Workshop 4 – Building a Full-stack Application

Agenda

- Create a JavaScript frontend

Couchbase
DEVELOPER COMMUNITY

Platforms

- Angular 2 (UI)

Couchbase
DEVELOPER COMMUNITY

Angular: Starting a new project

1. npm install –g @angular/cli
2. ng new <yourprojectname>
3. ng generate component <componentname>
4. ng build
5. ng serve

   access via http://localhost:4200

Angular / TypeScript

- TypeScript "compiles" to JavaScript
- Polyfills are included by default
- Compiled into a 'dist' folder

Dependencies

- app.module.ts
  - Declare the components
  - Setup the imports (dependencies)
  - Setup the providers
  - Setup the bootstrap component

src/app/app.module.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { FormsModule } from '@angular/forms';
import { HttpModule } from '@angular/http';
import { RouterModule } from '@angular/router';
import { Utility } from './utility';

import { AppComponent } from './app.component';
import { MyComponent } from './pathto/my.component';

@NgModule({
  declarations: [
    AppComponent,
    MyComponent
  ],
  imports: [
    BrowserModule,
    FormsModule,
    HttpModule,
    RouterModule.forRoot([
      { path: "foo/bar/:someId", component: MyComponent }
    ])
  ],
  providers: [Utility],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

src/app/thing/thing.component.ts

```
import { Component, OnInit } from '@angular/core';
import { Http } from '@angular/http';
import { Utility } from '../utility';

@Component({
  selector: 'app-item',
  templateUrl: './thing.component.html',
  styleUrls: ['./thing.component.css']
})
export class ThingComponent implements OnInit {

  public constructor(private http: Http, private utility: Utility) {

  }

  public ngOnInit() {

  }

  public whateverMethod() {

  }
}
```

GET using Http

```
import { Component, OnInit } from '@angular/core';
import { Http } from '@angular/http';
import { Item } from '../item';
import 'rxjs/add/operator/map';
import 'rxjs/add/operator/do';

@Component({
  selector: 'app-list',
  templateUrl: './list.component.html',
  styleUrls: ['./list.component.css']
})
export class ListComponent implements OnInit {

  public people: Array<Item>;

  public constructor(private http: Httpy) {
      this.people = [];
  }

  public ngOnInit() {
    this.http.get("http://localhost/api/getAll")
      .map(result => result.json())
      .subscribe(results => {
          this.people = results;
      }, error => {
          console.error(error);
      });
  }
}
```

List Template

```
<table class="table table-striped">
    <thead>
        <tr>
            <th>First Name</th>
            <th>Last Name</th>
            <th>Email</th>
        </tr>
    </thead>
    <tbody>
        <tr *ngFor="let person of people">
            <td>{{person.FirstName}}</td>
            <td>{{person.LastName}}</td>
            <td>{{person.Email}}</td>
        </tr>
    </tbody>
</table>
```

Events

```
<a href="#" (click)="bar("parameter1","etc")">do something</a>
```

Router

```
<a [routerLink]="['/foo', thing.foo]">edit</a>
```

Angular

- Look for TODOs

- Files included:
  - app.module.ts
  - utility.ts
  - item.ts
  - item.component.ts
  - item.component.html
  - list.component.ts
  - list.component.html

How to execute

Execute Angular 2 app:

- `ng serve`

Questions

Exercise: Fill in the blanks

- Fill in the blanks to make the application work
  - angular_workshop
  - Completed versions are in the angular folder

- Checkout the code from Github
  - https://github.com/couchbaselabs/aspnet-nosql-workshop/tree/master/04

- The source code is also available on USB sticks

Exercise: Getting Started

We're going to fill in one of the blanks together, one for each language/platform.

The rest of the them are up to you.

At the end of the lab, your app should be able to list, add, edit, and delete.

If you have questions or are running into a problem, we'll be walking around helping you individually.