FORMIDABLE

Anggota :

1. Alvin Pangantar (14.111.1286)
2. Adlin M. Hasri (14.111.1642)
3. Hen Hen (14.111.1260)
4. Muhammad Rinaldi Hutabarat (14.111.0451)
5. Santri Zahratul Fatmi (14.111.0485)

Jurusan Teknik Informatika

STMIK MIKROSKIL

Medan

2017

Formidable

Apa itu Formidable?

Formidable merupakan sebuah module yang terdapat didalam Node.js yang berfungsi untuk melakukan parsing form data, parsing form data ini biasanya digunakan dalam melakukan upload file.

Kenapa menggunakan Formidable?

Didalam Express 4 fungsi req.files tidak lagi tersedia secara default pada object Req, sehingga untuk mengakses file yang diupload melalui method req.files.object harus menggunakan multipart-handling middleware seperti busyboy, multer, formidable, multiparty atau pez. Diantara beberapa middleware tersebut yang paling sering digunakan berdasarkan rating di github.com adalah formidable, busyboy dan multiparty.

Terdapat beberapa keunggulan dari formidable yaitu :

1. Memiliki speed ~500mb/sec yang bersifat non-buffering multipart parser
2. Write file yang telah diupload secara otomatis
3. Memiliki memory footprint yang kecil
4. Memiliki cakupann test yang cukup luas

Cara instalasi Formidable

Untuk menggunakan formidable maka user perlu menginstall node.js terlebih dahulu dan selanjutnya menginput command *"npm install formidable –save"*

API Formidable

var form = new formidable.IncomingForm() → Membuat IncomingForm yang baru

form.encoding = 'utf-8' → Set tipe encoding untuk field pada IncomingForm

form.uploadDir = "/my/dir" → Sebuah path dimana file yang telah diupload akan diletakkan secara default terletak di temp file os

form.keepExtensions = True → Membuat file yang di write di directory path (form.uploadDir) memiliki ektensi file

form.type → Menentukan tipe antara multipart atau urlencoded yang akan disesuaikan dengan requsest yang akan dikirimkan

form.maxFieldsSize = 2 * 1024 * 1024 → Membatasi jumlah memori semua field (kecuali file) yang dapat dialokasikan dalam bytes. Jika nilai ini terlampaui, sebuah 'error' dimunculkan. Ukuran defaultnya adalah 2MB.

form.multiples = false → jika opsi ini diset True makan ketika kita menggunakan form.parse, objek file akan memiliki array untuk menginput data secara banyak menggunakan atribut multiple di HTTP5

Formidable.File

file.size = 0 → Size file yang diupload dalam satuan byte

file.path = null → path dimana file akan di write. Path dapat diedit melalui ‘fileBegin’

file.name = null → nama file yang diupload oleh user

file.lastModifiedDate = null → berisi date object yang mengindikasikan waktu file terakhir di write

Events

Progress → dikirimkan setiap kali data telah diparsing

Field → dikirimkan saat field / value telah diterima

fileBegin → dikirimkan saat file baru terdeteksi di upload stream, dengan mengganti ini kita dapat memodifikasi path dimana kita ingin meletakkan file

error → dikirmkan ketika terdapat error pada saat pemrosesan form yang sedang diterima

aborted → dikirimkan ketika request dibatalkan oleh user

end → dikirmkan ketika semua request telah diterima dan semua file yang dikirim telah di write kedalam disk.

Implementasi code

```
// Required Modules
var formidable = require('formidable'),
    http = require('http'),
    util = require('util'),
    fs   = require('fs-extra');

    // create server
    http.createServer(function(req, res) {
```

Implementasi formidable kami gunakan dengan server http, jadi pertama kita perlu mendeklarasi setiap variable yang nantinya diperlukan seperti formidable,http,util dan fs setelah itu server di buat dengan method *http.createserver*

```
console.log(req.url);
   if (req.url == '/upload' && req.method.toLowerCase() == 'post') {

 // creates a new incoming form.
 var form = new formidable.IncomingForm();

 // parse a file upload
    form.parse(req, function(err, fields, files) {
      res.writeHead(200, {'content-type': 'text/plain'});
      res.write('Upload received :\n');
      res.end(util.inspect({fields: fields, files: files}));
    });
 form.on('end', function(fields, files) {
        /* Temporary location of our uploaded file */
        var temp_path = this.openedFiles[0].path;
        /* The file name of the uploaded file */
        var file_name = this.openedFiles[0].name;
        /* Location where we want to copy the uploaded file */
        var new_location = __dirname + '/images/upload/';
        fs.copy(temp_path, new_location + file_name, function(err) {
            if (err) {
                console.error(err);
            } else {
                console.log("success!")
            }
        });
    });
    return;
  }
```

Pada bagian ini kita akan melakukan verifikasi untuk melihat route yang dituju, didalam code diatas jika route yang dituju adalah '*/upload*' maka akan dilakukan parsing data yang diinput oleh user dan diupload ke sebuah path. Path secara default berada di temp file windows, selanjutnya akan dilakukan pemindahan data dari temp file ke path yang diinginkan dengan menggunakan method fs.copy dengan parameter temp_path dan new_location sebagai destinasi.

```
else if(req.url.indexOf('.js') != -1){ //req.url has the pathname, check if it conatins '.js'

    fs.readFile(__dirname + req.url, function (err, data) {
      if (err) console.log(err);
      res.writeHead(200, {'Content-Type': 'text/javascript'});
      res.write(data);
      res.end();
    });
}
else if(req.url.indexOf('.css') != -1){ //req.url has the pathname, check if it conatins '.css'

console.log(__dirname+req.url);
console.log(req.url);
  fs.readFile(__dirname + req.url, function (err, data) {
  if (err) console.log(err);
  res.writeHead(200, {'Content-Type': 'text/css'});
  res.write(data);
  res.end();
  });
// res.end()

}
else if(req.url.indexOf('.png') != -1 || req.url.indexOf('.jpg') != -1 || req.url.indexOf('.gif') != -1){ //req.url ha

console.log(__dirname+req.url);
console.log(req.url);
  fs.readFile(__dirname + req.url, function (err, data) {
  if (err) console.log(err);
  res.writeHead(200, {'Content-Type': 'image/jpg'});
  res.write(data);
  res.end();
  });
// res.end()

}
```

Pada code dibagian ini kita melakukan pengecekan file yang akan direquest oleh server, Karena setiap data gambar yang telah diupload akan ditampilkan dengan menggunakan lightbox.js maka diperlukan parsing data setiap file dr lightbox baik html, css dan javascript. Pengecekan dilakukan dengan menggunakan method fs.readFile

```
  else if(req.url == '/'){
    fs.readFile('./index.html',function(err, html){
    console.log(fs);
      if (err) {
        throw err;
      } else {
    /* Displaying file upload form. */
      res.writeHead(200, {'content-type': 'text/html'});
      res.write(html);
    }
    res.end();
    })
```

Dibagian ini kita menentukan route default yang akan diakses oleh server ketika pertama kali dijalankan oleh user. Berdasarkan code diatas maka ketika user mengakses localhost:8080 maka file index.html akan direnderi di route tersebut.

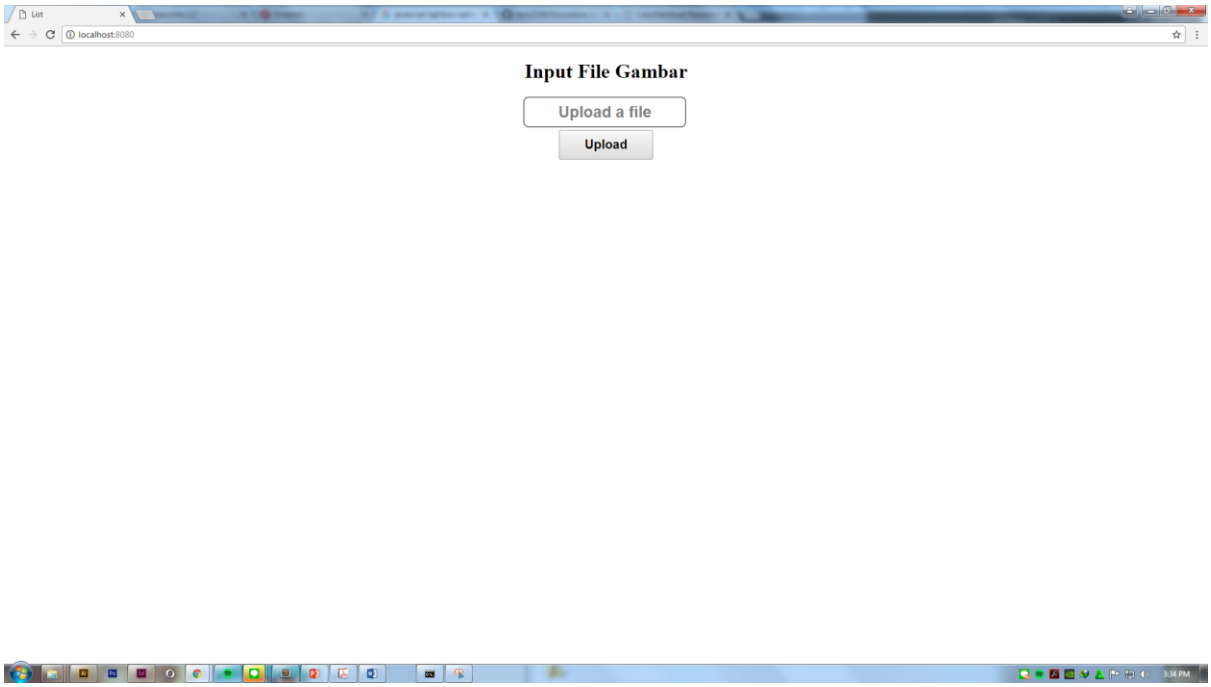
```
  } else if(req.url == '/list'){
    fs.readFile('./lightbox.html',function(err, html){
  console.log(html);
      if (err) {
    console(err);
        throw err;
    res.end();
      } else {
    res.writeHead(200, {'content-type': 'text/html'});
    res.write(html);
    res.end();
    }
    })
  }
}).listen(8080);
```

Dibagian terakhir kita membuat rute untuk melihat setiap gambar yang telah diupload, setiap gambar akan ditampilkan menggunakan lightbox melalui file lightbox.html yang akan dirender di rute tersebut

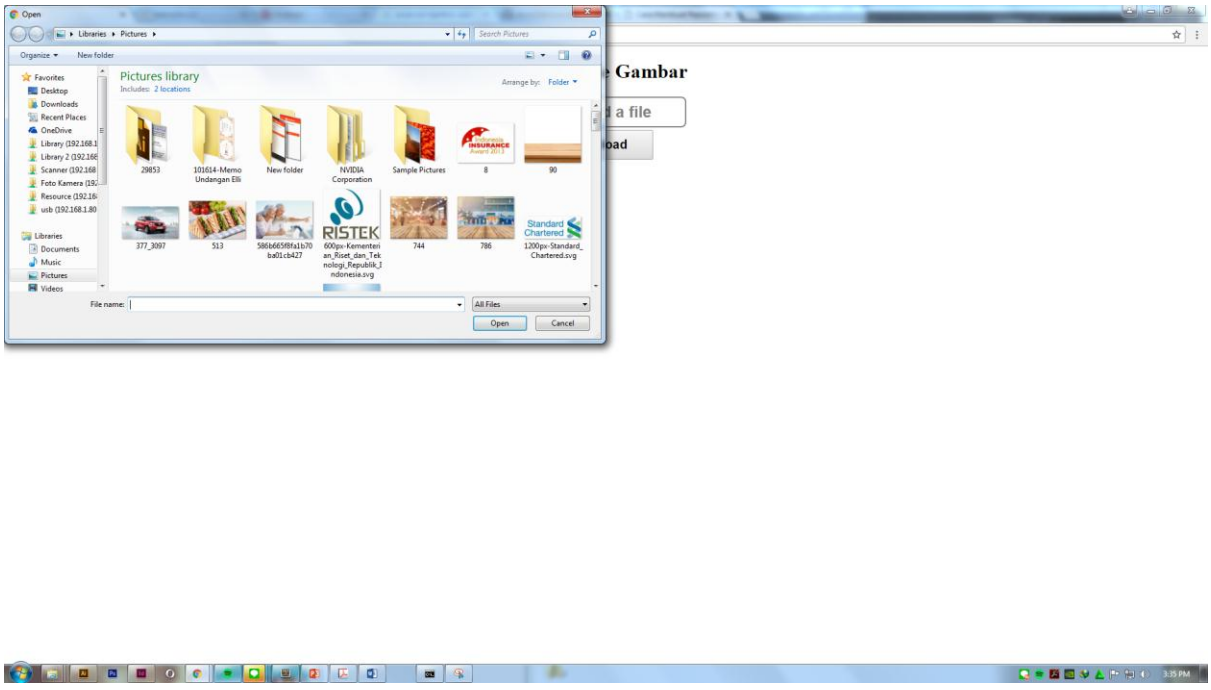
```
    res.writeHead(200, {'content-type': 'text/html'});
    console.log(res.body);
    // res.write(html);
    fs.readdir(__dirname + '/uploads', function(err, filenames)
    {
      //console.log(__dirname + '/images/upload/');
      var headerS = fs.readFileSync("header.html", "utf8");
      var footerS = fs.readFileSync("footer.html", "utf8");
      var image_text = "";
      filenames.forEach(function(filename) {
        // console.log(filename)
        var ext = filename.split(".")[1];
        if(ext== "jpg" || ext== "png" || ext== "gif")
          image_text += '<a class="example-image-link" href="uploads/'+filename+'" data
      });
      res.write(headerS);
      res.write(image_text);
      res.write(footerS);
      res.end();
    });
    // res.writeHead(200, {'content-type': 'text/html'});
    }
    })
  }
}).listen(8080);
```

Dibagian ini dilakukan looping untuk menyimpan setiap file yang telah diupload kedalam sebuah array sehingga dapat diakses ketika ingin ditampilkan menggunakan lightbox.js

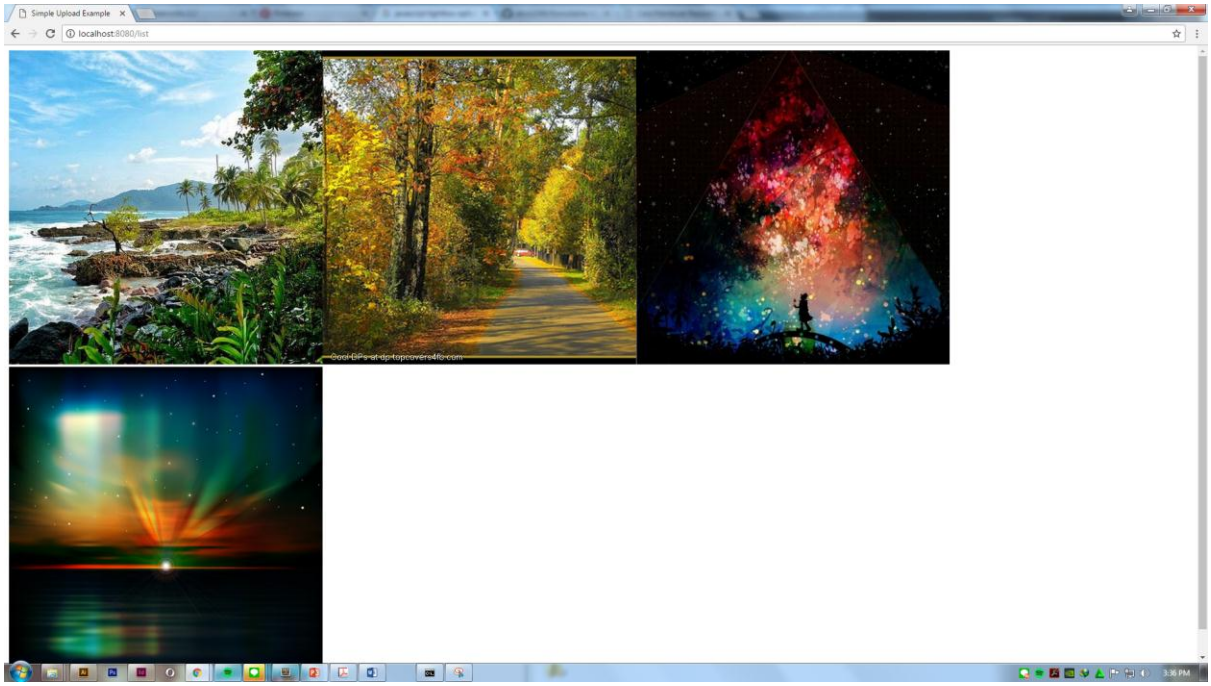
Demo Program



Route default localhost:8080 untuk melakukan upload file



Upload dialog akan muncul ketika kita menekan button Upload a file



Route localhost:8080/list untuk melihat setiap list gambar yang telah diupload, gambar yang terakhir diupload akan berada diposisi kiri atas.