Angular 2
Part 2

www.nodesen.se
Gopalakrishnan Subramani
gs@nodesen.se
+91 9886991146
gopalakrishnan.subramani@gmail.com

https://github.com/nodesense/ngapp

Agenda

- Router
- HTTP

Router

- Routing
- Navigation

```
import {RouterModule} from "@angular/router";

@NgModule( {
 imports: [
      BrowserModule,
      RouterModule,
      appRouting
 ],
```

Routing Strategies

- HashLocationStrategy (“#/”)
- PathLocationStrategy (HTML5 Mode)

Hash#

```
import {LocationStrategy, HashLocationStrategy}
from "@angular/common";


 @NgModule()({
providers: [
      {
          provide: LocationStrategy,
          useClass: HashLocationStrategy
      }
]
..
```

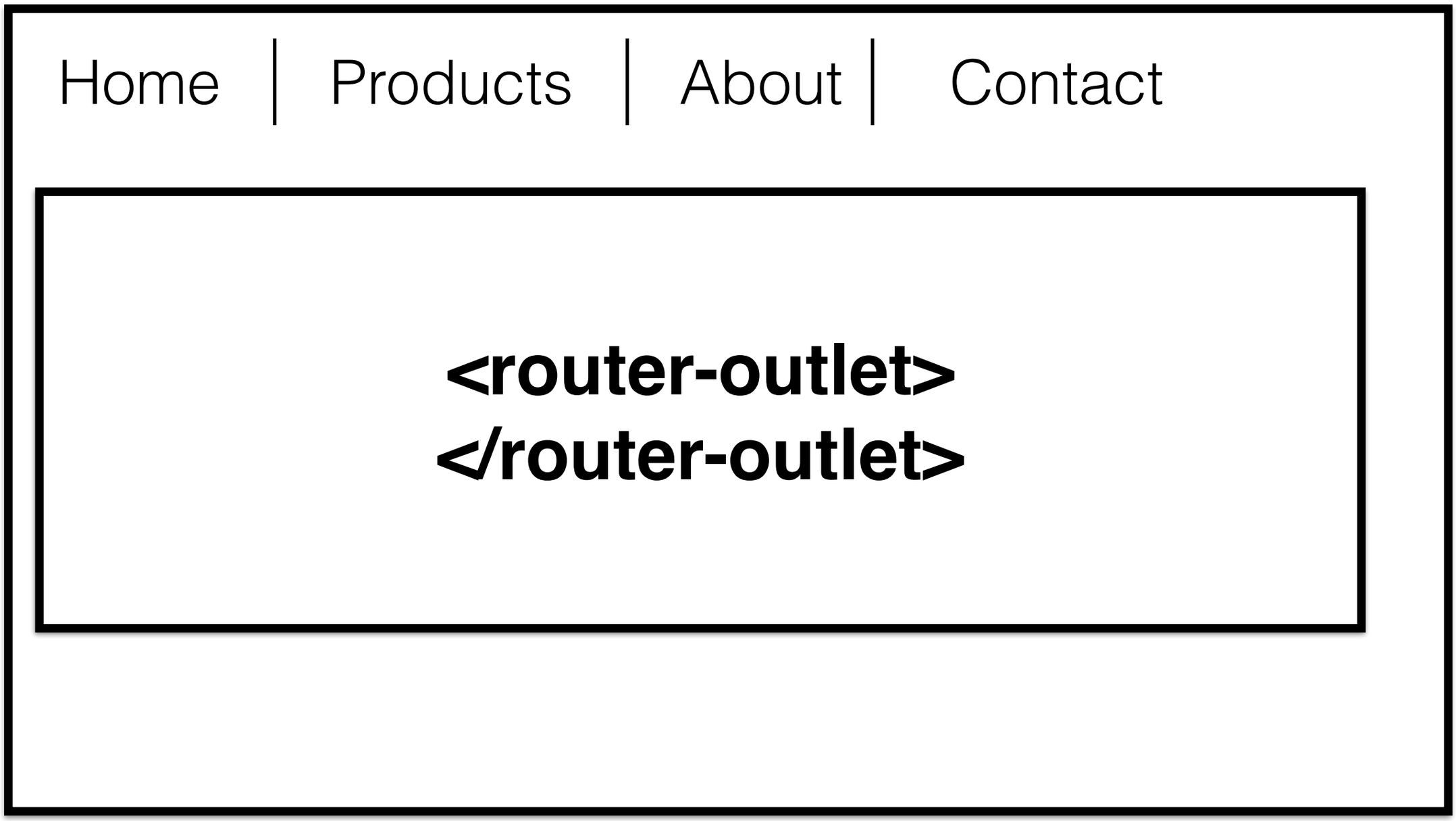
HTML5 Mode

- Enabled By Default
- Must Set **<base href="/" >**

```
import {LocationStrategy, PathLocationStrategy}
from "@angular/common";

 providers: [
      {
          provide: LocationStrategy,
          useClass: PathLocationStrategy
      }
```

RouterLink & RouterOutlet




```
<a [routerLink]="['products']">Products</a>
```

```
router.navigate(['products'])
```

Navigation

By Code

```
import {Router} from "@angular/router";

export class ProductEditComponent {
    constructor(private router: Router) {
    }

    goToList() {
      this.router.navigate([‘products’])
    }
}
```

By HTML

```
<a [routerLink]="['products']">Products</a>
```

Nested Navigation

Home | **Products** | About | Contact

List | Create | Search

<router-outlet>

</router-outlet>

Nested Navigation

```
export const routes:Routes = [
        {
            path: ‘products’,
            component: ProductHomeComponent,

            children: [
                {
                    path: ‘list’, // products/list
                    component: ProductListComponent
                },
                {
                    path: ‘edit/:id', //brands/edit/123
                    component: ProductEditComponent
                }
            ]
        }
    ]
```

Routes Configuration

```
import {Routes} from "@angular/router";
import {ProductListComponent} from "./product-list.component"
import {ProductEditComponent} from "./product-edit.component"

export const routes:Routes = [
    {
        path: ‘products/list',
        component: ProductListComponent
    },

    {
        path: 'products/edit/:id',
        component: ProductEditComponent
    }
]
```

Path is the URL on Address Bar

:id is a dynamic variable

Query Parameters

- We can pass route parameters through routerLink and router
- URL Parameter
- Route Parameter Object

```
export const routes:Routes = [
    {
        path: 'products/edit/:id',
        component: ProductEditComponent
    }
]
```

Route Data Handling

example.com/product/edit/12/apple

- ActivatedRoute

```
{
    path: ‘products/edit/:id/:name’,
    component: ProductEditComponent
}
```

```
import {ActivatedRoute, Params} from "@angular/router";

ngOnInit() {
        this.route.params.forEach((params: Params) => {
            let id:number = params["id"];
            let name:string = params["name"];
        });
    }
```

Matrix URL

- Matrix URL is not part of HTML, but part of HTTP URL Standard, can be used in browsers

localhost:3000/order/checkout;coupon=“save50”;expire=“60s”

```
{
        path: ‘cart/checkout’, component: …
}

this.router.navigate(["/cart/checkout",
        {'coupon': couponCode, ‘expire': “60s”  }]);
```

```
import {ActivatedRoute} from "@angular/router";

discount = route.snapshot.params["coupon"];
coupon =   route.snapshot.params["expire"];
```

CanActivate

- CanActivate is called **before** creating Component Object
- When **CanActivate Return false**, Component **shall not be loaded**, routes shall not change
- Good for Authentication and Authorization Handling

```
import { ActivatedRouteSnapshot, RouterStateSnapshot,
CanActivate} from '@angular/router';
```

CanActivate

```
@Injectable()
export class AdminGuard implements CanActivate {
    constructor(private router: Router,
                private authService: AuthService) {
    }
   canActivate(route: ActivatedRouteSnapshot,
               state: RouterStateSnapshot) {
        if (this.authService.isAdmin()) {
            return true;
        }
        return false;
}
```

```
{   Route
    path: 'edit/:id',
    component: ProductEditComponent,
    CanActivate: [AdminGuard]
}
```

CanDeactivate

- Called before leaving currently loaded Route, Component
- Useful for reminding ‘Work is not saved’ alerts

```
import {CanDeactivate} from "@angular/router";
```

CanDeactivate

```
export class SaveAlertGuard implements
CanDeactivate<BrandEditComponent> {

  canDeactivate(target: CheckoutComponent) :any {
     if(!target.isWorkSaved()){
         return window.confirm('Exit without saving?');
     }
     return true;
  }
}
```

```
{    Route
   path: 'edit/:id',
   component: ProductEditComponent,
   CanDeactivate: [SaveAlertGuard]
}
```

HTTP

HttpModule, AJAX, REST

HttpModule

- Ajax implementation
- Have support for get, post, put, delete methods
- Implementation is from HttpModule, must be imported

Import Module

```
import {HttpModule} from "@angular/http";
@NgModule({
    imports: [
        ...
        HttpModule
        ...
    ]
})
export class AppModule {

}
```

Inject Http

```
import {Injectable, Inject} from "@angular/core";
import {Http} from "@angular/http";

@Injectable()
export class ProductService {
      apiEndPoint: string = "http://localhost:7070";
      //Inject HTTP
      constructor(private http: Http) {
      }

      getProducts() {
            return this.http.get(this.apiEndPoint +
                                 "/api/products")
      }
}
```

REST

- Representational State Transfer (REST)
- For RESTful Web Services
- Uses HTTP Verb for CRUD/Web Services Implementation

GET Method

- Useful to fetch resource information from server
- Get list of resources
- Get a specific resource

Get List of products

GET /api/products

GET /api/products?q=htc

Get a Specific Product

GET /api/products/1

GET

```
@Injectable()
export class ProductService {

    //GET /api/products
    getProducts()  {
        return this.http.get(this.apiEndPoint +
                                "/api/products")
    }

    //GET /api/products/1
    getProduct(id :any)  {
        return this.http.get(this.apiEndPoint +
                                "/api/products/" + id)
    }
}
```

POST

- Useful to create new resource in the server
- Client may not know the unique id
- Server should respond with unique id of resource or send complete resource object with id after commit to DB
- POST /api/products

POST

```
//POST /api/products
//Content-Type: application/json
//{{data}}
saveProduct(product: any) {
    let headers: Headers = new Headers({
        "Content-Type": "application/json"
    })

   let requestOptions = new RequestOptions({
        'headers': headers
        })

        let jsonDataText = JSON.stringify(product);
        //POST /api/products
        return this.http.post(this.apiEndPoint
                               + "/api/products",
                              jsonDataText,
                              requestOptions)
    }
}
```

PUT

- For updating existing resource
- Client has id of the resource
- PUT /api/products/100
- Recommended to server to return complete resource after update or send status code 200 after successful commit

PUT

```
//PUT /api/products/1
//Content-Type: application/json
//{{data}}
saveProduct(product: any) {
    let headers: Headers = new Headers({
        "Content-Type": "application/json"
    })
    let requestOptions = new RequestOptions({
        'headers': headers
    })
    let jsonDataText = JSON.stringify(product);
    //PUT /api/products/1
        return this.http.put(this.apiEndPoint +
                "/api/products/" + product.id,
            jsonDataText,
            requestOptions
        )
}
```

DELETE

- Delete an existing resource from web service
- Client knows the ID

```
//DELETE /api/products/1

deleteProduct(id: any) {
        return this.http.delete(this.apiEndPoint
                                + "/api/products/" + id)
}
```

Handling JSON Data

- Angular shall not automatically parse JSON response

```
//GET /api/products/1
getProduct(id :any) : Promise<any> {
   return this.http.get(this.apiEndPoint +
                                "/api/products/" + id)
   .map( (response: Response) => response.json())
}
```