

Angular Advanced - Firebase – CRUD Operations



Peter Kassenaar –
info@kassenaar.com



CRUD operations in Firebase

- Options via AngularFire API – this is for `FirestoreListObservable` (!)
 - `push` – add new item
 - `update` – update item, use `$key` to indicate *which* item
 - `remove` – delete item. Also use `$key`.

API Summary

The table below highlights some of the common methods on the `FirestoreListObservable`.

method	
<code>push(value: any)</code>	Creates a new record on the list, using the Realtime Database's push-ids.
<code>update(keyRefOrSnap: string)</code>	Firebase
<code>remove(key: string?)</code>	Deletes the item by key. If no parameter is provided, the entire list will be deleted.

<https://github.com/angular/angularfire2/blob/master/docs/3-retrieving-data-as-lists.md>

Choice – add routing

- We add routing, so we can use Authentication and Guards later.
- Add `Routes`, `RouterModule` and routing table to application
- Add different component for different actions.
- Move logic and UI from `AppComponent` to `HomeComponent`.
- Add `<router-outlet>` and navigation menu to `AppComponent`.

```
<a class="btn btn-default" routerLink="home">
  <i class="fa fa-home"></i>
  Home
</a>
<a class="btn btn-default" routerLink="add">
  <i class="fa fa-plus"></i>
  Add City
</a>
<router-outlet></router-outlet>
```

Edit app.module.ts

```
const routes: Routes = [  
  {path: '', pathMatch: 'full', redirectTo: 'home'},  
  {path: 'home', component: HomeComponent},  
  {path: 'add', component: AddComponent},  
];
```

Add Routes

```
@NgModule({  
  imports      : [  
    ...  
    RouterModule.forRoot(routes)  
  ],  
  declarations: [  
    AppComponent,  
    HomeComponent,  
    AddComponent  
  ],  
  providers    : [],  
  bootstrap    : [AppComponent]  
})  
export class AppModule {  
}
```

Add RouterModule

Add Components

Create CityService

- Create a Service, responsible for handling CRUD-operations.
- We *could* also create a Store, but leave that as an exercise to the reader ;-)
- Add `city.service.ts`, and CRUD-skeleton
- We also added a `City` interface as a Model

```
// city.model.ts
export interface City {
  id: number;
  name: string;
  province: string;
  highlights: string[],
  rating: number
}
```

CityService

```
@Injectable()
export class CityService {

  constructor(private af: AngularFireDatabase) {
  }

  getCities(): FirebaseListObservable<City[]> {
    return this.af.list('/cities');
  }

  addCity(city: City): void {
    //...
  }

  updateCity(city: City): void {
    //...
  }

  removeCity(city: City): void {
    //...
  }
}
```



TODO: fill this
methods

HomeComponent

- Retrieve Cities via service

```
export class HomeComponent implements OnInit {  
  cities$: FirebaseListObservable<City[]>;  
  
  constructor(private cityService:CityService) {  
  
  }  
  
  ngOnInit() {  
    this.cities$ = this.cityService.getCities();  
  }  
}
```

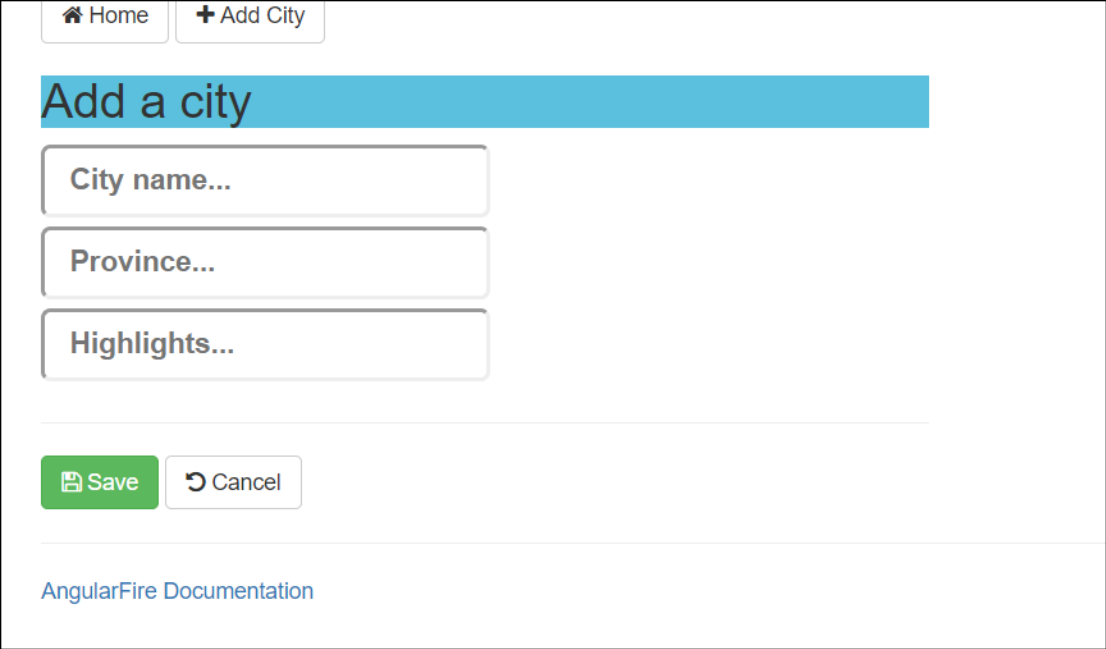
Inject CityService and
set up Observable

Inject CityService and
set up Observable

Adding a City

<!--Simple form to add a city. NO VALIDATION AT THIS MOMENT!-->

```
<input type="text" #cityName class="input-lg"
      placeholder="City name...">
<input type="text" #cityProvince class="input-lg"
      placeholder="Province...">
<input type="text" #cityHighlights class="input-lg"
      placeholder="Highlights...">
<button class="btn btn-success"
      (click)="addCity(cityName.value, cityProvince.value, cityHighlights.value)">
  Save
</button>
```



The screenshot shows a web application interface for adding a city. At the top, there are two buttons: 'Home' with a house icon and 'Add City' with a plus icon. Below these is a blue header bar with the text 'Add a city'. Under the header, there are three stacked text input fields with placeholders 'City name...', 'Province...', and 'Highlights...'. At the bottom of the form, there are two buttons: a green 'Save' button with a floppy disk icon and a white 'Cancel' button with a circular arrow icon. The footer of the application shows the text 'AngularFire Documentation'.

Add logic

```
addCity(name: string, province: string, highlights: string): void {  
    let newCity: City = {  
        id          : this.cityService.getRandomId(),  
        name        : name,  
        province    : province,  
        highlights: [],  
        rating      : 0  
    };  
    this.cityService.addCity(newCity);  
    this.router.navigate(['/']);  
}
```

Component

```
addCity(city: City): any {  
    return this.af.list('/cities')  
        .push(city);  
}
```

Service

Deleting cities

- Pass `$key` for unique reference
 - Remember to update your Model with property `$key`.

```
<li class="list-group-item"
  *ngFor="let city of cities$ | async ">
  {{ city.id }} - {{ city.name }}
  <button (click)="remove(city)" class="btn btn-sm btn-danger pull-right">
    <i class="fa fa-remove"></i>
  </button>
</li>
```

HTML

```
removeCity(city: City): void {
  this.af.list('/cities/' + city.$key)
    .remove();
}
```

Service

[Home](#) [+ Add City](#)

List of Cities from Firebase

1 - Groningen

2 - Hengelo

3 - Den Haag

4 - Enschede

5 - Heerlen

[AngularFire Documentation](#)

[Home](#) [+ Add City](#)

Add a city

Amsterdam

NH

Rijksmuseum

Save

Cancel

[AngularFire Documentation](#)

Workshop

- Add Routing to your project, create a `/home` route and `/add` route
- Add Component to add a new city
- Add Service and logic to Add and Delete cities in firebase
- Example: `/510-Firebase-crud`

