BENVGSC4: Spatial Data Capture and Analysis
Workshop 9 – Interactive Visualisation 3: Node.JS

Welcome to Workshop 9 of Spatial Data Capture and Analysis. Today you're going to learn how to construct a server side API using Node.JS to connect to your MySQL database and develop your JavaScript skills further. We will be using the Flickr Photo Explorer we completed last week to view the new metadata API endpoints that you will create in today's workshop, which will make our app more interactive and more informative for the user.

1 Connecting to the Server using SSH

We are going to develop our Node.JS application on the server, as that's where our database and web application is installed. Users will visit the website and their browser will then connect to our Node.JS API to get the data for the visualisation. To connect to the server we will run a program called Putty (on Windows) that allows developers to connect to the server via the command line which will allow us developers to run the Node.JS program on the server. Use the following instructions to connect to the server depending on the system your using:

Windows Users: Open Putty (http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html) which is installed on the lab machines and type in the following into Hostname:

```
[username]@ dev.spatialdatacapture.org
```

Make sure ssh is selected and that the port number is 22. Then press Open. To disconnect from the server just close the putty window that appears.

Mac Users: Use the Terminal App (Macintosh HD ▸ Applications ▸ Utilities ▸ Terminal.app) which is pre installed on all Macs. To connect to the course server use your UCL username and type removing the square brackets:

```
ssh [username]@dev.spatialdatacapture.org
```

The server will ask for your password (which is your UCL password). Note that as you type, your password will not be displayed but the text is being sent to the server. Press Enter to submit your password and if correct you'll be connected to the server. To disconnect from the server just close the window.

2 Basic UNIX Commands

You may or may not be used to using a command line driven interface to edit and interact with files so I have included a basic guide on how to navigate around the server using the command line. It is important to note that in the UNIX command line everything is either a file or a folder and everything you can do with a GUI driven interface (such as Windows or Mac Finder) can be done with the command line.

Here is a list of common commands you'll need to interact with the server:

```
touch foo     – Creates a file called foo
rm foo        – Deletes a file called foo
mkdir bar     – Creates a folder called bar
rmdir foo     – Removes a folder called foo
cd bar        – Change to directory bar
cd ..         – Go back 1 directory
ls .          – Prints all files in current directory
nano bar.js   – Opens (or creates if it doesn't exist) a file called bar.txt in editor
```

Node Specific

```
node bar.js   – Executes bar.js file in node
```

```
forever start bar.js  - Runs bar.js in node forever (if it crashes forever will
                         restart the app automatically.
```

We expect you to use the command line to run your node programs rather than running them locally due to configuration issues. Also this will allow you learn more about the process of remotely debugging applications. To edit files on a remote server use either WinSCP or CyberDuck (Mac) using the instructions from Workshop 8, Section 6. We recommend you use `nano` to edit the files on the web server, as this will avoid confusing local files with remote files.

3 Running your first Node.JS Program

As you have learned in the lecture, Node.JS allows developers to write fast and scalable network applications using JavaScript. Unlike JavaScript code running inside the browser, Node.JS is **not** sandboxed which means that your JavaScript program, which is running with Node, can access all parts of your system. Care should be taken when running code that you've copied from the web. Understand what the code does before running, as any malicious code will have full access to your system. We are not going to cover the installation of Node on your local system but if you want to install node you can download and setup by following the instructions at `https://nodejs.org`.

When you have connected to the server, create a folder called node. Inside that folder, create another folder called firstProgram. Type in the following code into a file called first.js and save the file:

```
// First Node.JS Program
// Author:  Your Name

var colors = require("colors");
var title = "My First Node.JS Program";
console.log(("*********** " + title + " ***********").red);
console.log("Welcome to your first node program".green);
```

(**Note:** All Node.JS programs must have the extension **.js**)

Run this program by navigating to the folder using the command list in Section 2 and type:

```
node first.js
```

You'll notice that the code will not run, as we need install an external library called `colors`. All of node's packages are stored in a single repository called `npm` (https://www.npmjs.com) and you can install any of these packages by typing:

```
npm install [package]
```

When you install a package with npm it will create a folder in your current directory called `node_modules`. This makes node applications very portable as all the libraries or packages you want to use can be installed in the users local folder rather than system wide. Also it allows developers to distribute their code to other developers or different machines extremely easily.

Install colors by typing:

```
npm install colors
```

Now try and run your Node.JS program again.

If your code runs then download todays workshop code by typing the following commands into the command line on the server:

```
wget http://dev.spatialdatacapture.org/data/workshop_9/server.zip
unzip server.zip
```

4 Building your first API

For the rest of today's workshop we will build an API using node's express server and we will connect this to our database. The code you have downloaded is the server API we used last week to return the Flickr photos from our database tables. We will extend the API to include some functions to get specific data about the types of phones used to take the photographs and create some clickable buttons on our interface to select the different data points, which will make our visualisation even more interactive.

We've explained how this code works during the lecture and you should also have a port number, which was given to you at the start of the workshop. In the API code sample you have just downloaded replace the relevant variables (username, password, and database) with your user account details given to you on Week 1. Also at the top of the application change the relevant port number to the port number on your sheet of paper.

Install the packages `moment, mysql, express and ejs` inside the folder you have just extracted (**`flickrServer)`**

Run the server using the command:

```
node flickrDataServer.js
```

(Note: to stop the server press ctrl-c)

When you run this code it will create a webserver on that port number, which will respond to requests you send it. To test that your API is working visit the following page in your browser. Make sure you remove the square brackets.

`http://dev.spatialdatacapture.org:[yourPortNumber]`

Using the code that is already provided in the sample, extend the API to respond to the endpoint "date" and have the API return the current date and time of the server. **Hint**: Look at the Date function in JavaScript (http://www.w3schools.com/jsref/jsref_obj_date.asp) and you can copy, paste and edit the code that responds to the "/" event.

Once you have completed the changes restart the server (to stop the server press `cntrl-c`) and make sure API responds correctly in your browser on before moving on in the workshop.

```
http://dev.spatialdatacapture.org:[yourPortNumber]/date
```

Now navigate into the views folder in the API and update the **`index.ejs`** file to include some html reference for your new API endpoint. It is always good development practise to have a document that lets users know what endpoints are available for your API so that other developers can access your data.

The reason that the file extension is ejs rather than .html is that node uses an engine called **ejs** to replace variables inside your html file before rendering the html that gets returned to the browser. This is known as a template engine. This allows the developer to reuse code by building templates to make a page more dynamic depending on values passed to the page. If you want to learn more look at: http://www.embeddedjs.com/

There are also some good tutorials here:
https://scotch.io/tutorials/use-ejs-to-template-your-node-application

Now, using MySQL Workbench and the SQL skills you learned in weeks 1-3, build a query which gets a list of the top 10 most popular used camera makes within your metadata table. Make a note of your findings, as you'll need this list in section 4 to create your interface inside the web app.

Create another new endpoint using the code in the sample to return all the photos that has been taken by a specific camera. You'll need to come up with an SQL statement that takes a type of camera and returns all the photos that has been taken by that camera – `(pid, lat, lon)`. Build the query in MySQL workbench first to test and then form the API endpoint, which will take a string as the input and concatenates the value into the SQL statement. The new API end point should have the name **/data/cameraType/[value]**. Also make sure you make your SQL statement safe by parsing the string value with the function **mysql_real_escape_string**, which will prevent SQL injections. Use the code that we have provided and adapt it to create the new end point.

Again before moving onto section 4 make sure that if you call your end point, the correct data is returned from the database.

5 Integrating the API with the Web Visualisation

Now you should have a working end point that looks something like this:

```
http://dev.spatialdatacapture.org:[yourPortNumber]/data/cameraType/[value]
http://dev.spatialdatacapture.org:[yourPortNumber]/data/cameraType/nokia
            (for example)
```

When you have confirmed your API is working correctly and is returning the correct data move back to your local machine and download the W

We can now edit our Web Application to call the new endpoint to get the data from the API and display it on the map. We want to add a few links (10 in total) into our information div on the left hand side of the page. When we click the links we want the browser to first clear the map fetch the data from the API then display the data on the page. The photo markers on the page will show the individual markers on the page and allow you click to see the photos. Also we want to change the colour of the markers to show the different type of photographs shown. If you look in the **img** folder of this workshop (downloaded from Moodle) you'll see at least 5 different colours of marker that you can use for this. The `getCameraData` function takes a value for the camera type and an image name for the marker image: e.g.

```
getCameraData("nokia", "blue_marker.png");
```

If you look at the JavaScript code in web application you'll see that we've defined a click element that will call a function when clicked (using jQuery). Your final task in this workshop is to create the 10 links that when clicked will clear the map and get the new data using your API end point. Have a look at the getCameraData() function and understand what is happening inside the function. Ensure you define your API endpoint inside this function with the correct port number for your API, and create all 10 links for the Top 10 camera links you discovered from the data analysis you performed in Section 3.

All the code to allow you do this is in this file, just hack around the code that we have provided to get it working. Don't worry about not finishing the workshop by the end of today; just like last week we will explain what is happening inside the code at the end of the workshop. Good Luck.

See you all next week for the final lecture and workshop of the course!