HttpConn

member

- **m_sockfd** — sockfd
- **sockaddr_in m_addr** — sock地址
- **m_http_request** — **http_request**
  - **member**
    - 解析出来的信息
      - m_method
      - m_version
      - m_body
      - m_path
      - m_header
    - 存储登录用户的信息 — m_user_info
    - 存储上传文件的信息 — m_file_info
  - **function**
    - init — 可能keep-alive要在使用同一个request对象时接收不同响应，需要init
    - **parse**
      - parse_request_line
      - parse_headers
      - parse_body
      - ParsePath — 继续解析path，加上文件后缀,如果是list.json，写入list.json，传给客户端
      - parse_post
        - parse_urlencoded
        - parse_formdata
    - GetFiles — 获取上传上来的文件列表，写入一个vector<string>
    - WriteJson
- **m_http_response** — **http_response**
  - **member**
    - m_stat — 文件信息
    - m_mmfile — 共享内存指针
    - SUFFIX_TYPE — 文件类型
    - CODE_STATUS
  - **function**
    - init — 接受httprequest解析完毕的信息，初始化
    - **make_response**
      - add_line — 添加line到write_buff
      - add_header — 添加header到write_buf
      - add_body — 打开文件映射到内存，获得mmap指针，由HttpConn的writebuffer写
    - GetfileType — 判断文件类型
    - error_content — 打开文件错误时填充body内容
    - error_HTML — request传来的code是错误码400或者对文件内容进行判断的404,403，将文件信息修改为对应的错误HTML
- **m_read_buf** — httprequest的读缓冲区
- **m_write_buf** — httpresponse的写缓冲区

function

- InitConn — 初始化连接
- **ReadBuffer** — 核心函数：将数据从sockfd读到m_read_buf
- HandleHttp
  - 1.用http_request对象解析在m_read_buf里的请求，结果用成员变量保存
  - 2.将解析结果传给http_response对象
  - 3.将回复信息写到m_write_buf，m_write_buf当信息载体
  - 4.将m_write_buf的信息和文件内容填充iov结构体
- **WriteBuff** — 核心函数：将iov里的内容分散写到对端

流程图

- ReadBuffer → Parse
  - NO_REQUEST → epoll重置EPOLLIN
  - GET_REQUEST → response.init(200)
    - file OK → add_status-add_header → add_body → error_content?
      - no → status和header填充给iov[0],获得mmfilePTR,填充给iov[1] → WriteBuffer
      - yes → 写入error的body信息,和status和header共同填充给iov[0] → WriteBuffer
    - file not exist → error_HTML → add_status-add_header
    - file fib → error_HTML → add_status-add_header
  - BAD_REQUEST → reponse.init(400) → error_HTML