CC_Design and Implementation of Client – Server communication.

Nils Tolleshaug 2017-09-06

DESIGN CONSIDERATION

The CC specifications define one module for Client access to Server when Server is located on separate Server (“Server access”) and a simpler version (“Bridge access”) of the communication module if the Client and Server are located on the same PC. The intention in the proposed design in the document, are to have only one type of module for Client Server communication between Client and Server and over the IP interface. Therefore in the Client Local Connection Requester and Remote Connection Requester will be implemented as one common Connection Request Handler. In the Server there will be one Connection Listener Handler instead of having both Local Connection Listener and Remote Connection Listener.

In the local PC case where Client and Server resides on the same PC, the IP port interface will be via “localhost – 127.0.0.1” while the communication in the remote case will be on a network defined IP port interface either on the local LAN or WAN. The advantage of this approach will be that the more complex Client to Server communication over Internet can be implemented in Java and tested early in the HRE development process. If it turns out that communication via “localhost” will not work according to the intention, it is easy to later implement the “Bridge access” to server on one PC as a modified subset of first implementation of (“Server access”) communication.

The proposed objective is to implement the Client Request Manager and Server Request Listener all in Java. The Client - Server Connection shall offer parallel communication of data which implies that the Server Request Manager and partly the Client Request Manger and other included class implementations of communication processes must be to running as Thread’s (implements Runnable). Please note that Java SWING is no Thread safe and therefore “parallel” processing of GUI processes should be avoided. Consequently database requests (database request API) in the Client Connection Request Handler can be implemented as pure Java class method call.

SCOPE

Design and implement in Java a Client Request Manager that handles communication requests from Client processes to the Server Request Listener which pass the requests to the Server processes. These to modules communicate over the IP interface on a permanent basis on a fixed port number called the management channel. The Client Request Manager and Server Request Listener will use the management channel to be able to establish various communication services between Client and Server. For each communication service the Client Request Manager and Server Request Listener will establishing pairs of Connection Request Handler and Connection Listener Handler that use a specific port number for each functional area of communication. The pairs of Connection Request Handler and Connection

Listener Handler will be dynamic and established when the Client requests a specific service or action from the Server and terminated the communication process are completed.

The communication shall use class definitions for Connection Request Handler and Connection Listener Handler specific for STATUS, DATABASE and BATCH requests as specifies in 3.2 Architecture - Client Component and 3.3 Architecture – Server Component.

STATUS - communication process is a command and status report channel between Client and Server. The channel will exist as long as the user communicates with the HRE implementation. STATUS will have regular a “handshake” to monitor the status of the server communication.

DATABASE – To be used by GUI SWING components that are not Thread safe. GUI process must wait for response before continuing to process and present data retrieved from database. Only the listener process in Server need to running as Thread.

BATCH – Transfer of a data file from in both directions between Client and Server. Both Client and Server process need to be running as Thread’s

The first version of the Client Server communication implementation will have STATUS and DATABASE processes implemented to be able to demonstrate the Client – Sever communication.

ACTIONS

Client Request Manager shall handle following communication to/from the Server Request Listener:

1. Connect with the Server and establish the STATUS communication channel.
2. Query the Server Status and send commands to/from Client and Server over STATUS channel.
3. Large database intensive processes can be initiated over STATUS channel -
4. Disconnect from Server and terminate STATUS communication.
5. Transfer DATABASE requests to update the database – transfer and receive strings.
6. Transfer DATABASE requests to retrieve data from the database – transfer and receive strings.
7. BATCH File transfer process that transmits files in both directions between the Client and Server.
   The file transfers should use background asynchronous FTP transfers.
   BATCH processes can only be initiated by the GUI and read and store files in the filesystem.

USED BY

Business Rules in Client will in the DATABASE case prepare queryData String to the database and interpret the received responseData String.
See APPLICATION PROGRAMMING INTERFACES (API)

STATUS, DATABASE and BATCH processes in Server will receive data from Server Request Listener and respond with data codes as a Java String or reference to a Java ReadStream.

AUXILIARY DATA

List of available port number in the Client and Server is needed in order to dynamically assign free port numbers for communication between Connection Request Handler and Connection Listener Handler.

HRE project should prepare the definition of a basis communication handshake (protocol) over the STATUS channel.

REQUIRE SERVICES

Connection Request Handler and Connection Listener Handler need IP communication modules and localhost implementation.

Charset handler – Java uses charset according to national language while Linux servers use UTF-8

The network communication need Secure port communication based on class SLLSocket and class SLLServerSocket.

APPLICATION PROGRAMMING INTERFACES (API)

Example of DATABASE API

- API for BUSINESS RULES database request In ClientRequestManager

```
String responseData = ClientRequestManager.requestDatabaseData(String queryData)  {
        // Receive Send String queryData from BUSINESS layer and send  to
        DatabaseRequestListener
        // Wait for String responseData to be returned to BUSINESS layer
}
```

```
API for requesting data from DATABASE submitted by from Server Request Listener
String responseData = DatabaseHandler.requestDatabaseData(String queryData) {
        // Send String queryData to DATABASE.
        // Wait for String responseData to be returned from DATABASE
}
```

Alternatively implement an API that delivers a ReadStream to the BusinessLayer process.
The ReadStream are in BusinessLayer used for reading and converting the character stream to a JSON Object or a DOM object reference.

Further API specifications of STATUS, DATABASE and BATCH communication are needed.

WARNING CONDITIONS

The ClientRequestManager and ServerRequestListener must communicate to inform the user of certain problems in the communication. The ClientRequestManager can throw a HREexception that is picked up by the BusinessLayer prosesses while the ServerRequestListener must use the STATUS channel to communicate failures over to the STATUS process in ClientRequestManager which in turn throw a a HREexception.

Typical examples of reported error conditions:

- STATUS communication failed between Client and Server
- Not possible to use a specific port – port in use.

More to be added!

ERROR CONDITIONS

User Error messages should be as understandable as possible for ordinary PC users. As ex. error messages should refer to the superior process that failed rather than referring to the name or abbreviation of the software component with the problem. Please see point 6 in Design Guidelines.