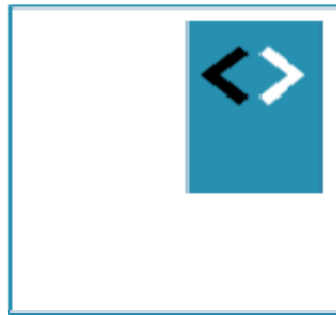


Angular Advanced @ngrx/store – Using Http Directly



Peter Kassenaar –
info@kassenaar.com

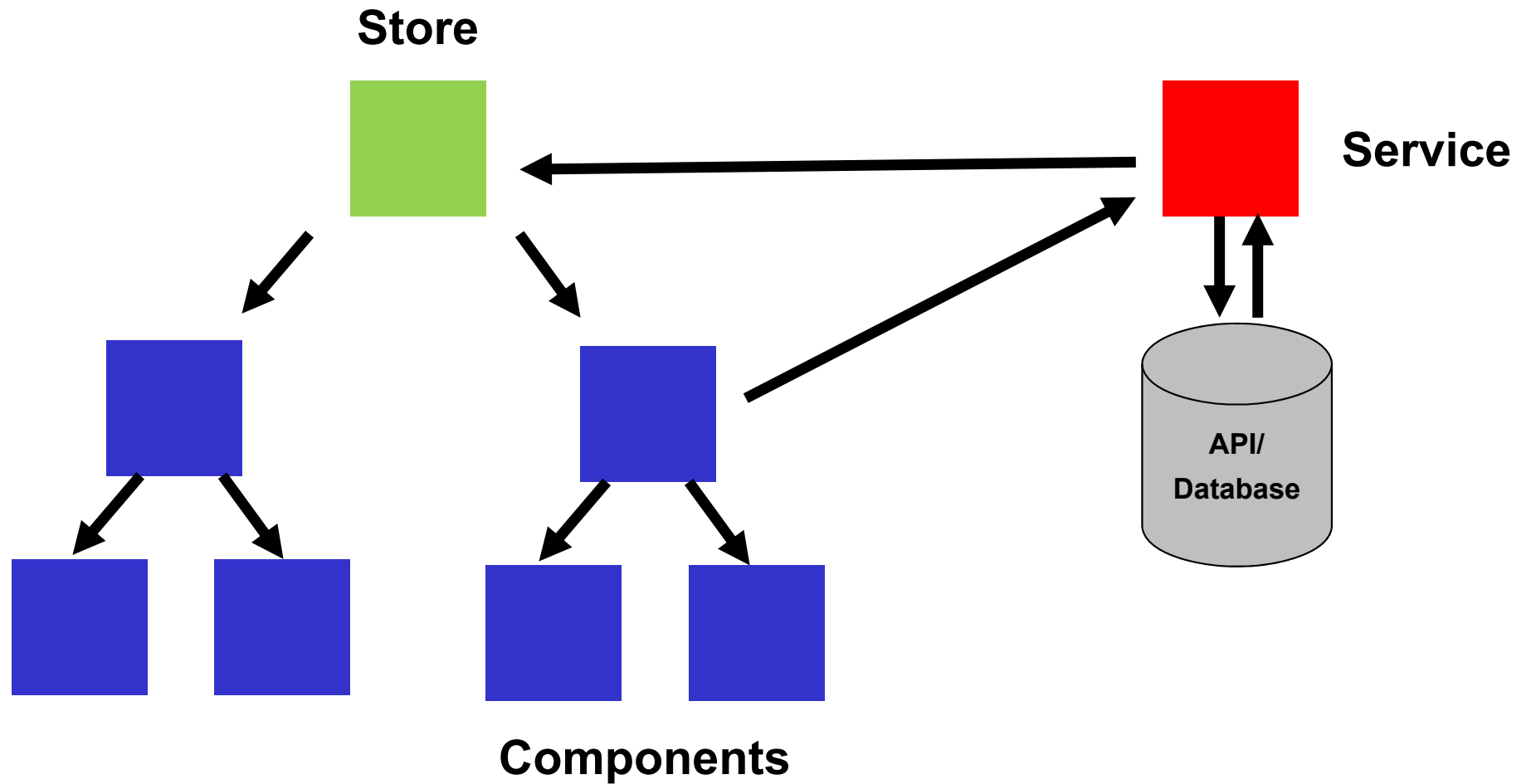


Using http directly

Talking RESTful to real API's – plain and simple!

Architecture

Call API in Service, dispatch to Store, subscribe in Components

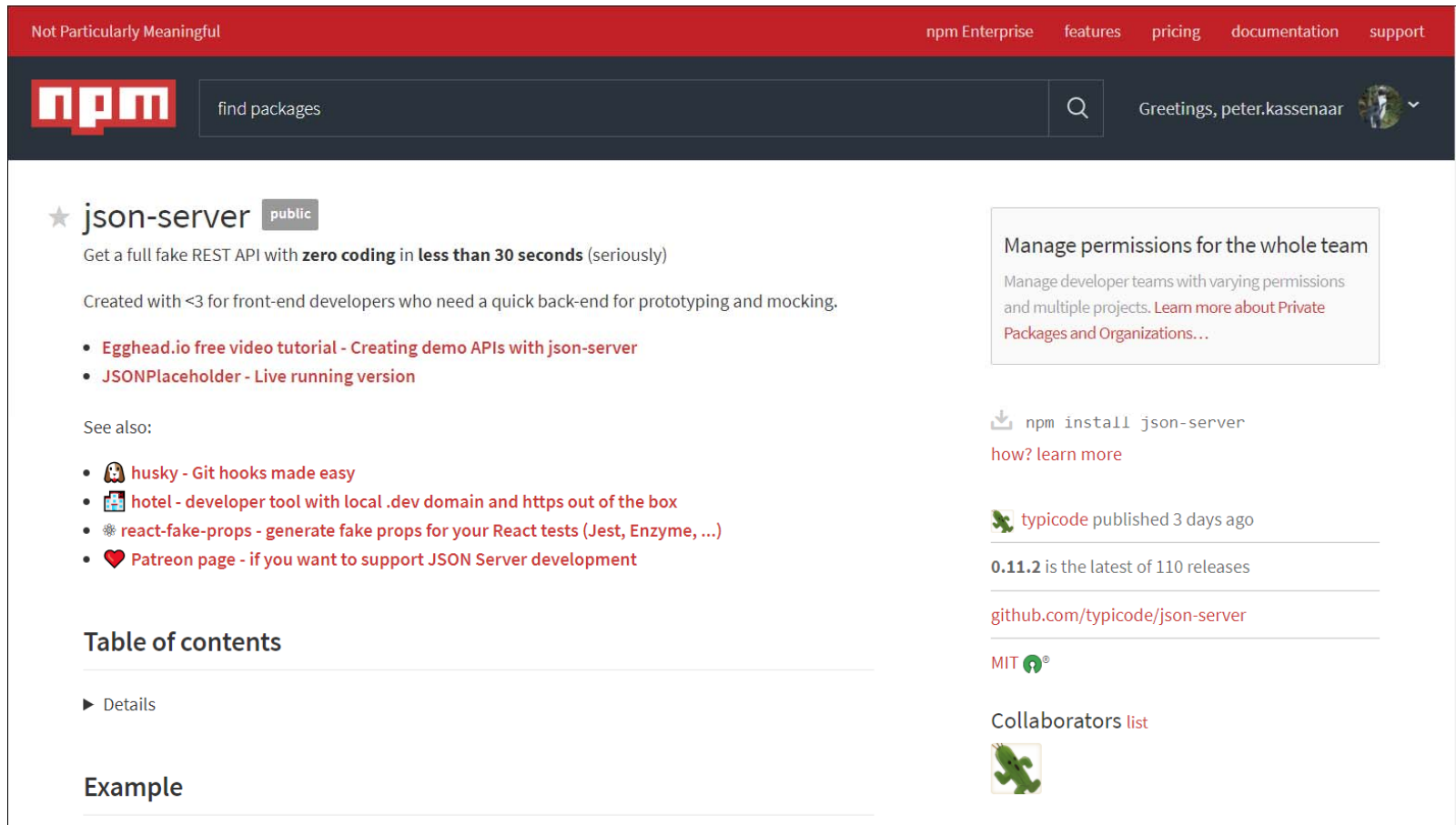


Actions and Reducers

- No changes on Actions, Action Creators and reducers.
- Add a service (if you haven't done so already) that talks to the outside world
- When a result comes back, dispatch the result to the store.

First – add a server

- We're using json-server here
- Provides a simple RESTful API, based on .json-file in webroot



The screenshot shows the npm website interface for the 'json-server' package. The header is red with the npm logo and navigation links. The main content area is white and features the package name 'json-server' with a 'public' badge. Below the name, there's a description: 'Get a full fake REST API with zero coding in less than 30 seconds (seriously)'. It also mentions it was created with <3 for front-end developers. A list of links includes 'Egghead.io free video tutorial - Creating demo APIs with json-server' and 'JSONPlaceholder - Live running version'. A 'See also' section lists other related packages like 'husky', 'hotel', 'react-fake-props', and a 'Patreon page'. On the right, there's a sidebar with a 'Manage permissions for the whole team' section, a command 'npm install json-server', and information about the latest release '0.11.2' published 3 days ago. The bottom of the page shows a 'Table of contents' with links to 'Details' and 'Example'.

Not Particularly Meaningful

npm Enterprise features pricing documentation support

npm find packages

Greetings, peter.kassenaar

★ json-server public

Get a full fake REST API with **zero coding** in **less than 30 seconds** (seriously)

Created with <3 for front-end developers who need a quick back-end for prototyping and mocking.

- [Egghead.io free video tutorial - Creating demo APIs with json-server](#)
- [JSONPlaceholder - Live running version](#)

See also:

- [husky - Git hooks made easy](#)
- [hotel - developer tool with local .dev domain and https out of the box](#)
- [react-fake-props - generate fake props for your React tests \(Jest, Enzyme, ...\)](#)
- [Patreon page - if you want to support JSON Server development](#)

Table of contents

- Details

Example

Manage permissions for the whole team

Manage developer teams with varying permissions and multiple projects. [Learn more about Private Packages and Organizations...](#)

npm install json-server

[how?](#) [learn more](#)

[typicode](#) published 3 days ago

0.11.2 is the latest of 110 releases

[github.com/typicode/json-server](#)

MIT

Collaborators [list](#)

[typicode](#)

<https://www.npmjs.com/package/json-server>

Add HttpClientModule to application

- Update `app.module.ts` and `city.service.ts`
- Since we're using services, the HTML and Component are unaltered
- Angular 5+ : use `HttpClientModule`, of course

```
import {http[Client]Module} from '@angular/[common]/http';  
...  
  
@NgModule({  
  ...  
  imports      : [  
    http[Client]Module,  
  ],  
  ...  
})
```

Edit city.service.ts

Add `Http` and call API in `loadCities()`.

Upon subscription, dispatch data to the store

```
const BASE_URL = 'http://localhost:3000/cities';
const HEADERS = {headers: new HttpHeaders().set('Content-Type', 'application/json')};

@Injectable()
export class CityService {

  constructor(private store: Store<AppState>,
               private http: HttpClient) {
    this.loadCities();
  }

  loadCities() {
    return this.http.get(BASE_URL).pipe(
      tap(res => console.log('just received', res))
    ).subscribe(cities => this.store.dispatch(new fromActions.LoadCities(cities)),
      err => console.log('Error: start json-server '),
      () => console.log('Getting cities complete'))
  }
}
```

Adding and deleting cities

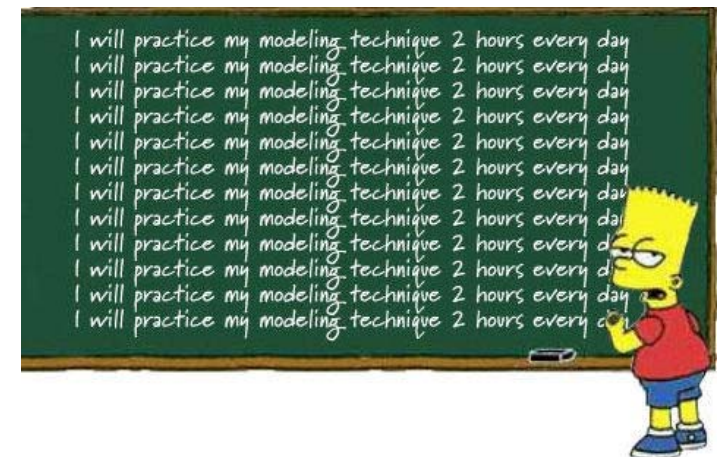
- Same procedure...

```
// add a city to the store
addCity(city: City): void {
    this.http.post(BASE_URL, JSON.stringify(city), HEADERS)
        .map(res => res.json())
        .subscribe(payload =>
            this.store.dispatch(new fromActions.AddCity(payload)));
}
```

```
// Remove city from store
removeCity(city: City) {
    this.http.delete(`${BASE_URL}/${city.id}`)
        .subscribe(action =>
            this.store.dispatch(new fromActions.RemoveCity(city)));
}
```


Workshop

- Use your own app, add a service and call HTTP to load .json-data
- OR: Start from `../220-ngrx-store-http`
- Make yourself familiar with the store concepts and http-flow. Study the example code.
- Add the `addCity()` method on the service, that adds a city to the .json file via json-server
- Add the `editCity()` method on the service, to edit an existing city



Next Steps

- [@ngrx/effects](#) - Side Effect model for @ngrx/store to model event sources as actions.
- [@ngrx/router-store](#) - Bindings to connect the Angular Router to @ngrx/store
- [@ngrx/store-devtools](#) - Store instrumentation that enables a powerful time-travelling debugger
- [@ngrx/entity](#) - Entity State adapter for managing record collections.

<https://github.com/ngrx/platform>