <dd>dl/dd是围绕标题的描述</dd>
    </dl>
    <hr/>

    form表单标签
    <form action="对应的url请求路径" method="POST"> <!--form是表单域，表示表单域内的元素数据是要提交的
表单数据-->
        <input type="text" placeholder="input标签默认是文本框"> <br/>
        <input type="password" placeholder="type属性为password是密码框"> <br/>
        <input type="radio" name="sex">type属性为radio是单选框,name属性相同则默认为同一组-男 <br/>
        <input type="radio" name="sex" checked="checked">type属性为radio是单选框-女<br/>
```

```
        <input type="checkbox"> checkbox是复选框-吃饭 <br/>
        <input type="checkbox"> checkbox是复选框-睡觉 <br/>
        <input type="checkbox"> checkbox是复选框-打游戏<br/>
        <input type="checkbox" id="testid"> <label for="testid">label标签for属性与对应的输入框id
对应起来，这时候点击文字也能选中</label><br/>
        <input type="button" value="button是普通按钮" onclick="alert('alert是提示框调用函数')">
<br/>
        <input type="submit" value="submit是提交按钮"><br/>
        <input type="file" value="file是文件选择按钮框"><br/>
        <input type="reset" value="reset是清空按钮，会清空表单域的所有数据"><br>
    </form>
    <hr>

    select下拉菜单
    <select>
        <option selected="selected">--请选择年份--</option>
        <option>1990</option>
        <option>1991</option>
        <option>1992</option>
    </select>
    <hr>

    textarea文本域
    <textarea name="文本域标签" id="" cols="30" rows="10" placeholder="textarea是文本域标签">
</textarea>
    <hr>

    无语义标签div和span,通常用于进行页面布局
    <div>div是个大盒子独占一行</div>
    <span>span是个小盒子并不独占一行</span>
    <span>span是个小盒子并不独占一行</span>
    <hr>

    html5的语义化标签，便于页面容器比如div和span这种无语义标签的分析优化
    <header>头部容器标签</header>
    <nav>导航容器标签</nav>
    <article>内容容器标签</article>
    <section>某个区域的容器标签</section>
    <aside>侧边栏容器标签</aside>
    <footer>尾部容器标签</footer>
    <hr>

    多媒体标签
    <video src="https://www.runoob.com/try/demo_source/movie.mp4" controls="controls"
type="video/mp4">video是视频控件标签</video>
    <audio src="https://www.runoob.com/try/demo_source/horse.mp3" controls="controls"
type="audio/mp3">audio是语音或者说音乐等控件标签</audio>
    <hr>
</body>
</html>
```

26. 前端界面模块实现-CSS基础认识

层叠样式表 (`Cascading Style Sheets`).

`CSS` 能够对网页中元素位置的排版进行像素级精确控制, 实现美化页面的效果. 能够做到页面的样式和结构分离.

基本语法规范

选择器 + {一条/N条声明}

- 选择器决定针对谁修改 (找谁)
- 声明决定修改啥. (干啥)
- 声明的属性是键值对. 使用 ; 区分键值对, 使用 : 区分键和值.

```
<style>
    p {
        /* 设置字体颜色 */
        color: red;
        /* 设置字体大小 */
        font-size: 30px;
    }
</style>
<p>hello</p>
```

- `CSS` 要写到 `style` 标签中(后面还会介绍其他写法)
- `style` 标签可以放到页面任意位置. 一般放到 `head` 标签内.
- `CSS` 使用 `/* */` 作为注释.
- `CSS` 不区分大小写, 我们开发时统一使用小写字母

选择器的种类

1. 基础选择器: 单个选择器构成的

- 标签选择器
- 类选择器
- id 选择器
- 通配符选择器

2. 复合选择器: 把多种基础选择器综合运用起来.

- 后代选择器
- 子选择器
- 并集选择器
- 伪类选择器

```
/*通配选择器-对所有的标签产生效果*/
* {
    margin: 0px;
    padding: 3px;
}
```

```
/*p这种与html标签同名的称之为标签选择器，同类标签都具有这个属性*/
p {
    color: red;
    font-size: 30px;
}
<p>这是一个标题</p>
```

```
/*类选择器,类名以.开头，多个类属性可以在一个标签内使用*/
.green {
    color: green;
}
.big {
    font-size: 40px;
}
<p class="green big">这是一个类选择器修饰的属性</p>
```

```
/*id选择器，名称以#开头，修饰指定id的标签容器，只能被一个标签使用，因为id唯一*/
#font {
    font-size: 20px;
    font-family:sans-serif;
    color: aqua;
}
<p id="font">这个是id选择器弄的样式</p>
```

27. 前端界面模块实现-`vue.js`基础认识

安装

```
<!-- 开发环境版本，包含了有帮助的命令行警告 -->
<script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>

<!-- 生产环境版本，优化了尺寸和速度 -->
<script src="https://cdn.jsdelivr.net/npm/vue"></script>
```

入门案例

使用 vue 实现一个简单的 hello world 程序

```
<div id="app">{{message}}</div>

<script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>
<script>
    var app = new Vue({
        el: '#app',
        data: {
            message: 'hello',
        },
    });
</script>
```

代码解释:

1. 创建了一个 div, id 为 app
2. 在 div 中使用 "插值表达式" `{{message}}` 来指定 div 内部的内容.
3. js 中创建了一个名为 app 的 Vue 实例. 构造函数中的参数是一个对象.
4. 参数中的 el 字段是一个选择器, 对应到 html 中的具体元素id.
5. 参数中的 data 字段是一个特定的对象, 用来放置数据.
6. data 中的 message 属性值, 就对应到 {{message}} 中的内容. Vue 会自动解析 {{message}} , 并把 data 中对应的名字为 message 的属性值替换进去.

理解响应式: 修改 message 的值, 界面显示的内容就会发生改变.

插值操作: `{{}}`

```
<div id="app"> {{str1}} {{str2}} </div>

<script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>
<script>
    let app = new Vue({
        el: '#app',
        data: {
            str1: 'hello',
            str2: 'world'
        }
    });
</script>
```

遮罩: `v-cloak`

> cloak 意思是 "斗篷", 用来遮罩被渲染之前的插值表达式.

HTML 解析代码的时候是从上往下解析. 如果加载 `vue` 的速度比较慢, 那么就会在界面上看到 `{{ }}` 这样的内容.

```
<style>
    [v-cloak] {
        display: none;
    }
</style>

<div id="app" v-cloak> {{message}} </div>
```

注意: `v-cloak` 相当于 `div` 标签的一个属性. 这个属性在 `Vue` 接管 `div` 之前会存在, 但是 `Vue` 执行之后这个 `v-cloak` 就会被 `Vue` 去掉. 此时 `display` 样式就不再生效了.

> `[v-cloak]` 是属性选择器(是 `CSS` 的基本选择器之一). 选择了所有包含 `v-cloak` 属性的元素.

绑定属性: `v-bind`

很多标签的属性都是需要动态进行设置的. 比如 `<a>` 标签的 `href`, `<img>` 的 `src` 等.

此时使用插值表达式在属性中是不能使用的.

```
<img v-bind:src="url" alt="">

<script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>
<script>
    let app = new Vue({
        el: '#app',
        data: {
            url: "一个图片路径链接"
        }
    });
</script>
```

事件监听: `v-on`

v-on 后面使用 : 连接后续的事件名称.

```
<div id="app">
    <button v-on:click="dialog('点击了按钮')">按钮</button> <!--当按钮被点击就会触发事件函数的调用-->
</div>

<script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>
<script>
    let app = new Vue({
        el: '#app',
        data: {
        },
        methods: {
            dialog: function (str) {
                alert(str);
            }
        }
```

```
    });
</script>
```

- `methods`： `vue`对象的函数或方法都放在其中

```
<div id="app">
    <form action="http://www.sogou.com">
        <input type="submit" value="提交" v-on:click.prevent="click()">
    </form>
</div>

<script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>
<script>
    let app = new Vue({
        el: '#app',
        data: {
        },
        methods: {
            click: function () {
                console.log('hello');
            },
        }
    });
</script>
```

- `v-on:click.prevent`： 阻止元素默认行为. 比如这里则进制了默认的form表单提交操作

条件显示： `v-show`

`v-show`, 条件为 `true` 的时候显示元素, 条件为 `false` 时不显示.

```
<div id="app">
    <h3 v-show="flag">hello</h3>
</div>

<script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>
<script>
    let app = new Vue({
        el: '#app',
        data: {
            flag: true,
        }
    });
</script>
```

条件指令： `v-if`

通过一个表达式决定某个元素是否被渲染.

```
<div id="app">
    <div v-if="score > 90">学神</div>
    <div v-else-if="score > 80">学霸</div>
    <div v-else-if="score > 60">普通</div>
    <div v-else>学渣</div>
</div>

<script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>
<script>
    let app = new Vue({
        el: '#app',
        data: {
            score: 95,
        }
    });
</script>
```

`v-show` 和 `v-if` 的区别：当把 `flag` 置为 `false` 时, v-if 对应的元素不会被构建到 `dom` 树中, 而 `v-show` 的元素构建到 `dom` 树中了, 只是通过 `display:none` 隐藏了.

循环指令：`v-for`

`v-for` 可以绑定数据到数组来渲染一个标签

```
<div id="app">
    <div v-for="hero in heros">{{hero}}</div>
</div>

<script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>
<script>
    let app = new Vue({
        el: '#app',
        data: {
            heros: ['小乔', '曹操', '李白'],
        }
    });
</script>
```

双向绑定：`v-model`

表单是实际开发中和用户交互的重要手段. 通过 `v-model` 可以将一个 `vue` 数据与标签数据关联起来，实现一荣俱荣一损俱损的效果。

```
<div id="app">
    <input type="text" v-model="message">
    {{message}}
</div>

<script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>
<script>
    let app = new Vue({
```

```
        el: '#app',
        data: {
            message: 'hello',
        }
    });
</script>
```

通过 v-model 命令就把 input 标签的 value 和 message 关联起来了.

> 此时, 通过修改输入框的内容, app.message 就会发生改变.
>
> 修改 app.message 的值, 界面也会随之发生改变.
>
> 这个操作就称为 **双向绑定**

28. 前端界面模块实现-`jquery.ajax`基础认识

AJAX 是与服务器交换数据的技术，它在不重载全部页面的情况下，实现了对部分网页的更新。

其实简单来说，`ajax`就是一个`http`客户端，可以异步请求服务器。

```
<html>
    <body>
        <div id="app">
            <button v-on:click="myclick()">提交</button>
        </div>
    </body>
</html>
<script src="https://cdn.staticfile.org/jquery/1.10.2/jquery.min.js"></script>
<script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>
<script>
    let app = new Vue({
        el: '#app',
        data: {
            numbers: 0,
            videos: []
        },
        methods: {
            myclick: function() {
                $.ajax({
                    url: "http://192.168.122.137:9090/video",
                    type: "get",
                    context: this,//这里是将vue对象传入ajax作为this对象
                    success: function(result,status,xhr) {//请求成功后的处理函数
                        this.videos = result;
                        alert(result);
                    }
                })
            }
        }
```

```
    });
</script>
```

29. 前端界面模块实现-前端视频展示页面

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<meta name="description" content="">
<meta name="author" content="OrcasThemes">
<meta http-equiv="X-UA-Compatible" content="IE=Edge" />
<title>Home</title>
<link href="css/bootstrap.css" rel="stylesheet">
<link rel="stylesheet" href="css/screen.css">
<link rel="stylesheet" href="css/animation.css">
<link rel="stylesheet" href="css/font-awesome.css">
<link href="css/lity.css" rel="stylesheet">

<style>
    [v-cloak] {
        display: none;
    }
</style>
</head>
<body>
<div id="app">
    <div id="home1" class="container-fluid standard-bg">
    <div class="row header-top">
        <div class="col-lg-3 col-md-6 col-sm-5 col-xs-8">
        <a class="main-logo" href="#"><img src="img/main-logo.png" class="main-logo img-
responsive" alt="Muvee Reviews" title="Muvee Reviews"></a>
        </div>
        <div class="col-lg-6 hidden-md text-center hidden-sm hidden-xs">
        </div>
        <div class="col-lg-3 col-md-6 col-sm-7 hidden-xs">
        <div class="right-box">
            <button type="button" class="access-btn" data-toggle="modal" data-
target="#enquirypopup">上传视频</button>
        </div>
        </div>
    </div>
    <!-- MENU -->
    <div class="row home-mega-menu ">
        <div class="col-md-12">
        <nav class="navbar navbar-default">
            <div class="navbar-header">
                <button class="navbar-toggle" type="button" data-toggle="collapse" data-
target=".js-navbar-collapse">
                <span class="sr-only">Toggle navigation</span>
                <span class="icon-bar"></span>
                <span class="icon-bar"></span>
```

```
                <span class="icon-bar"></span>
                </button>
            </div>
            <div class="collapse navbar-collapse js-navbar-collapse megabg dropshd ">
                <ul class="nav navbar-nav">
                    <li><a href="index.html">Home</a></li>
                </ul>
                <div class="search-block">
                    <form>
                    <input type="search" placeholder="Search">
                    </form>
                </div>
            </div>
            <!-- /.nav-collapse -->
        </nav>
        </div>
    </div>
    <!-- CORE -->
    <div class="row">
        <!-- SIDEBAR -->
        <div class="col-lg-2 col-md-4 hidden-sm hidden-xs">
        </div>
        <!-- HOME MAIN POSTS -->
        <div class="col-lg-10 col-md-8">
        <section id="home-main">
            <h2 class="icon"><i class="fa fa-television" aria-hidden="true"></i>最新视频</h2>
            <div class="row">
                <!-- ARTICLES -->
                <div class="col-lg-9 col-md-12 col-sm-12">
                    <div class="row auto-clear">
                    <article class="col-lg-3 col-md-6 col-sm-4" v-for="video in videos" v-
cloak>
                        <!-- POST L size -->
                        <div class="post post-medium">
                            <div class="thumbr">
                                <a class="afterglow post-thumb" v-bind:href="'/single-
video.html?id='+video.id" target="_blank">
                                    <span class="play-btn-border" title="Play">
                                        <i class="fa fa-play-circle headline-round" aria-
hidden="true"></i>
                                    </span>
                                    <img class="img-responsive" v-bind:src="video.image"
alt="#">
                                </a>
                            </div>
                            <div class="infor">
                                <h4><a class="title" href="#">{{video.name}}</a></h4>
                            </div>
                        </div>
                    </article>
                    </div>
                    <div class="clearfix spacer"></div>
                </div>
```

```
                <!-- RIGHT ASIDE -->

            </div>
        </section>
        </div>
    </div>
    </div>
    <!-- TABS -->
    <div id="tabs" class="container-fluid featured-bg">

    </div>
    <!-- MAIN -->
    <div id="main" class="container-fluid">

    <div class="clearfix"></div>
    </div>
    <!-- CHANNELS -->
    <div id="channels-block" class="container-fluid channels-bg">
    <div class="container-fluid ">
        <div class="col-md-12">
        <!-- CHANNELS -->
        <section id="channels">
        </section>
        <div class="clearfix"></div>
        </div>
    </div>
    </div>
    <!-- FOOTER -->
    <div id="footer" class="container-fluid footer-background">
    <div class="container">
        <footer>
        <div class="row copyright-bottom text-center">
            <div class="col-md-12 text-center">
                <a href="" class="footer-logo" title="Video Magazine Bootstrap HTML5
template">
                <img src="img/footer-logo.png" class="img-responsive text-center" alt="Video
Magazine Bootstrap HTML5 template">
                </a>
                <p>Copyright &copy; 2022.Author by {{author}}</p>
            </div>
        </div>
        </footer>
    </div>
    </div>
    <!-- MODAL -->
    <div id="enquirypopup" class="modal fade in " role="dialog">
    <div class="modal-dialog">
        <!-- Modal content-->
        <div class="modal-content row">
        <div class="modal-header custom-modal-header">
            <button type="button" class="close" data-dismiss="modal">x</button>
            <h2 class="icon"><i class="fa fa-television" aria-hidden="true"></i>上传视频</h2>
        </div>
```

```
        <div class="modal-body" >
            <form name="info_form" class="form-inline" action="/video" method="post"
enctype="multipart/form-data">
                <div class="form-group col-sm-12">
                    <input type="text"  name="name" placeholder="视频名称">
                </div>
                <div class="form-group col-sm-12">
                    <input type="text"  name="info" placeholder="视频描述">
                </div>
                <div class="form-group col-sm-12">
                    <input type="file"  name="video" placeholder="视频文件">
                </div>
                <div class="form-group col-sm-12">
                    <input type="file"  name="image" placeholder="封面图片">
                </div>
                <div class="form-group col-sm-12">
                    <input type="submit"  name="submit" value="提交">
                </div>
            </form>
        </div>
        </div>
    </div>
    </div>
</div>
<script src="js/jquery-1.12.1.min.js"></script>
<script src="js/bootstrap.min.js"></script>
<script src="js/lity.js"></script>
<script src="https://cdn.jsdelivr.net/npm/vue@2/dist/vue.js"></script>
<script>
 $(".nav .dropdown").hover(function() {
   $(this).find(".dropdown-toggle").dropdown("toggle");
 });
</script>
<script>
    var app = new Vue({
        el: '#app',
        data: {
            author: 'Zhang',
            videos: []
        },
        methods: {
            get_allvideos: function() {
                $.ajax({
                    type: "get",
                    url: "/video",
                    context: this,
                    success:function(result,status,xhr){
                        this.videos = result;
                    }
                })
            }
        }
    })
```

```
    app.get_allvideos();
</script>
</body>
</html>
```

30. 前端界面模块实现-前端视频观看页面

```
<!DOCTYPE html>
<html lang="en">
   <head>
      <meta charset="utf-8">
      <meta name="viewport" content="width=device-width, initial-scale=1">
      <meta name="description" content="">
      <meta name="author" content="OrcasThemes">
      <meta http-equiv="X-UA-Compatible" content="IE=Edge" />
      <title></title>
      <link href="css/bootstrap.css" rel="stylesheet">
      <link rel="stylesheet" href="css/screen.css">
      <link rel="stylesheet" href="css/animation.css">
      <link rel="stylesheet" href="css/font-awesome.css">
      <link href="css/lity.css" rel="stylesheet">
   </head>
   <body>
   <div id="app">
      <!-- SINGLE VIDEO -->
      <div id="single-video" class="container-fluid standard-bg">
         <!-- HEADER -->
         <div class="row header-top">
            <div class="col-lg-3 col-md-6 col-sm-5">
               <a class="main-logo" href="#"><img src="img/main-logo.png" class="main-logo"
alt="Muvee Reviews" title="Muvee Reviews"></a>
            </div>
            <div class="col-lg-6 hidden-md text-center hidden-sm hidden-xs">
            </div>
            <div class="col-lg-3 col-md-6 col-sm-7 hidden-xs">
               <div class="right-box">
                  <button type="button" class="access-btn" data-toggle="modal" data-
target="#enquirypopup">修改信息</button>
               </div>
            </div>
         </div>
         <!-- MENU -->
         <div class="row home-mega-menu ">
            <div class="col-md-12">
               <nav class="navbar navbar-default">
                  <div class="navbar-header">
                     <button class="navbar-toggle" type="button" data-toggle="collapse"
data-target=".js-navbar-collapse">
                     <span class="sr-only">Toggle navigation</span>
                     <span class="icon-bar"></span>
                     <span class="icon-bar"></span>
```

```
                     <span class="icon-bar"></span>
                     </button>
                  </div>
                  <div class="collapse navbar-collapse js-navbar-collapse megabg dropshd ">
                     <ul class="nav navbar-nav">
                        <li><a href="index.html">Home</a></li>
                     </ul>
                     <div class="search-block">
                     </div>
                  </div>
                  <!-- /.nav-collapse -->
               </nav>
            </div>
         </div>
         <!-- SINGLE VIDEO -->
         <div class="row">
            <!-- SIDEBAR -->
            <div class="col-lg-2 col-md-4 hidden-sm hidden-xs">
            </div>
            <!-- SINGLE VIDEO -->
            <div id="single-video-wrapper" class="col-lg-10 col-md-8">
               <div class="row">
                  <!-- VIDEO SINGLE POST -->
                  <div class="col-lg-10 col-md-12 col-sm-12">
                     <!-- POST L size -->
                     <article class="post-video">
                        <!-- VIDEO INFO -->
                        <div class="video-info">
                           <!-- 16:9 aspect ratio -->
                           <div class="embed-responsive embed-responsive-16by9 video-embed-
box">
                              <iframe v-bind:src="video.video"  class="embed-responsive-
item"></iframe>
                           </div>
                           <h2 class="title main-head-title">{{video.name}}</h2>
                           <div class="metabox">
                              <span class="meta-i">
                              <i class="fa fa-thumbs-up" aria-hidden="true"></i>20.895
                              </span>
                              <span class="meta-i">
                              <i class="fa fa-thumbs-down" aria-hidden="true"></i>3.981
                              </span>
                              <span class="meta-i">
                              <i class="fa fa-user"></i><a href="#" class="author"
title="John Doe">John Doe</a>
                              </span>
                              <span class="meta-i">
                              <i class="fa fa-clock-o"></i>March 16. 2017
                              </span>
                              <span class="meta-i">
                              <i class="fa fa-eye"></i>1,347,912 views
                              </span>
                              <div class="ratings">
```

```
                        <i class="fa fa-star" aria-hidden="true"></i>
                        <i class="fa fa-star" aria-hidden="true"></i>
                        <i class="fa fa-star-half-o" aria-hidden="true"></i>
                        <i class="fa fa-star-o"></i>
                        <i class="fa fa-star-half"></i>
                     </div>
                  </div>
                  <div class="share-input">
                     <span class="fa fa-chain sharelinkicon"></span>
                  </div>
               </div>
               <div class="clearfix spacer"></div>
               <!-- DETAILS -->
               <div class="video-content">
                  <h2 class="title main-head-title">视频描述</h2>
                  <p>{{video.info}}</p>
               </div>
               <div class="clearfix spacer"></div>
            </article>
             </div>

         <!-- VIDEO SIDE BANNERS -->
         <div class="col-lg-2 col-md-4 hidden-sm hidden-xs">
         </div>
      </div>
   </div>
</div>
<!-- CHANNELS -->
<div id="channels-block" class="container-fluid channels-bg">
   <div class="container-fluid ">
      <div class="col-md-12">
         <!-- CHANNELS -->
         <section id="channels">
         </section>
         <div class="clearfix"></div>
      </div>
   </div>
</div>
<!-- BOTTOM BANNER -->
<div id="bottom-banner" class="container text-center">
</div>
<!-- FOOTER -->
<div id="footer" class="container-fluid footer-background">
   <div class="container">
      <footer>
         <!-- SECTION FOOTER -->
         <div class="row">
            <!-- SOCIAL -->
            <div class="col-lg-3 col-md-3 col-sm-6 col-xs-12">
               <div class="row auto-clear">
                  <div class="col-md-12">
                  </div>
```

```
                        <div class="col-md-12">
                        </div>
                        <div class="col-md-12">
                        </div>
                     </div>
                  </div>
                  <!-- TAGS -->
                  <div class="col-lg-3 col-md-3 col-sm-6 col-xs-12">
                  </div>
                  <!-- POST -->
                  <div class="col-lg-3 col-md-3 col-sm-6 col-xs-12">
                  </div>
                  <!-- LINKS -->
                  <div class="col-lg-3 col-md-3 col-sm-6 col-xs-12">
                  </div>
               </div>
               <div class="row copyright-bottom text-center">
                  <div class="col-md-12 text-center">
                     <a href="" class="footer-logo" title="Video Magazine Bootstrap HTML5
template">
                     <img src="img/footer-logo.png" class="img-responsive text-center"
alt="Video Magazine Bootstrap HTML5 template">
                     </a>
                     <p>Copyright &copy; 2022.Author by Zhang.</p>

                  </div>
               </div>
            </footer>
         </div>
      </div>
      <!-- MODAL -->
      <div id="enquirypopup" class="modal fade in " role="dialog">
         <div class="modal-dialog">
            <!-- Modal content-->
            <div class="modal-content row">
               <div class="modal-header custom-modal-header">
                  <button type="button" class="close" data-dismiss="modal">x</button>
                  <h2 class="icon"><i class="fa fa-television" aria-hidden="true"></i>修改信息
</h2>
               </div>
               <div class="modal-body">
                  <form name="info_form" class="form-inline" action="#" method="post">
                     <div class="form-group col-sm-12">
                        <input type="text" class="form-control" name="name" id="name" v-
model="video.name">
                     </div>
                     <div class="form-group col-sm-12">
                        <input type="text" class="form-control" name="" id="email" v-
model="video.info">
                     </div>
                     <div class="form-group col-sm-12">
                        <button class="subscribe-btn pull-right" type="submit"
title="Subscribe" v-on:click.prevent="update_video()">提交</button>
```

```
                    </div>
                </form>
            </div>
        </div>
    </div>
</div>


    <!-- JAVA SCRIPT -->
    <!-- jQuery (necessary for Bootstrap's JavaScript plugins) -->
    <script src="js/jquery-1.12.1.min.js"></script>
    <script src="js/bootstrap.min.js"></script>
    <script src="js/lity.js"></script>
    <script src="https://cdn.jsdelivr.net/npm/vue@2/dist/vue.js"></script>
    <script>
        $(".nav .dropdown").hover(function() {
          $(this).find(".dropdown-toggle").dropdown("toggle");
        });
    </script>
    <script>
        var app = new Vue({
            el: '#app',
            data: {
                video: {}
            },
            methods: {
                get_param: function(name) {
                    return decodeURIComponent((new RegExp('[?|&]' + name + '=' + '([^&;]+?)
(&|#|;|$)').exec(location.href) || [, ""])[1].replace(/\+/g, '%20')) || null
                },
                get_video: function() {
                    var id = this.get_param("id");
                    $.ajax({
                        type: "get",
                        url: "/video/" + id,
                        context: this,
                        success: function(result,status,xhr) {
                            this.video = result;
                        }
                    })
                },
                update_video: function() {
                    $.ajax({
                        type: "put",
                        url: "/video/" + this.video.id,
                        data:JSON.stringify(this.video),
                        context: this,
                        success: function(result,status,xhr) {
                            window.location.reload();
                        }
                    })
                }
```

```
            }
        });
        app.get_video();
    </script>
  </body>
</html>
```

31. 项目总结

项目名称：视频共享点播系统

项目功能：搭建一个共享点播系统，服务器支持用户通过前端浏览器访问服务器，获取展示与观看和操作的界面，最终实现视频的上传以及观看和删改查等基础管理功能。

开发环境：`centos7.6/vim、g++、gdb、makefile`

技术特点：`http`服务器搭建，`restful`风格接口设计，`json`序列化，线程池，`html+css+js`基础

项目模块：

- 数据管理模块：基于`MYSQL`进行数据管理，封装数据管理类进行数据统一访问
- 业务处理模块：基于`HTTPLIB`搭建`HTTP`服务器，使用`restful`风格进行接口设计处理客户端业务请求
- 前端界面模块：基于基础的`HTML+CSS+JS`完成基于简单模板前端界面的修改与功能完成。

项目扩展：

2. 添加用户管理以及视频分类管理
3. 添加视频的评论，打分功能。
4. 服务器上的视频或图片采用压缩处理节省空间，进行热点与非热点的管理

常见问题：

1. 说说你的项目
2. 为什么做这个项目
3. 项目中的某个技术点你是怎么实现的，为什么要用它
4. 服务器怎么搭建的，为什么不自己实现
5. 多个客户端同时视频上传怎么处理
6. 你的服务器支持多少个客户端，如何进行测试的