Call APL+Win Functions from Browser-based Javascript

Contents

Overview

In the current information technology environment users expect application software systems to

- Be accessible anywhere there is Internet access
- Be usable on any of the popular hardware and software platforms, e.g. Windows, Android, Apple – desktop computer, laptop computer, tablet computer or smart phone
- Require no local installation or updates

The solution for APL+Win programmers is to web-enable their application systems by creating a browser-based user interface which accesses server-side processing implemented in APL+Win as a web service. This design will preserve and enhance the algorithms and business rules supported by APL+Win and extend the availability of this valuable asset to additional users and venues.

The GUI that runs anywhere

The graphical user interface (GUI) of an application system can be designed to run in an Internet browser on a client workstation, laptop, tablet or smart phone. A significant benefit of a browser-based application system GUI is that it can run on any hardware and operating system platform which provides an Internet browser. This means a user can access the application system in virtually any modern client-side environment from any location which has Internet access.

The main functions of such a GUI are to:

- Collect application input information from the user
- Provide a means to request application system processing of user input information
- Display the result of application system processing

The GUI supports these purposes via 'controls', e.g. edit box, list, option button, check box, buttons, links, menus, etc., that the user can manipulate in a browser window.

The underlying programming language of a browser-based GUI is html. Although it is possible to develop a browser-based GUI using only html, most browser-based application system programmers use Javascript and in many cases such programmers also use a Javascript library to improve productivity and benefit from the combined efforts of many other programmers.

The jQuery library is one of the most popular and well-supported Javascript libraries.

What is jQuery

Quote from the jQuery web site:

> *jQuery is a fast, small, and feature-rich JavaScript library. It makes things like HTML document traversal and manipulation, event handling, animation, and Ajax much simpler with an easy-to-use API that works across a multitude of browsers. With a combination of versatility and extensibility, jQuery has changed the way that millions of people write JavaScript.*

Using jQuery to access a web service

A browser-based application system may perform simple data manipulation and validation entirely on the client-side, but generally, significant processing of user input is done on the server-side. Using jQuery it is possible to transmit user input data gathered from the browser-based GUI to a web server, cause a server-side function to process that data, receive the processing results from the server-side and display those results in the client browser.

The jQuery Javascript library supports Ajax which means that information can be asynchronously exchanged between the client-side GUI and the server-side programs by updating only a portion of the current web page in the client-side browser rather than loading the entire web page.

The server-side function must be properly designed to receive the user input data and respond with processing results in formats which are suitable for the browser-based, client-side program. Proper formatting of client-side input data and server-side processing results means using the http/https Internet protocol. This protocol serializes data in an XML format or other formats such as json or html form format which can be recognized by both the client- and server-side of an application system.

APL+Win Functions can be exposed as web services

APL+Win functions in an APL+Win workspace on the server-side can be exposed as web services using the APLNext Application Server tool to provide a high-performance and secure implementation of application system algorithms and business rules.

Using APLNext Application server the arguments and results of APL+Win functions are automatically serialized so that this information can be transmitted between a browser-based, client-side GUI and a Microsoft Windows server-side installation of APL+Win.

The client-side GUI is indifferent to the hardware and operating system platform of the server-side portion of the application system. Using APLNext Application Server all interaction between the client-side GUI and the server-side APL+Win functions is done using industry-standard protocols that exchange data using industry-standard serialization. APLNext Application Server provides the http/https protocol and the XML serialization so APL+Win functions can be exposed as web services very easily.

Learn by example

The remainder of this document provides two examples of using jQuery to access APL+Win functions exposed as web services.

As you will see in this example, creating the APL+Win functions to be exposed as web services is easy and will be familiar to an APL+Win programmer. Setting up the APLNext Application Server presents choices which the application system developer must make based upon the desired interaction between the client-side, browser-based user and the server-side APL+Win functions which perform the desired processing of the user input.

To appropriately make these choices, the web page containing the client-side GUI is created using the syntax of the html, Javascript and jQuery. The actions in the GUI by which the user requests server-side processing are usually supported by buttons or links in the web page. In this example buttons are used and labelled ‘Submit’ and ‘Add’. These buttons have associated event handler functions that are included in the web page and are written in Javascript called ‘sendForm()’ and ‘sendAdd()’ respectively.

These Javascript functions use jQuery to call APL+Win functions exposed as web services by the APLNext Application Server. The user-input fields in the client-side GUI will have data types which are recorded in the APLNext Application Server configuration so that they can be de-serialized to values which are suitable as APL+Win function arguments. The APLNext Application server will also appropriately serialize the result of an APL+Win function so that I can be de-serialized by the browser-based GUI.

Notes on these examples:

- The layout of the GUI in the browser has been simplified for clarity. In a production application the GUI layout would be established using cascading style sheets and jQuery techniques.
- The name of the server machine and the physical paths containing the elements of the example application may need to be modified to suit your environment.

Setup the APLNext Application Server

The setup of the APLNext Application Server will associate a 'virtual path' with an APL+Win function so that the APL+Win function has a unique web service address (url).

- Obtain a server, i.e. a Microsoft Windows server or workstation which has a connection to the Internet or enterprise intranet.

- Install APL+Win on the server and register APL+Win as an ActiveX component on that server

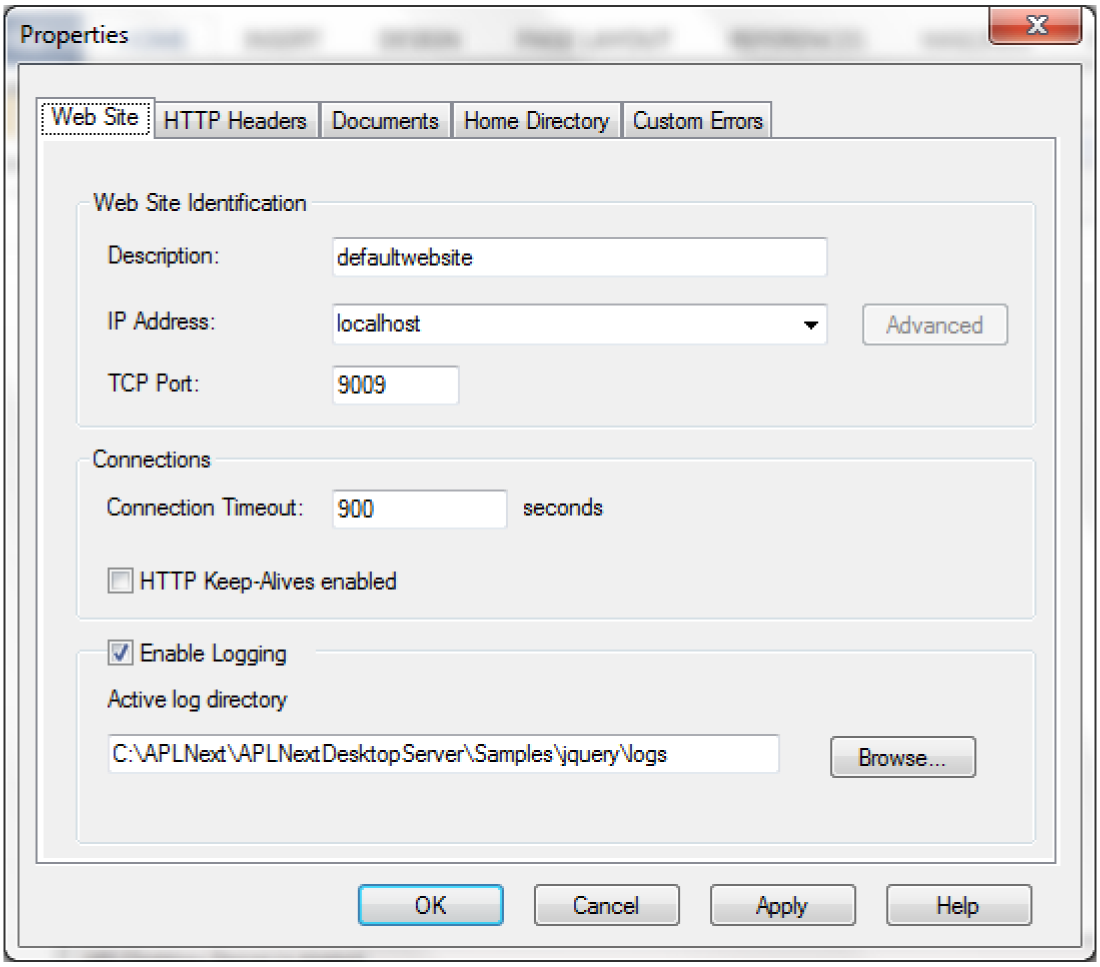
- Install APLNext Application Server on the server. There are several versions of the APLNext Application Server. Contact sales@apl2000.com for more information.
  - Desktop Server which runs as a Windows application and is suitable for implementing and testing by the APL+Win programmer
  - Application Server independent of Microsoft Internet Information Server (IIS) runs as a Windows service and is suitable for production operation of the application system. This version uses custom web server software to communicate with clients.
  - Application Server integrated with Microsoft IIS runs as a Windows service and is suitable for production operation of the application system. This version uses the Microsoft IIS web server software to communicate with clients.

- Using the APLNext Application Server Administration tool
  - Create the default web site
  - Define the home directory
  - Establish the virtual paths for the APL+Win functions
  - Identify the APL+Win workspace(s) containing the APL+Win functions

Create the default website:

In this example:

- APLNext Application Server is installed on the local workstation, which is convenient for implementing and testing an APL+Win web enabled application system. In a production environment APLNext Application Server would be installed on a Windows server or workstation which was connected to the public Internet or a private intranet.
- TCP port 9009 is used. In a production environment any port not already used by another web service can be selected.
- The log directory is where APLNext Application Server will create log files indicating web service activity.


Properties

Web Site | HTTP Headers | Documents | Home Directory | Custom Errors

Web Site Identification

Description: defaultwebsite

IP Address: localhost

TCP Port: 9009

Connections

Connection Timeout: 900 seconds

[ ] HTTP Keep-Alives enabled

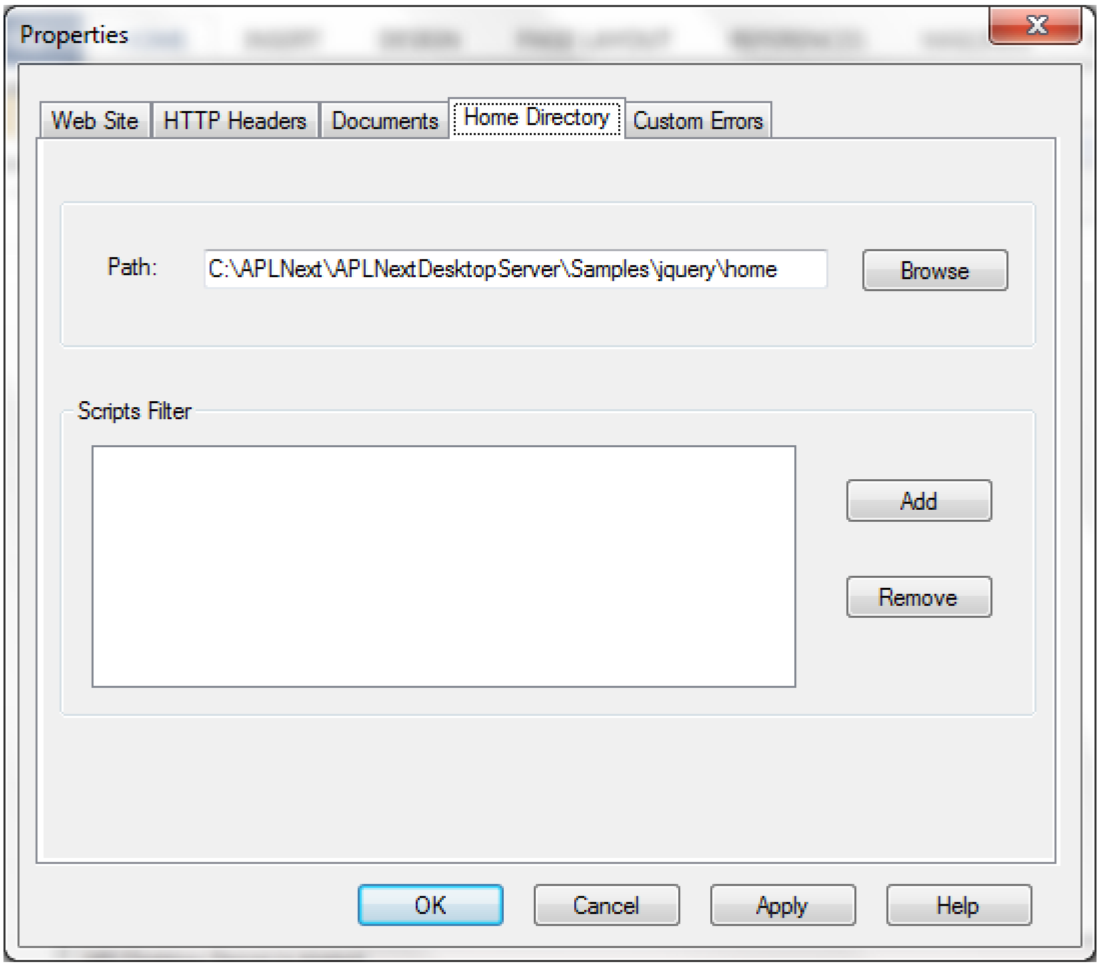
[✓] Enable Logging

Active log directory

C:\APLNext\APLNextDesktopServer\Samples\jquery\logs

Define the home directory:

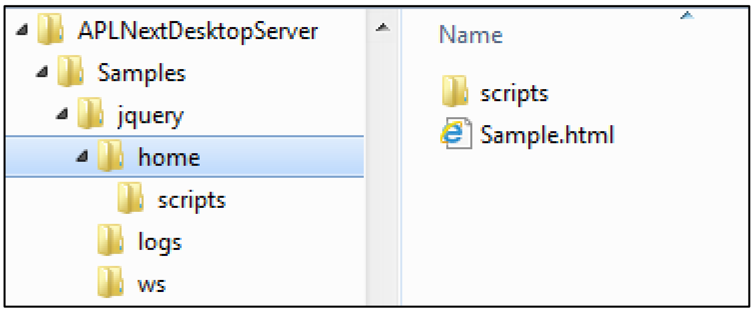
The home directory contains the html-format ‘sample.html’ web page document and the jQuery library.

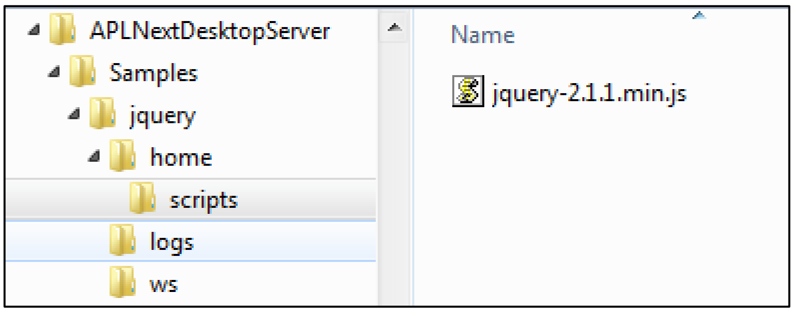


Path: C:\APLNext\APLNextDesktopServer\Samples\jquery\home

Home directory structure

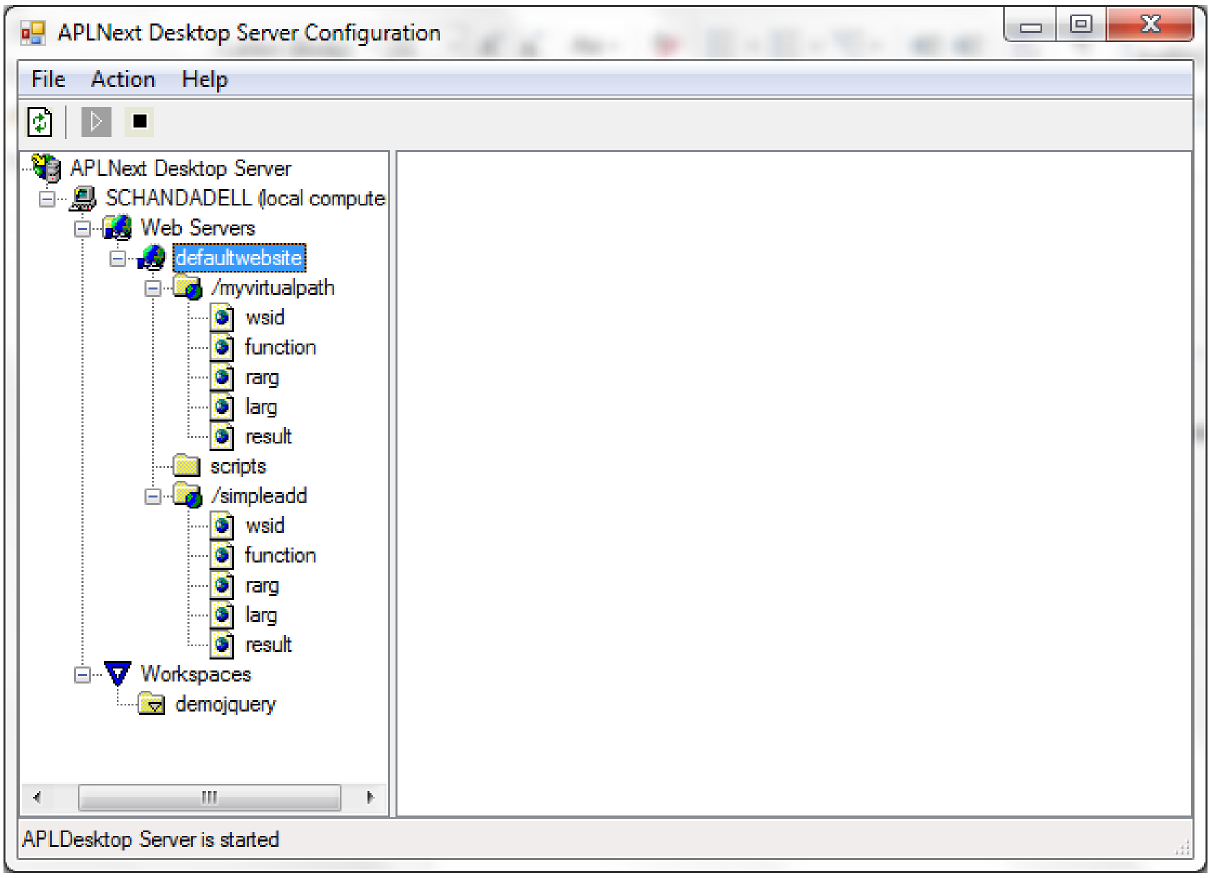
The home directory contains the 'Sample.html' file for this example and the 'scripts' folder which contains the 'jquery-2.1.1.min.js' tool kit file.





Establish the virtual paths:

Each virtual path is associated with a web service method implemented as an APL+Win function.



APLNext Desktop Server
SCHANDADELL (local compute
Web Servers
defaultwebsite
/myvirtualpath
wsid
function
rarg
larg
result
scripts
/simpleadd
wsid
function
rarg
larg
result
Workspaces
demojquery

APLDesktop Server is started

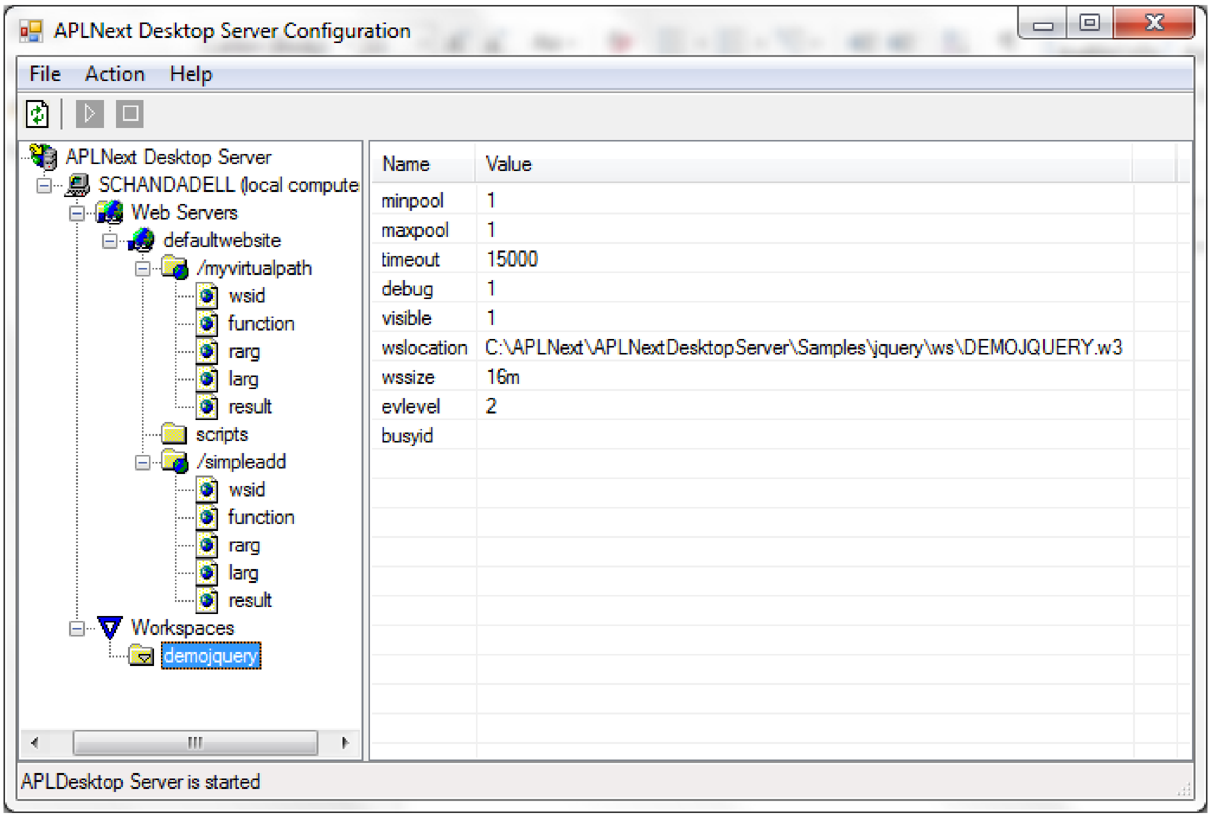
Below is a portion of the APLNext Application Server exported configuration for this example illustrating the virtual path definitions. The APLNext Application Server provides standard Internet 'types' for the APL+Win function arguments and result.

```
<virtualpath name="/myvirtualpath">
    <access />
    <wsid>demojquery</wsid>
    <function>DemoForm</function>
    <rarg type="entity-body">document</rarg>
    <result type="document">r</result>
  </virtualpath>
  <virtualpath name="/simpleadd">
    <access />
    <wsid>demojquery</wsid>
    <function>Add</function>
    <rarg type="int">a</rarg>
    <rarg type="int">b</rarg>
    <larg type="entity-body">data</larg>
    <result type="document">r</result>
  </virtualpath>
```

Identify the APL+Win workspace:

In this example:

- a pool of one APL+Win ActiveX instance supports the web service. In a production environment the 'minpool' and 'maxpool' settings would be tuned to the anticipated user load. For extreme user loads the 'busyid' may be used to send excess processing requests to other APLNext Application Servers.
- The 'timeout' setting indicates that if the server-side processing time exceeds 15 milliseconds, APLNext Application Server will respond with a timeout message to the browser-based client. Timeouts are necessary to prevent a defective client request from tying up server time.
- The 'debug' setting of '1' disables the 'timeout' setting which is useful for testing the server side of the application system.
- The 'visible' setting of '1' makes the APL+Win session visible which is useful for testing the server side of the application system.



Name	Value
minpool	1
maxpool	1
timeout	15000
debug	1
visible	1
wslocation	C:\APLNext\APLNextDesktopServer\Samples\jquery\ws\DEMOJQUERY.w3
wssize	16m
evlevel	2
busyid	

APLDesktop Server is started

Import the Sample APLNext Application Server Configuration

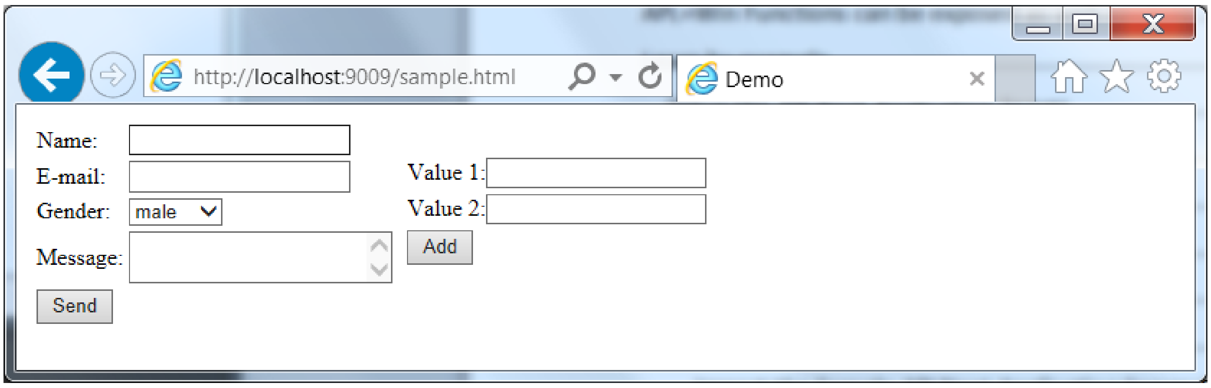
The steps outlined below can be replaced by using the APLNext Application Server Administration tool 'Action > Import' menu item to import the configuration file, 'JQUERYSAMPLE-Exported Configuration from APLNext Application Server.XML', provided with the example.

How Everything Works

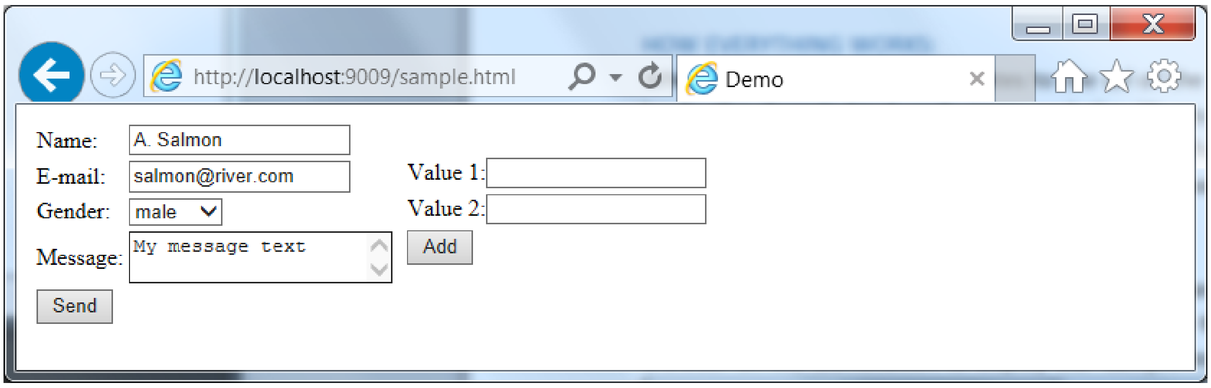
When the client-side user navigates to the url of the 'Sample.html' web page, the client-side browser loads the Sample.html web page and the jQuery javascript library from the server-side. This webpage creates two HTML FORM elements ('my_form' on the left and 'Form2' on the right), which are used to gather user input data in a simple way and have that data processed by server-side APL+Win functions.

Overall Web Page Workflow

Assuming that the components of this example have been installed on the local workstation and APLNext Web Server has been started, navigate to the 'http://localhost:9009/sample.html' url in the browser on the workstation to display the 'sample.html' web page.

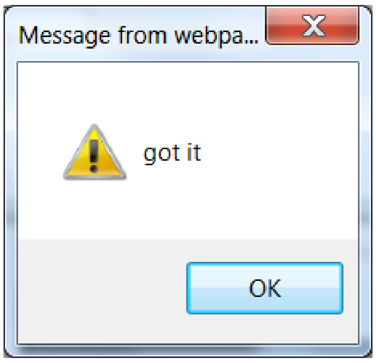


Enter information into the 'Name', 'E-mail', 'Gender' and 'Message' GUI controls and click the 'Send' button.



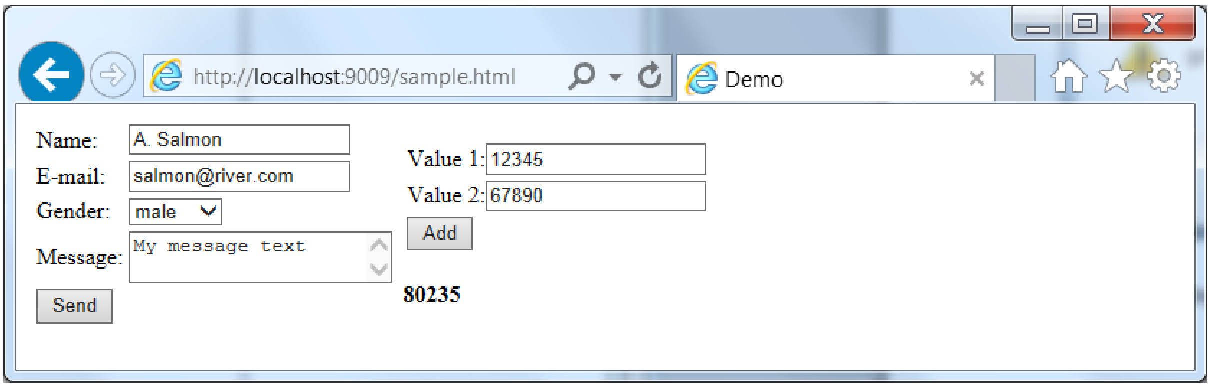
Name: A. Salmon
E-mail: salmon@river.com
Gender: male
Message: My message text

APLNext Application Server will respond and jQuery will display the response in an html modal 'alert' box. Click the 'OK' button to return control to the 'sample.html' web page.


Message from webpa...

got it

Enter an integer value into each of the ‘Value 1’ and ‘Value 2’ GUI controls and click the ‘Add’ button. APLNext Application Server will respond and jQuery will display the response in the html ‘addResult’ label GUI control:



http://localhost:9009/sample.html

Demo

Name: A. Salmon

E-mail: salmon@river.com

Gender: male

Message: My message text

Send

Value 1: 12345

Value 2: 67890

Add

80235

How ‘my_form’ works

When the user enters values into the ‘Name’, ‘E-mail’, ‘Gender’ and ‘Message’ GUI input controls and clicks the ‘Send’ button, jQuery in the browser serializes the data to form-url-encoded format and sends the data to the server-side at the virtual path defined in the APLNext Application Server. APLNext Application Server deserializes this data and provides it to the APL+Win ‘DemoForm’ function. The ‘DemoForm’ function displays this data in the APL+Win session on the server-side so that is can be seen by the APL+Win programmer developing the application system. The ‘DemoForm’ function also has a result which, for the purposes of this simple example, is the message ‘Got It’. In a production application system the APL+Win function might persist the user-input data to a database and provide a response verifying that the record had been received and was properly stored on the server-side. APLNext Application Server serializes this result as a simple entity-body of the HTTP result and sends it to the client-side browser. When the APL+Win response is received by the client-side browser, jQuery in the browser displays it in an ‘alert’ (modal message box).

The ‘my_form’ element of the web page illustrates the use of several types of html GUI ‘controls’:

- table: Control container
- text: user editable box with label ‘Name:’
- text: user editable box with label ‘E-mail:’
- select: list of options ‘male’, ‘female’ with label ‘Gender:’
- textarea: multi-line user editable box with label ‘Message:’
- button: user clickable button with caption ‘Send’ and click event handler ‘sendForm()’

```
<form id="my_form">
            <table>
              <tr><td>Name:</td><td><input type='text' name='name'></td></tr>
              <tr><td>E-mail:</td><td><input type='text' name='email'></td></tr>
              <tr><td>Gender:</td><td><select name='gender'>
              <option value='male'>male</option>
              <option value='female'>female</option>
              </select></td></tr>
              <tr><td>Message:</td><td><textarea name='about'></textarea></td></tr>
              <tr><td colspan="2">
              <input type="button" value="Send" onclick="sendForm()"/></td></tr>
               </table>
          </form>
```

When the ‘Send’ button is clicked the ’sendForm()’ Javascript function is executed on the client-side. This function will POST the serialized contents of ‘my_form’ to the server directed to the ‘myvirtualpath’ virtual path. If the POST succeeds a simple alert box is shown with the data returned by the server:

```
function sendForm() {
        $.ajax({
          type: "POST",
          url: "myvirtualpath",
          data: jQuery("#my_form").serialize(),
          cache: false,
          success: function (data) {
           alert(data);
```

```
            }
        });
    }
```

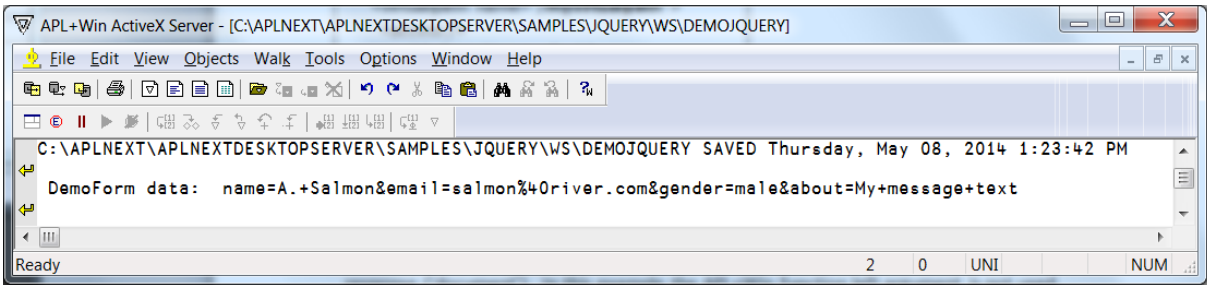
The server-side virtual path configuration is used by the APLNext Application Server de-serialize the user input data and direct the user input to the associated APL+Win function as APL+Win function arguments.

```
    <virtualpath name="/myvirtualpath">
      <access />
      <wsid>demojquery</wsid>
      <function>DemoForm</function>
      <rarg type="entity-body">document</rarg>
      <result type="document">r</result>
    </virtualpath>
```

When the POST reaches the server, the APL+Win function ‘DemoForm’, in the wsid defined by the workspace ‘demojquery’, will be executed. The right argument value for this function will be all the user-input data unmodified (HTTP entity-body) and the result of this function is a simple HTTP response (‘document’). In this example the APL+Win function left argument is not used.

The APL functions always returns the text message ‘Got It’, and displays the data sent from the browser to the server-side APL+Win session.

```
   ∇ r←DemoForm form
[1]   ⎕←'DemoForm data: ' form
[2]   r←'got it'
   ∇
```


APL+Win ActiveX Server - [C:\APLNEXT\APLNEXTDESKTOPSERVER\SAMPLES\JQUERY\WS\DEMOJQUERY]
File Edit View Objects Walk Tools Options Window Help
Ready
2
0
UNI
NUM

```
C:\APLNEXT\APLNEXTDESKTOPSERVER\SAMPLES\JQUERY\WS\DEMOJQUERY SAVED Thursday, May 08, 2014 1:23:42 PM

 DemoForm data:  name=A.+Salmon&email=salmon%40river.com&gender=male&about=My+message+text
```

How ‘Form2’ works

When the user enters values into the ‘Value 1’ and ‘Value2’ GUI input controls and clicks the ‘Add’ button, jQuery in the browser serializes the data to form-url-encoded format and sends that data stream to the server-side at the virtual path defined in the APLNext Application Server. APLNext Application Server de-serializes this data and provides it to the APL+Win ‘Add’ function. The APL+Win ‘Add’ function displays the left argument in the APL+Win session on the server side so that it can be seen by the APL+Win programmer developing the application system. This example demonstrates that an APL+Win left argument value can be passed from the javascript web page to the APLNext Application Server. The APL+Win ‘Add’ function processes the right-argument data by adding the values and returns the total as its result. APLNext Application Server serializes this result to a simple entity-body of the HTTP response and sends it to the client-side browser. When the APL+Win response is received by the client-side browser, jQuery in the browser deserializes the response from APLNext Application Server and uses Ajax technology to update the web page without reloading the entire page.

The definition of the ‘Form2’ implements two html input text boxes ‘a’ and ‘b’ which will become the right argument of the APL+Win ‘Add’ function and button with ‘sendAdd()’ click event handler.

```
<form id="Form2">
              <table>
                <tr><td>Value 1:<input type='text' name='a'></td></tr>
                <tr><td>Value 2:<input type='text' name='b'></td></tr>
                <tr><td><input type="button" value="Add" onclick="sendAdd()"/></td></tr>
              </table>
            </form>
```

If the client-side javascript ‘sendAdd()’ function is successful it will update the ‘addResult’ html element rather than reloading the entire web page.

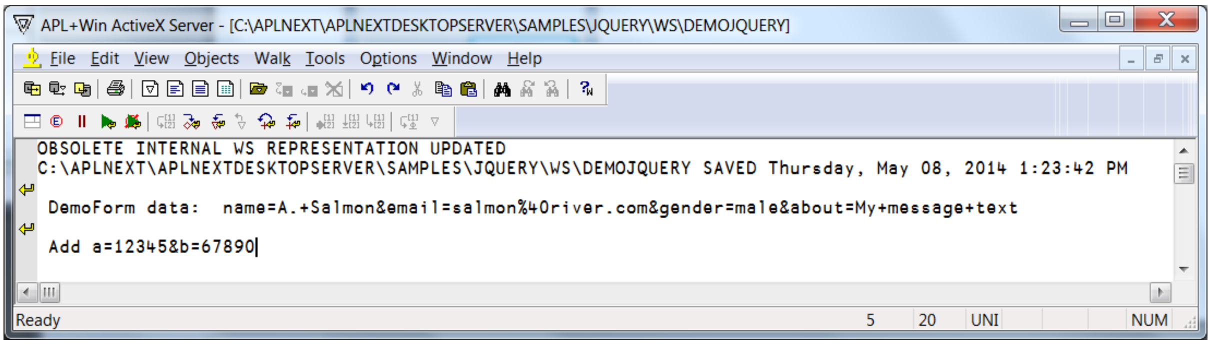
```
function sendAdd() {
        $.ajax({
           type: "POST",
           url: "SimpleAdd",
           data: jQuery("#Form2").serialize(),
           cache: false,
           success: function (data) {
              $("#addResult").html("<strong>" + data + "</strong>");
           }
        });
     }
```

The definition of the virtual path for the APL+Win ‘Add’ function in the APLNext Application Server indicates how the user input fields ‘a’ and ‘b’ will be de-serialized to integers and provided (both) as the right argument to the APL+Win function. The raw data sent from the browser, ‘entity-body’, is used to set the left argument to the APL+Win ‘Add’ function to illustrate that the left argument of an APL+Win function can also be used from javascript.

```
<virtualpath name="/simpleadd">
      <access />
      <wsid>demojquery</wsid>
      <function>Add</function>
      <rarg type="int">a</rarg>
      <rarg type="int">b</rarg>
      <larg type="entity-body">data</larg>
      <result type="document">r</result>
    </virtualpath>
```

The APL+Win 'Add' function displays the left argument of the function (entity-body sent by jQuery) and returns the total of the elements contained in the right argument (12345 and 67890) as the function result (67890).

```
    ∇ r←larg Add data
[1]   ⎕←'Add' larg
[2]   r←+/data
    ∇
```


APL+Win ActiveX Server - [C:\APLNEXT\APLNEXTDESKTOPSERVER\SAMPLES\JQUERY\WS\DEMOJQUERY]
File Edit View Objects Walk Tools Options Window Help
Ready 5 20 UNI NUM

```
OBSOLETE INTERNAL WS REPRESENTATION UPDATED
C:\APLNEXT\APLNEXTDESKTOPSERVER\SAMPLES\JQUERY\WS\DEMOJQUERY SAVED Thursday, May 08, 2014 1:23:42 PM

 DemoForm data:  name=A.+Salmon&email=salmon%40river.com&gender=male&about=My+message+text

 Add a=12345&b=67890
```

How Sample.html works

'Sample.html' contains the html-format specification of the web page for this example with the following elements:

- <!DOCTYPE html> remark which indicates that html5 is being used
- <html>… </html> element containing all the page elements
    - <head>… </head> container element
        - <meta… /> element containing metadata about the html document.
        - <title>…</title> element
    - <body>… </body> container element
        - <script>… </script> elements indicating that javascript and jQuery are being used in this web page
            - Javascript function elements 'sendForm()' and 'sendAdd()'
        - <table>… </table> container element
        - <form>… </form> elements 'my_form' and 'Form2' containing the html GUI controls

```
<!DOCTYPE html>

<html>
<head>
  <meta charset="utf-8" />
  <title>Demo</title>
</head>
<body>
  <script src="/scripts/jquery-2.1.1.min.js"></script>
  <script type="text/javascript">
    function sendForm() {
      $.ajax({
        type: "POST",
        url: "myvirtualpath",
        data: jQuery("#my_form").serialize(),
        cache: false,
        success: function (data) {
          alert(data);
        }
      });
    }
    function sendAdd() {
      $.ajax({
        type: "POST",
        url: "SimpleAdd",
        data: jQuery("#Form2").serialize(),
        cache: false,
        success: function (data) {
          $("#addResult").html("<strong>" + data + "</strong>");
        }
      });
    }
  </script>
  <table>
```

```
      <tr>
        <td>
          <form id="my_form">
            <table>
              <tr><td>Name:</td><td><input type='text' name='name'></td></tr>
              <tr><td>E-mail:</td><td><input type='text' name='email'></td></tr>
              <tr><td>Gender:</td><td><select name='gender'>
              <option value='male'>male</option>
              <option value='female'>female</option>
              </select></td></tr>
              <tr><td>Message:</td><td><textarea name='about'></textarea></td></tr>
              <tr><td colspan="2"><input type="button" value="Send"
onclick="sendForm()"/></td></tr>
               </table>
           </form>
        </td>
        <td>
          <form id="Form2">
            <table>
              <tr><td>Value 1:<input type='text' name='a'></td></tr>
              <tr><td>Value 2:<input type='text' name='b'></td></tr>
              <tr><td><input type="button" value="Add" onclick="sendAdd()"/></td></tr>
            </table>
           </form>
          <br />
          <label id="addResult" />
        </td>
      </tr>
   </table>

</body>
</html>
```

Need more help or more information?

APL2000 Consulting Services

APL2000 Consulting Services can help APL+Win application programmers to design and implement an effective browser-based GUI, using html, javascript, jScript or ASP.Net, and configure APLNext Application Server to expose APL+Win functions as secure, high-performance web services.

Try APLNext Application Server

A demonstration/evaluation version of APLNext Application Server is available for download.

APLNext Application Server Features

APLNext Application Server has additional features which make it the best way to incorporate APL+Win functions into web-based application systems including:

- Access APL+Win calculation/business rules across the Internet or private intranet network
- Use html web pages or pdf forms to create web-based application system GUI
- Support http/https protocol including SOAP and binary serialization
- Direct access to advance pooling and load balancing of Microsoft IIS
- Programmer-controlled APL+Win instance pooling for load balancing
- Fully-scalable to multiple servers when client request levels require it
- Active logging of server activity and client requests for testing & accounting
- Programmer-customizable timeouts, headers, document types & error messages
- Clear separation of APL+Win functions and http protocol technology so ‘standard' APL+Win functions can be used with virtually no modification
- Visible server option for program development & testing
- Administration tool included for server setup and maintenance
- Based on .Net for multi-threaded APL+Win instance operation
- Programmer-controlled stateless or stateful processing of client requests
- Automatic queueing of client requests for reliability & maximum through-put
- Fully documented with detailed programming examples
- APL2000 forum for APLNext Application Server

Learn jQuery

- HTTP://WWW.LEARNINGJQUERY.COM/
- HTTP://LEARN.JQUERY.COM/
- HTTP://WWW.W3SCHOOLS.COM/JQUERY/DEFAULT.ASP