Zaawansowane

Programowanie Webowe

Dokumentacja Projektów

Łukasz Brewczyński
AGH EAIiIB
Informatyka
Środa godz. 15:30

Spis Treści

Wstęp 3
Projekt 1 - Generator wykresów z XML (HTML5 & XML) 4
Wykorzystane Technologie 4
Szczegóły implementacyjne 4
Projekt 2 - Dynamiczna aktualizacja wykresów (AJAX, jQuery) 7
Wykorzystane Technologie 7
Szczegóły implementacyjne 7
Projekt 3 – Konfigurator OpenVPN dla urządzenia Raspberry Pi (PHP) 10
Wykorzystane Technologie 10
Szczegóły implementacyjne 10
Wnioski 12
Źródła 12

Wstęp

Dokumentacja zawiera dane techniczne na temat realizacji projektów w ramach kursu "Zaawansowane Programowanie Webowe. W trakcie kursu zadaniem postawionym studentom było przygotowanie trzech mini projektów wykonanych w następujących technologiach:

- HTML5, XML
- AJAX, jQuery
- Dowolna technologia backend'owa

Do hostowania zrealizowanych projektów posłużył serwer student.agh.edu.pl jak również wykorzystane urządzenie Raspberry Pi – Model B działający pod systemem Raspian Wheezy, opartym na dystrybucji Debian'a.

Edytory wykorzystane do utworzenia kodu to :

- Microsoft WebMatrix 3
- Notepad++
- vim

Do przechowywania kodu wytworzonego w trakcie realizacji projektu wykorzystany został system kontroli wersji – Git (https://github.com).

Projekt 1 - Generator wykresów z XML (HTML5 & XML)

Celem pierwszego projektu było wykorzystanie znaczników HTML dodanych do standardu w wersji 5 oraz ewentualnie zapoznanie się ze strukturami dokumentów XML. Zadanie polegało na przygotowaniu serwisu pozwalającego na wysyłanie plików w formacie XML i na tej podstawie generowanie wykresów wykorzystując przy tym HTML5.

Wykorzystane Technologie

Z zakresu wykładu:

1) HTML5
2) JavaScript
3) XML
4) DOM Tree

Spoza zakresu wykładu:

1) Bootstrap v. 3.0.3
2) Charts.js (biblioteka do generowanie wykresów)

Szczegóły implementacyjne

W zrealizowanym projekcie wczytywanie plików XML jest możliwe na dwa sposoby – za pomocą dedykowanego button'a, bądź poprzez przeciągniecie plików na wyróżniony obszar. Za obsługę plików odpowiada klasa FileReader.

```
function handleFileSelectDrop(evt) {
    evt.stopPropagation();
    evt.preventDefault();
    var file = evt.dataTransfer.files[0]; // FileList object.
    var reader = new FileReader();
    waitForTextReadComplete(reader);
    reader.readAsText(file);
  }
```

Generowane wykresy pojawiają się na stronie w miejscu wykorzystania elementu canvas z HTML5.

```
<canvas id="chart" height="450" width="600" style="display: none;"></canvas>
```

XML Schema plików reprezentujących wykresy przedstawiają się następująco:

```
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="chart">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="data">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="point" maxOccurs="unbounded" minOccurs="0">
                <xs:complexType>
                  <xs:simpleContent>
                    <xs:extension base="xs:string">
                      <xs:attribute type="xs:byte" name="value" use="optional"/>
                      <xs:attribute type="xs:short" name="r" use="optional"/>
                      <xs:attribute type="xs:short" name="g" use="optional"/>
                      <xs:attribute type="xs:byte" name="b" use="optional"/>
                      <xs:attribute type="xs:byte" name="a" use="optional"/>
                    </xs:extension>
                  </xs:simpleContent>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
      <xs:attribute type="xs:string" name="name"/>
      <xs:attribute type="xs:string" name="type"/>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Wczytany plik musi zostać sparsowany, wykorzystano do tego obiekt DOMParser.

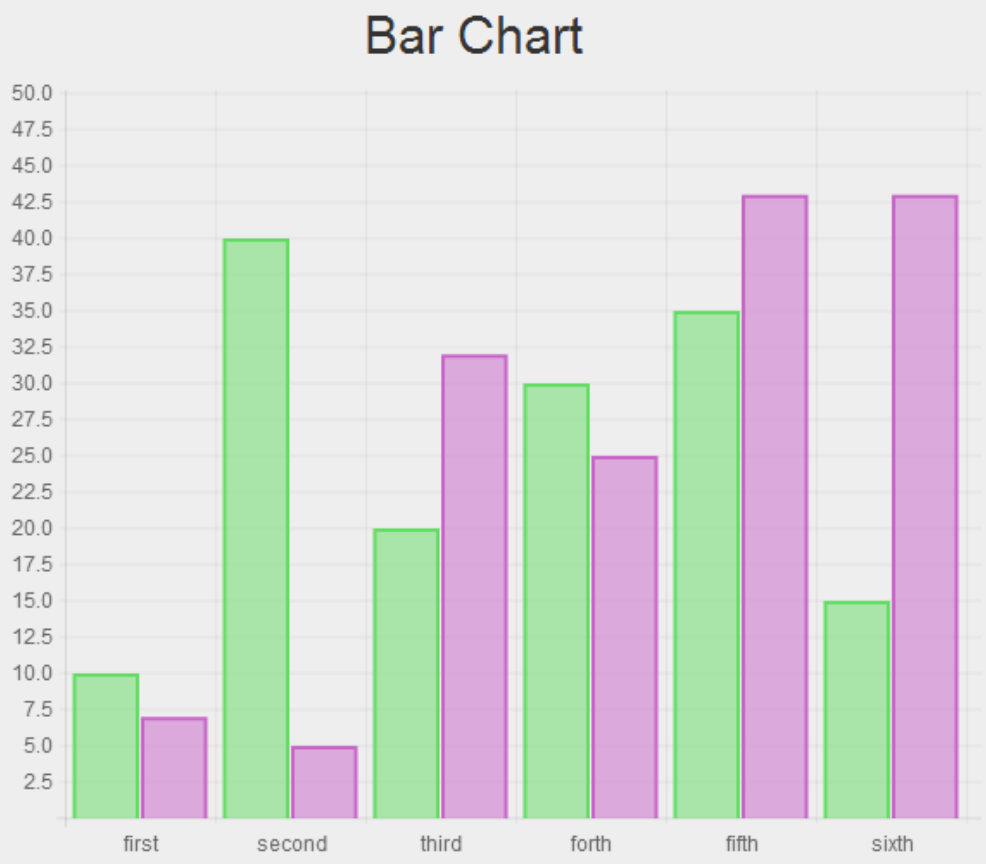
```
function parseTextAsXml(text) {
    var parser = new DOMParser();
    var xmlDOM = parser.parseFromString(text, "text/xml");
}
```

Po sparsowaniu pliku, odpowiednie parametry pobrane z obiektu DOM, można było przekazać do metod generujących wykres w obszarze canvas. Wykorzystana biblioteka przeznaczona do generowania wykresów to Charts.js, która pozwala na rysowanie 6 typów wykresów oraz proste animacje na nich.

W celu poprawienia wyglądu serwisu zastosowano bibliotekę styli Bootstrap w wersji 3.0.3.

Chart Maker

Home | Upload XML | Clock | BitCoin | RPi-Data | RPi-CPU | RP-CPUL



Show as a Picture

Projekt 2 - Dynamiczna aktualizacja wykresów (AJAX, jQuery)

Celem drugiego projektu było wykorzystanie technologii AJAX, tak, aby wykresy przygotowane w projekcie 1, odświeżały się na bieżąco, bez przeładowywania strony. Jako dane wejściowe zostały wykorzystane dane pochodzące z serwisu `https://bitpay.com/api/rates`, przedstawiające obecną wartość bitcoin'a w różnych walutach oraz dane umieszczone w bazie CouchDB udostępnione przez urządzenie Raspberry Pi.

Wykorzystane Technologie

Z zakresu wykładu:

1) HTML5
2) JavaScript
3) jQuery v. 2.0.3
4) JSON
5) DOM Tree

Spoza zakresu wykładu:

1) CouchDB
2) Bash scripts

Szczegóły implementacyjne

Do przechowywania danych została wykorzystana nierelacyjna baza CouchDB, ze względu na stosunkowo prostą możliwość udostępnienia danych jako WebService'ów. Dla bazy przygotowana została struktura dokumentu, przechowującego dane z Raspberry Pi oraz następujący widok:

```
{
    "_id": "_design/widoki",
    "language": "javascript",
    "views": {
        "wszystkie": {
            "map": "function(doc){if(doc.type && doc.type === 'rpdata'){emit(doc.created, doc);}}"
        }
    }
}
```

Przygotowany widok generował dane w następującym formacie:

```
{
    "_id": "20131211133304",
    "_rev": "1-ffba20728f9cb71d4dbc8bb7e34effca",
    "type": "rpdata",
    "Time": "20131211133304",
    "Memory": "18.29",
    "CPU": "13.0"
}
```

Baza została umieszczona na urządzeniu Raspberry Pi, które zasilało ją danymi o obecnym stanie pamięci oraz pracy jednostki CPU. Za umieszczanie danych odpowiadał bash'owy skypt, uruchamiany co 1 sekundę, wykorzystując komendę curl:

```
curl -X PUT http://192.168.0.30:5984/rpdata/${TIME} -H "Content-Type: application/json" -d
'{"type":"rpdata","Time":"'"$TIME"'", "Memory":"'"$MEM"'", "CPU":"'"$PROC"'"}'
```

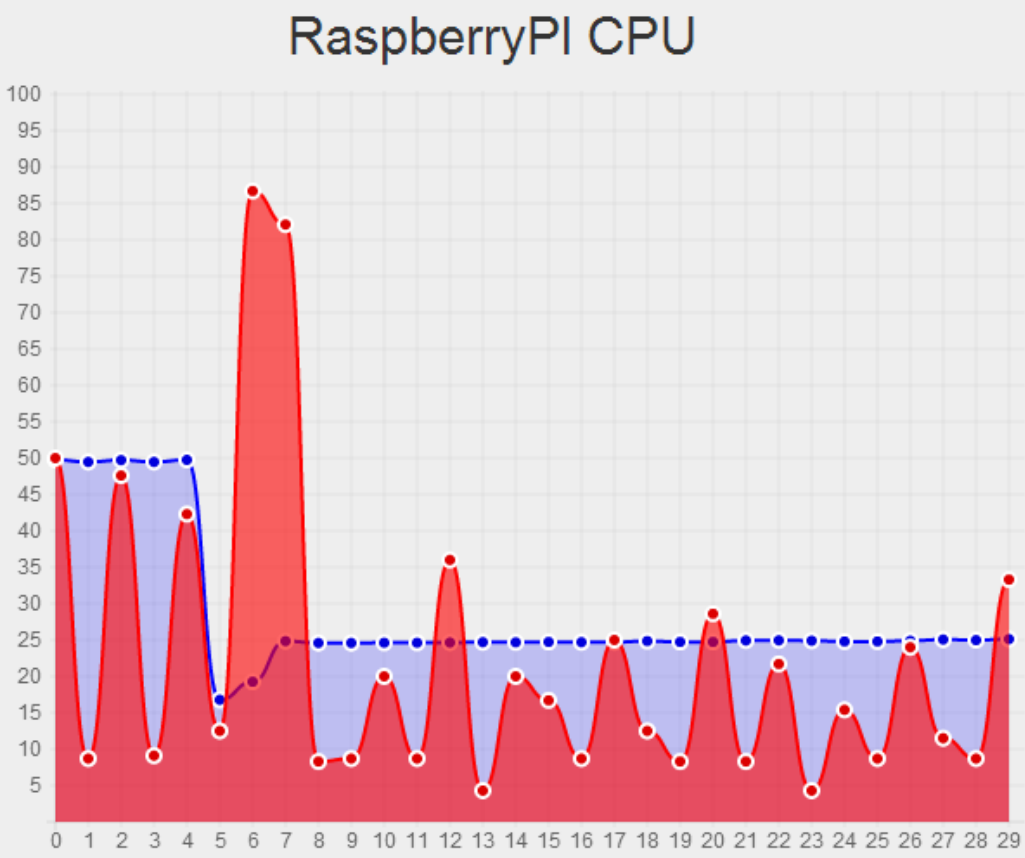
Po przygotowaniu danych należało wykorzystać te dane. Wykorzystano technologie AJAX używając metody getJSON oraz setTimeout, aby ponawiać zapytania.

```
getRemoteData: function () {
   console.log("getRemoteData");
   var requestURL = 'http://89.79.177.93:5984/rpdata/_design/widoki/_view/wszystkie?callback=?';

   $.getJSON(requestURL, {
      limit: 10,
      descending: true
   }, function (data) {
      console.log(data);
      RPData.parseRemoteData(data);
   });

   RPData.timeout = setTimeout(RPData.getRemoteData, RPData.delay);
},
```

W odniesieniu do projektu 1 należało uaktualnić metody parsujące, gdyż w tym przypadku dane wejściowe były w formacie JSON, a nie XML.

Chart Maker

Home | Upload XML | Clock | BitCoin | RPi-Data | RPi-CPU | RP-CPUL



Show as a Picture

Projekt 3 – Konfigurator OpenVPN dla urządzenia Raspberry Pi (PHP)

Celem trzeciego projektu było utworzenie serwisu webowego pozwalającego na instalację oprogramowania OpenVPN i konfigurację urządzenia Raspberry Pi, jako serwera VPN. Biorąc pod uwagę specyfikację sprzętową urządzenia, które zarówno miał zostać serwerem VPN, jak również hostować witrynę konfiguracyjną, do realizacji wykorzystano język PHP.

Wykorzystane Technologie

Z zakresu wykładu:

1) PHP
2) Cookie's

Spoza zakresu wykładu:

1) CouchDB
2) Bash scripts
3) OpenVPN

Szczegóły implementacyjne

Implementacja konfiguratora opiera się na kolejnym wykonywaniu komend bashowych po stronie serwera PHP. W celu przygotowania urządzenia do pracy z konfugartorem należy zainstalować serwer apache, php w wersji 5, a także nadać prawa sudo użytkownikowi www-data.

Większość czynności opiera się o wykorzystanie prostej metody, która wykonuje komendę bash'ową po stronie serwera, dodatkowo przekierowując strumień błędu do strumienia wyjściowego:

```
function executeCMD($cmd) {
        $output = shell_exec($cmd." 2>&1");
        echo $output;
        echo "<br/>";
}
```

Generowanie certyfikatów i kluczy wymagało wykorzystania zmiennych środowiskowych. Jednak PHP nie ma dostępu do nich, dlatego też niektóre skrypty należało połączyć z plikiem ustawiającym zmienne. Umożliwiało to uzyskanie dostępu do zmiennych wygenerowanych lokalnie podczas wykonywania skryptu.

Część kroków wymagała edycji plików konfiguracyjnych (tekstowych). Do edytowania poszczególnych linii wykorzystano następujący fragment kodu, pozwalający na podmianę poszczególnych linii:

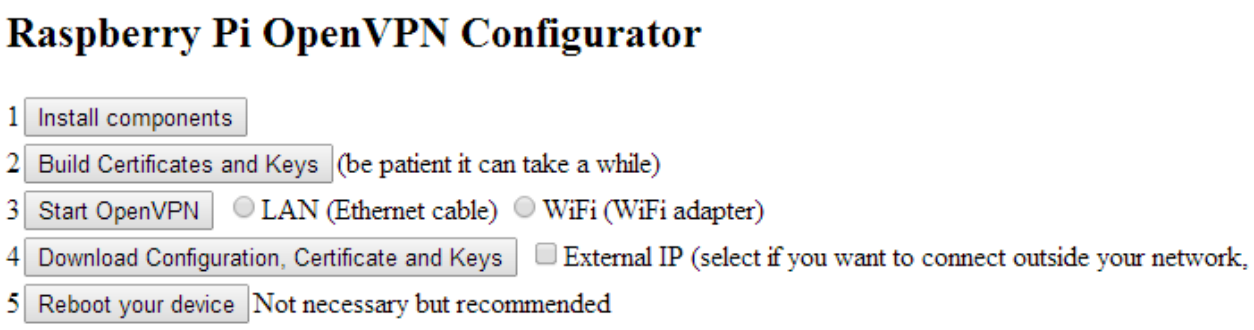
```
        while (!feof($sh)) {
                $line=fgets($sh);
                if (strpos($line, 'export EASY_RSA')!==false) {
                        $line='export EASY_RSA="/etc/openvpn/easy-rsa"'.PHP_EOL;
                }
                fwrite($th, $line);
        }
```

Funkcjonalność pobierania archiwum kluczy oraz certyfikatów została zrealizowana za pomocą następującego fragmentu:

```
if (file_exists($file)) {
        header('Content-Description: File Transfer');
        header('Content-Type: application/octet-stream');
        header('Content-Disposition: attachment; filename='.basename($file));
        header('Content-Transfer-Encoding: binary');
        header('Expires: 0');
        header('Cache-Control: must-revalidate');
        header('Pragma: public');
        header('Content-Length: ' . filesize($file));
        ob_clean();
        flush();
        readfile($file);
        exit;
}
```

W przypadku potrzeby przechowywania danych pomiędzy poszczególnymi krokami, wykorzystano mechanizm cookies, pozwalający na zapis mały danych po stronie przeglądarki.

Aplikacja pozwala na konfigurację OpenVPN po stronie serwera i wygenerowanie certyfikatu i kluczy dla 1 klienta. Po wykonaniu wszystkich kroków i otworzeniu portu 1194 na routerze, można nawiązać szyfrowane połączenie z urządzeniem spoza sieci lokalnej.



Wnioski

Realizacja dwóch pierwszych projektów pozwoliła na zapoznanie się z obecnie popularnymi technologiami wykorzystywanymi w trakcie budowania witryn WWW, czyli HTML5, JavaScript, JQuery, AJAX. Wiedza ta pozwoliła na zbudowanie strony pozwalającej na prosty monitoring urządzeń działających na systemach bazujących na Debianie.

Realizacja trzeciego projektu, pozwoliła na szczegółowe zapoznanie się z oprogramowaniem OpenVPN, poznanie metod PHP, na wprowadzanie zmian w plikach znajdujących się po stronie serwera, zarządzających Cookie's oraz pogłębieniu wiedzy z zakresu skryptów bash'owych.

Źródła

1) http://www.w3schools.com/
2) http://www.chartjs.org/
3) http://getbootstrap.com/
4) http://raspberrypihelp.net/tutorials/1-openvpn-server-tutorial