#Chapter 8: Watson Discovery

Return to Table of Contents

This is an update to the previous Chapter 8, where we used Alchemy. This will require that you activate the Watson Discovery service in Bluemix, replacing the Alchemy service if you had it. Much of this chapter is unchanged. There is a structural change to the application and changes in four files, described below.

Structural change

The original Chapter 8 called the alchemy service directly from the browser. This was, in part, because most of the Alchemy documentation showed the service being accessed from curl commands, which require that you call a URL rather than using the SDK. Equally, it was an opportunity for you to experiment with accessing these services directly from the browser. This does have a notable downside that the apikey is exposed in the browser side javascript.

In this updated version, we are removing the direct call from the browser and using the same approach as used in other chapters, which is to call Watson services from the nodeJS server. This means that we need to add code to the server *discovery.js* which handles access to the news service and all credential requirements. The browser will still call a URL and process the same information, it will just call a different url and will need less code.

Server Side changes

- **controller/features/discovery.js**

This file is new and has two routines exported from it, one is intended to be informative, but is not used. That's the 'getID' function. The other is, appropriately, 'getNews', which is explained in the javascript file nearly

line by line.

- **env.json**

Because we're using Discovery, rather than Alchemy, we'll add Discovery to our env.json file:

```
        "discovery": {
           "version": "v1",
           "version_date": "2017-10-16",
           "url":
"https://gateway.watsonplatform.net/discovery/api",
           "password": "<Your password here>",
           "username": "<your username here>"
        },
```

- **controller/router.js**

There are two changes to this file, one is the necessary statements for the new get and post commands. The other is a routine which logs to the terminal window each request to the server. I've started adding this to all of my demonstrations, primarily to help me both understand and explain what is going on between the client (browser) and the server. That code is the following:

```
var count = 0;
router.use(function(req, res, next) {
 count++;
 console.log('['+count+'] at:
'+format.asString('hh:mm:ss.SSS', new Date())+' Url
is: ' + req.url);
 next(); // make sure we go to the next routes and
don't stop here
});
```

- the count variable increments for every call and enables simple sequence tracking.
- format.asString takes the current date and provides a timestamp on the request
- req.url is the inbound request from the browser to the server.

Client (Browser) changes

- **z2c-alchemy.js** is replaced by **z2c-discovery.js**

In the original file, we made api calls directly from the browser to the alchemy service; so one of the responsibilities of the browser javascript code was to correctly format the alchemy request url. We are now using a server side routine to handle that, so the following changes have been made:

- The browser code now takes the data from the browser page and sends it directly to the server.
- The browser code now determines the start and end date, rather than passing the number of days. While we could, still, just pass the number of days, the code to calculate the current and offset date is identical regardless of where we execute it.
- The browser code now executes a POST command to request the news information where previously it issued a GET command
- We have one new line of code in the display routine, which now includes the summary text of the returned articles in the accordion.