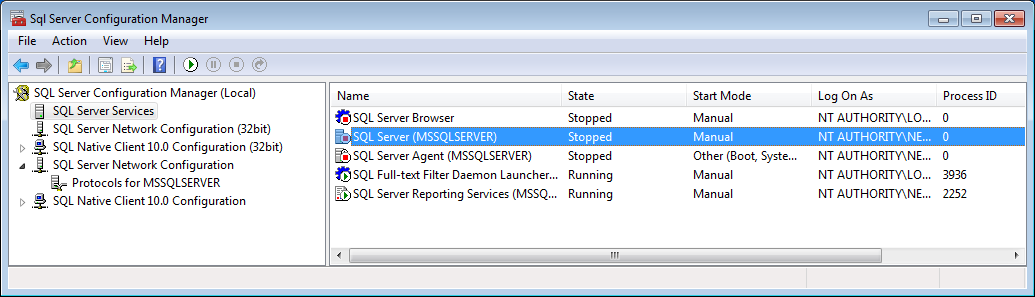
Start SQL Server Services

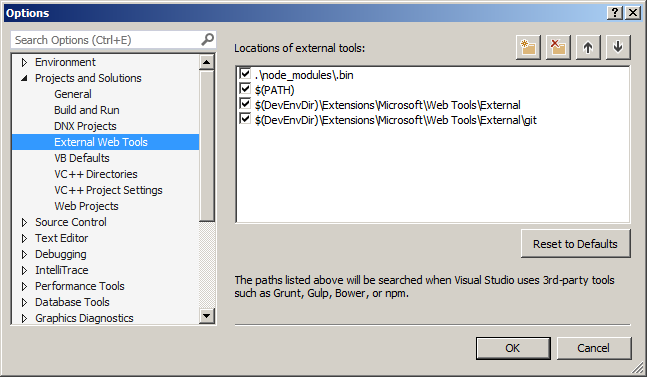
1. Turn on SQL Server services from Windows Start -> SQL Server Configuration Manager
2. SQL Server Network Configuration -> Protocols for MSSQLSERVER -> TCP/IP-> (RHC) Enabled



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

Setup VS2015 to Use TypeScript

1. Open the UCB application (in a VS2015)
2. Make sure it builds and runs (F5) and opens up UCB in a browser.
3. Select Tools->Options->Projects and Solutions->External Web Tools and make sure order of external tools is as shown (use arrow keys to move items).



1. Close VS2015
2. Download load and install <https://www.microsoft.com/en-us/download/details.aspx?id=48593>
3. In a command window navigate to D:\Projects\DotNet and type:

npm install typings –g

Add Angular 4.0.0 to UCB

1. In the command window paste following to clone the Angular 4.0.0 quickstart template

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

1. Change directory to ‘clientucb’
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. From ‘clientucb’ folder copy all 4 files & 1 folder into D:\Projects\DotNet\Ucb\UcbWeb

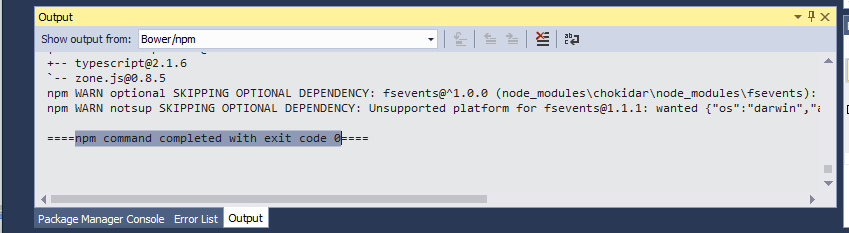
(Replace/update all files when prompted)

1. Open the UCB application (in VS2015)

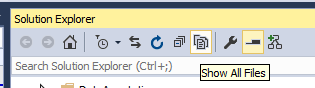
VS2015 should automatically be installing node\_modules from the package.json file you have just copied from the ‘clientucb’.

Select Output window -> show output from: Bower/npm, after a minute or two you should see message

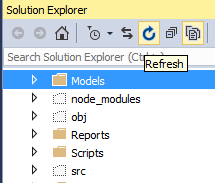
npm command completed with exit code 0



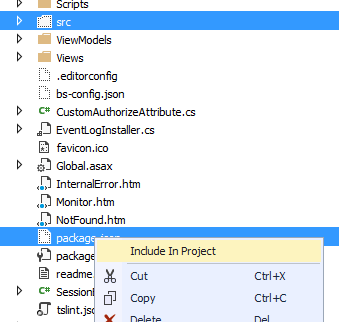
1. Click on ‘Show All Files’ icon



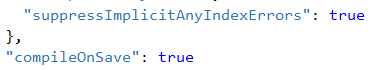
1. You should see node\_modules now – if you don’t click the ‘Refresh’ icon



1. Select hidden folder/file ‘src’ & ‘package.json’ ->RHC ->Include In project



1. Open file src\tsconfig.json and add compileOnSave option



1. Open file src\systemjs.config.js and append ‘src’ to following:

'app': '**src**/app

loader: '**src/**systemjs-angular-loader.js'

1. Rebuild solution:
   1. Build->Clean Solution
   2. Build->Build Solution
2. Run app to ensure it runs ok

Add Default Angular page

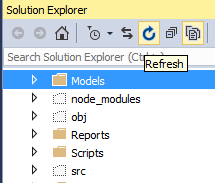
In this section you will add a new

* Controller
* View
* Modify \_Layout.cshtml where Angular will be injected

1. Add new View by

Highlight Views->RHC->Add->New Folder->Angular

Refresh screen



1. Move file src\Index.html into this new Views\Angular folder
2. Rename it Index.**cs**html
3. Update Index.cshtml to include bootstrap and script locations under ‘src’

<!DOCTYPE html>

<html>

<head>

<title>Angular UCB</title>

<base href="/">

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1">

<link rel="stylesheet" href="~/src/styles.css">

<!-- 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>

<!-- Polyfill(s) for older browsers -->

<script src="node\_modules/core-js/client/shim.min.js"></script>

<script src="node\_modules/zone.js/dist/zone.js"></script>

<script src="node\_modules/systemjs/dist/system.src.js"></script>

<script src="~/src/systemjs.config.js"></script>

<script>

System.import('../src/main.js').catch(function(err){ console.error(err); });

</script>

</head>

<body>

<my-app>Loading AppComponent content here ...</my-app>

</body>

</html>

1. Highlight Controllers->RHC->Add Controller->Add->AngularController->Add

Refresh screen

1. Copy following so AngularController inherits from BaseController with a default action Index() which is called when the controller is called without an action (i.e. <http://url/Angular>), which calls the index page.

using Dwp.Adep.Ucb.ResourceLibrary;

using AutoMapper;

using UcbWeb.Helpers;

using UcbWeb.ViewModels;

using UcbWeb.Models;

using System.Web.Mvc;

using UcbWeb.UcbService;

using System;

using System.Net;

using Newtonsoft.Json;

using Newtonsoft.Json.Serialization;

using System.Collections.Generic;

namespace UcbWeb.Controllers

{

public class AngularController : BaseController

{

private IUcbService UcbService;

public AngularController()

: this(new UcbServiceClient(), new SessionManager(), new CacheManager())

{

}

public AngularController(IUcbService UcbService, ISessionManager sessionManager, ICacheManager cacheManager)

: base(sessionManager, cacheManager)

{

this.UcbService = UcbService;

}

public ActionResult Index()

{

ViewBag.IsAngular = true;

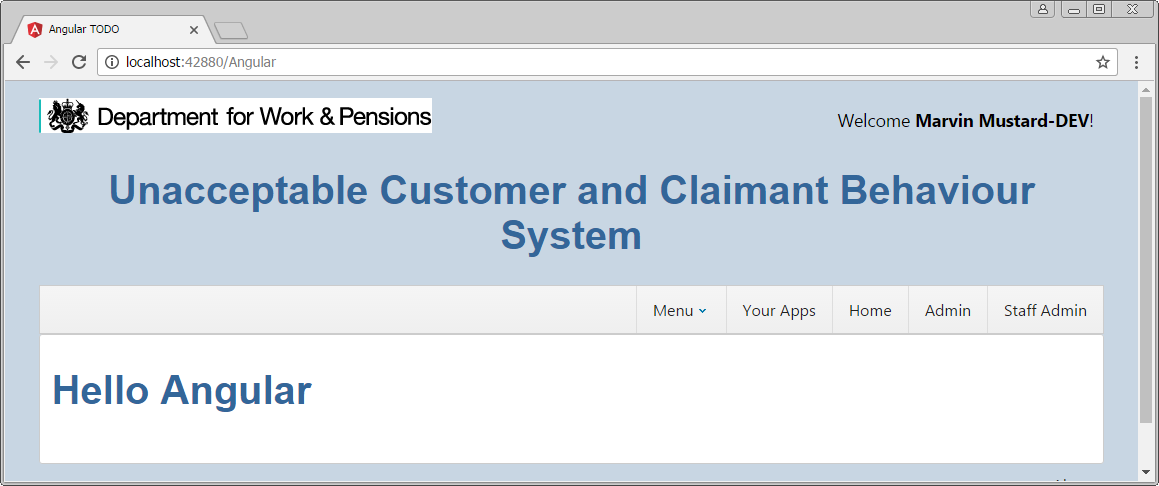
return View();

}

}

}

1. Rebuild
   1. Build->Clean Solution
   2. Build->Build Solution
2. Run app
3. Navigate to this controller http://localhost:42880/Angular , you should see default ‘hello’ message from scr\app\app.component.ts



1. Add a menu link by opening Views\Shared\\_Layout.cshtml and wrap an IsAngular condition around the <scripts…> (these scripts are for .Net only not for Angular)



1. Wrap another IsAngular condition
   1. below the SYS\_NAME
   2. removing the ‘@’ by the string.IsNullOr..
   3. add ActionLink



1. Refresh browser you should now have a clickable link ‘Angular’.

Click link the ‘Hello..’ message should appear again.

Add Angular Menu

1. In src\app\app.compoent.ts 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' }

    ]),

  ],

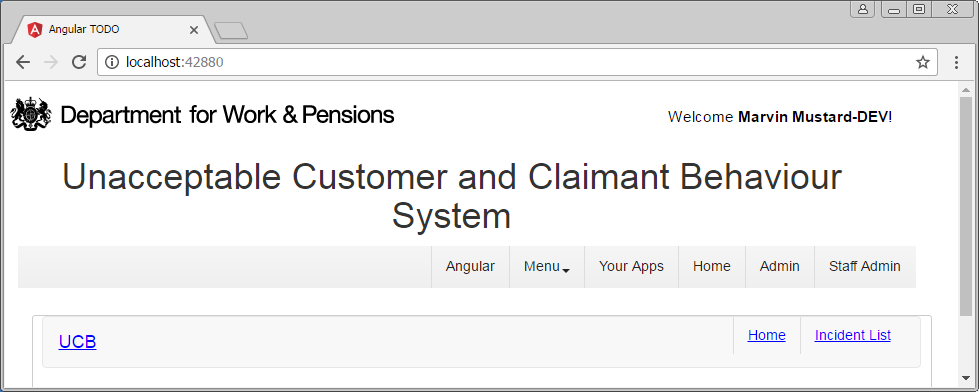
  declarations: [ AppComponent ],

  bootstrap:    [ AppComponent ]

})

export class AppModule { }

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



Add Incident Service & Controller

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 {

// MVC pattern is http://Url/Controller

private \_incidentUrlBase = '/Angular/';

errorMessage: string;

// inject Http component

constructor(private \_http: Http) {

console.log('IncidentService constructor');

}

// Public methods

getIncidents(): Observable<any[]> {

// Http returns an Observable

// MVC pattern is http://Url/Controller/action

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 by GUID not ID

getIncident(guid: string): Observable<any> {

return this.\_http.get(this.\_incidentUrlBase + 'getIncident?guid='+guid)

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

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

.catch(this.handleError);

}

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

return this.\_http.post(this.\_incidentUrlBase + 'updateIncident', incidentVMDetails)

.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. Update app.module.ts so that the Http module is available



1. Add getAllincidents() to AngularController.cs

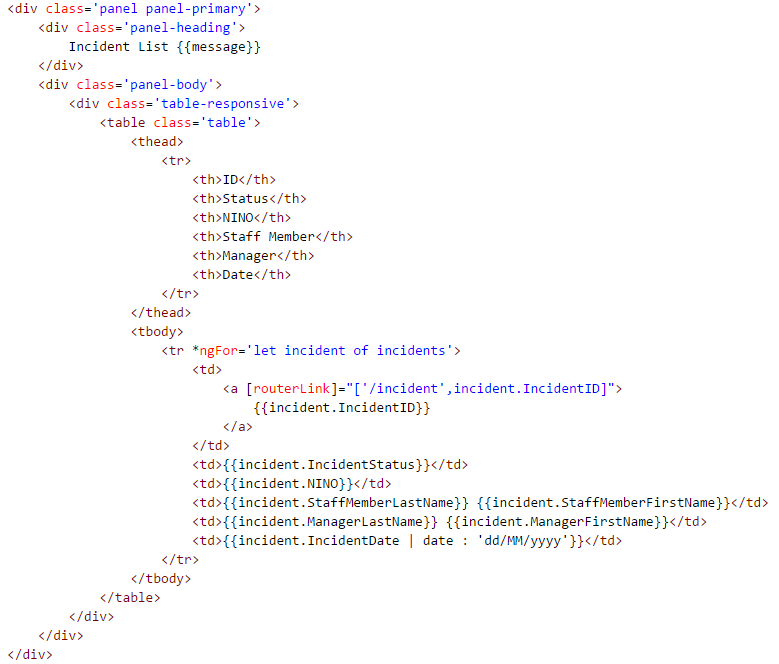


Add Incident List View

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



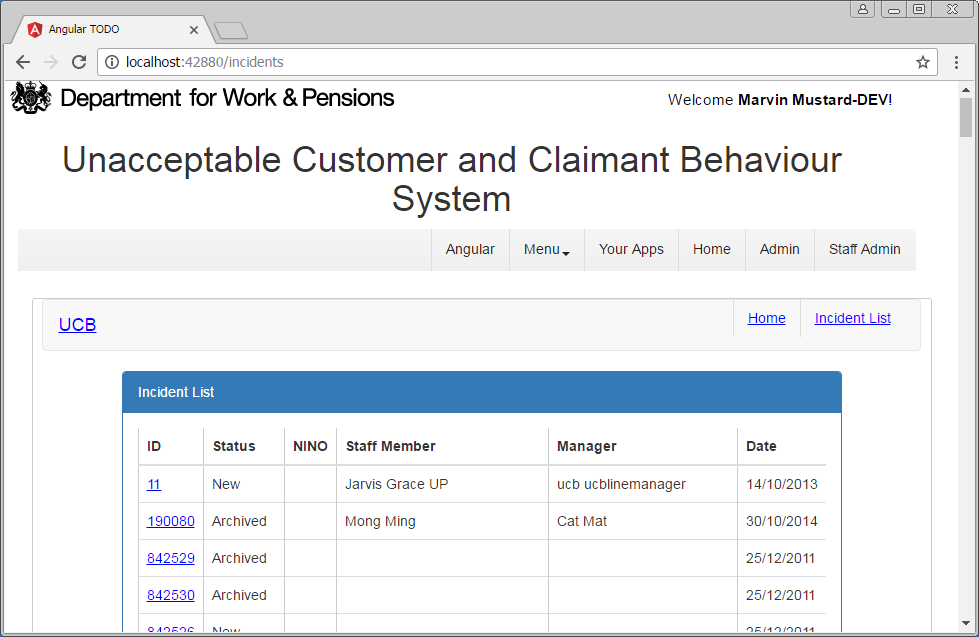
1. Add incident-list.component.html copy table code



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



Congratulations this is an Angular screen which is hosted in IIS running a .Net application

Note NINO empty because the .NET version of the service layer does not include it.