## TCP/IP协议组

## HTTP

### 后退按钮访问浏览器缓存

https://blog.csdn.net/yangshijin1988/article/details/44418587

当点击后退按钮时，默认情况下浏览器不是从Web服务器上重新获取页面，而是从浏览器缓存中载入页面。

HTTP头信息“**Expires**”和“**Cache-Control**”为应用程序服务器提供了一个控制浏览器和代理服务器上缓存的机制。

HTTP头信息**Expires**告诉代理服务器它的缓存页面何时将过期。   
HTTP1.1规范中新定义的头信息**Cache-Contro**l可以通知浏览器不缓存任何页面。

如下是使用Cache-Control的基本方法：   
 1) no-cache:强制缓存从服务器上获取新的页面   
 2) no-store: 在任何环境下缓存不保存任何页面

例：

对于HTML网页，加入：

<meta HTTP-EQUIV="pragma" CONTENT="no-cache">   
<meta HTTP-EQUIV="Cache-Control" CONTENT="no-cache, must-revalidate">   
<meta HTTP-EQUIV="expires" CONTENT="0">

对于JSP页面，加入：  
<%   
response.setHeader("Cache-Control","no-store");   
response.setHeader("Pragrma","no-cache");   
response.setDateHeader("Expires",0);   
%>

## HTTPS

SSL协议（Secure Sockets Layer [安全套接层](https://baike.baidu.com/item/%E5%AE%89%E5%85%A8%E5%A5%97%E6%8E%A5%E5%B1%82" \t "https://baike.baidu.com/item/ssl/_blank)）

TLS协议（Transport Layer Security 安全传输层协议）

### 证书

尽管有 非对称算法 与 对称算法 对数据进行加密与解密。但为了防止公钥被人恶意劫持、替换，以取代通信双方进行交流，于是需要公证机构（**CA**）颁布一个 **证书** 证明某个 公钥 属于 某个机构。

### 数字签名

同样为防止 证书 被劫持，故需要通过 Hash 算法，将公钥与网站信息通过 Hash 算法单向加密，生成 消息摘要。

由于 Hash 算法只要输入数据有一点变化，那么生成的消息摘要就会发生巨变，防止他人修改原始数据。

公钥及网站信息

CA

消息摘要

Hash 加密

数字签名

CA密钥 加密

公钥及网站信息

数字签名

数字证书

当 浏览器 得到数字证书时，将用 Hash 算法对 公钥及网站信息 进行加密生成**消息摘要1**，再用CA的公钥对数字前面进行解密得到**消息摘要2**，将两者对比以知是否被修改。

注意：CA有也可能被人劫持

在操作系统或浏览器中预置顶层的CA证书链，以互相验证CA。

## FTP（文件传输协议）

FTP与HTTP均为文件传输协议，具有很多共同点。

均运行在TCP上。

FTP协议使用两个并行的TCP连接来传输文件：控制连接 与 数据连接

### 控制连接

用于 客户端 与 服务端 之间传输控制信息，比如用户标识、口令、改变远程目录以及“存放（put）”和“获取（get）”文件的命令。

### 数据连接

TCP控制连接端口21

TCP数据连接端口20

服务端

客户端

传输文件

### 带外传送 与 带内传送

由于FTP使用独立的 控制连接 来传输控制信息，所以我们称FTP是 带外（out-of-band）传送；

HTTP则通过 带内（in-band） 发送控制信息

### 流程

用户主机 与 远程主机 开始一个FTP会话时。

客户端首先在向服务器21端口与服务器发起一个用于控制的 TCP 连接。FTP客户端也通过该控制连接发送用户的标识和口令，发送改变远程目录的命令。

FTP服务器端从该连接上收到一个文件传输的命令后，就发起一个到客户端的TCP数据连接。

FTP在该数据连接上准确地传送一个文件，然后关闭该连接。

故每次传输文件，TCP都是一个新的 数据连接。

注意，FTP服务器必须在整个会话过程中保持用户的状态。特别是服务器必须把用户账户与控制连接联系起来，随着用户在远程目录树上徘徊，用户必须追踪用户在远程目录上的当前位置。

对每个进行中的用户会话状态信息进行追踪，大大限制了FTP同时维持的会话总数。

### Python 服务器代码

<https://www.cnblogs.com/huangxm/p/6274645.html>

from pyftpdlib.authorizers import DummyAuthorizer #傻瓜式授权

from pyftpdlib.handlers import FTPHandler #FTP句柄

from pyftpdlib.servers import FTPServer #FTP服务

print("Hello world")

authorizer = DummyAuthorizer() #新建一个用户组

authorizer.add\_user("fan", "root", "D:/", perm="elr")#将用户名，密码，指定目录，权限 添加到里面

#这个是添加匿名用户,任何人都可以访问，如果去掉的话，需要输入用户名和密码，可以自己尝试

#authorizer.add\_anonymous("D:/")

#初始化FTP句柄

handler = FTPHandler

handler.authorizer = authorizer

#开启服务器

server = FTPServer(("127.0.0.1", 21), handler)

server.serve\_forever()

### Python客户端代码

'''

Created on 2018年9月1日

https://blog.csdn.net/qq\_39091609/article/details/79338906

@author: Administrator

'''

import ftplib

import sys

print("Hello FTP client:")

#获取服务器的ip地址（如192.168.1.107），使用sys.argv可以从命令行参数里面获取

if len(sys.argv) < 2:

#tmp = input("please input server address:")

tmp = "127.0.0.1"

sys.argv.append(tmp)

server\_address = sys.argv[1]

#创建FTP实例，并显示欢迎界面

ftp = ftplib.FTP(server\_address)

print(ftp.getwelcome())

#登录，输入服务器里添加过的用户名和口令

#ftp.login('user', 'pass')

#文件上传

def upload(fname):

fd = open(fname, 'rb')

new\_name = input("input new name:")

#以二进制的形式上传

ftp.storbinary("STOR %s" % new\_name, fd)

fd.close()

print("upload finished")

#文件下载

def download(fname):

#构建文件的存储路径，这里用的是D盘,可以自行设置

new\_path = "D:\\FTPdownload\\" + fname

fd = open(new\_path, 'wb')

#以二进制形式下载，注意第二个参数是fd.write，上传时是fd

ftp.retrbinary("RETR %s" % fname, fd.write)

fd.close()

print("download finished")

def main():

#选择操作，上传、下载、退出

op = input("what do you want?(u/d/q)")

if op == "u":

#输入文件完整路径，必要时可以用绝对路径

fname = input("input the file of path:")

upload(fname)

elif op == "d":

fname = input("input the file name:")

download(fname)

else:

print("quit now!")

ftp.quit()

if \_\_name\_\_ == '\_\_main\_\_':

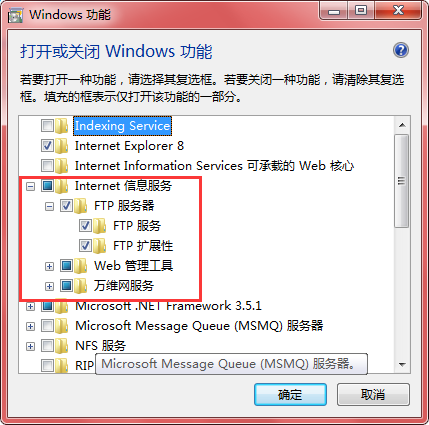
main()

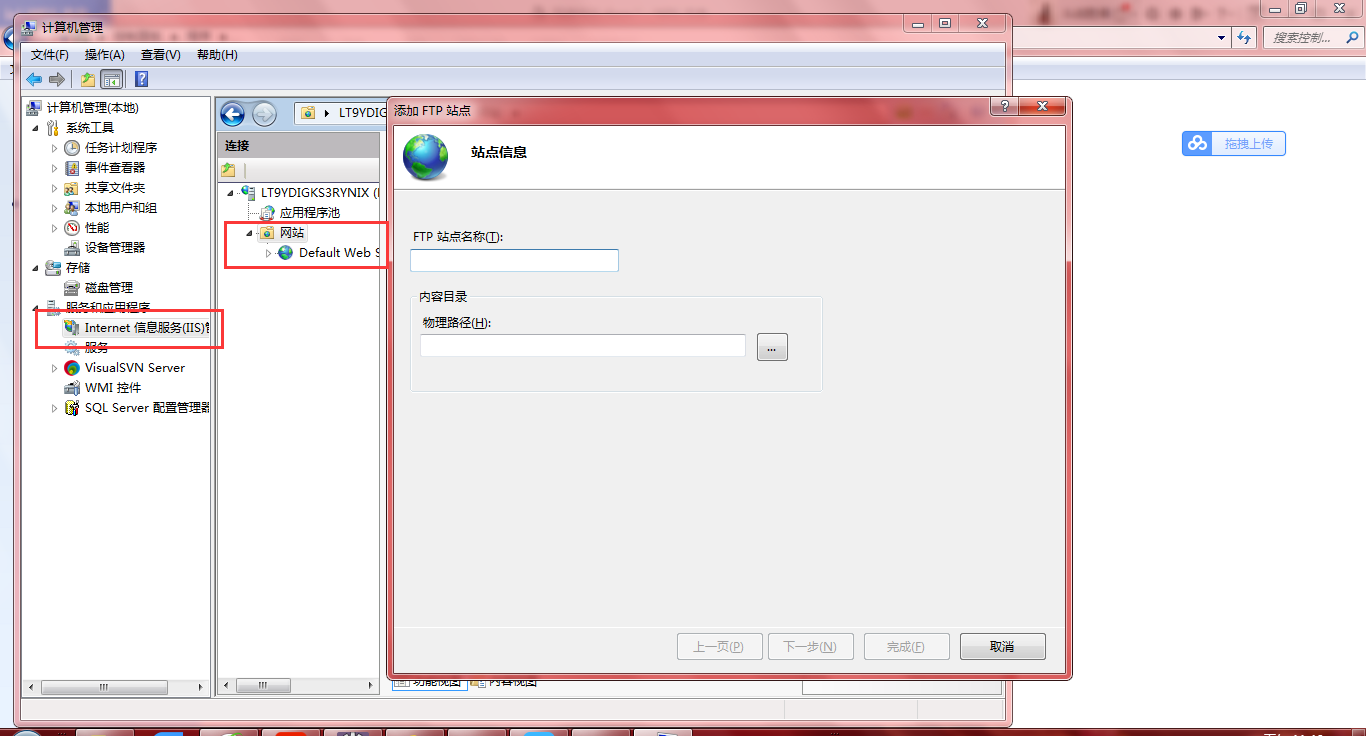
### Windows开启FTP Server服务

笔者使用的是windows7操作系统

https://www.cnblogs.com/liangxuru/p/6148212.html

1、打开控制面板->程序，选择“打开或关闭Windows功能”。

2、

1. 
2. 