Express Cont...

ExpressJS - Serving static files

- Static files are files that clients download as they are from the server. Create a new directory, **images**. Express, by default does not allow you to serve static files. You need to enable it using the following built-in middleware.
- **app.use(express.static(‘images'));**
- **Note** – Express looks up the files relative to the static directory, so the name of the static directory is not part of the URL.

- Note that the root route is now set to your **images** dir, so all static files you load will be considering **images** as root. To test that this is working fine, add any image file in your new **images** dir and change its name to "**test.jpg**“ .
- Now create your express file as follows-

```
var express = require('express');
var app = express();
const path = require('path');
app.use(express.static('images'));

app.get('/', function(req, res){
  res.sendFile(path.join(__dirname,  'a.html'));
});

app.listen(3000);
```

a.Html –

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

</head>
<body>

<h1>This is a heading</h1>
<p>This is a paragraph.</p>
<img src="test.png" alt="Smiley face" height="42" width="42">
</body>
</html>
```

- Note that the src is set to test.png, there is no need to define the exact path(images/test.png)

- **Multiple Static Directories**
- We can also set multiple static assets directories using the following program −
- In this program we will use a css file and an image as static resources to be used in our program –
- Create a directory – **public,** within it create a html file and css file like this –

a.Html –

```
<!DOCTYPE html>
<html>
<head>
<link rel="stylesheet" type="text/css" href="my1.css">
</head>
<body>

<h1>This is a heading</h1>
<p>This is a paragraph.</p>
<img src="test.png" alt="Smiley face" height="42" width="42">
</body>
</html>
```

My1.css –

```
body {
  background-color: lightblue;
}

h1 {
  color: red;
  margin-left: 20px;
}
```

Index.js –

```
var express = require('express');
var app = express();
const path = require('path');
app.use(express.static('public'));
app.use(express.static('images'));

app.get('/', function(req, res){
   res.sendFile(path.join(__dirname, 'public/a.html'));
});

app.listen(3000);
```

- **The sendFile() Method**
- The Express framework provides a **sendFile()** method available on the response object which can be used to send static files to the client.

Note - path must be an absolute path to the file. To define it we can use path.join() method like this –

res.sendFile(**path.join(__dirname, 'public/a.html')**);

__dirname is the absolute path to the directory containing the source file that is being executed

- (path.join will take care of unneccessary delimiters, that may occur if the given pathes come from unknown sources (eg. user input, 3rd party APIs etc.).
  So path.join('a/','b') path.join('a/','/b'), path.join('a','b') and path.join('a','/b') will all give a/b.

)

- **res.redirect([status,] path)**
- Redirects to the URL derived from the specified path, with specified status, a positive integer that corresponds to an HTTP status code . If not specified, status defaults to “302 “Found”.
- res.redirect('/foo/bar')
  res.redirect('http://example.com')
  res.redirect(301, 'http://example.com')
  res.redirect('../login') Redirects can be a fully-qualified URL for redirecting to a different site:
- res.redirect('http://google.com')

- **Difference between sendFile() and redirect() - ?**

ExpressJS - Form data

- Forms are an integral part of the web. Almost every website we visit offers us forms that submit or fetch some information for us. To get started with forms, we will first install the *body-parser*(for parsing JSON and url-encoded data) and multer(for parsing multipart/form data – generaly used to upload files) middleware.
- To install the *body-parser* and *multer*, go to your terminal and use –
- **npm install --save body-parser multer**

- 1 - **Handling POST forms**
- Express can handle text data of forms as follows -
- **a. Handling text-only data**
- If you want to handle text data submitted via the POST method, you are in luck. You don't have to install any packages. Express has an in-built middleware express.urlencoded, which you can use to process the text data.
- The processed form data is available on the body property of the request object.
- The form:

- Create a html file (login,html) in public dir as follows –

```
<form method="post" action="/handler">
  <input type="text" name="username" /><br />
  <input type="password" name="password" /><br />
  <input type="submit" />
</form>
```

Note - The HTML form element has an attribute named enctype, if not specified, its value defaults to "application/x-www-form-urlencoded" (URL encoded form). The express.urlencoded middleware can handle URL encoded forms only.

- The app:

```
var path = require('path');
var express = require('express');
var app = express();
app.use(express.urlencoded({ extended: true }));
app.use(express.static(path.join(__dirname, 'public')));

app.post('/handler', function (req, res) {
  console.log(req.body);

  res.write(req.body.username);
  res.write(req.body.password);
  res.send();      // or res.end

});

app.listen(3000);
```

- Note – res.send() send the response and end it. So whenever you are required to send multiple data as response, use write() method and finally send it using send() or end() method.

- Now run the app.js and enter the following url in your browser –

The URL: http://localhost:3000/login.html

- Example - If user enter correct name and password, he should be redirected to home page otherwise to login page with some error message.
- Create Home.html in public dir. And then modify app.js as follows –

```
var path = require('path');
var express = require('express');
var app = express();
var fs = require('fs');

app.use(express.urlencoded({ extended: true }));
app.use(express.static(path.join(__dirname, 'public')));

app.post('/LoginChk', function (req, res) {
  console.log(req.body);
```

```
  if(req.body.username == 'admin' && Number(req.body.password) == 123)
  {
        res.sendFile(path.join(__dirname, 'public/home.html'));
  }
  else
  {

        fs.readFile('./public/Login.html', function(err, text){
              text = text+"<p>Wrong user id or password</p>";

             res.send(text);

    });
  }
});

app.listen(3000);
```

- **1. Handling GET forms**
- Login.html –

```
<form method="get" action="/handler">
  <input type="text" name="username" /><br />
  <input type="password" name="password"
    /><br />
  <input type="submit" />
</form>
```