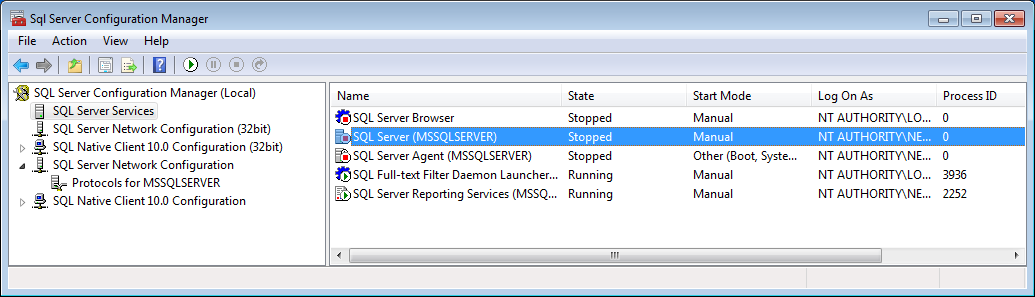
Start SQL Server Services

1. Turn on SQL Server services from Windows Start -> SQL Server Configuration Manager
2. SQL Server Network Configuration -> Protocols for MSSQLSERVER -> TCP/IP-> (RHC) Enabled



1. SQL Server Services -> SQL Server (MSSQLSERVER) -> (RHC) Start
2. SQL Server Services -> SQL Server Browser -> (RHC) Start

Update Express Routing

Carrying on from Session 12.

1. Open up application in VSC (app sql – from previous session).
2. Create a folder called ‘client’ – this is where the Angular code will eventually go.
3. Inside ‘client’ create file ‘index.html’ and enter following code:

<h1>Hello from index</h1>

1. Inside index.js, add default route to index.html; also Express needs to know where to look for html files (inside ‘client’) so update file t:

let express = require('express');

let sql = require('./routes/sqlServerSetup');

let ucbDBRoute = require('./routes/ucb');

let app = express();

let PORT = 1337;

// Folders where Express looks for static files - client side HTML/images/JS/node\_modules

app.use(express.static('client/src'));

app.use(express.static('client'));

// Route for web address /ucb/...

app.use('/ucb',ucbDBRoute);

// Default route

app.get('/', (req,res) => {

// Note will look in '/client', '/client/src', for html file

res.render('index.html');

});

// Catch all

app.get( '\*', (req,res) => {

res.redirect('/');

});

app.listen(PORT, () => {

console.log('Listening on port: ' + PORT);

});

1. Open an Integrated Terminal and type

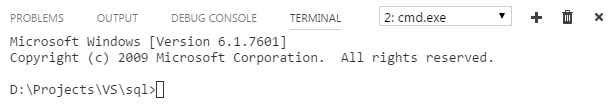
nodemon

1. Open a browser and navigate to application and you should see the ‘Hello..’ message.
2. Try commenting out the two ‘app.use(express…)’ lines and refreshing the browser, you should see an error. This is because express need to know where the html files are.

Put the two lines back in.

Add Angular 4.0.0 Quickstart to app

1. Delete index.html (this is because Angular generator needs an empty directory)
2. In the Integrated Terminal press the ‘+’ button. This will create a second Integrated Terminal (2:cmd.exe) (nodemon will still be running in terminal 1).



1. Paste following to clone the Angular 4.0.0 quickstart template

git clone https://github.com/angular/quickstart.git client

1. Change directory to ‘client’
2. Delete non-essential files by typing:

for /f %i in (non-essential-files.txt) do del %i /F /S /Q

rd .git /s /q

rd e2e /s /q

1. Install the Angular modules by typing:

npm install

If you refresh VSC Explorer you should see two node\_modules, one on the root and one inside ‘client’ which are the Angular modules

1. Try refreshing the browser, you should see ‘Loading AppComponent content here..’ message. It is trying to load JavaScript files, but the TypeScript (ts) files have not been transcompiled into JavaScript (js) yet. Do this by typing:

npm run build:watch

1. Try refreshing browser – you should see ‘Hello Angular’ message

Add UCB Menu

1. Find index.html under client/src and change the title to:

<title>MEAN UCB</title>

1. Add reference to bootstrap just above Polyfill comment

<!-- Include bootstrap style -->

<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.1.1/jquery.min.js"></script>

<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>

1. In app.component.ts (under client/src/app) add the menu navigation

import { Component } from '@angular/core';

@Component({

  selector: 'my-app',

  template: `

    <div>

        <nav class='navbar navbar-default'>

            <div class='container-fluid'>

                <a class='navbar-brand'>{{pageTitle}}</a>

                <ul class='nav navbar-nav'>

                    <li><a [routerLink]="['/welcome']">Home</a></li>

                    <li><a [routerLink]="['/incidents']">Incident List</a></li>

                </ul>

            </div>

        </nav>

        <div class='container'>

            <router-outlet></router-outlet>

        </div>

     </div>

     `,

})

export class AppComponent  {

   pageTitle = 'UCB';

  }

1. While in app.module.ts add the routing module:

import { NgModule }      from '@angular/core';

import { BrowserModule } from '@angular/platform-browser';

import { RouterModule } from '@angular/router';

import { AppComponent }  from './app.component';

@NgModule({

  imports: [

    BrowserModule,

    RouterModule.forRoot([

      { path: '', redirectTo: 'welcome', pathMatch: 'full' },

      { path: '\*\*', redirectTo: 'welcome', pathMatch: 'full' }

    ]),

  ],

  declarations: [ AppComponent ],

  bootstrap:    [ AppComponent ]

})

export class AppModule { }

1. Refresh the browser and you should see the UCB nav bar.

Add Incident Service

1. Under app create directory incidents with file incident.service.ts inside it.
2. Copy following code into incident.service.ts:

import { Injectable } from '@angular/core';

import { Http, Response } from '@angular/http';

import { Observable } from 'rxjs/Observable';

import 'rxjs/add/operator/do';

import 'rxjs/add/operator/catch';

import 'rxjs/add/operator/map';

@Injectable()

export class IncidentService {

private \_incidentUrlBase = '/ucb/';

// Reference Data

titles: any;

incidentCategories: any;

incidentTypes: any;

abuseTypes: any;

controlMeasures: any;

errorMessage: string;

// inject Http component

constructor(private \_http: Http) {

console.log('IncidentService constructor');

// Injectable components do not have ngOnInit, so call directly

this.Initialize();

}

// Injectable components do not have ngOnInit

private Initialize(): void {

console.log('IncidentService.ngOnInit');

// getReferenceData returns an Observable so need to Subscribe to results

this.getReferenceData()

.subscribe( data => { // success

console.log('IncidentService.ngOnInit.getReferenceData.subscribe - got data now');

this.titles = data.title;

this.incidentCategories = data.incidentCategory;

this.incidentTypes = data.incidentType;

this.abuseTypes = data.abuseType;

this.controlMeasures = data.controlMeasure;

},

error => this.errorMessage = <any>error

);

}

// Public methods

getReferenceData(): Observable<any> {

// Http returns an Observable

return this.\_http.get(this.\_incidentUrlBase + 'referenceData')

.map( (response: Response) => <any[]> response.json())

.do(data => console.log('All IncidentService.getReferenceData: ' + JSON.stringify(data)))

.catch(this.handleError);

}

getIncidents(): Observable<any[]> {

return this.\_http.get(this.\_incidentUrlBase + 'getAllIncidents')

.map((response: Response) => <any[]> response.json())

.do(data => console.log('All IncidentService.getIncidents: ' + JSON.stringify(data)))

.catch(this.handleError);

}

getIncident(id: number ): Observable<any> {

return this.\_http.get(this.\_incidentUrlBase + id )

.map((response: Response) => response.json())

.do(data => console.log('This IncidentService.getIncident(id): ' + JSON.stringify(data)))

.catch(this.handleError);

}

updateIncident(incident: any): Observable<any> {

return this.\_http.put(this.\_incidentUrlBase + 'updateIncident', incident)

.map((res: Response) => {

const data = res.json();

console.log('updateIncident status: ' + data.message);

return data.message;

})

.catch(this.handleError);

}

private handleError(error: Response) {

console.error(error);

return Observable.throw(error.json().error || 'Server error');

}

}

1. Update app.component.ts so incident.service is injected into it:

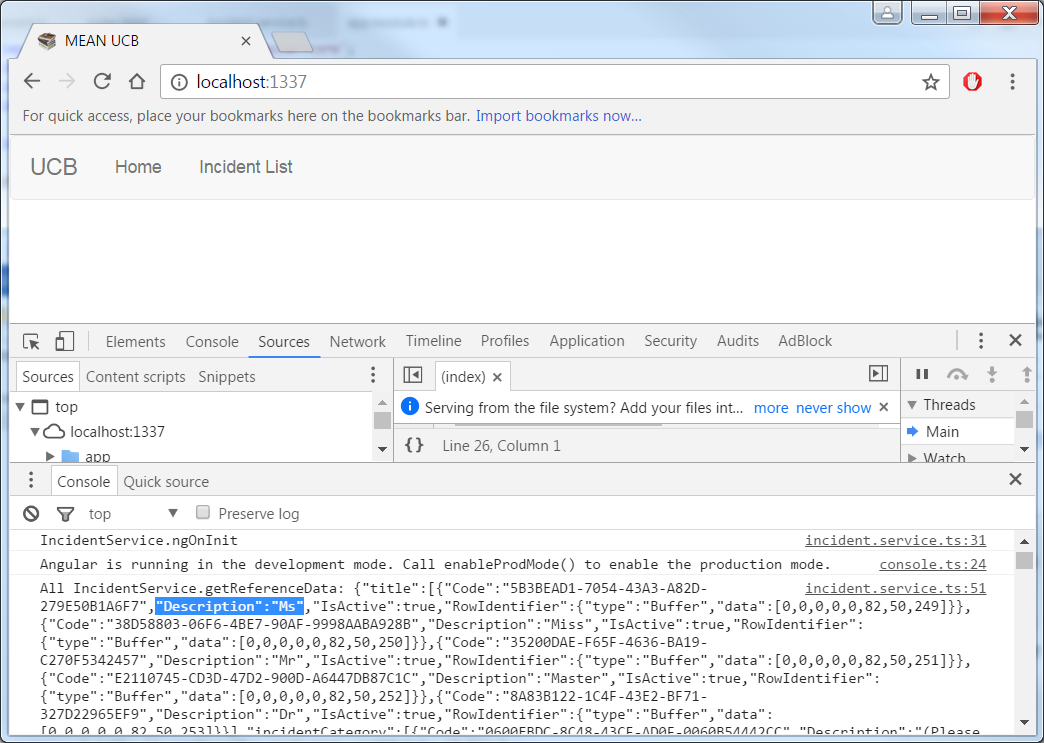


1. Finally update app.module.ts so that the Http module is available



1. Refresh browser.

Nothing appeals to have happened, however in Chrome press F5 (F12 in IE – I think) and notice in the Console window all the Reference data – this is because when app.component is created it gets IncidentService inject into it which retrieves all the Reference data (via Initialize() method)



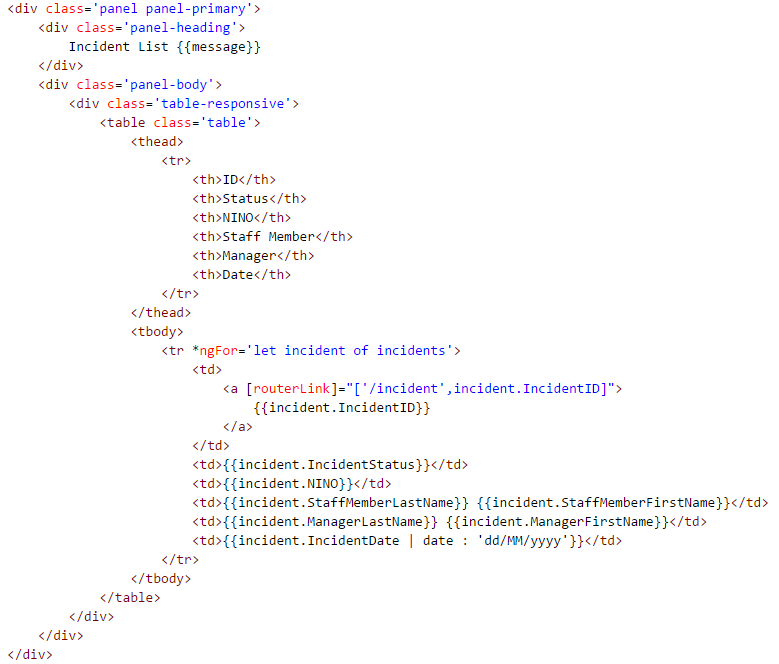
Congratulations you now have Sql Server, Express, Angular & Node all linked up

Add Incident List View

1. In incidents folder add file incident-list.component.ts and copy following code



1. Add incident-list.component.html copy table code



1. Finally import the IncidentListComponent into app.module



1. Refresh the browser and click the Incident List link and you should see a table of incidents

