To compile: compile.bat

To run: run.bat

Design Notes

- **Client Side:** I use C# to write a WPF program because it is much natural and easier.
- **Communication between WPF program and C++ library:** I write a DLL as the intermediate layer which allow WPF program to transfer data with my C++ lib with JSON format data. It is just like a B/S application which allow browser to transfer data with backend using json or xml data.
- **HttpClient and HttpServer:** I have the HttpClient and HttpServer which is to send and receive HTTP style message. There are three steps for HttpServer:
  1. Receive a message and deserialize it to a HttpRequest instance
  2. Find route table which match the resource identifier in the request instance (URL), redirect the request to the registered handler
  3. Get the response from handler, serialize it and return it to client
- Currently I use asynchronous message between client and server, it means that it may block the client if the server spends some time on analyzing the dependency relation.

Requirement List:

- Req1
- Req2
- Req3
  - You can upload a folder as a module to server
  - My program will automatically get all .h and .cpp as module files and folder name as module name
  - During the check-in process, my program will set the module's dependency by going through all of **existing closed** module
  - All information of a module is saved in the <metadata.json> file, including the module name, version, closed, dependency relation and file list
- Req4
  - I use **my own** project #3 program to analysis the dependency relation in the check-in process
- Req5
  - I create a new folder for each new module with the name <modulename__yyyyMMddhhmmss>
  - I use check-in date for both creating folder and the module version because it is much easier to locate a module and its version
  - The repository folder is <ProjectFolder/Code_Repository>
  - The <metadata.json> contains all information of a module, including the name, version, closed, dependency relation and file list
- Req6
  - The <closed> is set to false by default
  - When user upload to an opened module, the new files will replace the old files and I will generate dependency relation again
  - You can use <Close Module> function in the <Detail View> of client program to close a module

    - When user upload to an closed module, my program will create a new folder with new version for this module
    - **My dependency analysis will only use these closed module**
- Req7
    - There will be a download message box when you click download button in detail view. You can choose to download with or without dependency
    - The file will save to local disk and you will see a message box when the download is finished
- Req8
    - This program use C/S mode. The client send HTTP style message to server and the server will response with some HTTP style message as well
- Req9
    - I totally simulate a HTTP message
      <Method > <Resource>
      <Content-Type>; <Boundary>
      <Empty Line>
      <Boundary>
      <Content-Disposition>; <data-name>
      <Empty Line>
      <data>
    - In this message, boundary is used can split different data in a request **so I don't set content-length for the request**
    - The response message is almost same except it doesn't use boundary and the first line is the status code and status text

- Req10
    - I use http style message so the file content is actually in the request body and I use boundary to extract them. It is an elegant just what HTTP does
- Req11
    - There are some data in the repository already for test. Also, you can set HttpClient and HttpServer **as start project to run automated test instead** of using client program

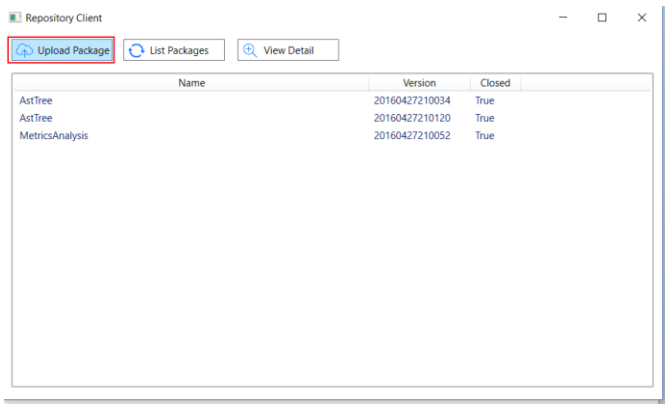
Usage:

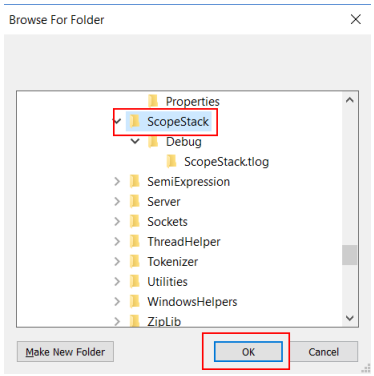
1. Upload a new module

a. Click <upload package> button

b. Select the module folder and click <OK> button

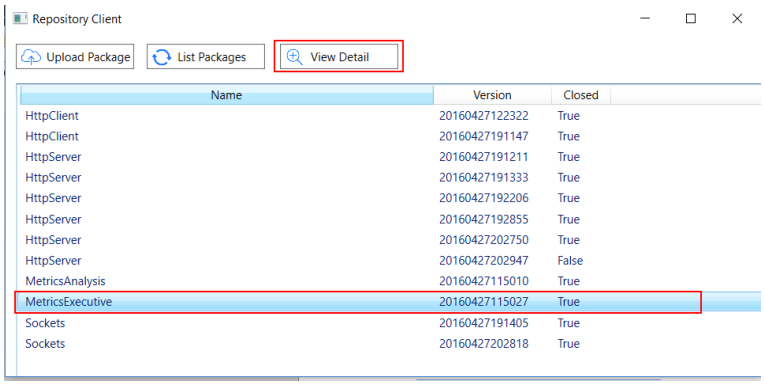
c. Wait for the server response and you can see the new module information in the list view





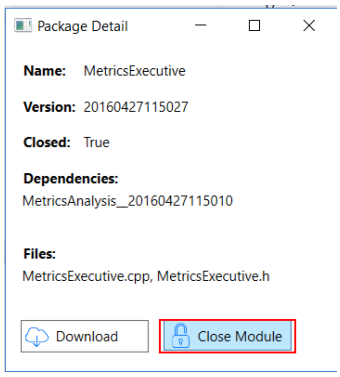
2. View Module Detail

Select an item in the module list view and click <View Detail> button.



3. Close module for modification

In the detail view, click the <Close Module> button. The list view will refresh after this operation and you can see the value in the <Closed> column has been changed.



4. Download module

In the detail view, click the <Download> button. After the download finish, the messagebox will show you the path of the download file and you can open it by clicking <open> button in the messagebox.