NODEJS 封装C++API学习总结

李晟泽

- 封装的是FtdcSysUserApi.h中提供的**CShfeFtdcSysUserApi** 和 **CShfeFtdcSysUserSpi**。

- **CShfeFtdcSysUserApi**：用于向服务器发起请求。
  - 数据的传输方向为： Js(前端) => V8 => C++(后台)

- **CShfeFtdcSysUserApi**：在接收到服务器的返回数据后，该类中的回调函数会被调用。
  - 数据的传输方向为： C++(后台) => V8 => Js(前端)

- 数据流的传输方向决定了需要解决的技术要点不同，导致封装的方法也不一样。

- **CShfeFtdcSysUserApi**： Js(前端) => V8 => C++(后台)

- Js(前端) 传输给V8的参数与C++后台对应的Api 形式是一致的，但需要通过V8的转换函数将其转换传输给C++ Api。

- 因为需要使用V8的主体函数进行封装，所以无法通过直接继承 **CShfeFtdcSysUserApi** 多态的重写原有的 虚函数完成。

- **这里采用的是**在类的内部定义了 **CShfeFtdcSysUserApi**的指针并重新定义了**CShfeFtdcSysUserApi**所有函数来实现封装。在每个函数内部通过**CShfeFtdcSysUserApi**的指针调用后台的Api。

- 技术要点： 不同版本的一致性。

- 在V8 用于转换的函数接受的参数形式是封装好的类型， Js传进的参数被封装到这个指定的类型中。

- 不同版本Nodejs对应的V8里面所设置的封装类型也不一致，所以为了避免不同版本的一致性，使用NAN_METHOD(FUNC) 来封装V8函数，

- **CShfeFtdcSysUserApi**： C++(后台) => V8 => Js(前端)
- 封装**CShfeFtdcSysUserApi** 数据的传输不需要完全使用V8函数的参数，所以可以直接派生，使用多态的重写虚函数的方式进行封装。
- 从后台传输回的数据会直接传输到重写的函数参数里。
- C++(后台) => V8 可以直接实现。

- 不同线程间的数据传递
- 实际使用过程中， 在直接封装重写的函数里使用V8转换数据传递给前端是无法实现的。
- 可能的原因在于，直接封装的函数与后台是作为一个IO线程单独运行，而前端是作为主线程在运行。
- 这是因为nodejs是一个非阻塞IO的事件驱动机制（data = fs.readFile; 与 data = fs.readFileSync）， 主线程和IO线程不是同一个线程， 根据不同的系统， windows（IOCP） 和 *nix(多线程)来实现IO线程。
- 事件驱动通过事件循环机制实现。

- 当做不同线程的数据处理的话，通过liuv的事情驱动机制来实现：

- 具体的实现就是：
- uv_async_t asyncOnFuncName;
- uv_async_send(&asyncOnFuncName);
- void OnFuncName(uv_async_t *handle)

- 可能的问题：
  - asyncOnFuncName.data 发送数据不是线程安全的，必须保证同一时间只有一个线
  - 程能够操作asyncOnFuncName

- 最后需要使用 Addon.cc 注册所有方法。
- 所有的函数信息可以在XML对应的找到。
- 下一周工作：
- 1. 研究libuv线程间消息传递的机制。
- 2. 掌握V8转换的技术要点，展开进一步的工作。