Arduino Inputs using Ajax with XML on the Arduino Web Server

Created on: 27 March 2013

Part 14 of the Arduino Ethernet Shield Web Server Tutorial

The Arduino web server hosts a web page (stored on the SD card) that displays the status of two push button switches and an analog (analogue) input.

The status of the two switches and the analog input are updated on the web page using Ajax. An XML file containing the switch statuses and the analog value is sent from the Arduino to the web browser.

This example produces the same output on the web page (with only the text changed) as part 9 of this tutorial – Analog Inputs and Switches using AJAX, with the following changes:

- Switch statuses and an analog value are sent in an XML file and not as a block of HTML.

- JavaScript responseXML is used instead of responseText to get the received values from the Arduino out of the XML file.
- The values from the XML file (Arduino inputs) are inserted into HTML paragraphs in the web page instead of replacing the entire paragraph.
- The web page is stored on the micro SD card of the Ethernet shield.

Why use Ajax with XML?

The advantage of using Ajax with an XML file is that individual values can easily be extracted by JavaScript on the web page, rather than having to write JavaScript code to extract values from a text file.

XML File Structure

An XML file uses tags like HTML or XHTML. The file has an initial tag that identifies it as an XML file. A main user defined tag follows that contains all other tags for the file.

This listing shows the structure of the XML file used in this example:

```
<?xml version = "1.0" ?>
<inputs>
    <button1></button1>
    <button2></button2>
    <analog1></analog1>
</inputs>
```

The **inputs** tag and all other tags contained in it are user defined names. The above XML could also be created as follows:

```
<?xml version = "1.0" ?>
<inputs>
    <button></button>
    <button></button>
```

```
    <analog></analog>
</inputs>
```

This file shows a button type and analog type that can be used to contain any button state or analog value. By adding more <button> or more <analog> tags, the state of additional buttons or analog inputs can be added.

The difference between the above two files is that the first uses unique names for all tags, whereas the second uses the tags to identify an input type.

Arduino XML File

In this example the Arduino creates an XML file and inserts the status of the switches and the analog input between the tags. The XML file is sent to the web browser in response to an Ajax request for data.

The image below shows an example of the XML file sent from the Arduino.

```
<?xml version = "1.0" ?>
<inputs>
    <button1>ON</button1>
    <button2>OFF</button2>
    <analog1>394</analog1>
</inputs>
```

XML File Sent by Arduino

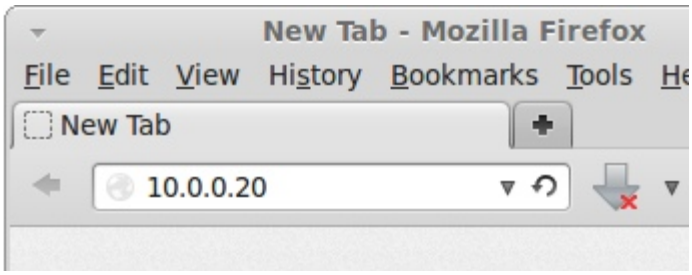
How Ajax with XML Works

If you have been following each part of this tutorial, then a lot of this will look familiar.

To update the Arduino input values on the web page, the following must occur:

1. Requesting a web page

As usual, the web browser is used to access the Arduino web server at the IP address that it has been set at in the Arduino sketch.



Connecting to the Arduino Web Server

This causes the web browser to sent an HTTP request:

```
GET / HTTP/1.1
Host: 10.0.0.20
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:1
Accept: text/html,application/xhtml+xml,application/xm
Accept-Language: en-ZA,en-GB;q=0.8,en-US;q=0.5,en;q=0.
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

2. Sending the web page

The Arduino web server receives the above request and responds with an HTTP header followed by the web page:

```
HTTP/1.1 200 OK
Content-Type: text/html
Connection: keep-alive
```

```
<!DOCTYPE html>
<html>
    <head>
        <title>Arduino SD Card Web Page using Ajax with XML</title>
        <script>
        function GetArduinoInputs()
        {
            nocache = "&nocache=" + Math.random() * 1000000;
            var request = new XMLHttpRequest();
            request.onreadystatechange = function()
            {
                if (this.readyState == 4) {
                    if (this.status == 200) {
                        if (this.responseXML != null) {
                            // extract XML data from XML file (containing switch states and analog value)
                            document.getElementById("input1").innerHTML =
                                this.responseXML.getElementsByTagName('button1')[0].childNodes[0].nodeValue;
                            document.getElementById("input2").innerHTML =
                                this.responseXML.getElementsByTagName('button2')[0].childNodes[0].nodeValue;
                            document.getElementById("input3").innerHTML =
                                this.responseXML.getElementsByTagName('analog1')[0].childNodes[0].nodeValue;
                        }
                    }
                }
            }
            request.open("GET", "ajax_inputs" + nocache, true);
            request.send(null);
            setTimeout('GetArduinoInputs()', 1000);
        }
    </script>
    </head>
    <body onload="GetArduinoInputs()">
        <h1>Arduino Inputs from SD Card Web Page using Ajax with XML</h1>
        <p>Button 1 (pin 7): <span id="input1">...</span></p>
        <p>Button 2 (pin 8): <span id="input2">...</span></p>
        <p>Analog (A2): <span id="input3">...</span></p>
    </body>
</html>
```

The Arduino reads the web page from the SD card and sends it to the web browser. After receiving the web page, it will be displayed in the web browser.

The web page contains JavaScript that is used as part of the Ajax process.

Note that the content type in the HTTP header for the HTML web page is text/html.

3. Ajax request

The JavaScript code on the web page sends an Ajax request to the Arduino (and continues to send a request every second).

```
GET /ajax_inputs&nocache=299105.2747379479 HTTP/1.1
Host: 10.0.0.20
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:1
Accept: text/html,application/xhtml+xml,application/xm
Accept-Language: en-ZA,en-GB;q=0.8,en-US;q=0.5,en;q=0.
Accept-Encoding: gzip
```

4. The Arduino responds to the Ajax request

After receiving the request for the XML file, the Arduino responds with an HTTP header followed by the XML file which contains input values from the Arduino.

```
HTTP/1.1 200 OK
Content-Type: text/xml
Connection: keep-alive
```

```
<?xml version = "1.0" ?>
<inputs>
    <button1>ON</button1>
    <button2>OFF</button2>
    <analog1>394</analog1>
</inputs>
```

Note that the content type in the HTTP header is now text/xml.

5. Displaying the data

Finally the JavaScript in the web page extracts the three values from the Arduino from the XML file and displays them on the web page.

Arduino Sketch and Web Page

Web Page

The Arduino hosts the following web page on the SD card:

```
<!DOCTYPE html>
<html>
    <head>
        <title>Arduino SD Card Web Page using Ajax with XML</title>
        <script>
        function GetArduinoInputs()
        {
            nocache = "&nocache=" + Math.random() * 1000000;
            var request = new XMLHttpRequest();
            request.onreadystatechange = function()
            {
                if (this.readyState == 4) {
                    if (this.status == 200) {
                        if (this.responseXML != null) {
                            // extract XML data from XML file (containing switch states and analog value)
                            document.getElementById("input1").innerHTML =
                                this.responseXML.getElementsByTagName('button1')[0].childNodes[0].nodeValue;
                            document.getElementById("input2").innerHTML =
                                this.responseXML.getElementsByTagName('button2')[0].childNodes[0].nodeValue;
                            document.getElementById("input3").innerHTML =
                                this.responseXML.getElementsByTagName('analog1')[0].childNodes[0].nodeValue;
                        }
                    }
                }
            }
            request.open("GET", "ajax_inputs" + nocache, true);
            request.send(null);
            setTimeout('GetArduinoInputs()', 1000);
        }
    </script>
    </head>
    <body onload="GetArduinoInputs()">
        <h1>Arduino Inputs from SD Card Web Page using Ajax with XML</h1>
        <p>Button 1 (pin 7): <span id="input1">...</span></p>
        <p>Button 2 (pin 8): <span id="input2">...</span></p>
        <p>Analog (A2): <span id="input3">...</span></p>
    </body>
</html>
```

Arduino Web Page index.htm - click for a bigger image

This is basically the same web page as sent by the Arduino in part 9 of this tutorial, but with the following changes (besides the text changes):

Function

The JavaScript function in the web page has been renamed to GetArduinoInputs().

The function still sends out an Ajax request every second. It now sends ajax_inputs with the GET request.

Because an XML file is being sent back from the Arduino, the function now checks if responseXML contains data instead of responseText:

```
if (this.responseXML != null) {
```

The data is extracted from the received XML as explained shortly.

HTML

The HTML is modified to display three paragraphs of text, one each for each value sent from the Arduino. Each paragraph contains an HTML span, each span has a unique ID.

The JavaScript function will insert the extracted values from the XML file into each span. This will replace only the default text (...) in each paragraph with the value from the Arduino.

The function uses the following code to get hold of each span for inserting data (code for getting "input1" shown here):

```
document.getElementById("input1").innerHTML =
```

Extracting the XML Data

The XML data is extracted from the received XML file using the following line of code:

```
this.responseXML.getElementsByTagName('button1')[0].ch
```

In this code, this.responseXML is used instead of this.responseText as used in previous examples.

Now every tag in the XML can be accessed using this.responseXML.getElementsByTagName('button1') as can be seen in the JavaScript function.

If you refer back to the top of this part of the tutorial under XML File Structure, and the second XML file example, you will see that there can be tags with the same name. If we used this for the button tags, then each button tag value can be accessed as follows:

```
this.responseXML.getElementsByTagName('button')[0].chi
this.responseXML.getElementsByTagName('button')[1].chi
```

This is usefull if there were a number of buttons that you did not want to give unique tags to. The values can then also be accessed in the JavaScript by using a loop.

The button values will then be extracted in the order that they have been inserted into the file.

The number of buttons in the XML file can then be obtained by using:

```
this.responseXML.getElementsByTagName('button').length
```

Arduino Sketch

The Arduino sketch for this example is shown below.

```
/*-----------------------------------------------
  Program:      eth_websrv_SD_Ajax_XML

  Description:  Arduino web server that serves up a
                that displays the status of two swit
                one analog input.
                The web page is stored on the SD car
                The web page contains JavaScript coc
                Ajax and XML to get the states of th
                and value of the analog input.

  Hardware:     Arduino Uno and official Arduino Eth
                shield. Should work with other Ardui
                compatible Ethernet shields.
                2Gb micro SD card formatted FAT16.
                Push button switches interfaced to p
                of the Arduino. Potentiometer interf
                analog input.

  Software:     Developed using Arduino 1.0.5 softwa
```

This sketch is basically a modified version of the sketch from the previous part of this tutorial.

Creating the XML File

The XML_response() function takes care of generating and sending the XML file in the format that has already been explained.

The switches and analog values are inserted into the XML file and sent to the web browser.

HTTP Response

Because the HTTP response must send a different content type for the HTML page and XML file (text/html or text/xml), it has been split up in the sketch to send the correct file type in each HTTP header.

As with the previous part of this tutorial, the web page is stored on the SD card as index.htm and sent when the browser accesses the Arduino web server.

Running the Sketch

Wire up the push button switches and the potentiometer as shown in the circuit diagram from part 9 of this tutorial.

Copy the index.htm file to a micro SD card and insert it into the micro SD card socket of the Ethernet shield. The index.htm file can be copied below.

Load the above sketch to the Arduino and connect to the Arduino with Ethernet shield using a web browser.

You will not seen any significant difference between this part of the tutorial and part 9 of the tutorial, but we now have an easy way of extracting values sent from the Arduino to be used on a web page.

Web Page Source Code

The web page can be copied here and pasted to a file called index.htm:

```
<!DOCTYPE html>
<html>
    <head>
```

```
<title>Arduino SD Card Web Page using Ajax v
<script>
function GetArduinoInputs()
{
    nocache = "&nocache=" + Math.random() *
    var request = new XMLHttpRequest();
    request.onreadystatechange = function()
    {
        if (this.readyState == 4) {
            if (this.status == 200) {
```

← Go back to Part 13 Go to Part 15 →

Comment (1)

Login

Sort by: **Date** Rating Last Activity

schel4ok · *20 weeks ago* 0

What is the reason for this piece of code?
// disable Ethernet chip
pinMode(10, OUTPUT);
digitalWrite(10, HIGH);

Reply

Post a new comment

Enter text right here!

Comment as a Guest, or login:

Name Email Website (optional)

Displayed next to your comments. *Not displayed publicly.* *If you have a website, link to it here.*

Subscribe to None **Submit Comment**