Implementation of a novel web form server in jsocket for adding forms to J function

Yuji Suda
g.ysuda@gmail.com

JAPLA Workshop
PALETTE KASHIWA, Kashiwa, Chiba, Japan
December 8, 2023

Abstract

A novel web form server which can work as form adding functionality for J function has been implemented. The server system is written in J and accompanying system tool of jsocket. Users can deploy mechanism of retrieving arguments for J functions through web form controls of textbox, textarea, checkbox, dropdown, checklist and radiobutton, and of displaying result of J function in the same web form page as plain text and one image file. Four major web browsers, i.e. Microsoft Edge, Google Chrome, Apple Safari and Firefox, were tested. They were all feasible in communication of plain text. As for image transmission through inline frame control, only Firefox was feasible in operation. With this server system, web browsers can be utilized for adding basic forms to J function.

Contents

1 Introduction

Adding graphical user interface (GUI) to J function has been addressed in GUI part 1 of J primer in the example of centigrade fahrenheit tempeature mutual conversion named as cfgui, initially with legacy window driver [3] and currently, with Qt window driver [4]. Since Qt window driver is not compatible with legacy window driver, a number of legacy GUI added J functions can not be executed in the current version of J9.4. And In the legacy system, form design is achievable graphically using built –in form editor, whereas in Qt window driver layout of form controls is achieved with a completely different bin command system. Rewriting GUI from legacy to current Qt GUI is not easy for users who have been adapted to legacy system. And if future J version should adopt another GUI system which is not lower compatible with Qt GUI, the similar situation may arise. Web browser based GUI for J function could well be an everlasting GUI for J function, since browsers are the very fundamental interface to computers in the present day and possibly everlasting. In this context in mind, a novel web form server to add forms to J function has been developed and implemented.

2 Materials and Methods

2.1 System requirement

This web form server has to be executed in jconsole and not in Qt IDE environment. It may run in Qt IDE but it does result in hang up phenomenon because of problems of asynchronous GUI mechanism of the Qt IDE of J [1]. The server behavior has been tested in jconsoles of j9.4 in Windows 10, Mac catalina, Linux Debian 12 (installed in VMware Workstation 17 Player for Windows, free for non–commercial use) and Windows Subsystem for Linux Debian. Also in jconsole of j903 and j902 in Mac High Sierra. The system may run in jconsoles of earlier version of J such as j602, j701, j807 and j901. But they are not fully tested in the system operation. Use of the latest release of j9.4 is highly recommended.

2.2 Text editor

In order to define web form application users must edit application definition file: web_form_app_definition_file.ijs located in the server system directory (see 2.5.2). The system is not equipped with an original editor for this purpose and users must use text editors external to the web form server system. Windows user can use OS built-in notepad editor. The way of using notepad editor is guided in the main menu of the web form server manager (see 2.5.3). Mac and linux user can use OS preinstalled nano editor in another independent terminal. The way of using nano editor is also guided in the main menu of the web form server manager.

And in addition, in macOS or Linux, if command line emacs editor is installed, web form application definition file can be opened with ease without leaving the web form server system. There is no need to open another terminal.

2.3 Implementation of infinite loop for receiving, parsing and sending

In J's Studio Lab, Jsocket server and Jsocket server – client are lectures of basic jsocket operations in step by step manner. In order to perform continuous loop of message receiving, parsing and sending through a socket, an infinite loop with flow control command of while was written. The following is just a frame of such a loop. The actual ijs script for the loop can be confirmed in the source file of jsocket_simple_server_jconsole.ijs described in the section of implementation of simple server and client application in J (see 2.4.2).

```
NB. --- A frame of jsocket initialization and infinite communication loop -----

NB. define a port number for a socket
NB. define buffer size for receiving message
NB. define a label. for resumimg operation
NB. clean up jsocket environment
NB. assign a new socket for listening of connect request
NB. bind the socket at the defined port number
NB. start listening at the socket
NB. while. 1 do.
NB.    while. 1 do.
```

```
NB.      monitoring of connetion request
NB.      if. connected do. break. end.
NB.    end.
NB.    creat a new socket for message communication
NB.    while. 1 do.
NB.      receive incomming message
NB.      if. the message is nil do. goto. resume label end.
NB.      if. client requests server halt do. set halt request flag and break end.
NB.      send server reply message in response to client's message
NB.    end.
NB.    if. halt request flag set do. break. end.
NB. end.
NB. clean up jsocket environment
NB. this ends jsocket server operation
```

2.4 Implementation of simple server and client application in J

In order to confirm feasibility of basic socket communication, a simple server with functions of echo back and one line J command response, and terminal client application were written in J.

2.4.1 Installation and activation

The source codes are zipped to making_basic_ver_0_1.zip Download the file from

```
https://www.smccake.net/japla/making_basic_ver_0_1.zip
(login id and password are japla and iverson, respectively).
```

Unzip the file at the download folder to obtain the main folder of making_basic as shown in 2.4.2. In Microsoft Windows, copy the folder making_basic to c:\, whereas in Apple Mac and Linux, copy the folder to home directory.

In order to activate simple server and client, open two jconsoles independently and issue the following command.

```
In Microsoft Windows,
   load 'c:/making_basic/begin_simple_server_or_client_jconsole.ijs'

In Microsoft Windows Subsystem for Linux,
   load '/mnt/c/making_basic/begin_simple_server_or_client_jconsole_wsl.ijs'

In both Apple Mac and Linux,
   load (_1}.2!:0 'echo $HOME'), '/making_basic/begin_simple_server_or_client_jconsole.ijs'

After loading begin script, system displays the following,

   please type either of the following commands and hit enter to begin each menu
   begin 'server'
   or
   begin 'client'

In one jconsole,       issue a command begin 'server' to start a server and
in the other jconsole, issue a command begin 'client' to start a client.
```

2.4.2 Folders and files of each folder of making_basic_ver_0_1.zip

```
  making_basic
    begin_simple_server_or_client_jconsole.ijs
```

```
    begin_simple_server_or_client_jconsole_wsl.ijs
  making_basic/jconsole
    jsocket folder only
  making_basic/jconsole/jsocket
    jsocket_simple_server_jconsole.ijs
    jsocket_simple_client_jconsole.ijs
```

2.4.3 Usage of simple server and client

On issuing a begin 'server' or 'client' command, the system displays job menu shown below. Each menu item is self–explanatory. On selection of menu item, a prompt with a guide is displayed.

simple server menu

```
======= simple server (port monitor) menu (ver 0.1) =======

current PORT number = 1500

1 change port number
E echo back mode
J J command mode (one line command only)

Q Quit

select your command (1, E, J, or Q)
```

simple client menu

```
======= jsocket simple client menu (ver 0.1) =======

current host IP = 127.0.0.1  PORT = 1500

1 change HOST IP address
2 change port number
C connect to server
Q quit menu

input your selection 1, 2, C or Q
```

2.5 Implementation of web form server system

As an extention of simple server and client, parser and responding mechanism for HTTP protocol GET and POST method [2] were written in J for an implementation of web from server to J functions.

2.5.1 Installation and activation

The source codes are zipped to making_ver_0_1.zip. Download the file from

```
 https://www.smccake.net/japla/making_ver_0_1.zip
(login id and password are japla and iverson, respectively)
```

Unzip the file at the download folder to obtain the main folder of making as shown in 2.5.2. In Microsoft Windows, copy the unzipped folder making to c:\, whereas in Apple Mac and Linux, copy the folder to home directory.

In order to activate web form server menu, open jconsole and issue the following command.

```
In Microsoft Windows,
   load 'c:/making/begin_web_server_jconsole.ijs'

In Microsoft Windows Subsystem for Linux,
   load '/mnt/c/making/begin_web_server_jconsole_wsl.ijs'

In both Apple Mac and Linux,
   load (_1}.2!:0 'echo $HOME'), '/making/begin_web_server_jconsole.ijs'

After loading begin script, system displays the following,

   please type begin 0 to start server
   begin 0

And issue a command begin 0 to start the web form server manager menu.
```

2.5.2 Folders and files of each folder of making_ver_0_1.zip

```
  making
    begin_web_server_jconsole.ijs
    begin_web_server_jconsole_wsl.ijs
    basic_system_tools.ijs
  making/usr
  making/jconsole
    jsocket folder only
  making/jconsole/jsocket
    global_variables_jconsole.ijs
    server_core_jconsole.ijs
    support_tools_jconsole.ijs
    jsocket_web_form_server_for_j_function_jconsole.ijs
    web_form_app_definition_file.ijs
    main_fixed_part_jconsole.ijs
    jconsole_applications_fixed_part_jconsole.ijs
    create_response_message_to_GET_method_part_1_jconsole.ijs
    create_response_message_to_GET_method_part_2_jconsole.ijs
    create_response_message_to_GET_method_part_3_jconsole.ijs
    create_response_message_to_GET_method_part_4_jconsole.ijs
    action_to_POST_method_part_1_jconsole.ijs
    action_to_POST_method_part_2_jconsole.ijs
    03x03_Orange.bmp
    addNR.awk
  making/jconsole/jsocket/systemApp/making_of_app_def_file_jconsole
    web_form_app_definition_file.ijs
    main_jconsole.ijs
    jconsole_applications_jconsole.ijs
    create_response_message_to_GET_method_jconsole.ijs
    action_to_POST_method_jconsole.ijs
  making/jconsole/jsocket/app
    00_centigrade_fahrenheit_mutual_converter
    01_demo_web_form_app_with_available_form_controls_created_by_utility_app
    02_00_demo_number_list_reformat_with_no_option_for_reformat
    02_01_demo_number_list_reformat_with_column_number_option
    02_02_demo_number_list_reformat_with_column_width_option_added
    02_03_demo_number_list_reformat_with_decimal_digits_option_added
    02_04_demo_number_list_reformat_with_file_save_option_added
    02_05_demo_number_list_reformat_with_table_title_option_added
```

```
    50_print_a_beautiful_triangle_with_figure_length_calculation
    51_barrier_option_pricing_model
    52_hokusai_komon_image_galary
```

2.5.3 Usage of web form server management application

On issuing a start command begin 0, the system displays web form server manager menu as follows. Values of system install directory and j_version are taken from the running environment.

```
 ================ jsocket web form server manager menu (ver 0.1) ======================

                    system install directory: /Users/foo/making
                                  j_version: j9.4

 1 select a web form server app registered in the app directory and run it

 2 make a new web form server app based on the definition file and execute a test run
    (definition file: web_form_app_definition_file.ijs)
    (use menu E to edit the file and do not forget to save it before leaving the editor)
    (at the end of the creation, a test run follows. )
    (check app behavior and revise the definition file as needed with menu E.)
    (repeat revision process until the applicatio is completed.)

 C create a new definition file with utility web form application and activate menu 2
    (definition file: web_form_app_definition_file.ijs)
    (will be created with the utility web form application.)
    (On finishing the creation, above menu 2 is issued in succession. )

 R reload app definition file from already registered app (use menu E for a revision)

 E start external Editor for revision of web form app definition file
    (windows user is guided to issue built-in notepad bat file)
    (in mac and linux emacs will be issued, if it has been installed)
    (if not, OS built in nano editor usage guide is displayed. )

 D delete application from the registration directory


 Q quit this menu

input your menu selection.
```

The manager menu items 1, 2, C, R, E, D, Q are self-explanatory. The followings are complementary comments.

- On selection 1, R and D, current web form applications are listed with item numbers starting with 0 to max – 1. To cancel the application selection, just hit enter key to go back to the menu.
- In R and D menu item, execution is prompt with Yes/No question as a final decision.
- In menu item 2, web form application definition file is parsed for its application name, that is the name of application folder, and it is checked for duplication of the folder name. If the same folder name exists, delete Yes/No prompt is displayed. Answering No, the further operation is canceled. Answering Yes, the existing folder is removed and the definition file is parsed if it has incompatiblity with definition rule stated in the following subsection. If there is an incompatibility, the very point is displayed and further operation is canceled. If everything is OK, the newly created application is executed just like menu selection 1.

- Menu item C is the utility web form application for creating a new web form application where user can write four parts needed for web form application. So this item may be the first menu selection for creation of web form application to add form controls to user's J function. On finishing menu item C, menu item 2 follows for evaluation.
- When revision of definition file is needed, menu item E is used to edit the current definition file.

2.6 Specification of web form application definition file

The name of web form application definition file is 'web_form_app_definition_file.ijs'. The file is in the working directory making/jconsole/jsocket. It can be opened with the menu item E in the web form server manager menu. Essentially, it is a J script file which is loaded during the execution of web form server application. The basic format for definition file is the following.

```
1. Line preceded by 'NB.' is a comment.  This is J's grammer.
2. Line preceded by 'NB./*/' is a special comment and used in the web form server system for
   creation of application script.
3. The other lines are parsed by J engine as ordinary j script.
```

The definition file is composed of four parts.

```
A. Nomination of web form application
B. Design of HTML form document
C. J functions to be used in the web form application
D. Definition of an actual job with use of posted values
   (a) Collection of posted values and adjustment of arguments for J functions
   (b) Execution of J functions using arguments gathered in (a)
   (c) Preparation of a return value as one string of CRLF ended lines
       using the result of J function
```

The detail of the specification is described in the following subsubsection.

2.6.1 A. Nomination of web form application

The application name is defined in the line with prefix of 'NB./*/APP_FOLDER_NAME/'. If application name is foo, the assignment line is as follows.

```
NB./*/APP_FOLDER_NAME/foo
```

When multiple words are used as an application name, use underscore between words. See sample application name in section 2.10.

2.6.2 B. Design of HTML form document

In this part, design of HTML document is described. The title and one line remark are defined with the prefix of

```
NB./*/HTML_TITLE/
```

and

```
NB./*/HTML_REMARK/
```

respectively. If the title is 'title foo', and remark is 'remark foo', the definition lines are as follows.

```
NB./*/HTML_TITLE/title foo
NB./*/HTML_REMARK/remark foo
```

Spaces between words are allowed but the content must be in ONE line, otherwise J engine breaks here with a message of value error and web form server system stops.

Following the definitions of title and remark, form controls are defined according to the specification described below. Available form controls are textbox, textarea, checkbox, checklist, dropdown and radiobutton. In addition, three auxially HTML document items for horizontal line, new line and plain text of one line are available. Definition statement should be preceeded by NB./*/. as in title and remark.

TEXTBOX

Control name of TEXTBOX, control ID of two number digit(00 to 98), layout of either HORIZONTAL or VERTICAL, number of column, label caption, and default value. The field separator is a slash. Examples are as follows.

```
NB./*/TEXTBOX/00/VERTICAL/20/lable for textbox 00/default text for textbox 00
NB./*/TEXTBOX/01/VERTICAL/20/lable for textbox 01/default text for textbox 01
```

TEXTAREA

Control name of TEXTAREA, control ID of two number digit(00 to 98), layout of either HORIZONTAL or VERTICAL, number of column, number of row, label caption, and default value. An example is as follows.

```
NB./*/TEXTAREA/00/VERTICAL/150/05/label for textarea 00/default text for textarea 00
```

CHECKBOX

Control name of CHECKBOX, control ID of two number digit(00 to 98), layout of either HORIZONTAL or VERTICAL, label caption, and a value. An example is as follows.

```
NB./*/CHECKBOX/00/VERTICAL/label for checkbox 00/value for checkbox 00
```

CHECKLIST

Control name of CHECKLIST, control ID of two number digit(00 to 98), layout of either HORIZONTAL or VERTICAL, number of total items, label caption, total number of pairs of 'item label+item value'. An example is as follows.

```
NB./*/CHECKLIST/00/VERTICAL/03/label for checklist 00/label for item 00+value for item 00
     /lable for item 01+value for item 01/lable for item 02+value for item 02
```

DROPDOWN

Control name of DROPDOWN, control ID of two number digit(00 to 98), layout of either HORIZONTAL or VERTICAL, label caption, 'item label+item value' pairs as needed. An example is as follows.

```
NB./*/DROPDOWN/00/VERTICAL/label for dropdown 00/label for item 00+value for item 00
     /lable for item 01+value for item 01/lable for item 02+value for item 02
```

RADIOBUTTON

Control name of RADIOBUTTON, control ID of two number digit(00 to 98), layout of button items either HORIZONTAL or VERTICAL, label caption, 'item label+item value' pairs as needed. An example is as follows.

```
NB./*/RADIOBUTTON/00/VERTICAL/label for radiobutton 00/label for item 00+value for item 00
     /lable for item 01+value for item 01/lable for item 02+value for item 02
```

HORIZONTAL LINE, NEW LINE and PLAIN TEXT

They are defined as follows.

```
NB./*/HORIZONTALLINE
NB./*/NEWLINE
NB./*/arbitrary plain text in one line
```

2.6.3 C. J functions to be used in the web form application

In this part of definition file, the scripts of J fucntions are written.

2.6.4 D. Definition of YOUR_JOB function: J function of actural job with use of posted values

(a) Collection of posted values and adjustment of arguments for J functions
Posted value of each form contorl is stored as a global variable in a uniform syntax as shown below.

```
CONTROL_ID_VALUE
where
   CONTROL is TEXTBOX, TEXTAREA, CHECKBOX, CHECKLIST, DROPDOWN or RADIO
   ID is two digits number: from 00 to 98
   VALUE is the fixed word

Followings are examples.

TEXTBOX_00_VALUE,   TEXTBOX_01_VALUE, and so on
TEXTAREA_00_VALUE,  TEXTAREA_01_VALUE, and so on
CHECKBOX_00_VALUE,  CHECKBOX_01_VALUE, and so on
CHECKLIST_00_VALUE, CHECKLIST_01_VALUE, and so on
DROPDOWN_00_VALUE,  DROPDOWN_01_VALUE, and so on
RADIO_00_VALUE,     RADIO_01_VALUE, and so on
```

And the web form server system is equippted with a support function to gather all the posted values. It is

```
monitor_entry_ID_values
```

And default job of

```
a_collection =. monitor_entry_ID_values 0
```

is executed at the beginning in YOUR_JOB function in a web form application created with the utility application.

(b) Execution of J function using using arguments gathered in (a)
Posted values are all in a string value. So if a argument of J function is number, the corresponding posted value must be converted from string to number with J primitive '"."' as shown below.

```
Examples of assignment are as follows.

a_string_argument =. TEXTBOX_00_VALUE
a_number_argument =. ". TEXTBOX_01_VALUE
```

(c) Preparation of a return value as one string of CRLF ended lines using the result of J function
At first, the result of J function must be reformated to one string value of CRLF ended lines. If the result is an atom (scalor) or list (vectgor), it can be changed to string value ended with CRLF with J primitive '":'. But if the result is either table (matrix) or report (multi dimension array), a special function is needed to convert it to one string data. In the web form server system, the support functions includes one for such a task. The function name is

```
save_array_number_data_as_text_file
```

Usage is as follows.

```
NB. when a_result is an array data
a_result save_array_number_data_as_text_file 'saved_here.txt'
a_result_in_str =. read_from_file 'saved_here.txt'
```

There is one limitation, though, that the dimension of array must be less than 4.

And finally, a return value of YOUR_JOB function must be prepared as a string value together with the string value reformatted as above. This string return value is the very message which is included in system reply message region of the HTML document to be sent to client's browser.

Reviewing the definition files of sample applications (see 2.10) with server manager menus R and E is recommended to get idea of the specification of definition file.

2.6.5 Summaries of HTML design rules and usage of posted values

Summary of design rule of web form control

Web form control is defined with prefix NB./*/ followed by general control name and necessary properties with field separator '/'. The first three fields are general control name (in upper case), serial number of control and layout style. They are common to all controls. The control serial number (Number) must be in the range 00 and 98 in two digit format. Layout of control is either VERTICAL or HORIZONTAL. The sizes of column and row are arbitray number which are adjusted to the need of a specific application. Checklist, dropdown and radiobutton need select options in a pair of option lable and its value conjugated with '+' character. In addition, checklist need a total number of options. The definition file is processed line by line, so no carriage return should be inserted even if a line length is greater than the width of editor.

- TEXTBOX
  NB./*/TEXTBOX/Number/Layout/Column size/Label/Default string value
- TEXTAREA
  NB./*/TEXTAREA/Number/Layout/Column size/Row size/Label/Default one line
- CHECKBOX
  NB./*/CHECKBOX/Number/Layout/Label/Value
- CHECKLIST
  NB./*/CHECKLIST/Number/Layout/Total Number of option pairs/Option label+Value/Option label+Value ...
- DROPDOWN
  NB./*/DROPDOWN/Number/Layout/Option label+Value/Option label+Value ...
- RADIOBUTTON
  NB./*/RADIOBUTTON/Number/Layout/Option label+Value/Option label+Value ...

The following three definitions are decorations and additional plain text remark.

- horizontal line
  NB./*/HORIZONTALLINE
  This definition is converted to HTML <hr >.
- new line
  NB./*/NEWLINE
  This definition is converted to HTML
.
- arbitrary one line text
  NB./*/any supplemental statement in the HTML document.
  The strings preceeded by NB./*/ is treated with <pre >tag.

Summary of fetching posted value and setting new value to web form control.

- feching posted values
  The posted value are all in string value and each value of web form control can be fetched as

```
TEXTBOX_00_VALUE,   TEXTBOX_01_VALUE, and so on
TEXTAREA_00_VALUE,  TEXTAREA_01_VALUE, and so on
CHECKBOX_00_VALUE,  CHECKBOX_01_VALUE, and so on
CHECKLIST_00_VALUE, CHECKLIST_01_VALUE, and so on
DROPDOWN_00_VALUE,  DROPDOWN_01_VALUE, and so on
RADIO_00_VALUE,     RADIO_01_VALUE, and so on
```

  The values are used as arguments and conditions for execution of J functions.

- setting a new value to textbox and textarea
  The result of J function can be send back to client as a new string value of the web form control textbox and textarea. If the result is string value, it can be set directly as new value of the web form control. If the result is numeric value, it must be converted to string value, using J primitive ": in case of atom and list. When the shape of result is array of number or character, the following support function in the web from server system may help. Usage is as follows.

```
NB. when a_result is an array data
a_result save_array_number_data_as_text_file 'saved_here.txt'
a_result_in_str =. read_from_file 'saved_here.txt'
```

  New values of web form control must be assigned with J primitive =: as follows,

```
TEXTBOX_00_VALUE   =: new_value_in_one_line
TEXTAREA_00_VALUE  =: new_value_in_CRLF_connected_multi_lines
```

2.7 System default core structure of web form applicaton

The web form server sends response message as HTML form document following HTTP header. The core structure of the web form application consists of a server control radiobutton set, a submit button and a server message in plain text (Figure 1. This part of HTML document is written as server system task. The core structure of web form application can be confirmed in the following definition file which contains no HTML design nor J functions, and definition of YOUR_JOB returns just a test string as shown in Figure 2).

```
NB. a web form application definition to show core structure
NB. web_form_app_definition_file.ijs

NB. ====  (1/4) name of application =======
NB./*/APP_FOLDER_NAME/core_structure_of_web_form_app

NB. ====  (2/4) design of HTML doc  =======
NB. None

NB. ====  (3/4) descrition of J function  =======
NB. None

NB. ====  (4/4) desctiption of YOUR_JOB based on posted data as argument of J function ==
YOUR_JOB =: 3 : 0
a_return =. 'test return value of YOUR_JOB function'
a_return
)
```

Figure 1: This is the core structure of the web form server. The radiobutton for server activity control, i.e.continue or halt and submit button are parts of the core structure of HTML document. This core documnet is created by the system in the process of creating web form application. There is no need of writing HTML source in the definition file for web form application.

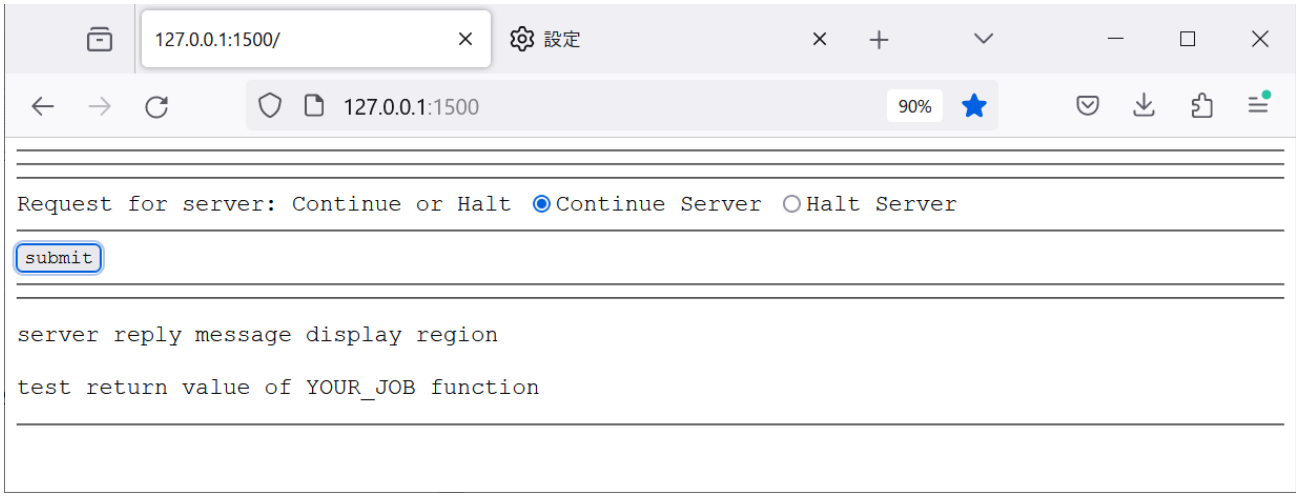

Figure 2: The screen capture of the HTML document received after submission. Note the content of the server reply message display region at the bottom of the document shows test reply message which is returned YOUR_JOB function written in the definition file.

2.8 Utility for creating a start up definition file of a new web form application

In order to help user to create a definition file, the system is equippted with a special utility web form application (Figure 3, 4, and 5). Although there are limitaions in the numbers of available form controls, i.e. three textboxes, one textarea, two checkboxes, one checklist, one dropdown, and one radiobutton, it does help to begin with writing a defintion file. This utility is available in the menu selection item C. The number of form controls can be increased in the process of revisioning. A sample web form application which was created with the utility is included in the registered sample applications (see 2.10.2).

```
01_demo_web_form_app_with_available_form_controls_created_by_utility_app
```

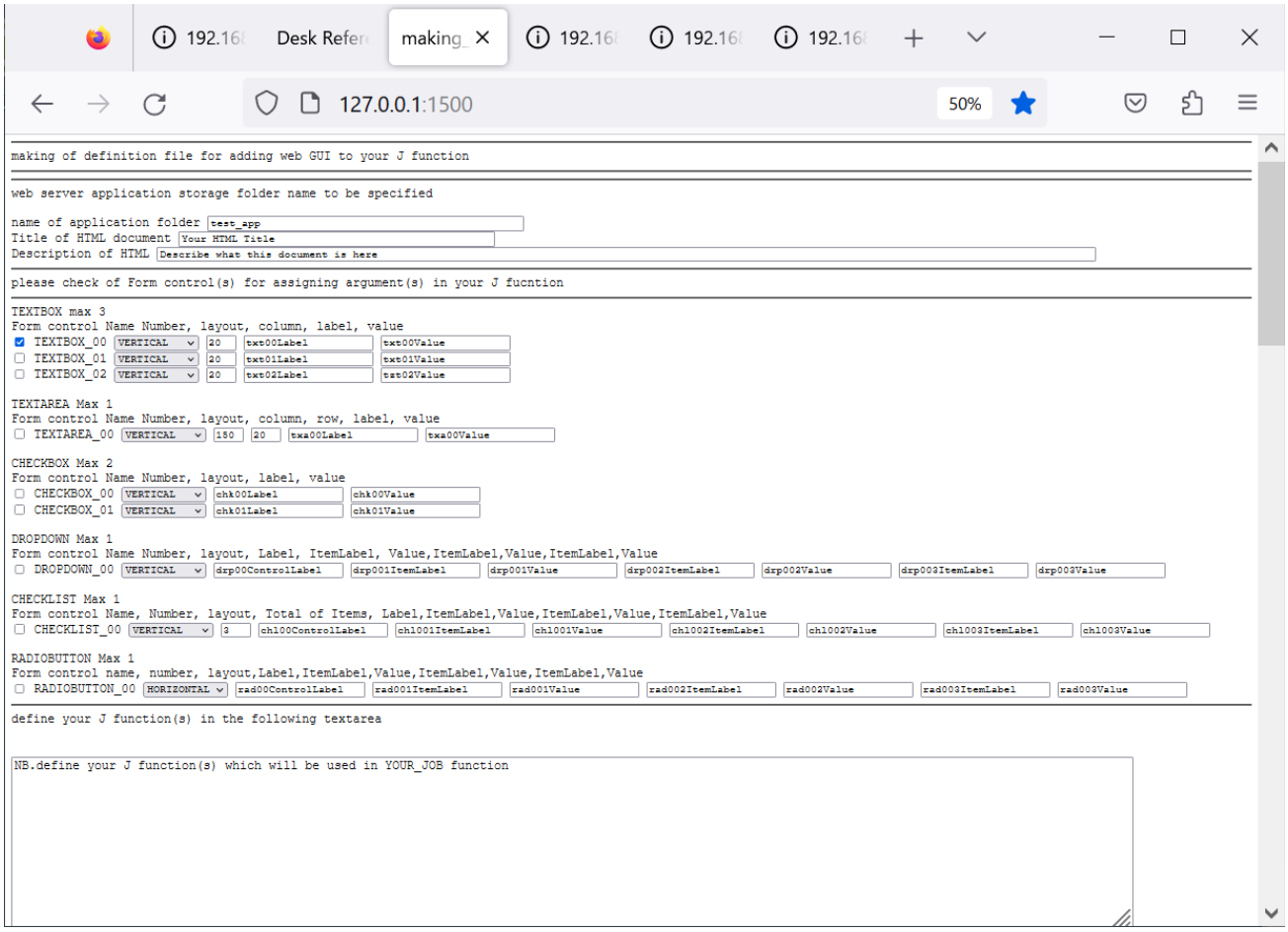

Figure 3: This is the upper half of screen capture of utility web form application for creating definition file. Note the first textbox is checked to use in the application. Note at the bottom there is a textarea for J function to be written.

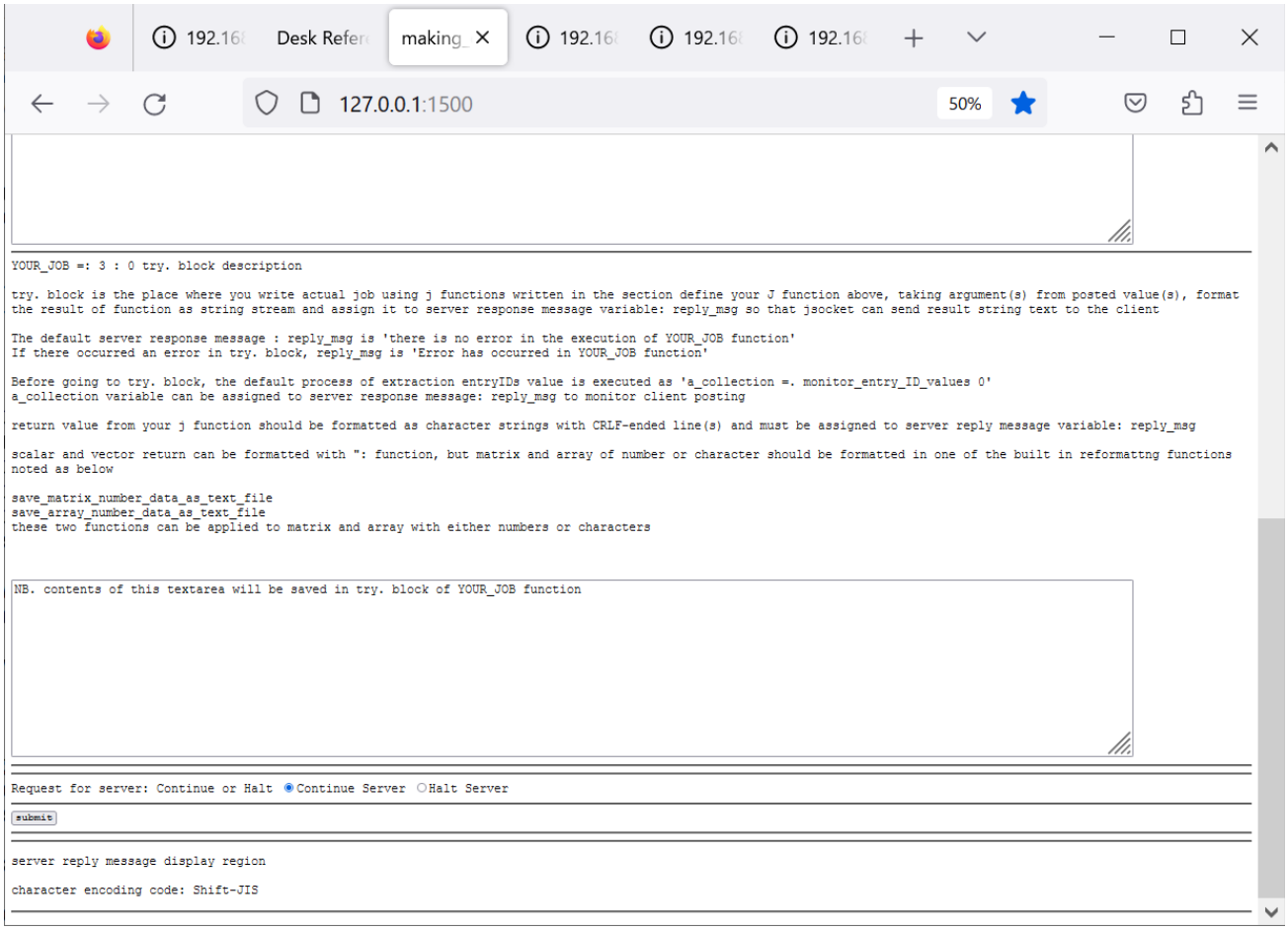

Figure 4: This is the lower half of screen capture of utility web form application for creating definition file. Note, in the lower region, there is another textarea for definition of YOUR_JOB function. The content of this textarea will be placed in try region in order to avoid an accidental system break at run time of web form application.

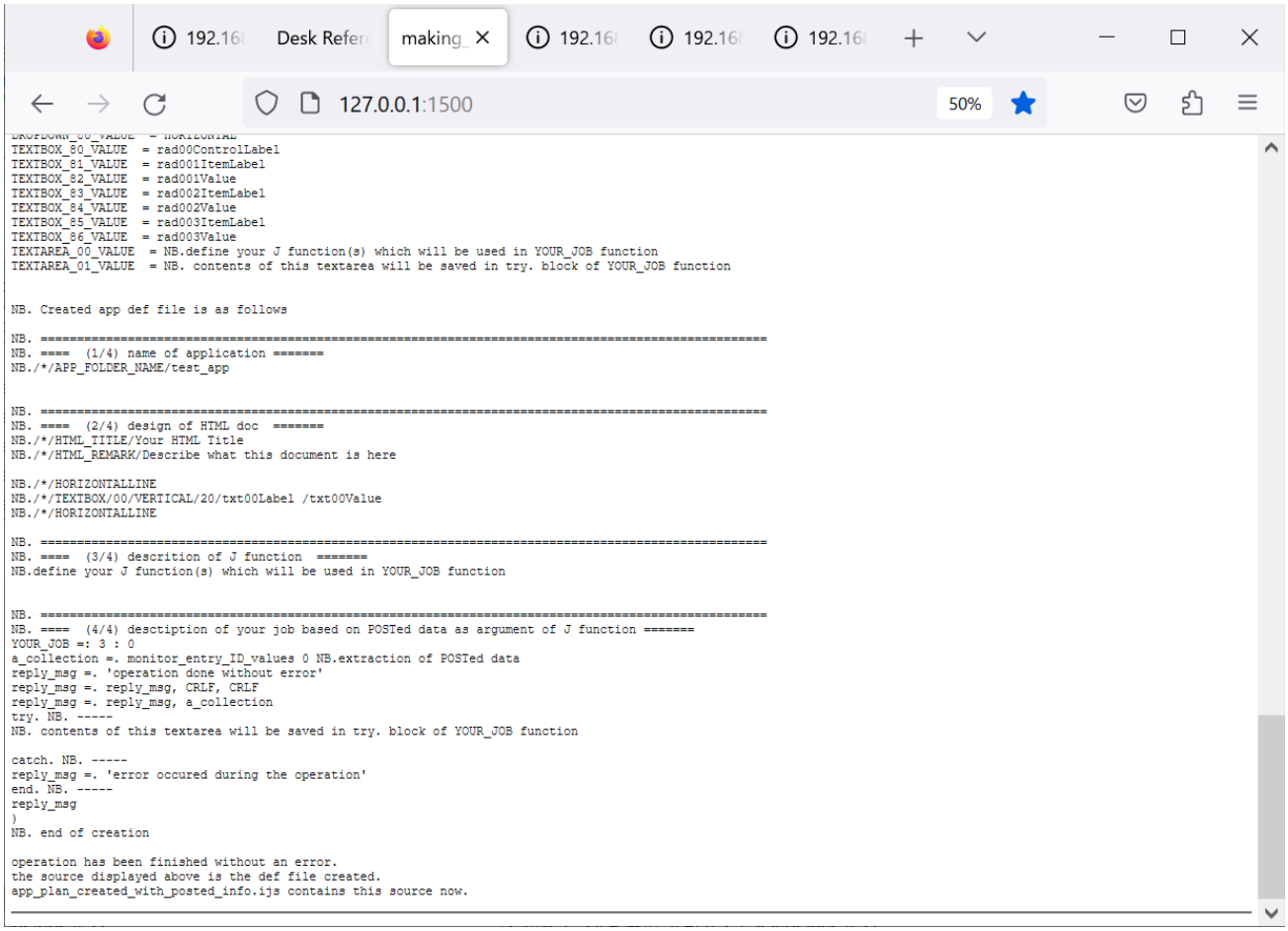

Figure 5: This is the lower part of the HTML document received after submission. Note the definition file created in the web form application appears at the bottom of system message region. Also note that application name, HTML title, HTML remark, and one TEXTBOX are defined. No description are seen in the regions of J function and YOUR_JOB since there are no description at the time of submission, and there are default system response message, i.e. reply_msg, with a collection of POSTed control values.

2.9 Use of a bmp image file as server response

The web form server can include one bitmap image in the response HTML form document. An image file must be created and saved as system default file name of 'response_image.bmp' in YOUR_JOB function. This bmp image is set in the HTML iframe contorl [6] and the image transmission is activated by setting RESPONSE_IMG_FLG global varaiable 1 as shown below

```
NB. default value of the following flag is 0
NB. if an image is used as a server response, prepare response_image.bmp and set the flag

RESPONSE_IMG_FLG =: 1

NB. iframe image size of width and height can be defined in the following global varaibles
NB. as a string value. Sample size below are '200', but any value is acceptable.

IFRAME_SIZE_W =: '200'
IFRAME_SIZE_H =: '200'
```

A sample application for image transmission is the following (see 2.10.10).

```
52\_hokusai\_komon\_image\_galary
```

2.10 Sample web form applications

In order to help users to grasp the idea of this web form server, there are several sample applications regsitered.

2.10.1 00_centigrade_fahrenheit_mutual_converter

This is a web form server version of the introductory sample GUI addressed in J's Help/Primer/GUI part 1 [3, 4].

2.10.2 01_demo_web_form_app_with_available_form_controls_created_by_utility_app

This is a sample application created by utility web form application issued by mananger menu item C. Is has no J function defined and the content of YOUR_JOB function is in the default state. It is just to show available web form controls in the utility web form application. This sample application can be used as a revision exercise, i.e. open definition file with menu item E, edit it and save, then menu item 2. In revision process, application name must be changed in order to trace revisions.

2.10.3 02_00_demo_number_list_reformat_with_no_option_for_reformat

The sample application of demo number list reformat is a series of 6 steps, i.e. from 02_00 to 02_05. The number list reformat application starts with just one textarea where data numbers are entered. And the numbers are reformatted as a matrix with fixed column number 6 with supplemental zero's as needed.

2.10.4 02_01_demo_number_list_reformat_with_column_number_option

In 02_01, a dropdown is added to change the number of columns.

2.10.5 02_02_demo_number_list_reformat_with_column_width_option_added

In 02_02, a checklist is added to change the column width.

2.10.6 02_03_demo_number_list_reformat_with_decimal_digits_option_added

In 02_03, another checklist is added to change decimal digits. This second checklist is layouted horizontally.

2.10.7 02_04_demo_number_list_reformat_with_file_save_option_added

In 02_04, additional textbox and radiobutton are added for a file save functionality. The textbox is for file name and the radiobutton is for file save options, i.e. no save, new and append.

2.10.8 02_05_demo_number_list_reformat_with_table_title_option_added

In 02_05, one checkbox and another textbox are added for adding option of title to the reformatted matrix. The checkbox is for on/off of a title addition and the textbox is for a title.

2.10.9 50_print_a_beautiful_triangle_with_figure_length_calculation

This is a web form application of an example in which form controls are added to J console function. The original J console function reported in the article 'Print a beautiful triangle with figure length calculation' was created by Toshio Nishikawa ([7]. It was modified to adjust to web form server application by Yuji Suda.

2.10.10 51_barrier_option_pricing_model

This is an example in which legacy form application was revised for the web form server. The original legacy form application reported in the article 'Barrier option pricing models' was created by Juichirou Takeuchi [8]. The main function for calcuration of barrier option model was adjusted for the web form server application by Yuji Suda.

2.10.11 52_hokusai_komon_image_galary

This is an example for sending image as a response. The hoskusai komon form application was written by Masato Shimura [9]. Becuase the web form server system does not support the drawing functions of hokusai komon, all the hokusai komon were drawn in advance using the original form application in j602a for Windows. The images were saved in png format with each image code in one of the system folder, 'hokusai_komon_in_png'. The web form server application parses the browser's request to identify each code of hokusai komon. And a selected image is converted to the default 'response_image.bmp' as an response image. Also RESPONSE_IMG_FLG is set as 1 to activate image transmission. The shape of the inline frame is a square and the size is selectable from a dropdown (options:400, 600, 800 and 1000).

3 Results

3.1 Simple server connection test: echo mode

On activation followed by echo mode selection, simple server waits for connection as shown in the Figure 6. Server connection were tested with 5 socket applications, i.e. simple client application in J, Microsoft Windows telnet.exe, free terminal application of teraterm for Windows, and Liunx telnet.

3.1.1 Simple client written in J

At first, simple jsocket client written in J was tested for connection to simple server. The connection was successful with message sending and receiving as shown in Figure 7.

3.1.2 Telnet.exe in Microsoft Windows

Then the windows telnet command was tested for connection. As shown Figure 8 and 9, connection was successful with message sending and receiving.

3.1.3 Teraterm free telent application for Microsoft Windows

Another telent application for windows, teraterm was also tested. This was also successful in connection, sending and receiving message as shown in Figure 10, 11 and 12.

3.1.4 telnet command in Linux

The final telnet test was done using Linux telnet (Figure 13). This Debian Linux version 12 was installed using VMware Workstation 17 Player for Windows in a windows machine where simple server is running in echo back mode. The host and guest machines are within the local area network 192.168.159.0, where the IP address of host machine is 192.168.159.1. So linux telnet accesses the host at 192.168.159.1 1500 instead of 127.0.0.1 1500. This was also successful in connection with sending and receiving message.

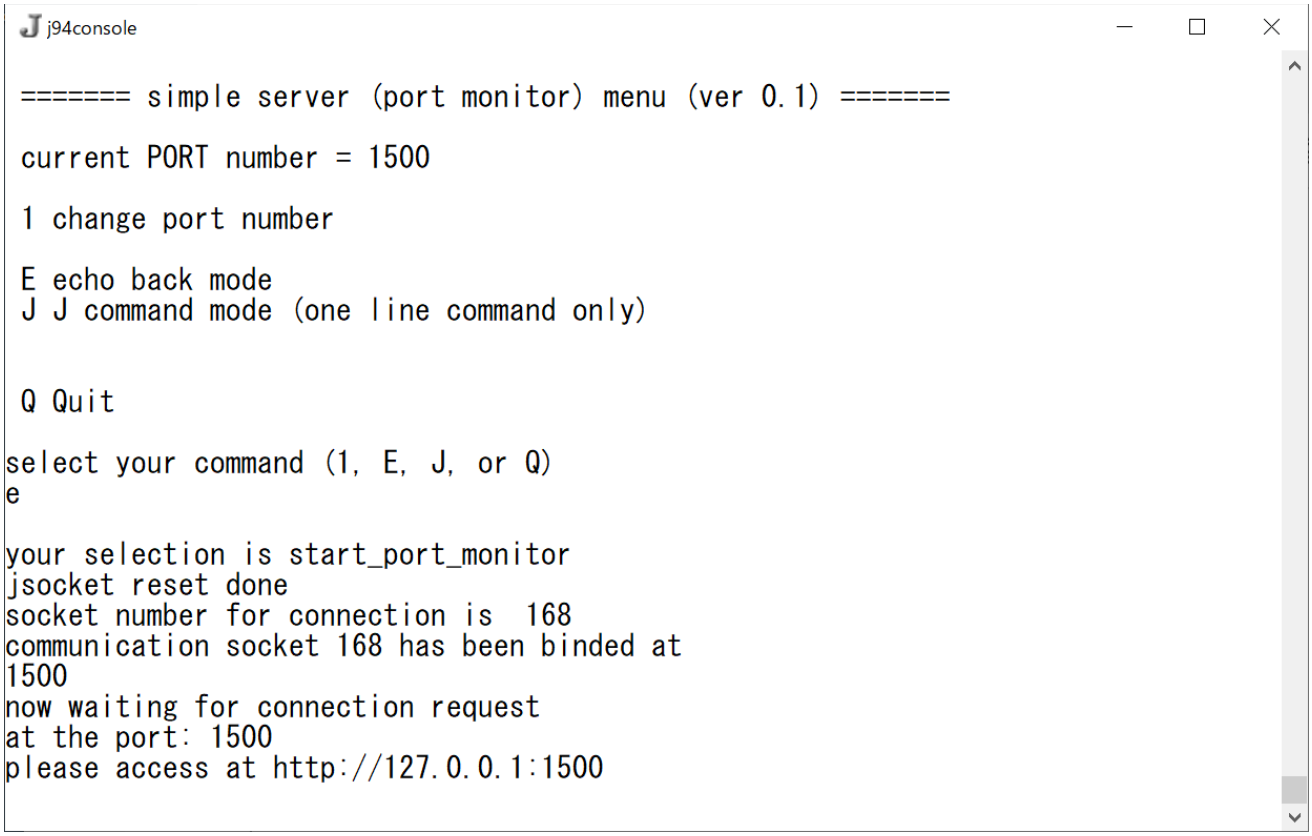

Figure 6: simple server menu and echo mode is selected, waiting for a connection. The server shows access URL of http://127.0.0.1:1500, i.e. local self IP address. But if the client is within local area network, it can access at the specific IP address assigned to the server machine.

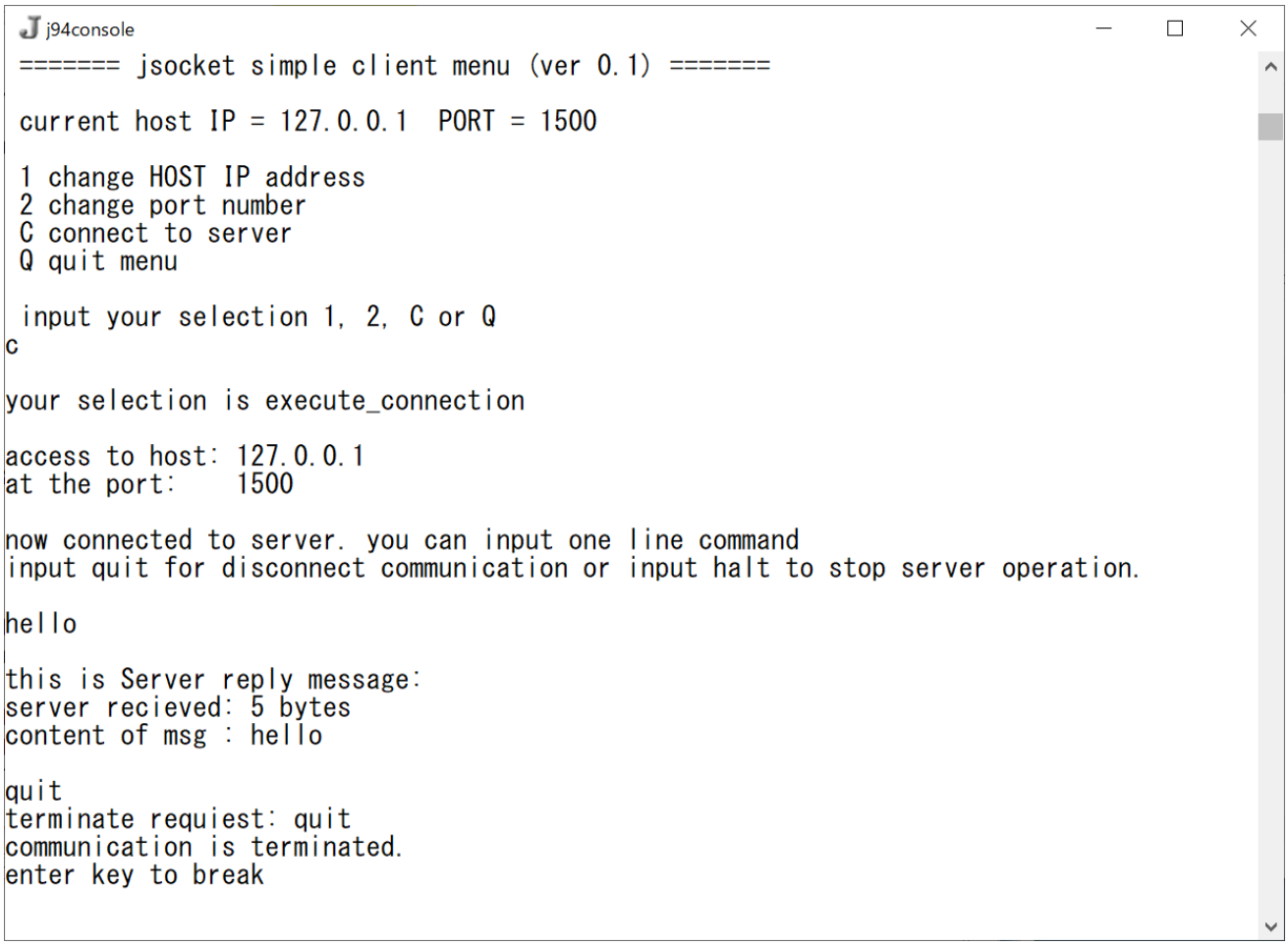

Figure 7: simple client menu C was issued and connected to the simple server. 'hello' was sent followed by server echo back message. Note the server counts the bytes of received message in addition to sending back the contents of the message received. 'quit' is the local command to quit connection.

Figure 8: Windows telnet application uses command prompt for open server, send message, and close connection.

Figure 9: This is the scrren shot of showing server reply message to windows telnet application. Again, server rely message includes the byte count of message received.

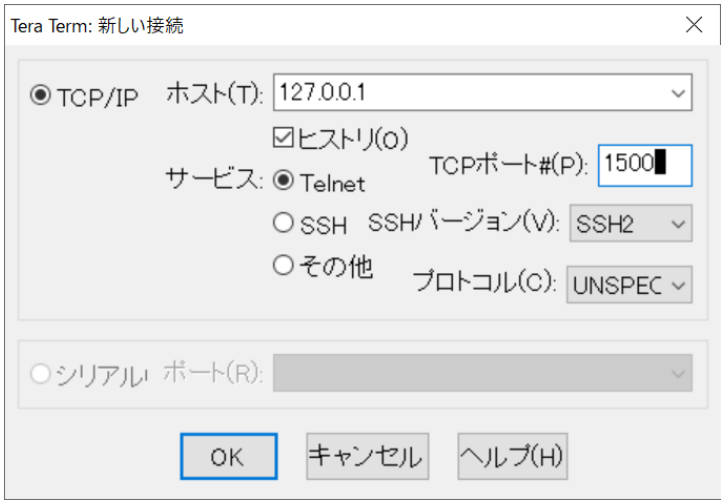

Figure 10: Freeware of telnet application for windows teraterm. Connection panel view. Note the port number has been set 1500.

Figure 11: In teraterm, local echo option is needed because the simple server does not echo back keyin characters in real time.

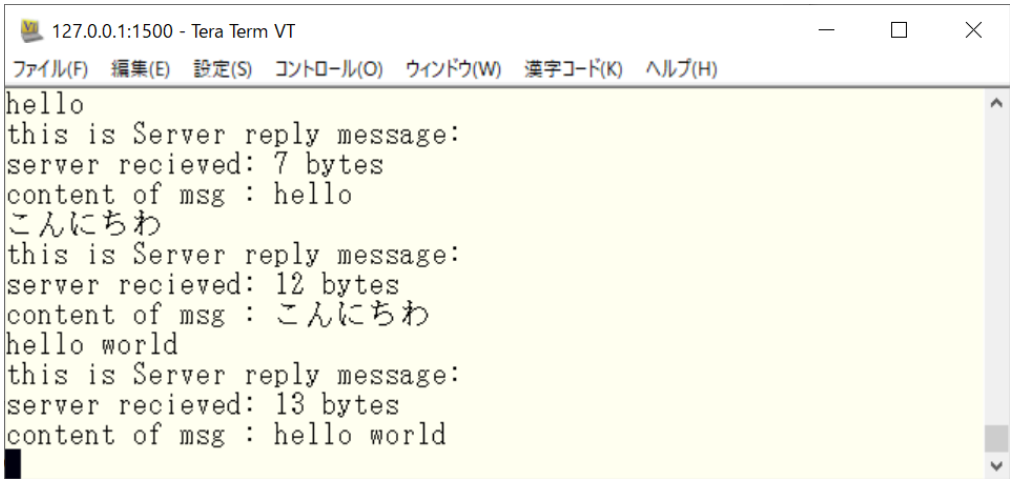

Figure 12: This is the scrren shot of communication between teraterm and the simple server. Note messages of hello, konnichiwa in Japanese and hello world were sent, and server return messages show two bytes greater than the length of words, meaning that teraterm seems to add end of line code, CRLF, in this case.

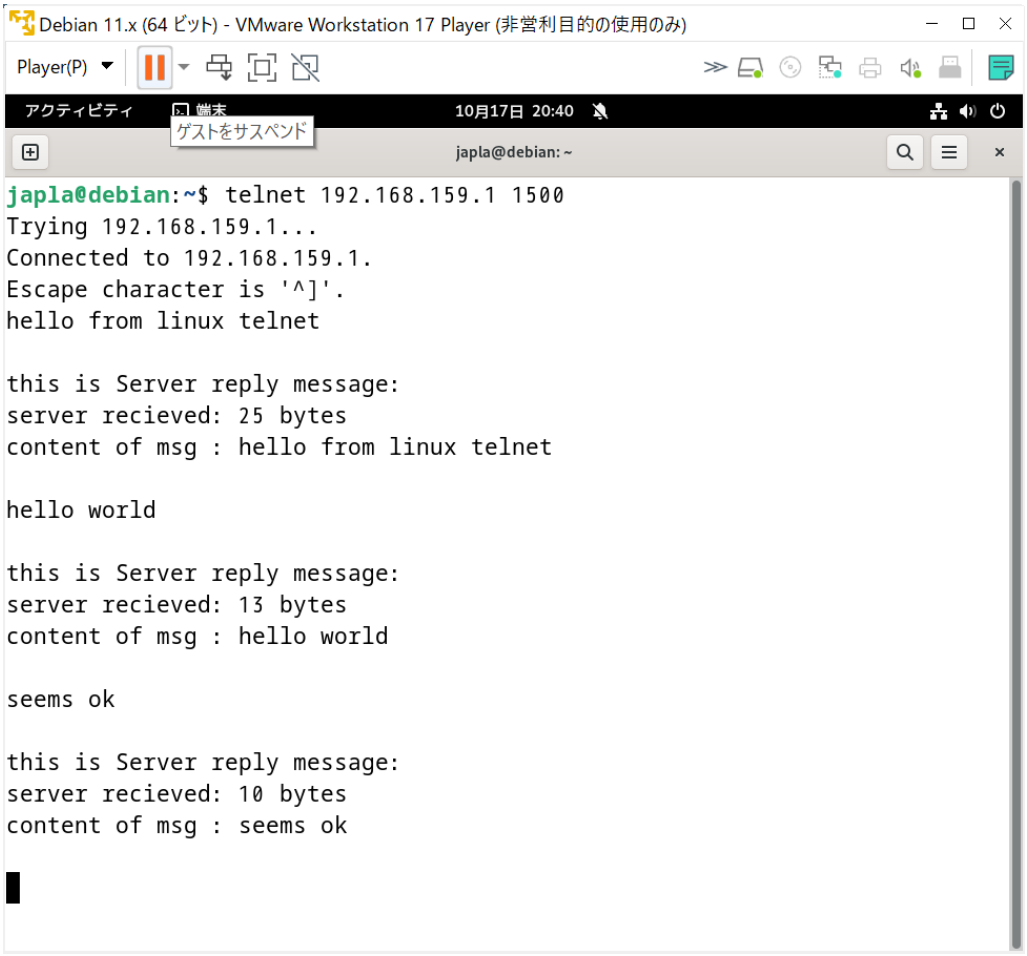

Figure 13: This is the screen capture of telnet connection to the simple server from Debian Linux version 12 installed in VMware Workstation 17 Player for windows. THe host IP address in the virtual network was 192.168.159.1 So telnet command is 'telnet 192.168.159.1 1500'. This also worked good.

3.2 Simple server connection test: one line J command mode

Then the simple server application was tested in one line J command mode (Figure 14). The access test was done with simple client writtenin J. The simple server now parses client's message as one line J command, executes it and sends back the result as reply message (Figure 15). In this limited version of simple server with one line J command, only one line command that results in atom and list can be handled. If a shape of an answer is matrix or array, the server returns domain error message, because the server does not support for converting an array to CRLF ended strings which is the very requisite in jsocket data transmission.

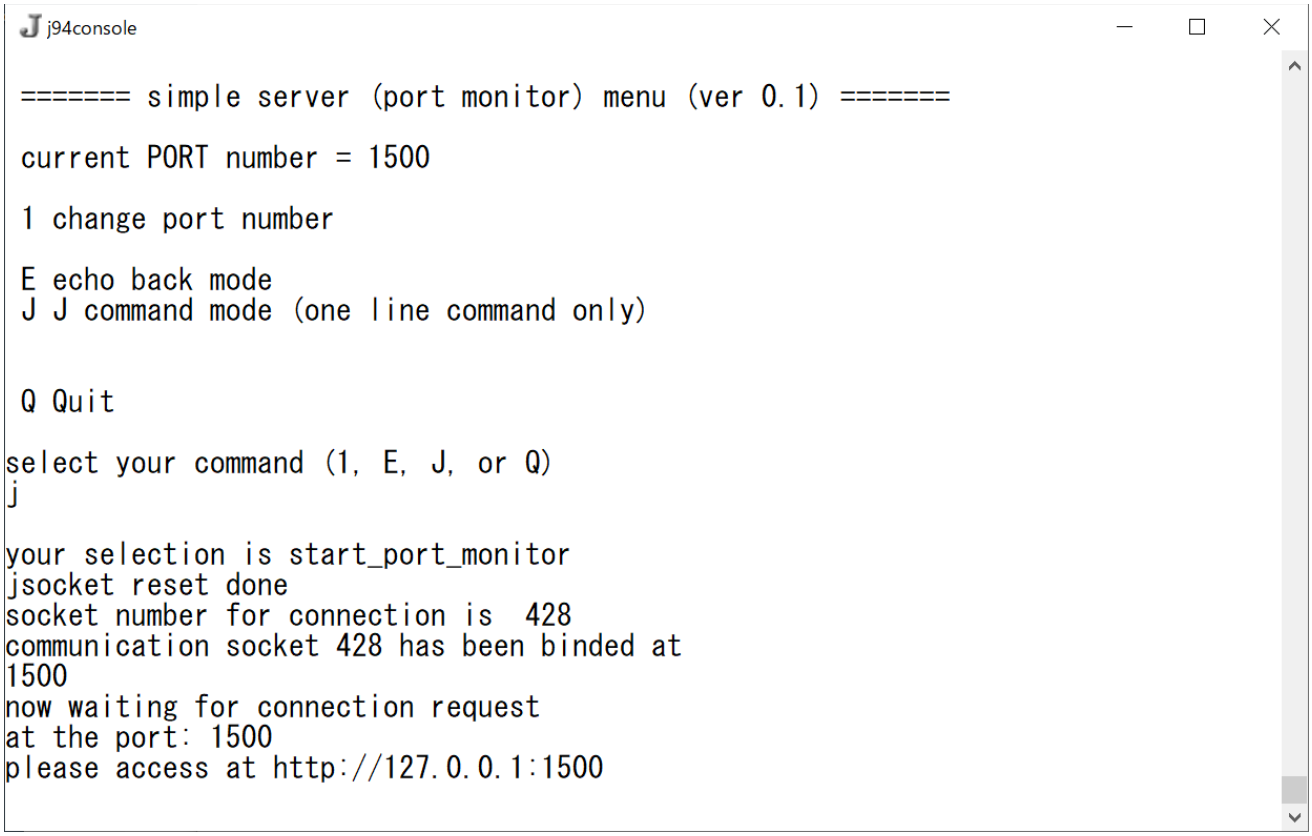

Figure 14: In simple server menu, J command mode was selected. The server is waiting for a connection.

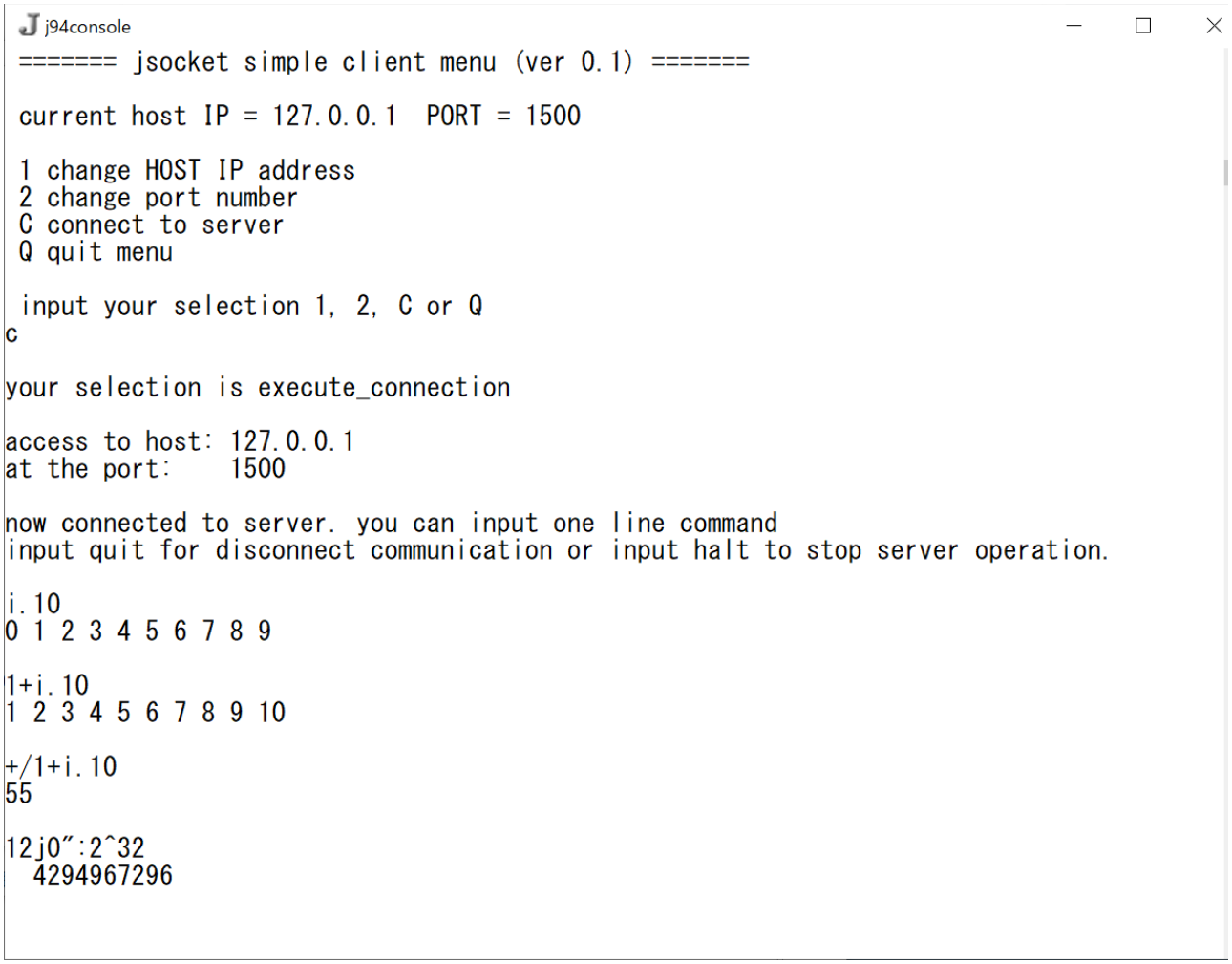

Figure 15: Simple client connected to the server running in one line J command mode. Some one line J commands were sent. Server replied sending result of each command.

3.3 Simple server port monitoring on web browser's connections

In addition, web browser's access to the simple server to monitor browsers HTTP GET method message. This test was performed to evaluate whether browsers can access simple server infinite communication loop which is applied to web form server. The simple server is, essentialy, a functionality of socket port monitor. It can be used to monitor how other process sends message to a host. The analysis of the message from clients is mandatory for development of parser mechanism. Figure 16 shows how to input access URL in a brouwser. Four major web browsers, i.e. Firefox, Microsoft Edge, Google Chrome and Apple Safari, were tested and incoming message from each web browser was monitored of its GET method message to the server. The screen shot of the incoming messages are shown in Figures 17 to 20 respectively. Note the first line of all the messages is GET / HTTP/1.1, and there are differences in content of HTTP header following GET method line.

Firefox
: Figure 17 shows the incoming message from Firefox.

Microsoft Edge
: Figure 18 shows the incoming message from Microsoft Edge.

Chrome
: Figure 19 shows the incoming message from Google Chrome.

Apple Safari
: Figure 20 shows the incoming message from Apple Mac Safari.

By parsing HTTP header of GET method, the server can identify the accessing browser.

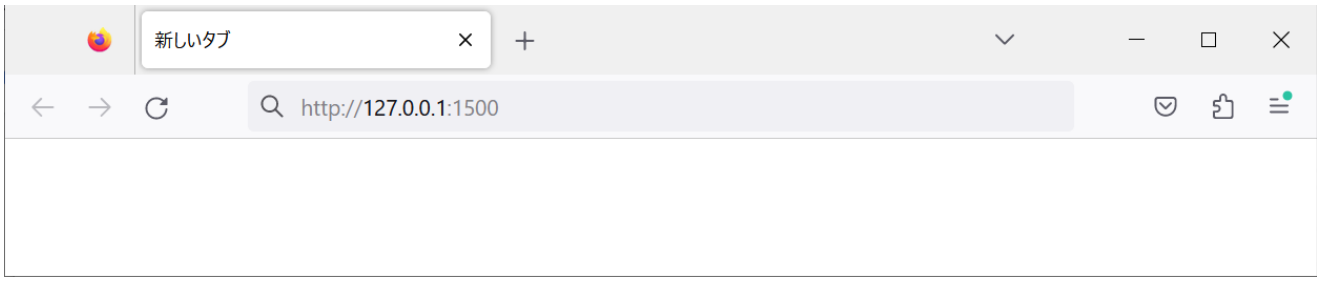

Figure 16: A web browser can access the simple echo back server as shown here. Inputting access URL as http://127.0.0.1:1500 and return key.

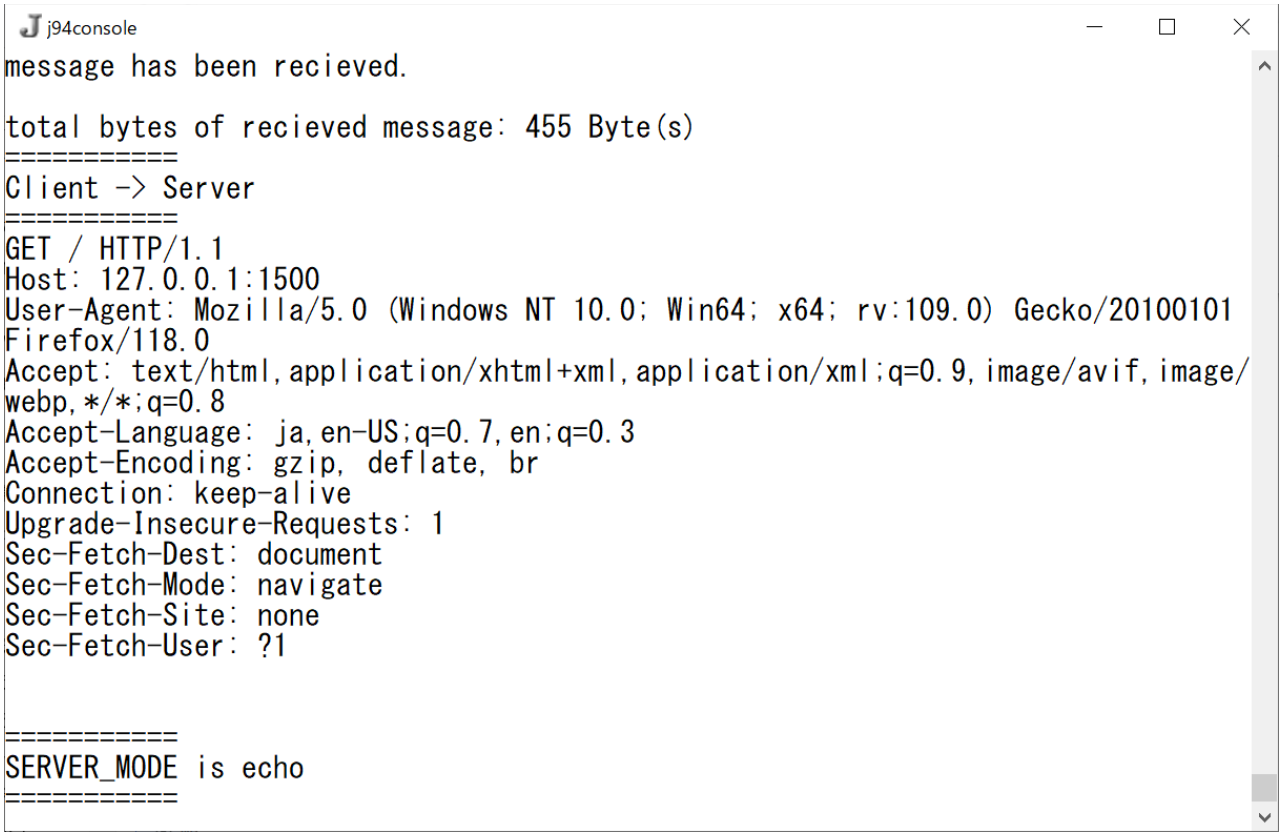

Figure 17: This is the monitoring screen shot of the simple server in echo mode on a connection from Firefox. Note the beginning word is GET. This is GET method of HTTP protocol. Also note the User-Agent contains Firefox. The server can check client's browser by parsing this line.

Figure 18: This is the monitoring screen shot of the simple server in echo mode on a connection from Microsoft Edge. Again, note the beginning word is GET. This is GET method of HTTP protocol. Also note the word Edge is confirmed in sec-ch-ua and User-Agent.

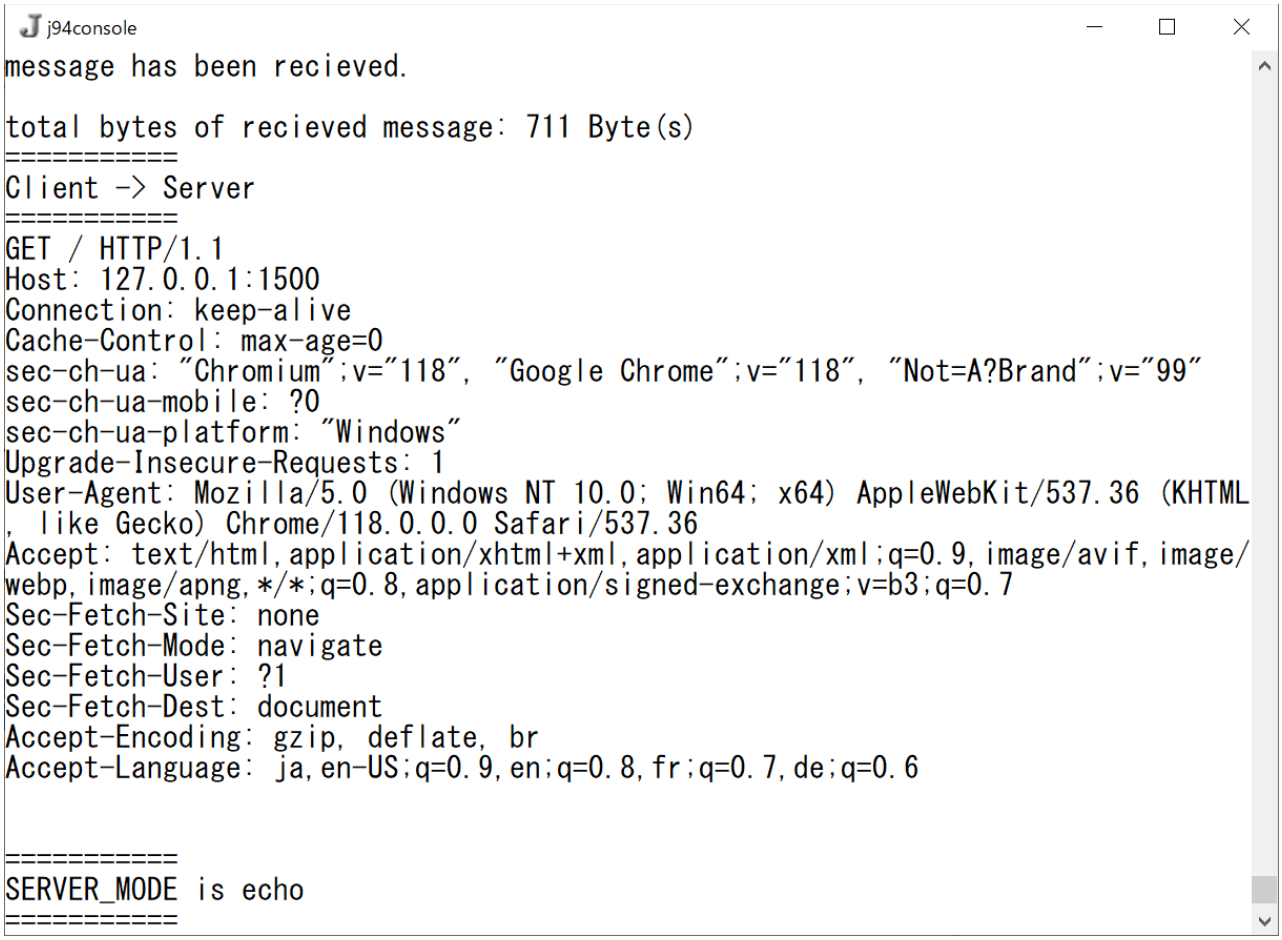

Figure 19: This is the monitoring screen shot of the simple server in echo mode on a connection from Google Chrome. Again, note the beginning word is GET. This is GET method of HTTP protocol. Also note the word Chrome is confirmed in sec-ch-ua and User-Agent.

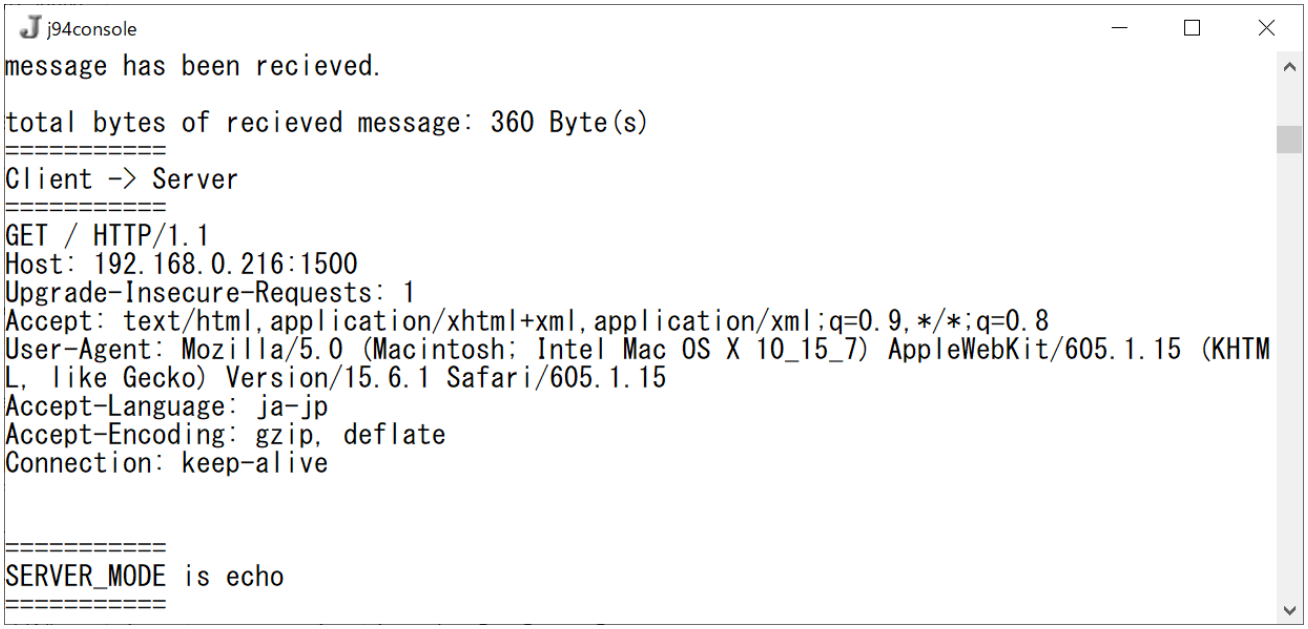

Figure 20: This is the monitoring screen shot of the simple server in echo mode on a connection from Apple Safari. Again, note the beginning word is GET. This is GET method of HTTP protocol. Also note the User-Agent contains Safari and 'host: 192.168.0.216:1500'. Apple mac and Windows are in a local network, 192.168.0.0, and the server IP address is 192.168.0.216.

3.4 Running test of sample web form applications

The following web form applications were tested for running.

```
00_centigrade_fahrenheit_mutual_converter
01_demo_web_form_app_with_available_form_controls_created_by_utility_app
02_00_demo_number_list_reformat_with_no_option_for_reformat
02_01_demo_number_list_reformat_with_column_number_option
02_02_demo_number_list_reformat_with_column_width_option_added
02_03_demo_number_list_reformat_with_decimal_digits_option_added
02_04_demo_number_list_reformat_with_file_save_option_added
02_05_demo_number_list_reformat_with_table_title_option_added
50_print_a_beautiful_triangle_with_figure_length_calculation
51_barrier_option_pricing_model
52_hokusai_komon_image_galary
```

Access test were performed with Firefox, Microsoft Edge, Google Chrome and Apple Safari. As a result, Firefox could communicate with all the web form applications, but the other three could only be good in plain text communication, resulting in socket disconnectiion or hang up in the last sample of

```
52_hokusai_komon_image_galary
```

The analysis to identify the reason for the failure was beyond the author's ability.

In the following, the full work flow of the first sample of

```
00_centigrade_fahrenheit_mutual_converter
```

is shown. They are the screen shots from the selection of the application to disconnection from the server sending server halt request. On issuing menu item 1, the web form server displayed the registered applications as shown in Figure 21. In this case issuing number 0, the server screen displayed application summary and waited for connection and entered in infinite loop of waiting connection request (Figure 22). Firefox client accessed the server at

```
http://127.0.0.1:1500
```

and server sent a reply web form document in response to GET method message (Figure 23). Client issued a request (POST method message) of a conversion of two values from centigrade to fahrenheit and received a response web form document (Figure 24). Then client requested halt of server and received disconnection message from the browser (Figure 25). The server showed a message of halt and back to menu message (Figure 26).

The screen shots of the other sample web form applications are placed at the end section of the report, and they include responses to GET and POST request only.

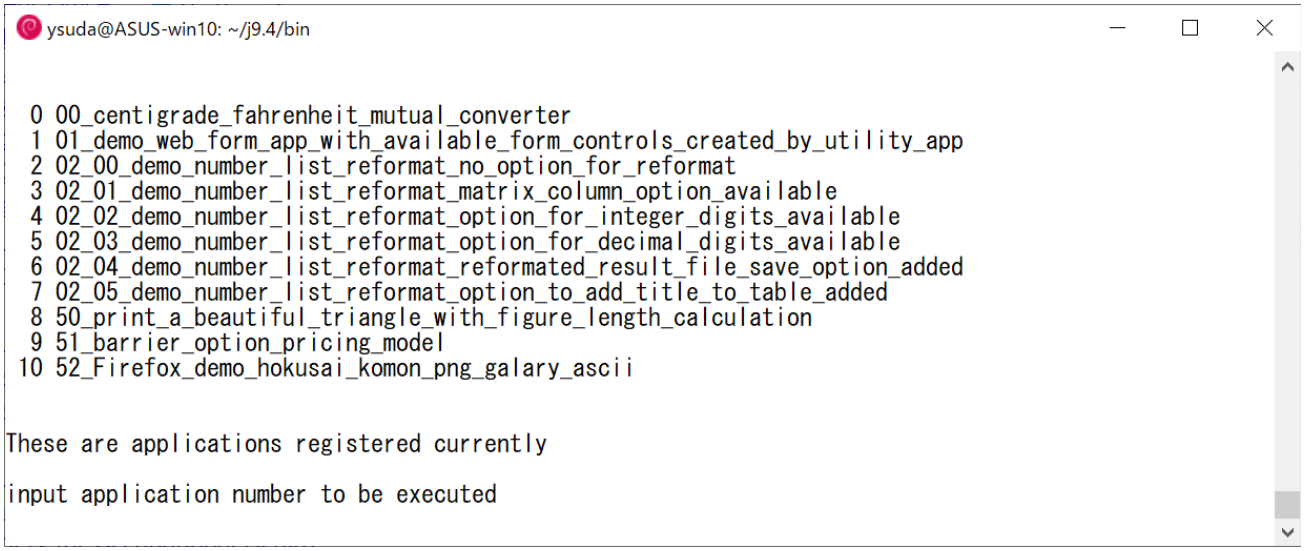

Figure 21: This is the screen shot of the list of web form applications registered. The names of application is sorted and serial number prefix starting 0 is added to each application.

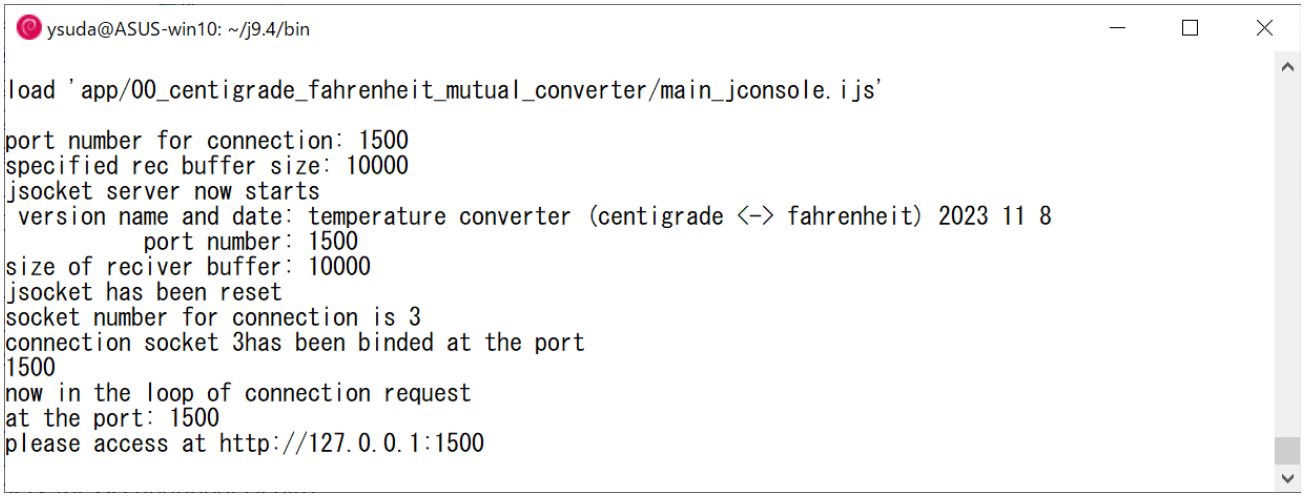

Figure 22: Inputting 0 loads the corresponding web form application, i.e. centigrade fahrenheit mutual converter. The server entered in infitite loop for connection.

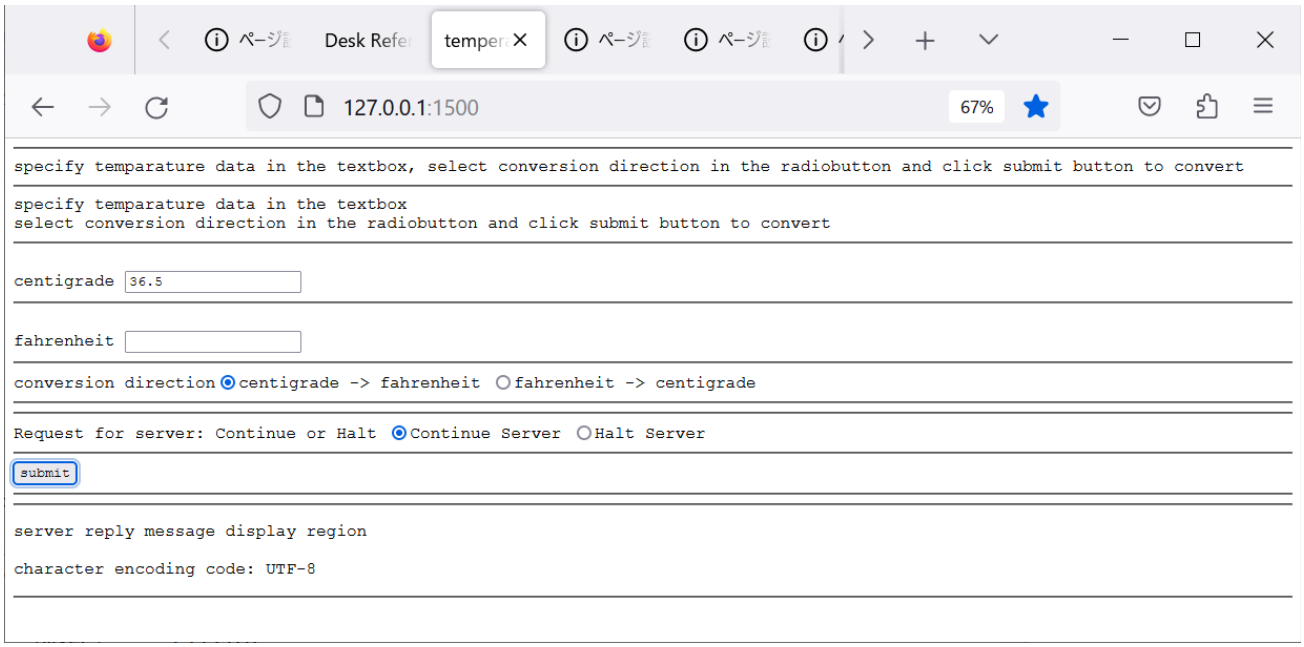

Figure 23: On connection with Firefox and its GET method, the server sent this response web form document. Note the default conversion direction is centigrade to fahrenheit and default value of centigrade is 36.5.

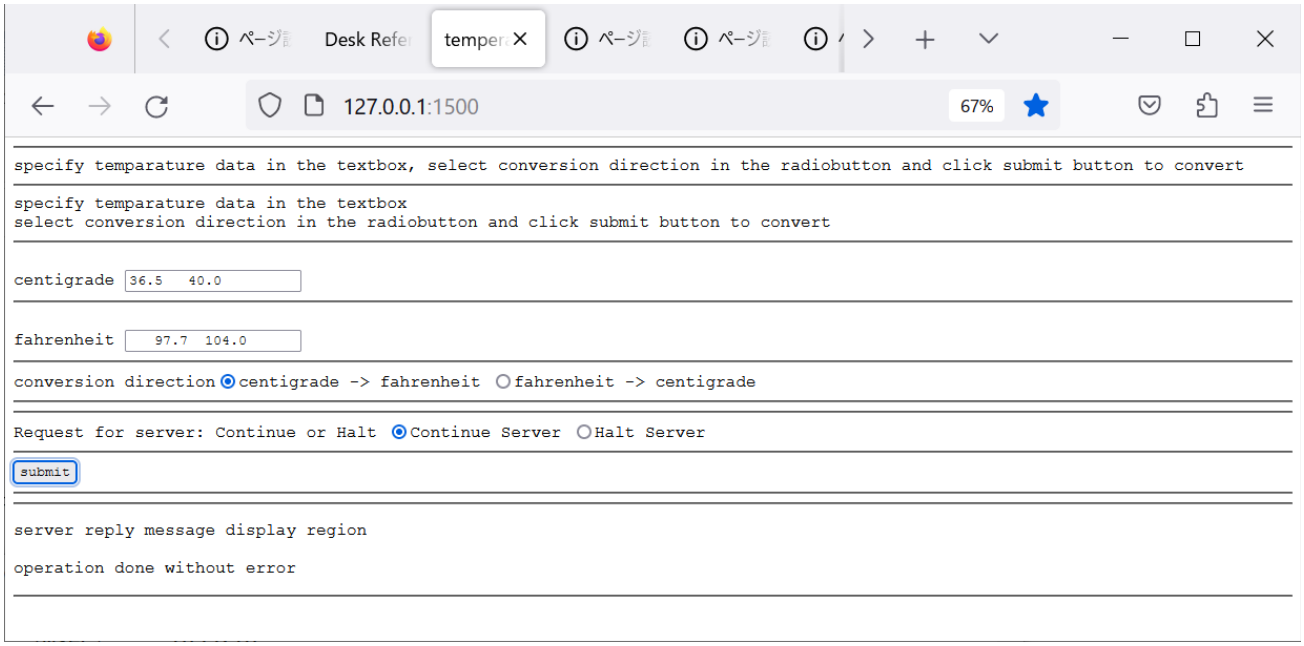

Figure 24: This is the screen shot of response document after submission with two values of centigrade entered in coresponding textbox. Note two conversion values with format of 7j1": in fahrenheit textbox. J engine can handle multiple temperature at a time.

Figure 25: This is the screen shot of Firefox on submitting server halt request. The server disconnects on halt request and the browser acknowledges the disconnection showing this kind of message.

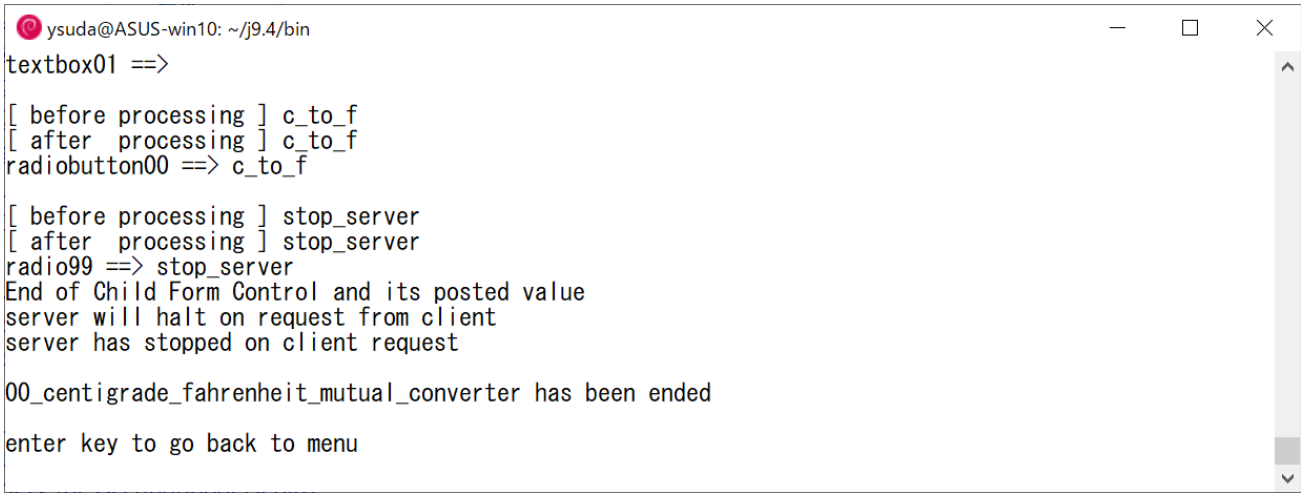

Figure 26: This is the screen shot of the server jconsole on receiving halt request. The web form application stops and the server waits for enter key to go back to main menu.

4 Discussion

A novel web form server for jconsole in jsocket was developed and implemented. Once an original J function is written and validated for its execution in jconsole, adding form contorls to finish as a GUI application can be acomplished with conventional HTML form controls of textbox, textarea, checkbox, checklist, dropdown and radiobutton. The result of J fucntion can be set as values of form controls of textbox and textarea in addition to HTML plain text content.

Moreover, an original J functiion can be developed step by step style using this web form server system, becuase revision process seems to be easy as seen in following sample web form applications for numeric data reformatting,

```
02_00_demo_number_list_reformat_with_no_option_for_reformat
02_01_demo_number_list_reformat_with_column_number_option
02_02_demo_number_list_reformat_with_column_width_option_added
02_03_demo_number_list_reformat_with_decimal_digits_option_added
02_04_demo_number_list_reformat_with_file_save_option_added
02_05_demo_number_list_reformat_with_table_title_option_added
```

The current version is also able to send bmp image in inline frame control. But there are two limitations in handling image response. The one is that the number of inline frame for image is just single. This may be increased in future version.

The other issue is that among browsers of Firefox, Microsoft Edge, Google Chrome and Apple Safari, only Firefox can receive inline image. Finding difference between Firefox and the other three browsers has been beyond the authors's ability.

And as for image preparation, the web form server itself does not provide any jconsole based tool for image creation, i.e. converting numeric result to image presentation. The images used in the application of

```
52_hokusai_komon_image_galary
```

were drawn with the original form application in j602a for Windows. The sample application just uses J's addon packages of bmp and png for converting the format of predrawn png image to bmp. In order to visualize numeric result of J function, J's addon packages of plot, viewmat, bmp, png, and others that are not dependent of Qt window driver are greatly helpful. But the packages that depend on Qt window driver can not be applicable, since the web form server is only executable in jconsole.

In summary, a novel web form based GUI environment for J function was implemented. Although the functionality is very fundamental, it is feasible and practical to add forms to J function.

5 Registered names for globals and functions

There are a number of global variables and functions in the web form server system. In order to avoid conflictions between user's and system's globals and functions, user must check the folloing list whether there occurs confliction. The task of reviewing through the list seems to be time consuming. And the easiest way to avoid confiction among names is to add an unique suffix of user's name initial (in may case, _ys) at the end of each useer's globals and functions described in the application definition file.

```
A
ACCESS_METHOD
APPLICATION_DATE
APPLICATION_NAME
APP_DEF_FILE_READY
APP_NAME_CONFLICTED_AND_DELETED_FLG
APP_NAME_CONFLICTED_FLG
APP_PLAN

B
BODY_END
BODY_TAG
BOXED_TWO_CHAR_DIGIT_00_TO_98
BOXED_TWO_DIGIT_CONTROL_ID_CHECKBOX
BOXED_TWO_DIGIT_CONTROL_ID_CHECKLIST
BOXED_TWO_DIGIT_CONTROL_ID_DROPDOWN
BOXED_TWO_DIGIT_CONTROL_ID_RADIOBUTTON
BOXED_TWO_DIGIT_CONTROL_ID_TEXTAREA
BOXED_TWO_DIGIT_CONTROL_ID_TEXTBOX

C
CHECKBOX_??_VALUE  (where ?? is 00 to 98)
CHECKLIST_??_VALUE (where ?? is 00 to 98)
CHECK_RESULT
COMMAND_NAME
CONTENT_OF_DIR
CRLF
CURRENT_APP_IN_ONE_STR_SEPARATED_WITH_CRLF
CURRENT_APP_IN_ONE_STR_SEPARATED_WITH_CRLF_with_number
CURRENT_OS

D
DEBUG_MSG
DROPDOWN_??_VALUE (where ?? is 00 to 98)

E
ENCODING_CODE
ENCODING_CODE_INFO
ERROR_MSG

F
FORM_CONTROL_DEF_ERR
FORM_END
FORM_TAG

H
HEAD_END
HEAD_TAG
```

HITLINE
HOST_IP
HOST_PORT
HTML_END
HTML_HEADER_PART
HTML_TAG

I
IFRAME_BORDER
IFRAME_SIZE_H
IFRAME_SIZE_W
IMAGE_BMP
INSTALL_DIR

M
MAC_LINUX_HOME_DIR
MAIN_MENU_QUIT_FLG
MSG_FROM_CLIENT
MSG_FROM_SERVER
MSG_FROM_SERVER_FLG
MY_Target_host_address

N
NEW_APP_DIR
NEXT_APP_NAME

P
PAGE_DESCRIPTION
PAGE_TITLE
PORT_FOR_SERVER
POSTED_NEWLINE
PRE_00

R
RADIO99_VALUE
RADIO_??_VALUE (where ?? is 00 to 98)
RADIO_99
REC_BUF_SIZE_FOR_TERM
RESPONSE_IMG_FLG
RESPONSE_IMG_TYP
RETURN_MSG
RGB_to_BGR

S
SDRECV_BUFFER_SIZE
SEND_BUTTON
SERVER_MODE
SKCLIENT
SKLISTEN
SKSERVER
SOURCE_DIRECTORY
SUB_DIR

T
TEXTAREA_??_VALUE (where ?? is 00 to 98)
TEXTBOX_??_VALUE (where ?? is 00 to 98)
TIME_AT_END_OF_DISPLAY_MSG

```
TIME_AT_END_OF_SEND_MSG
TIME_ON_ARRIVAL
TITLE_END
TITLE_TAG
TOTAL_BYTE_RECEIVED_IN_STR
TRANSLATE_SHELL_FLG
TXA_99

U
USER_AGENT_FIREFOX

W
WINDOWS_HOME_DIR

Y
YOUR_JOB
YOUR_OS

a
activate_menu_2
app_delete_question
app_name_check_and_rmapp_if_exist
app_reload_question
append_it
asciiChr

b
begin
begin_jsocket_simple_server_or_client_in_jconsole
begin_jsocket_web_form_server_for_j_function_in_jconsole
bye

c
cddir
cdir
change_dir
change_target_host_IP
change_target_host_port
change_working_directory
check_def_line_format
check_full_line_match_with_x_in_y_CRLF_separated_str
check_hankaku_number
check_layout
check_spec_checkbox_def
check_spec_checklist_def
check_spec_dropdown_def
check_spec_radiobutton_def
check_spec_textarea_def
check_spec_textbox_def
count_items_with_separator_chr_x
create_action_to_POST_method_jconsole_ijs
create_create_response_message_to_GET_method_jconsole_ijs
create_ijs_src_for_default_values_of_textbox_and_textarea_and_others
create_ijs_src_for_extract_entryID_value_for_all_the_form_controls_planned
create_jconsole_applications_jconsole_ijs
current_min_and_sec_in_msec
```

```
d
date_time
decode_escaped_ascii_char
define_function_for_monitoring_on_extracted_values
define_html_code_for_controls_based_on_plan_ijs
define_new_app_plan_src
dir
dispMsg
disp_client_terminal_menu
disp_menu
disp_port_monitor_server_menu

e
esc
exclude_specific_line_with_x_str_and_return_result
execute_connection
execute_your_job
exit
extract_POSTed_values
extract_entryID_value
extract_entryID_value_and_display
extract_func_name_in_ijs
extract_planed_app_html_remark
extract_planed_app_html_title
extract_planed_app_name_folder
extract_planned_app_name_folder_without_space
extract_specific_first_line_with_x_str
extract_specific_line_with_x_str
extract_specific_line_with_x_str_and_return_result
extract_specific_line_with_x_str_and_return_result_org
extract_web_app_def_lines
extract_web_control_def_lines

f
file_size
fs

g
get_current_J_version
get_current_OS
get_home_directory

h
hifen_to_underbar
hifen_to_underbar_all

i
initialize_ip_and_port
initialize_response_image_bmp
initialize_response_image_png
input_port_number_for_socket
issue_making_def_file_web_app_and_activate_menu_2

j
jserver_menu
jsocket_client_terminal
jsocket_simple_client_menu
```

```
jsocket_simple_server_menu

k
keyin

l
line_count_check_of_multi_line_str_y
list_current_app
list_current_app_without_app_exec
load_script

m
make_html_form_doc
make_html_header_part
make_main_jconsole_ijs
make_new_app_directory
make_new_web_form_app_for_jconsole
mkdir
monitor_entry_ID_values

n
num_of_LF_lines_in_y_str
num_of_items
num_of_items_separated_with_space_in_aLine
num_of_lines_in_y_str
number_of_items_separated_with_x_str_in_y_str

o
one_chr_menu_selection
one_chr_menu_selection_server_mode

p
parse_form_control_and_create_ijs_for_initial_values_for_textbox_and_textarea_and_others
parse_form_control_line_and_create_ijs_for_extract_entryID_value_statement
pick_up_items_all_with_separator_chr_2x
pick_up_items_all_with_separator_chr_x
pick_up_x_th_item_from_plus_separated_str_y
pick_up_x_th_item_from_slash_separated_str_y
pick_up_x_th_item_from_space_separated_str_y
plus_to_space
plus_to_space_all
position_of_str_x_in_y_str
prepare_boxed_two_char_digit_00_to_98
prepare_http_response_message_to_GET_method
prepare_http_response_message_to_GET_response_image_bmp_request
prepare_http_response_message_to_POST_method
prepare_http_response_status_and_header_field
prepare_response_message_echo_back_mode
prepare_response_message_j_one_line_command_mode
prepare_string_response_message_to_JCMD_method
print
print2
print_current_directory
pwd

r
read_fn_script
```

read_from_file
read_ijs_script
rec_msg
register_newly_created_app_def_file_to_working_dir
reload_app_def_file
replace_CRLF_with_space
replace_CRLF_with_space_all
replace_LF_with_CRLF
replace_LF_with_CRLF_all
replace_newline_code
replace_newline_code_all
reset_socket
resume
rff
rm
rmapp
rmapp2

s
save_array_data_as_CRLF_ended_text_file
save_array_number_data_as_text_file
save_current_hokusai_komon_img_as_bmpfile
save_current_jturtle_graph_as_bmp_file
save_current_jturtle_window_as_bmp_file
save_current_plot_as_bmp_file
save_gl3lab_as_bmp_file
save_matrix_number_data_as_text_file
save_opengl_parent_win_as_bmp_file
screen_out
screen_out_preceded_by_CRLF
scroll_up_screen
sdrecv_one_more
sdrecv_one_msg
show_all_lines
ssc
sss
start
start_port_monitor
start_server
start_simple_client
start_simple_server
start_socket_server
start_web_form_server_for_j_function
store_all_lines_in_boxes

t
take_LF_one_line_from_lines
take_first_word
take_first_word_plus
take_first_word_slash
take_first_word_with_sep_chr_x
take_one_line
take_one_line_from_lines
transform_checkbox_def
transform_checklist_def
transform_dropdown_def
transform_radiobutton_def

```
transform_textarea_def
transform_textbox_def
transform_web_def_line_to_html_src

u
underbar_to_hifen
underbar_to_hifen_all

v
view_one_line_all
view_x_th_line_in_y_multiline

w
write_str_to_file
wtf

x
x_pos_y
```

6 Screen shots of sample web form applications

The following is the list of the rest of web form sample applications tested. The screen shots of

```
00_centigrade_fahrenheit_mutual_converter
```

are excluded, since they have been included in the main text of Result section.

```
01_demo_web_form_app_with_available_form_controls_created_by_utility_app
02_00_demo_number_list_reformat_with_no_option_for_reformat
02_01_demo_number_list_reformat_with_column_number_option
02_02_demo_number_list_reformat_with_column_width_option_added
02_03_demo_number_list_reformat_with_decimal_digits_option_added
02_04_demo_number_list_reformat_with_file_save_option_added
02_05_demo_number_list_reformat_with_table_title_option_added
50_print_a_beautiful_triangle_with_figure_length_calculation
51_barrier_option_pricing_model
52_hokusai_komon_image_galary
```

The following figures are screen shots of server response HTML documents to GET and POST method from the browser.

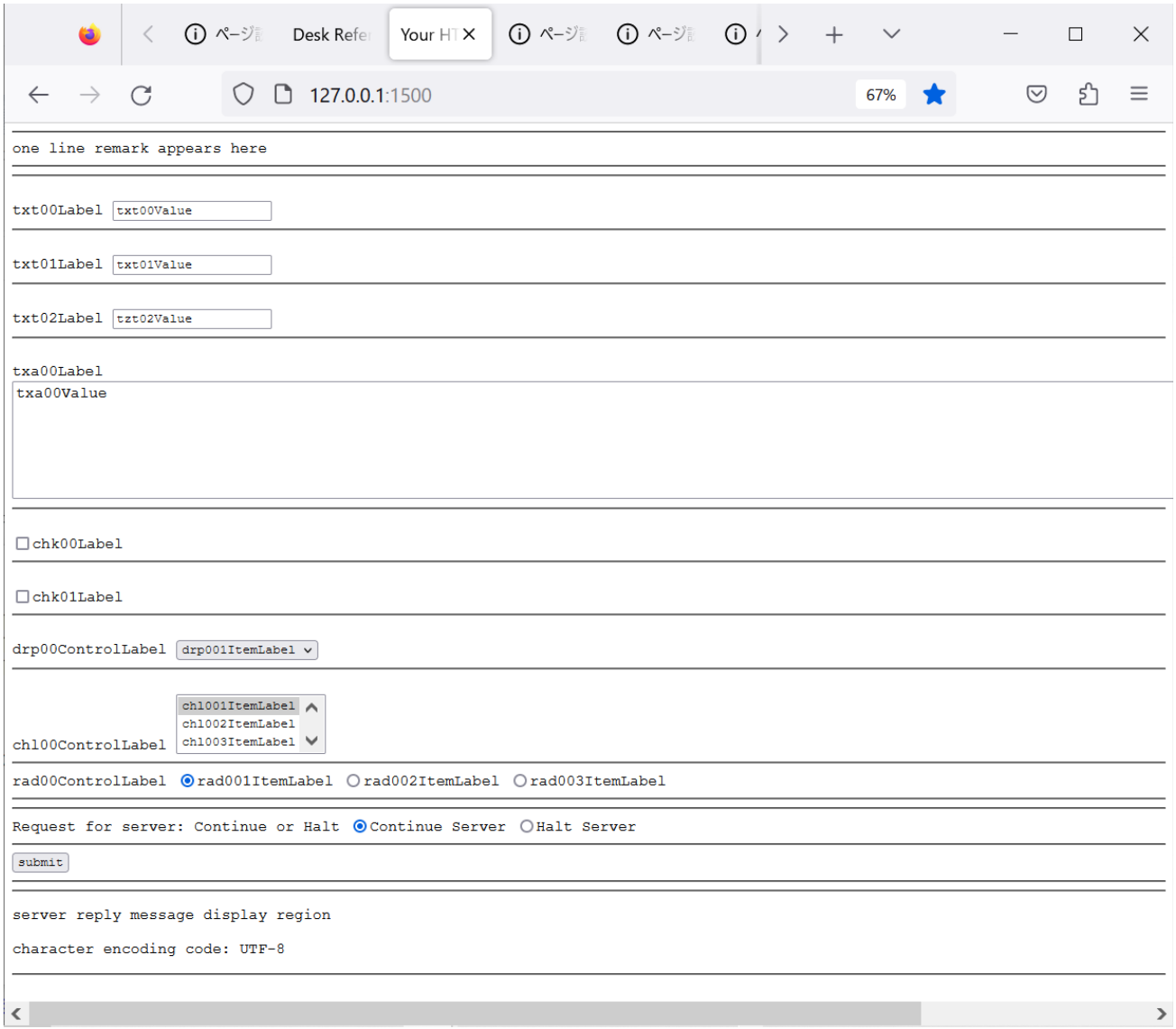

Figure 27: 01_demo_web_form_app_with_available_form_controls_created_by_utility_app_GET. The screen shot on accessing the server shows sample web form application created by utiltiy web form application. This application is just to show all the web form control available in the utility application. The labels and default values are not editted yet.

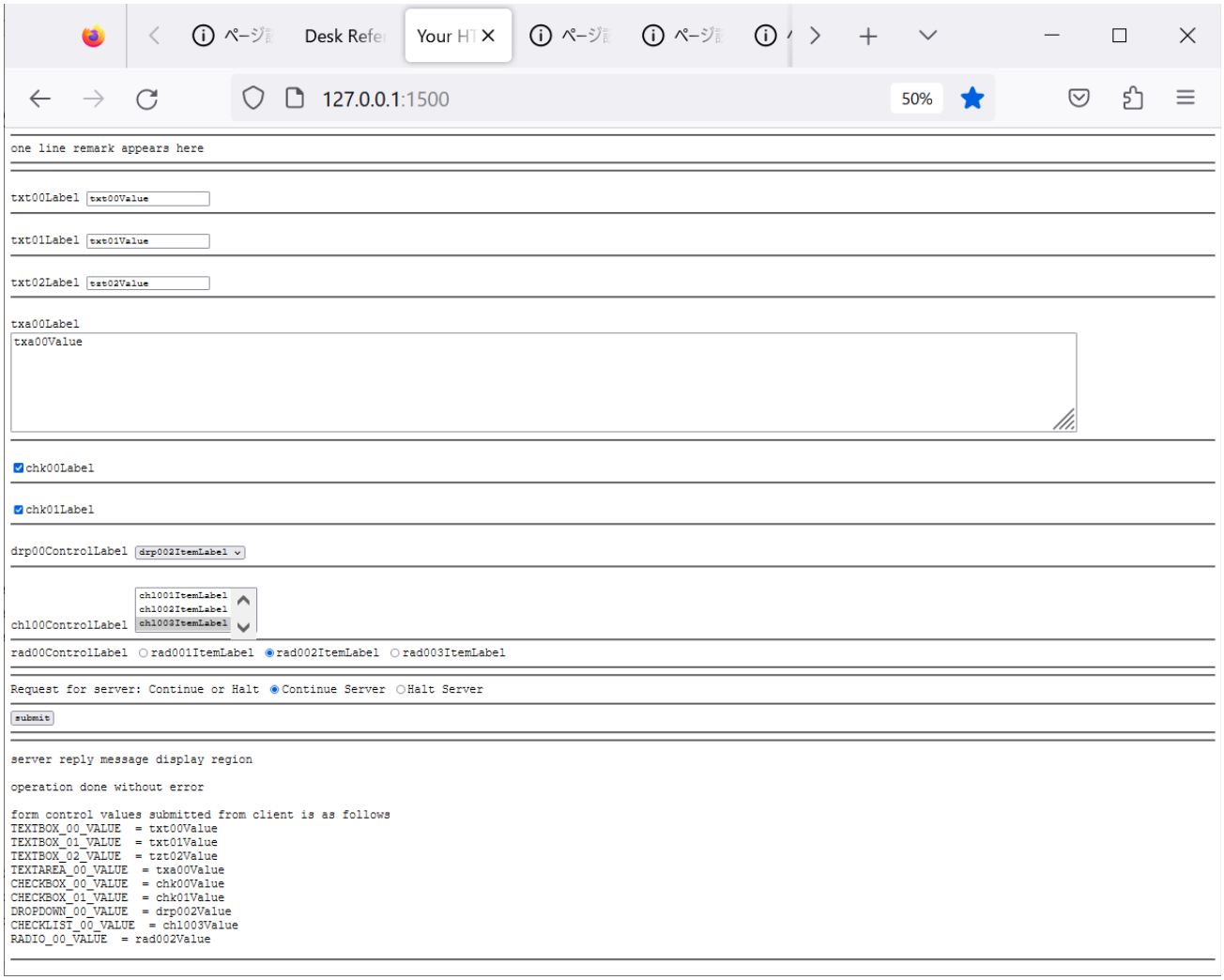

Figure 28: 01_demo_web_form_app_with_available_form_controls_created_by_utility_app_POST. Note, at the bottom of the document, there is a collection fo POSTed values of each web form control.

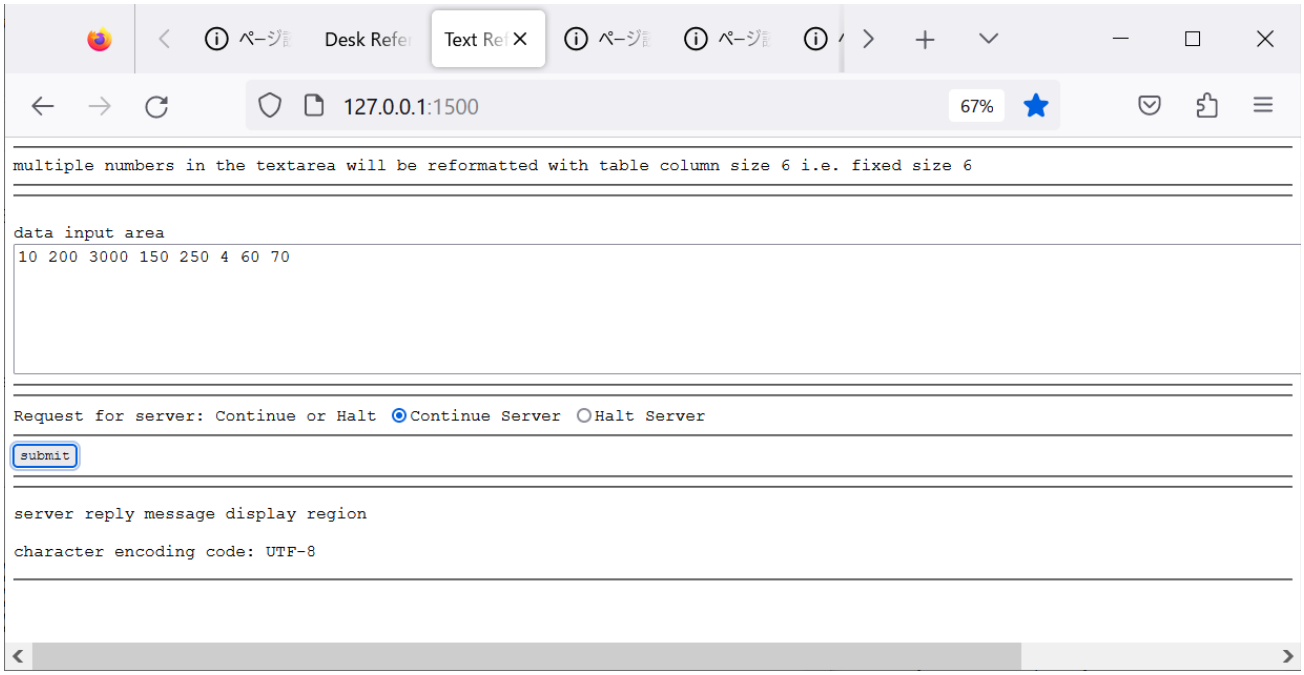

Figure 29: 02_00_demo_number_list_reformat_with_no_option_for_reformat_GET. Note there is one textarea for entry of numeric data to be reformatted.

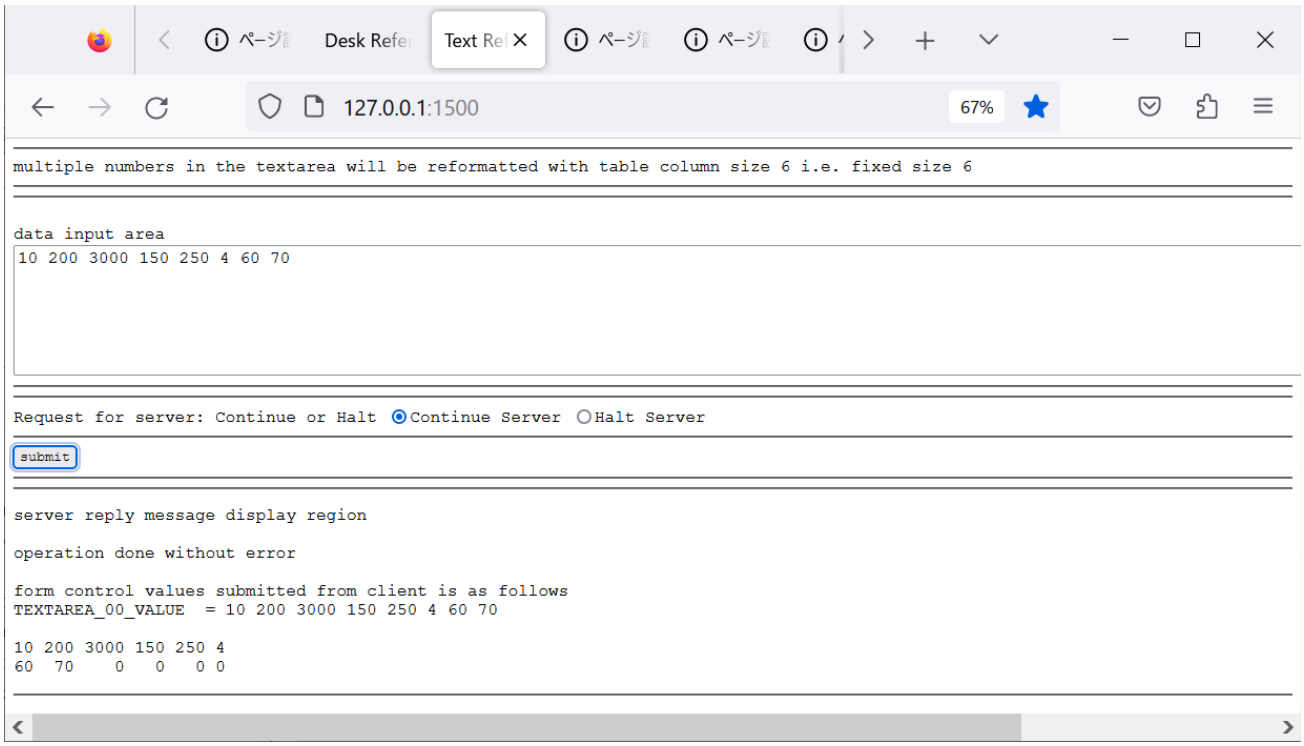

Figure 30: 02_00_demo_number_list_reformat_with_no_option_for_reformat_POST. Note at the bottom of HTML document there is server reply message display region that contains operation status (done without error), list of form control value and the operation result. After submission, the numbers are reformatted as 6 columns and 2 rows. In this web form application, there is no option for shape of reformat. The default reformat definition of this page is 6 column matrix adding dummy zero's as needed. Open the the definition file with main menu command R and E, and confirm the script of YOUR_JOB there.

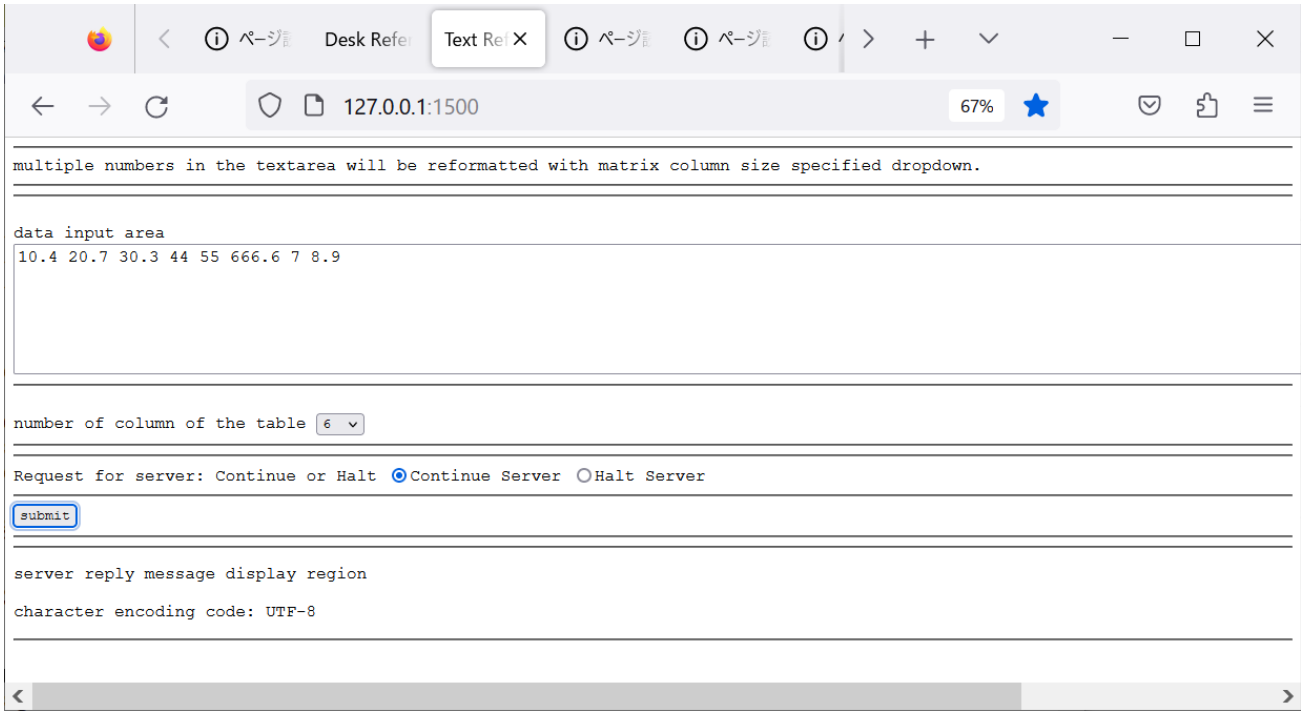

Figure 31: 02_01_demo_number_list_reformat_with_column_number_option_GET. Note in addition to the textarea there is one dropdown for number of column.

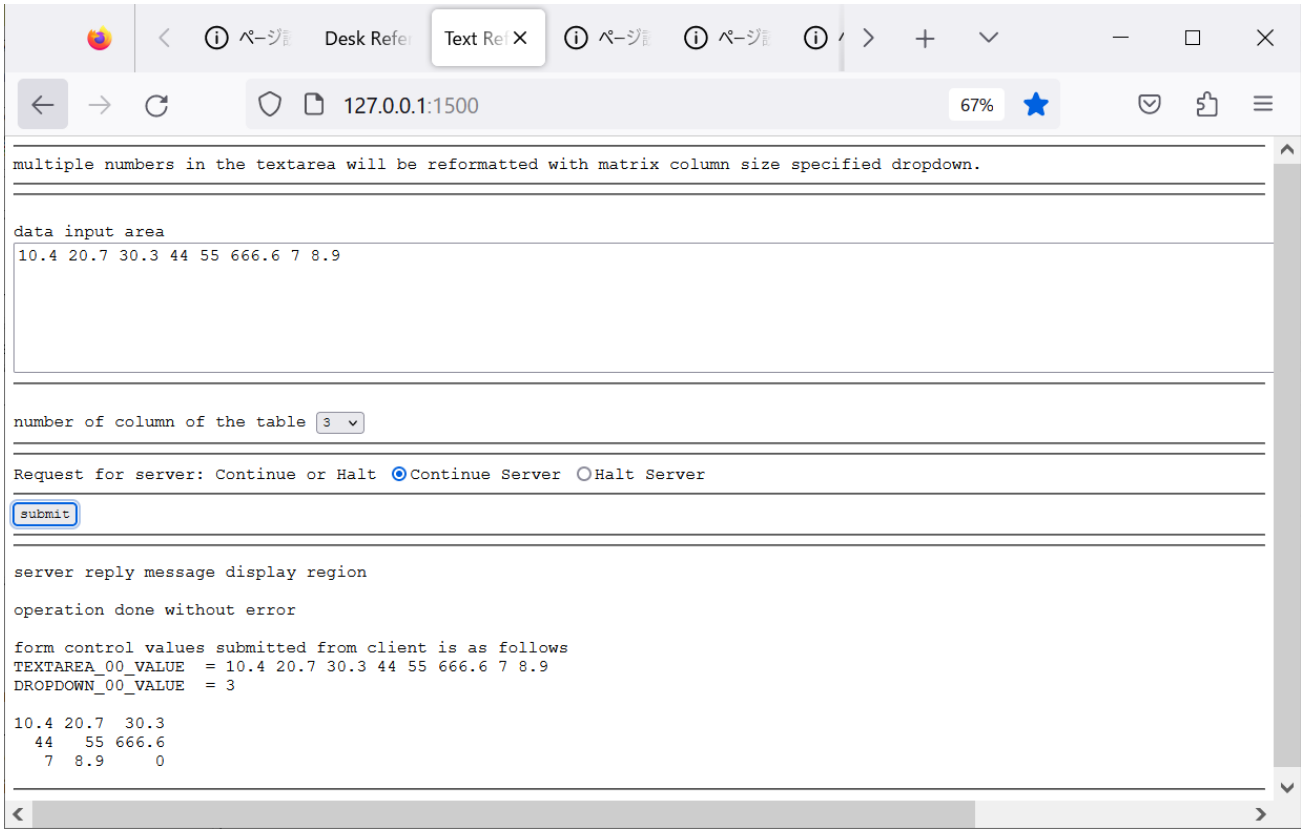

Figure 32: 02_01_demo_number_list_reformat_with_column_number_option_POST. Note the list of form control values are now textarea and dropdown. The selection of column size is 3 and the result is 3 columns and 3 rows with added zero.

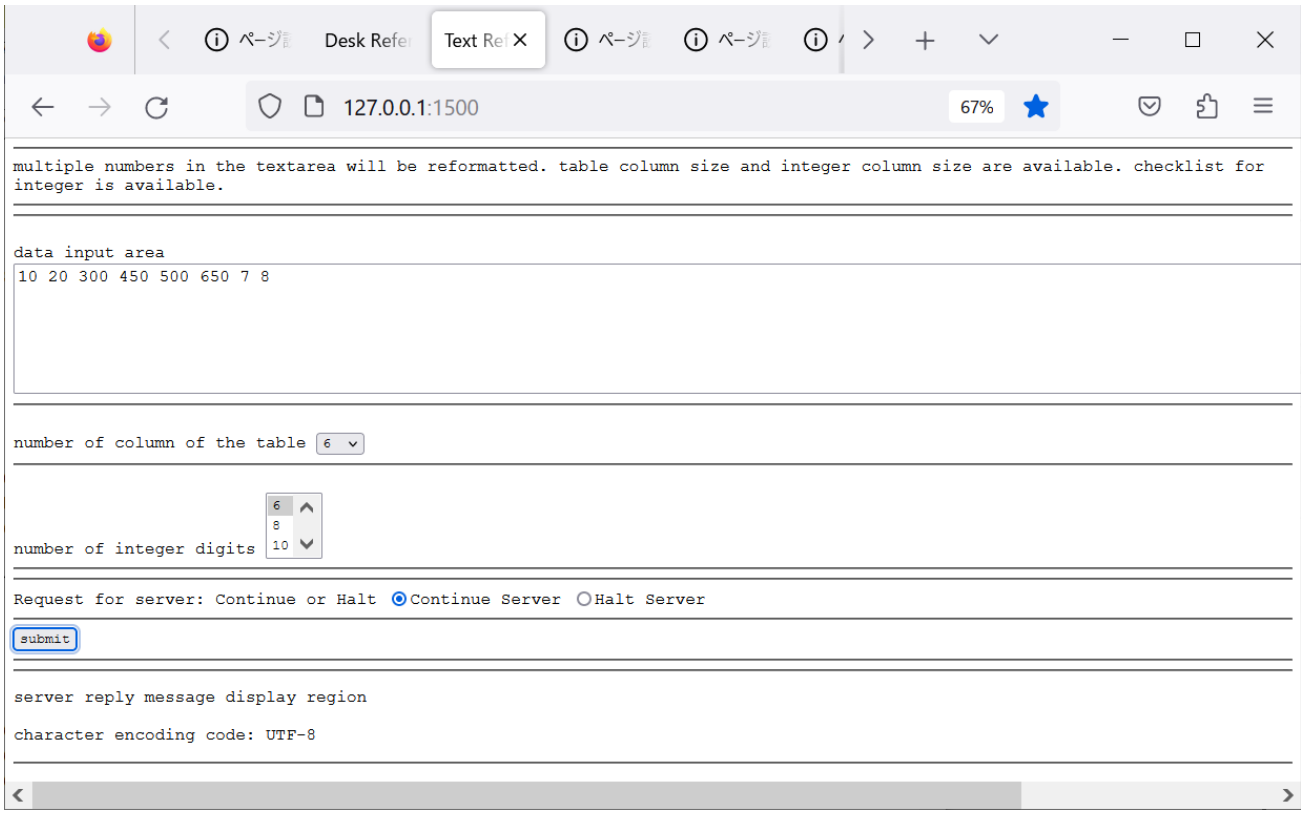

Figure 33: 02_02_demo_number_list_reformat_with_column_width_option_added_GET. Note One checklist for number of integer digit for refomatting is added below the dropdown for column size.

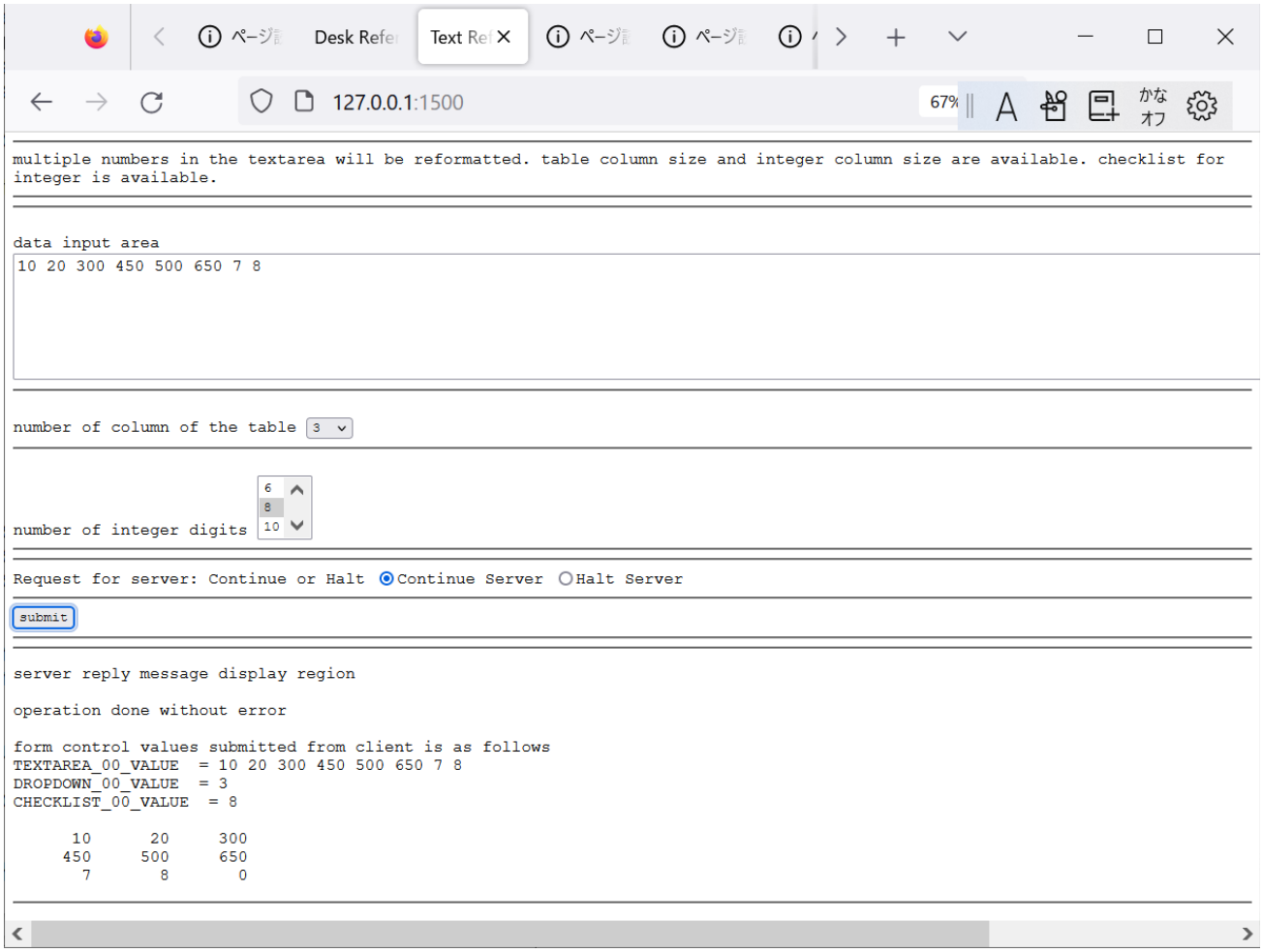

Figure 34: 02_02_demo_number_list_reformat_with_column_width_option_added_POST. Note the list of form control values has checklist value 8 in addition to textarea and dropdown, and the resulting reformat is 3 columns and 3 rows with added zero's.

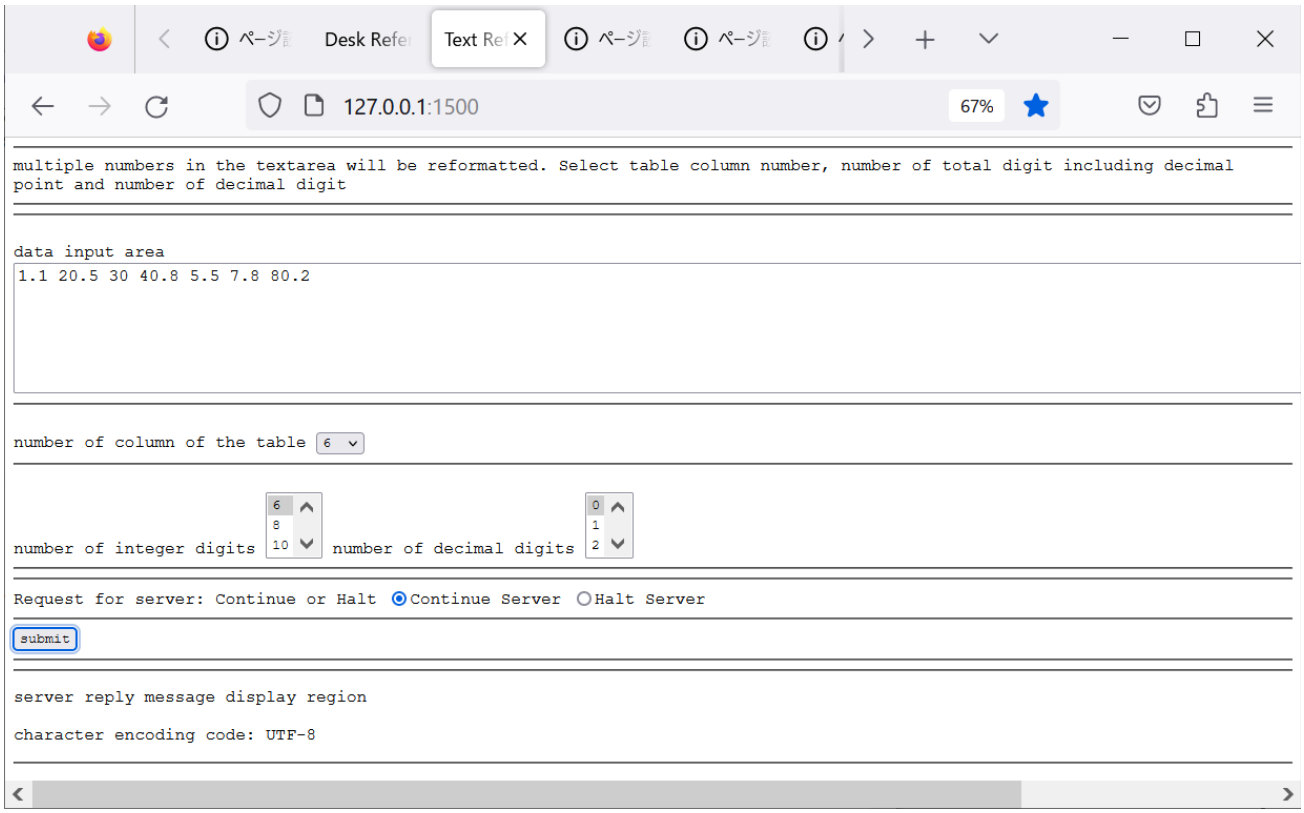

Figure 35: 02_03_demo_number_list_reformat_with_decimal_digits_option_added_GET. Note another checklist for decimal digit size is added with horizontal layout.

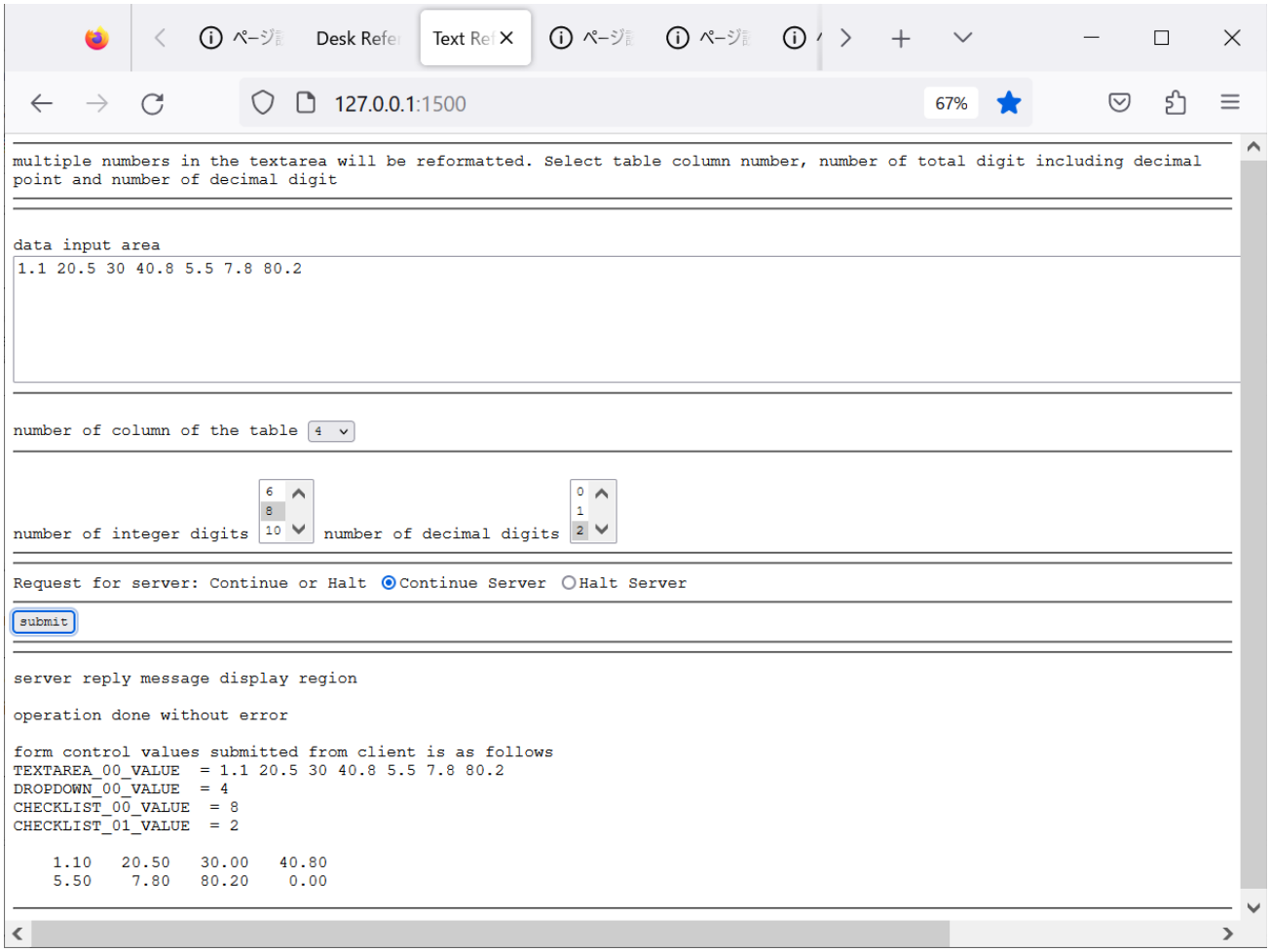

Figure 36: 02_03_demo_number_list_reformat_with_decimal_digits_option_added_POST. Note the second checklist value is added to the list of form control values and the resuling reformat is 4 columns, 2 rows, integer digit 8, and decimal digit 2. As integer digit value and decimal digit are used as 8j2 reformat function, actual ingeger digit is 5.

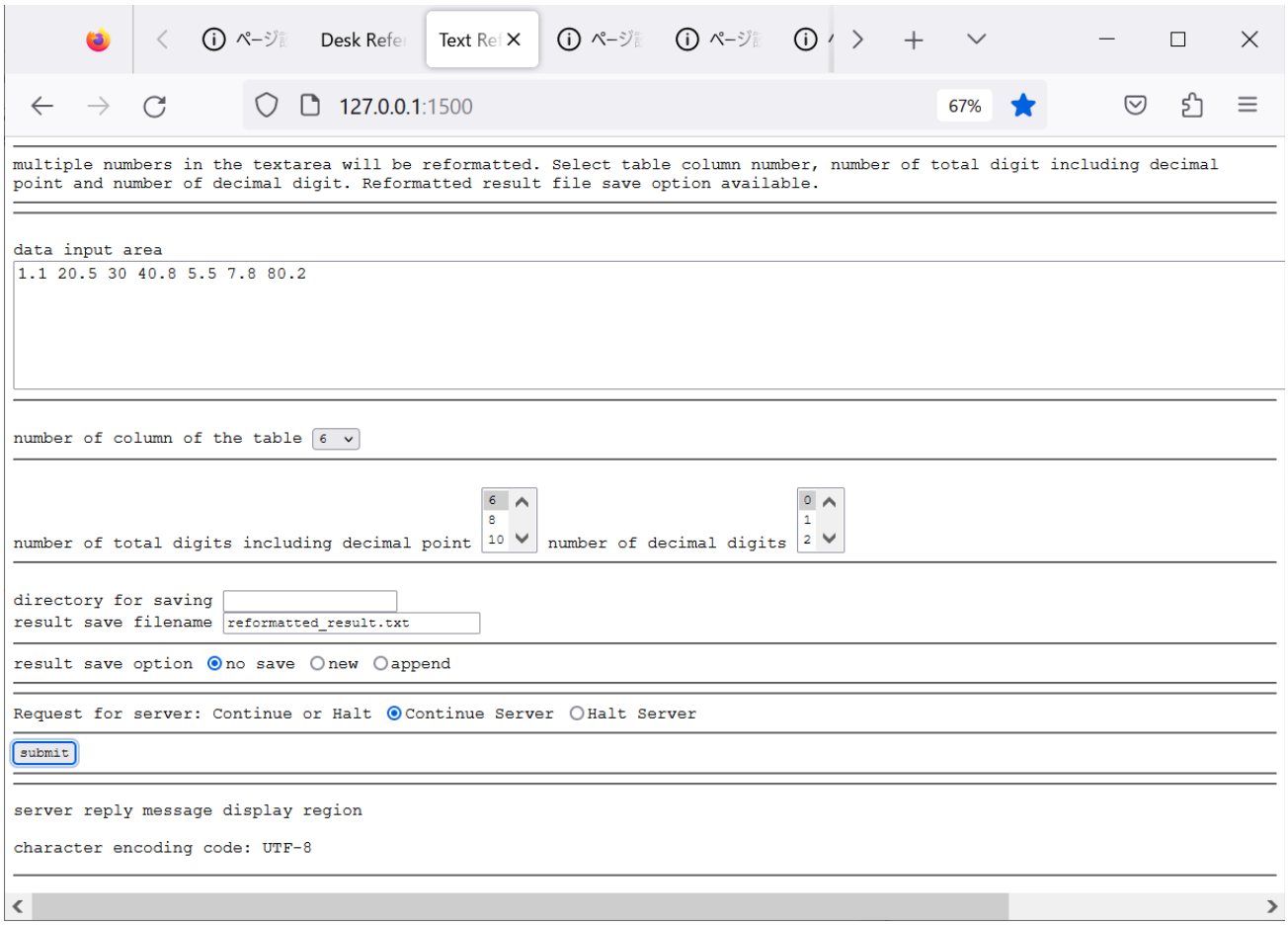

Figure 37: 02_04_demo_number_list_reformat_with_file_save_option_added_GET. Note two textboxes are added below two checklists. One is for directory for saving and the other is file name for saving. The formaer textbox is empty and the latter textbox shows default file name. And one radiobutton for options of saving (no save/new/append) is added, too.

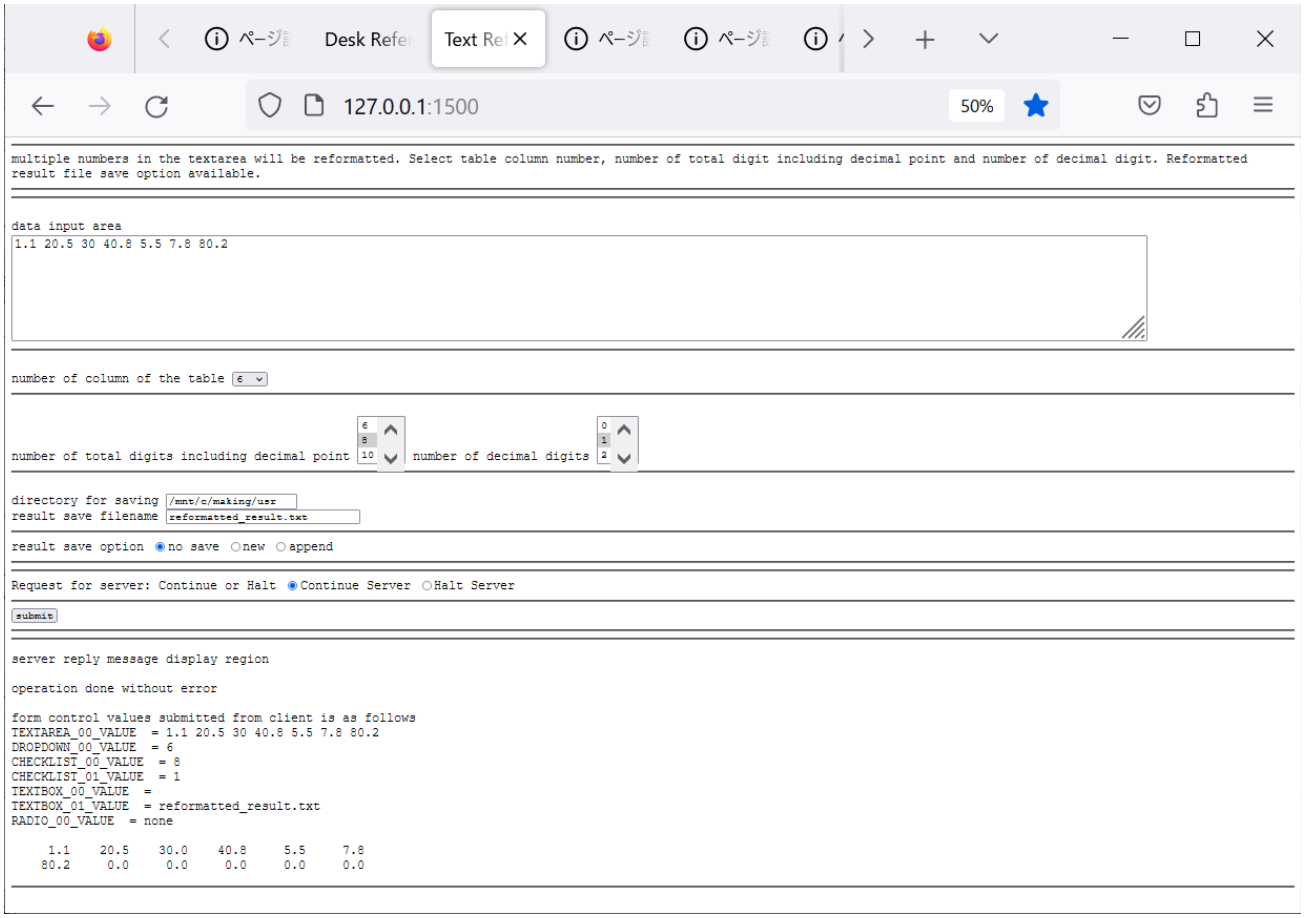

Figure 38: 02_04_demo_number_list_reformat_with_file_save_option_added_POST. Note the posted value list includes added textboxes and radiobutoon followed by resulting reformatted matrix. Also note the content of textbox for directory for saving is /mnt/c/making/usr. This is system default of user's saving directory when the system is used in Windows Subsystem for Linux. The file saving directory can be controlled in the script of YOUR_JOB of the definition file.

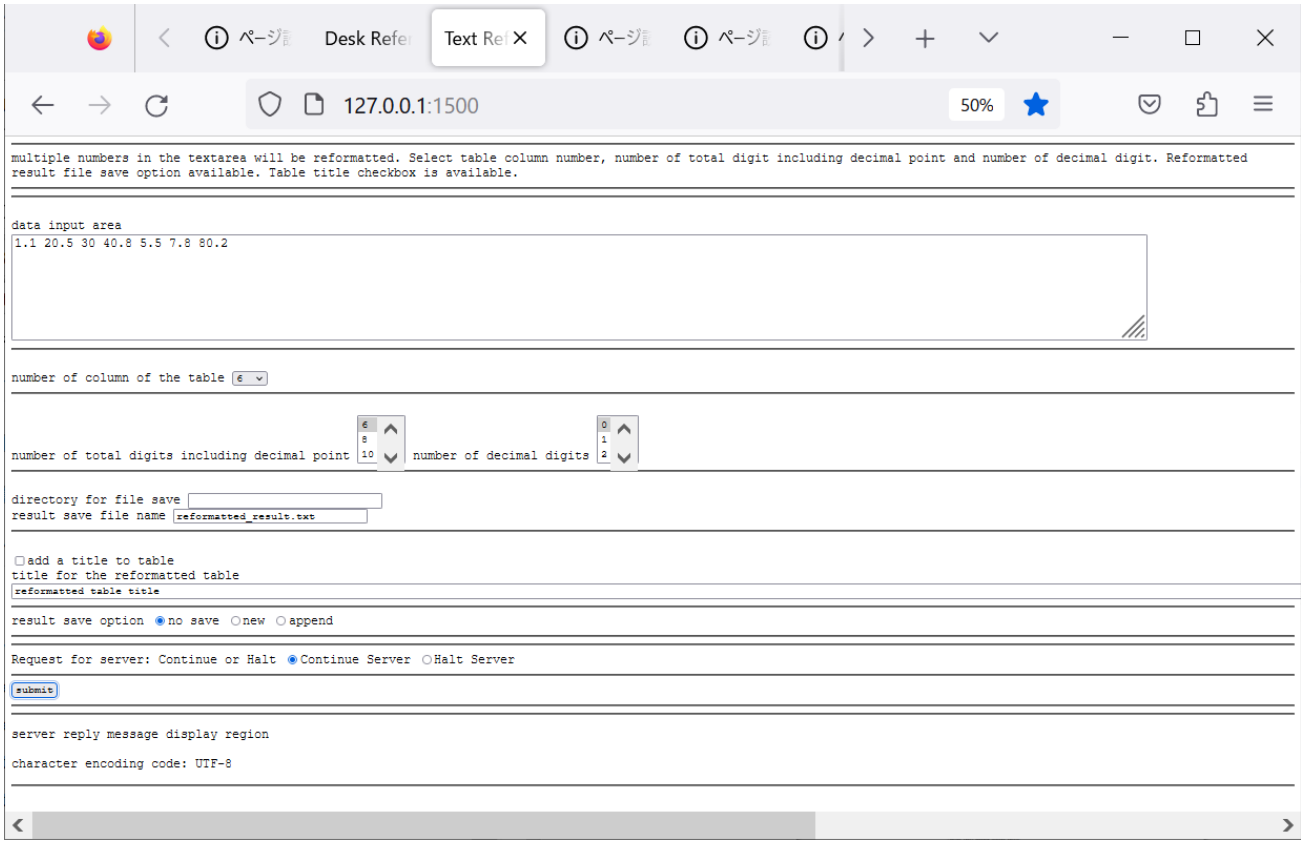

Figure 39: 02_05_demo_number_list_reformat_with_table_title_option_added_GET. Note following the textbox of result save file name, there are one checkbox whether to add a title to the result and one textbox for title itself.

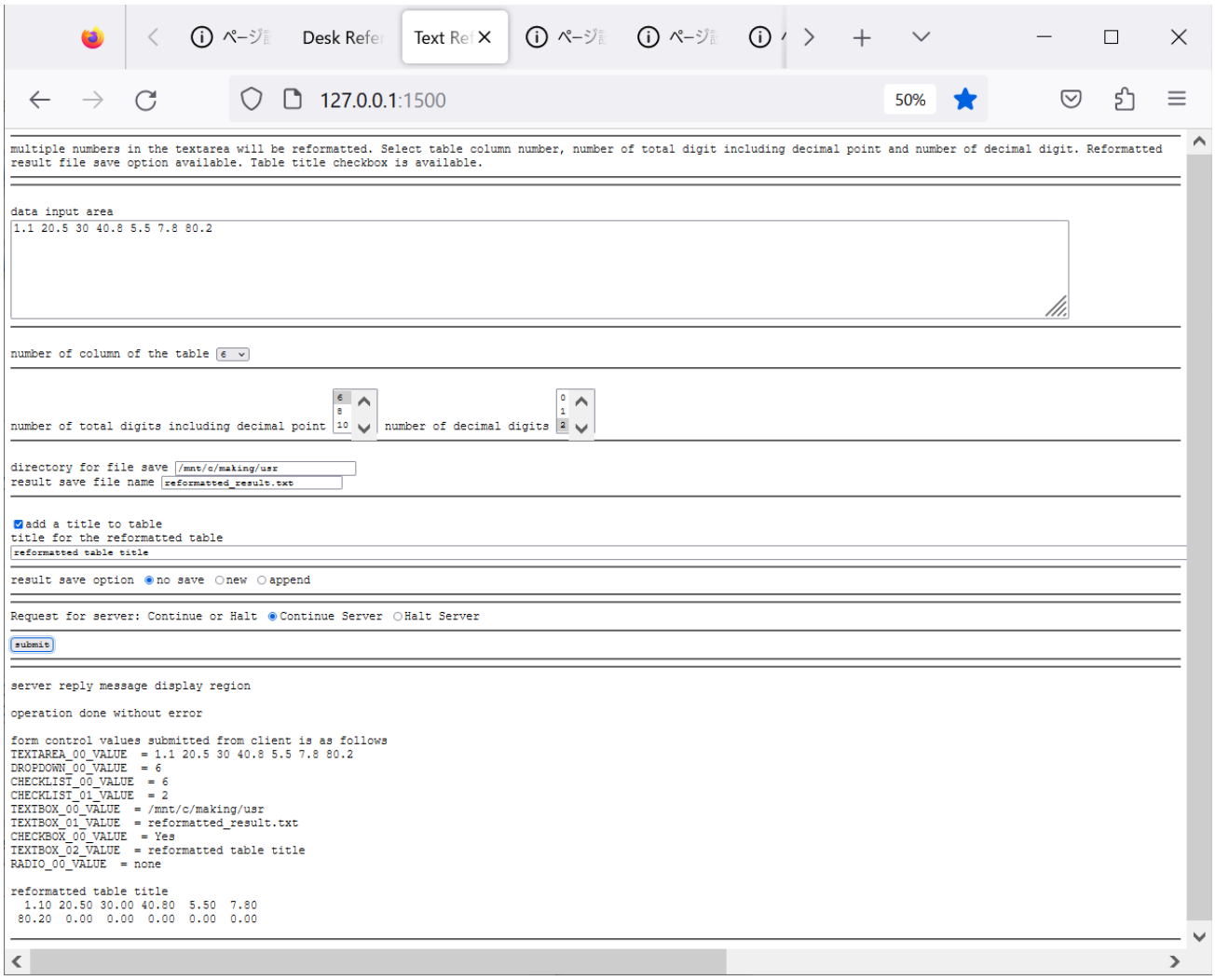

Figure 40: 02_05_demo_number_list_reformat_with_table_title_option_added_POST. At the end of server reply message display region, reformatted numbers is accompanied with the title.

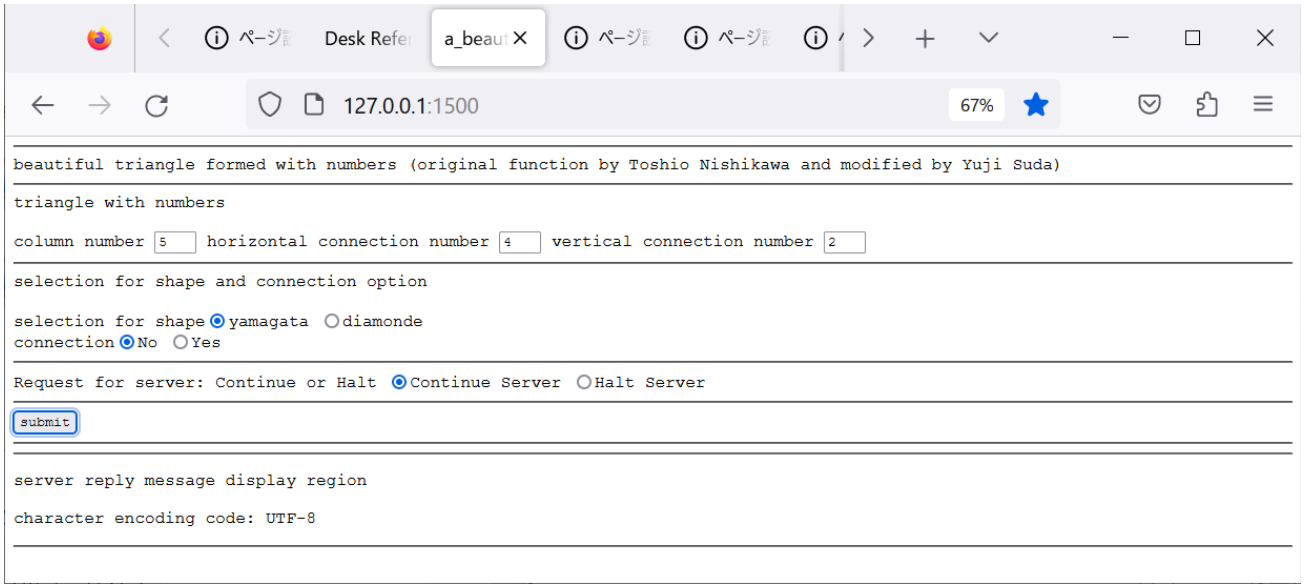

Figure 41: 50_print_a_beautiful_triangle_with_figure_length_calculation_GET. This is the web form application for 'Print a beautiful triangle with figure length calculation' by Toshio Nishikawa [7]. The original J function was modified by Yuji Suda for triangle reversal, forming diamonde shape and layout in matrix. Note three textboxes and two radiobuttons are used.

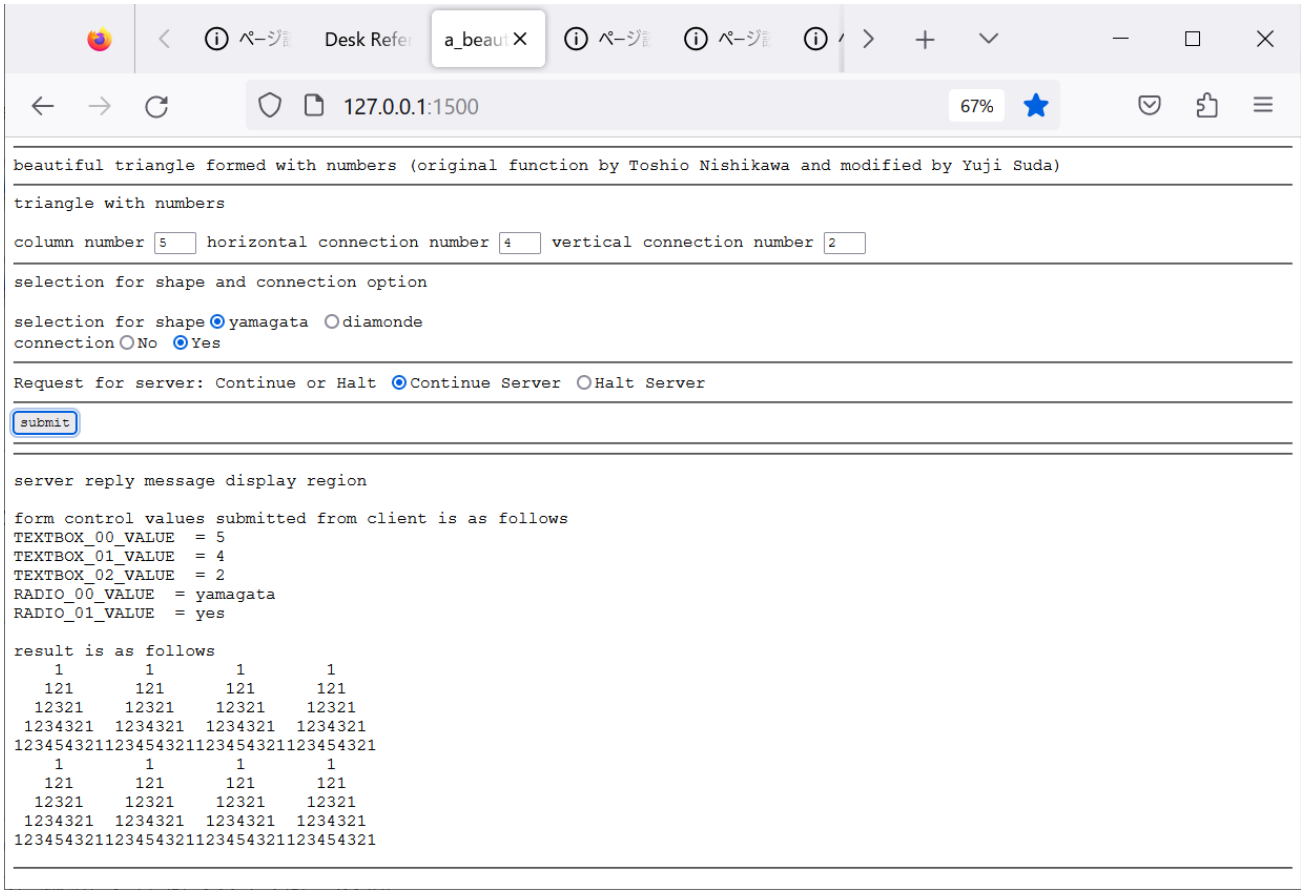

Figure 42: 50_print_a_beautiful_triangle_with_figure_length_calculation_POST. Note shape selection is yamagata and connection selection is Yes, so the result of triangle is in matrix based on the horizontal and vertical connection numbers.

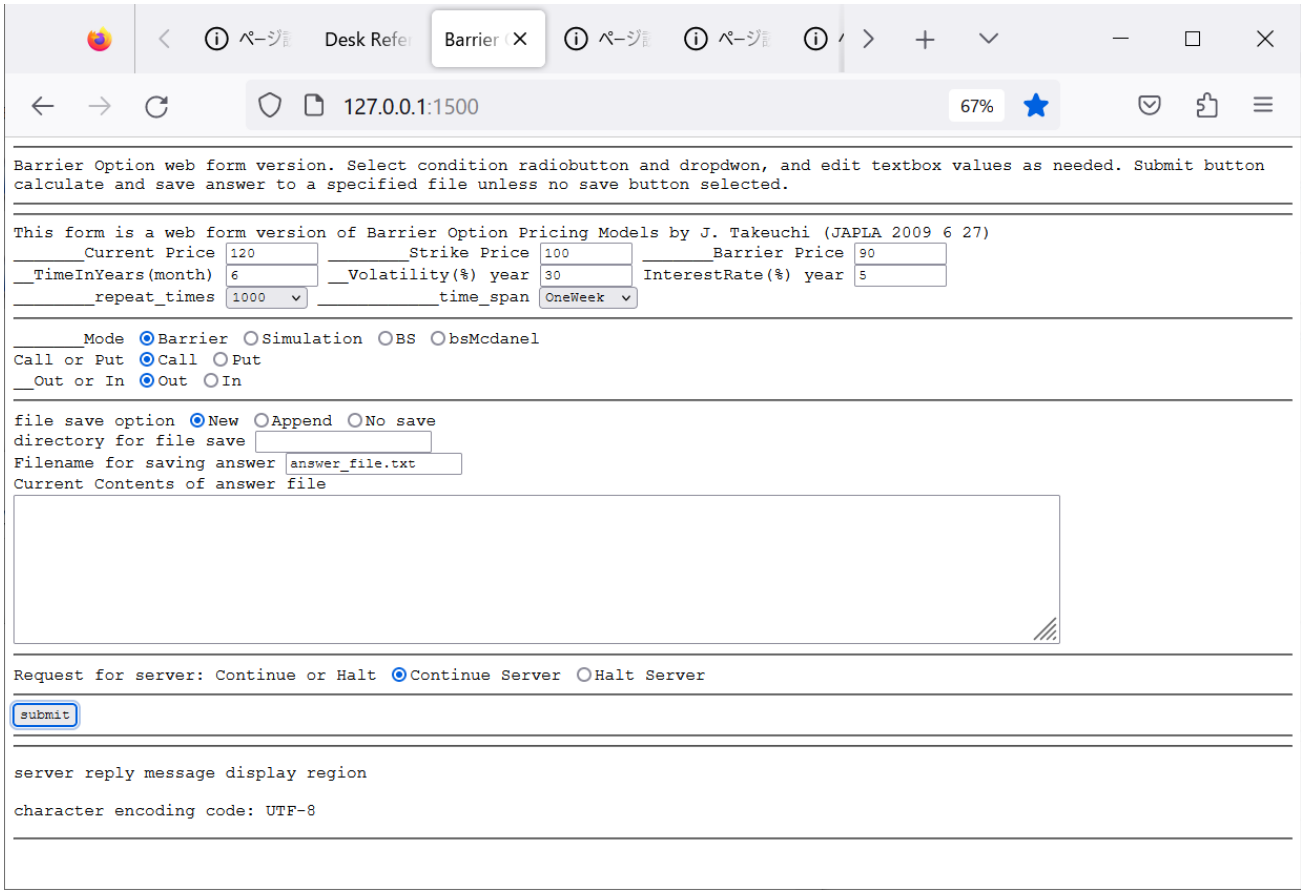

Figure 43: 51_barrier_option_pricing_model_GET. This is the web form application for 'The Form of Barrier Option Models.' by Juichirou Takeuchi [8]. The original J functions was modified to adjust for fetching various arguments from web form controls. File operation is also adjusted to the web form server system.

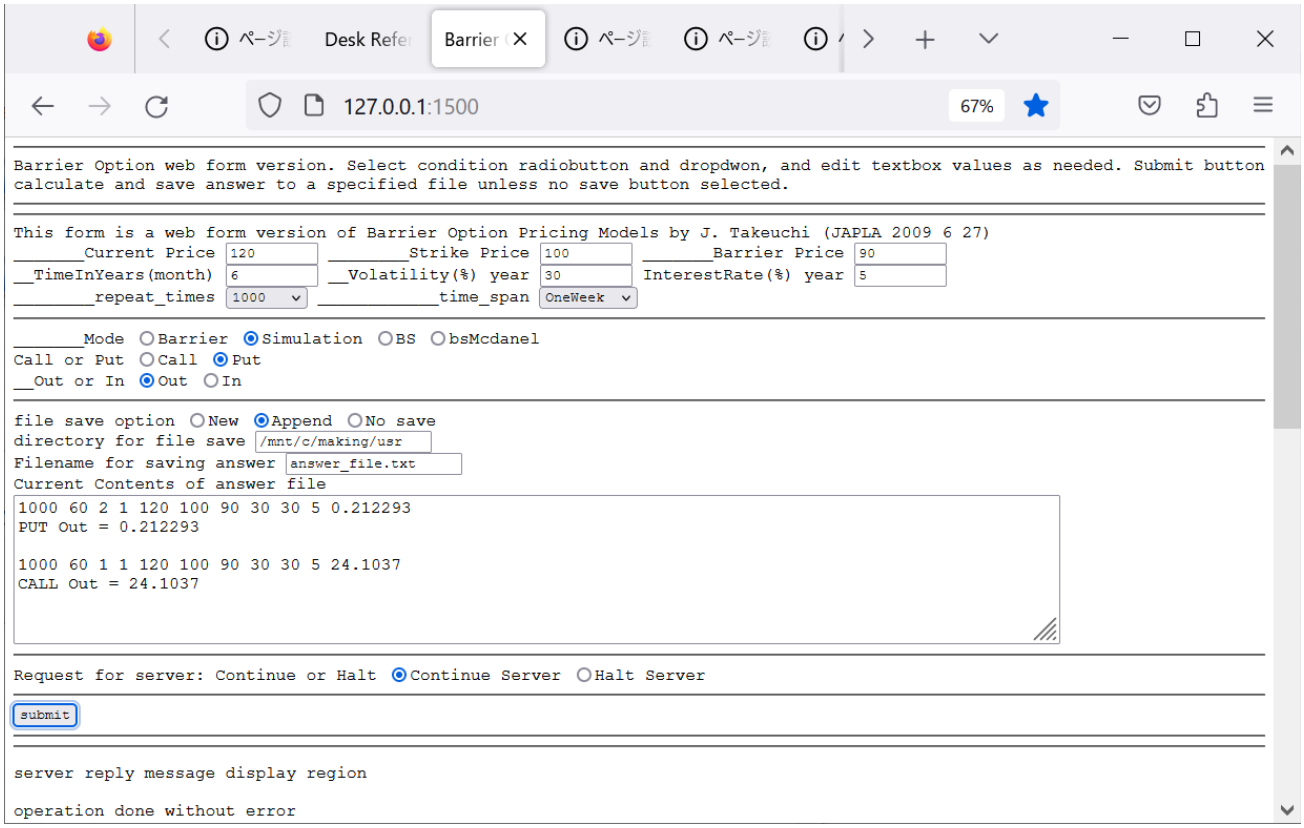

Figure 44: 51_barrier_option_pricing_model_POST. Note the textbox for current contents of answer file shows two caluculations appended.

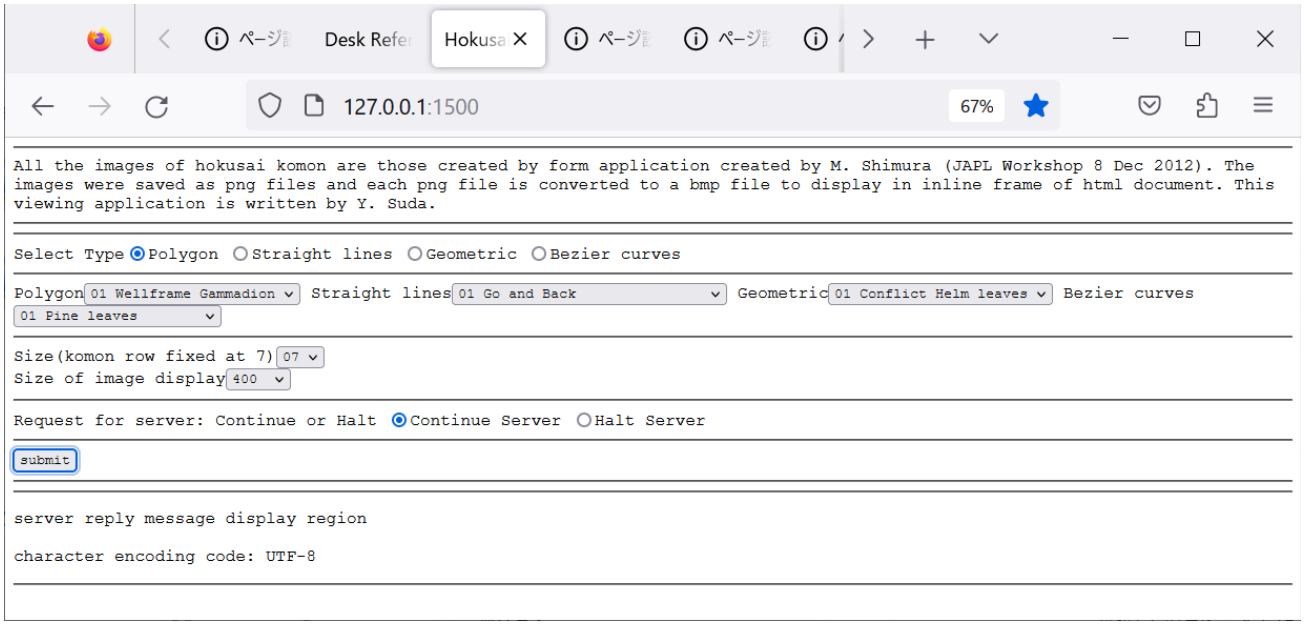

Figure 45: 52_hokusai_komon_image_galary_GET. This is the web form application of 'Print Form for Hokusai(for J6).' by Masato Shimura [9]. This is an example for iframe image transmission in the web form server system. Since the web server system runs in jconsole and the original function for drawing komons depends on legacy window driver J6 for windows, all the komon images have been created in png format in J602a for windows in advance. The web form application just fetches the requested image code and convert image format from png to bmp to be set as 'response_image.bmp' for iframe image file.

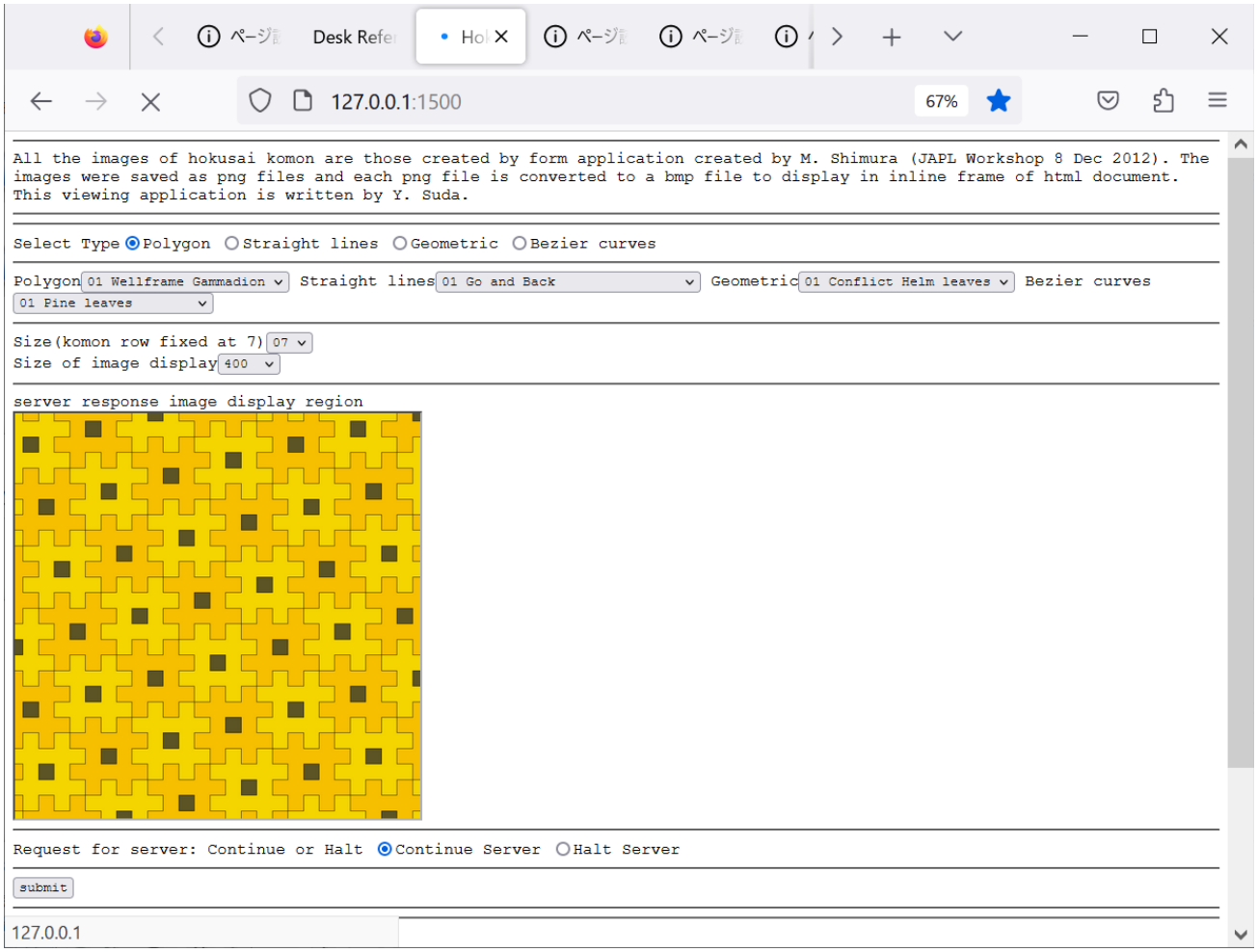

Figure 46: 52_hokusai_komon_image_galary_POST. Note type selection is Polygon and Polygon 01 Wellframe Gammdion is displayed in iframe control with size of 400x400 pixels, which is selected in the corresponding dropdown. Size of komon is fixed at 7, because the png images have been created only at this condition, whereas the options of frame size are 400, 600, 800 and 1000. In this case, 400 is selected.

References

[1] https://code.jsoftware.com/wiki/Guides/Asynchronous_GUI

[2] https://developer.mozilla.org/ja/docs/Web/HTTP/Session

[3] https://www.jsoftware.com/docs/help807/primer/gui.htm

[4] https://code.jsoftware.com/wiki/Help/Primer/100_GUI_part_1

[5] http://japla.sakura.ne.jp

[6] https://developer.mozilla.org/ja/docs/Web/HTML/Element/iframe

[7] Toshio Nishikawa Print a beautiful triangle with figure length calculation. JAPLA Workshop 27 December 2022

[8] Juichirou Takeuchi The Form of Barrier Option Models. JAPLA Workshop 27 June 2009

[9] Masato Shimura Print Form for Hokusai(for J6). JAPLA Workshop 8 December 2012