Lab 7: JavaScript HTTP Request to LabView

ELEC Computer Networks
Electrical Engineering and Technology, Wentworth Institute of Technology

Overview:

- Install LabView on your PC. Setup a LabView Web Service to act as a ***Server*** that will respond to HTTP Get requests to return a data value (such as, temperature value from a sensor).
- The ***Client*** will run in browser as HTML file coupled with JavaScript. The JavaScript will send HTTP requests to the Web Service (LabView program), obtain a data value (temperature), and then plot the value in the client web page canvas by drawing lines from point to point, as well as display the value in a slider.

- ***Do this lab as an individual*** (may consult with others, but work and demo result on your Laptop PC).

Install LabView 2013 and DAQ Drivers

- Connect PC to wired Ethernet, so can download from LConnect → Access your Software
- Download and Install
  - NI LabVIEW 2013 & MultiSIM
  - NI DAQ Drivers 2013
- During Download and Install,
  - the default options usually work, but ensure
    - LabView 2013
    - Driver for Elvis II
  - For "Cost Center" (after you input your name and e-mail) type "WIT", this will be used for the license.

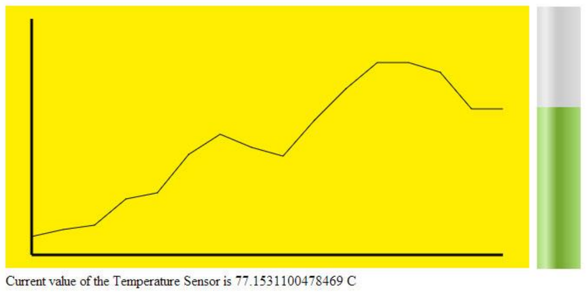
Setup LabView Web Service

- Create new Project
  - File → New → Project → Empty Project
  - File → Save as
    - Save this project in a new Lab7 folder, call the Project something like Lab7_Project_Dow, but replace Dow with your name or initials.
- In Project, right-click on My Computer → New → Web Service
  - Name this new service something like Lab7_WebService_Dow (but replace Dow with your name or initials).
- Setup the Server Web Resource that will be the LabView program that will be accessed by HTTP commands from the Client (HTML and JavaScript program).
  - From Black Board for Lab 7, copy the file, Lab7_LV_TempSensor.vi (a LabView program) into the same folder as your new project.
  - In LabView Project, right-click on Web Resources → Add File → Lab7_LV_TempSensor.vi
  - This file listed in the Project under Web Resources will have "(GET)" at the end of the name. This is as desired, since it will respond to HTTP GET requests

    (other types of HTTP requests include POST, PUT, DELETE, but are not used in this lab).
  - *You can open this LabView file. Notice that the Front Panel (the GUI) consists of a Slider, and a number box indicator of the value of the slider. This slider simulates (pretends to be) a temperature sensor that returns a value for temperature from 0°C to 125°C. For testing, you will manually move the slider so that the Client receives a dynamically changing series of values for temperature. In the future, this slider can be replaced with a DAQ (data acquisition module) connected to a circuit and analog value for temperature.*
- Test this new web service, using a client (web browser on same PC)
  - Start the Server to be ready for new HTTP requests
    - In LabView Project, right-click on the Lab7_WebService → Start
    - *Later you can stop the Server by selecting Stop (but leave it running for now).*
  - Get the URL of the LabView program that is to be called by HTTP requests
    - In LabView Project, right-click on the Lab7_LV_TempSensor.vi → Show Method URL
    - Then in pop-up (you can view the URL), click button Copy, and Close.
  - Open new tab on web browser, paste in this URL.
  - Then an HTTP Get request should be sent from the browser to the LabView program on the Server, and the value of the slider (the pretend temperature value) should be returned in XML format (Web Services typically return data in XML format, other options on JSON, plain text, etc.). The web browser should display something like:
    - This XML file does not appear to have any style information associated with it. The document tree is shown below.
    - `<Response>`
    - `<Terminal>`
    - `<Name>Temperature Slider Value</Name>`
    - `<Value>10</Value>`
    - `</Terminal>`
  - Notice the value of 10 in the center. This is the value of the Slider. To test this, change the value of the slider in the LabView Front Panel (GUI) by moving the slider. Then refresh the browser, and the returned XML data should reflect the new value of the slider. Do several times to ensure function is correct.
- Stop the web service
  - In LabView Project, right-click on the Lab7_WebService → Stop
- Setup Client accessible HTML and JavaScript files in public folder
  - In LabView Project, right-click on Lab7_WebService → Add Public Content Folder
    - In pop-up window, select New Folder, and name as Public Content, and select this folder.
  - From Black Board for Lab 7, copy in the *.html and *.js file and place into this new Public Content folder. In both file names, replace the “Starter” part of the name with your name or initials. Edit the *.html file, find the *.js file name and edit to reflect your new name.

Develop the Client HTML and JavaScript code

- Test these starter *.html and *.js files to ensure basic function correct.
  - Start the LabView Server to be ready for new HTTP requests
    - In LabView Project, right-click on the Lab7_WebService → Start
  - Get the URL of the HTML file
    - In Public Content folder of the LabView Project, right-click on the *.html file → Show Method URL
    - Then in pop-up, click button Copy, and Close.
  - Edit the *.js file, find the URL in there, and paste in your new URL
  - Open new tab on web browser, paste in this URL.
  - Three visual elements should appear on the browser web page:
    - Gray rectangle for the canvas
    - Right of canvas is vertical meter (scale slider)
    - Under canvas is text, such as "Current value of …"
  - Open JavaScript console to see messages displayed by the Java Script code related to the HTTP GET requests being sent once per second to the LabView web service progam (the temperature slider)
    - In Chrome, click on upper right icon "Customize and control Google Chrome" → Tools → JavaScript Console
  - Messages should report on the HTTP requests. Look for indications of errors, such as text in red. Typical problems include
    - Web Service not turned on (Start)
    - HTML file does not point to the correct JavaScript filename
    - JavaScript file does not point to the correct URL of the LabView Web Service (the URL of the LabView program with the Slider for Temperature).
- Modify the JavaScript code so that the graph has axis lines, new temperature data values are plotted, and the most recent temperature value is shown in both meter (scale slider) and text sentence below the canvas.
- Example image of the Browser Plot and Meter is as follows.


Current value of the Temperature Sensor is 77.1531100478469 C

- Study the starter HTML and JavaScript files. Easiest place to start is with the meter and text sentence (the plot on the canvas takes more thought and work).
- Modify HTML file so that your name appears, such as at the top (can just type text under the <body> tag, followed by a <br> for new line.

Challenge 1: Meter and Text Sentence under Canvas

- View the following Lynda video to see how the meter and text sentence works. Both have elements defined in the HTML file, with an ID, such that the JavaScript file can find and manipulate.
    - Lynda Course: HTML Essential Training with Bill Weinman
    - Chapter 11: HTML5 Data Elements
    - Video: Creating Dynamic Progress Indicators
    - Link:
        - http://www.lynda.com/HTML-tutorials/HTML-Essential-Training-2012/99326-2.html?org=wit.edu
    - After understanding the basic concepts and details on how to use:
        - Look at function init() in the starter *.js file. Follow the hints on how to have the JavaScript variables point to the defined elements in the HTML file.
        - Look at function plotNewTempPt(newTempVal). Here write code so that the passed in variable newTempVal is sent to both the meter and the text sentence under the canvas.
        - Test to ensure function correct (both meter and text sentence should show the current value of temperature that is set by the slider in the LabView program).

Challenge 2: Draw Lines for Graph X-axis and Y-axis

- Look at the function init(). Follow the hints to draw graph grid lines for the axis. Try to be similar to the example image shown above.
- Wise to slow down and make a plan for where on the canvas should be the origin of the axis, the maximum Y value, the maximum X value. Best to make variables for these values, since should use later to properly translate (add or subtract) and scale (multiple or divide) the new data values when plotting the new data lines on the graph.
- Test to ensure the axis shows up correctly in the web browser. This may take several attempts.

Challenge 3: Draw a Line on Graph for each new Temperature Data Point

- Look at the function plotNewTempPt(newTempVal). Follow the hints to draw lines on the graph for each new data point (draw line from the prior point to the new point). This may take some time and many tries to do correctly. The more thought you put up-front to make a plan on how to translate and scale the data points on the graph in the canvas, the more smooth this task will become.
- Make a counter to keep track of how many points have been plotted. This counter will help determine the X-value for each data point. When the X-axis is completely filled with data points, then turn off the timer interval (look at the hints and code at the bottom of the plotNewTempPt(newTempVal) function to do this. A Lynda video on Timers is at
    - Lynda Course: JavaScript Essential Training with Simon Allardice
    - Chapter 6: Working with Events and Event Listeners
    - Video: Working with timers
    - http://www.lynda.com/JavaScript-tutorials/JavaScript-Essential-Training/81266-2.html?org=wit.edu

Demo the functions to the instructor *(make sure any optional parts are shown)*.

- Submit to BlackBoard a document that contains
  - Two files and two images
    - HTML file
    - JavaScript file
    - Image of completed web browser page showing full graph lines of changing temperature values (ensure your name appears on page).
    - Image of Project (Web Service) in LabView