Angular Routing

Agenda

- Routing Basics
- Parameterized Routes
- Child Routes
- Route Gards
- Module Lazy Loading

Routing Basics

Routing

- Routing is the process of navigation from one "page" (view) to another
- Routing can be initated by:
  - Entering an URL
  - Using the browser Back Button or History
  - Clicking a Link
  - Programatically
- Routes are declared in app.module.ts by default

Angular Router

- Angular Router enables navigation from one view to the next
- Depends on RouterModule, Routes from @angular/router
- Has one singleton instance of the Router service

```
@NgModule({
declarations: [
AppComponent,
HomeComponent,
…
],
imports: [
BrowserModule,
RouterModule.forRoot(appRoutes, { enableTracing: false})
],
```

```
import { RouterModule, Routes } from '@angular/router';
import { HomeComponent } from './home/home.component';
import { VouchersComponent } from './vouchers/vouchers.component';
…
const appRoutes: Routes = [
{ path: '',
component: HomeComponent
},
{ path: 'vouchers',
component: VouchersComponent
},
{ path: 'voucher/:id',
component: VoucherComponent
},
```

Redirects & Wildcards

- Allows to redirect from one route to another (catch misspelled route)
- Requires a pathMatch property to tell the router how to match a URL
- ** corresponds to a wildcard route

```
const appRoutes: Routes = [
{ path: '',
component: HomeComponent
},
{ path: 'vouchers',
component: VouchersComponent
},
{
path: 'wotschers',
redirectTo: 'vouchers',
pathMatch: 'full'
},
{ path: '**',
component: PageNotFoundComponent
}
];
```

Router Link

- Replacement for HTML anchor tags using Angular Route component
- Support event model (NavigationStart, NavigationEnd, ….)

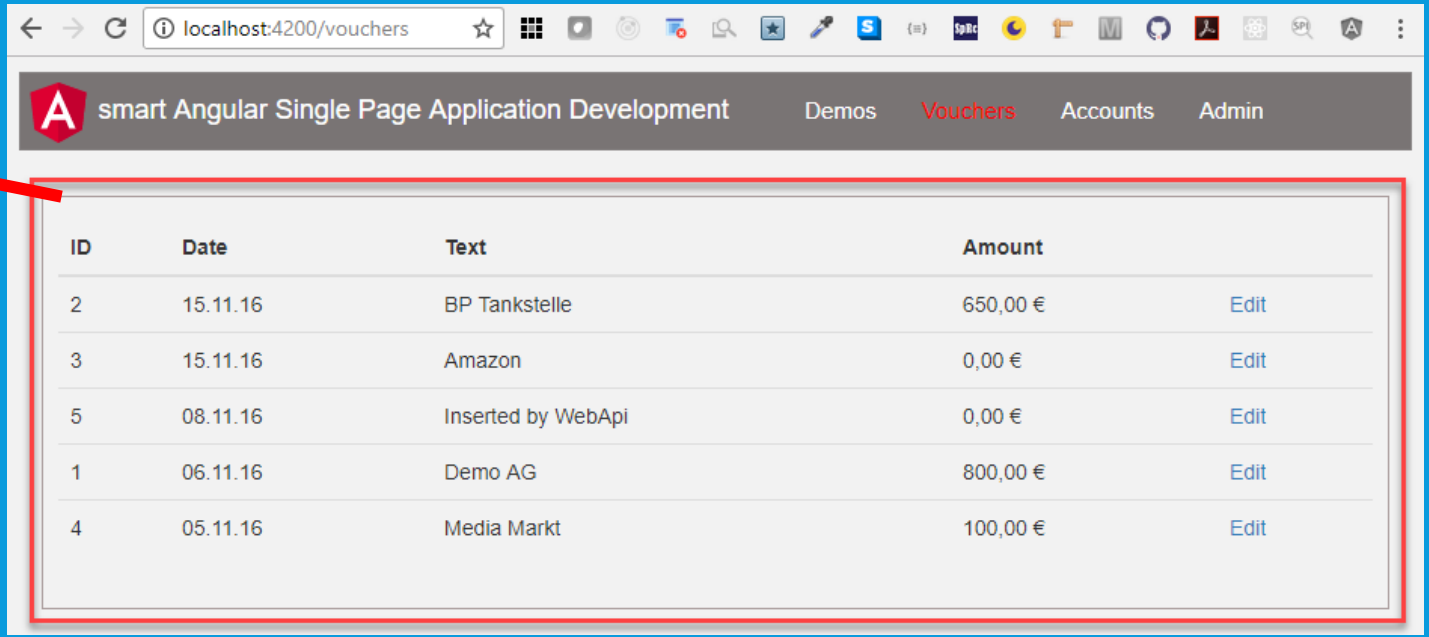
```
<nav>
    <a routerLink="/" routerLinkActive="active">Home</a>
    <a routerLink="/vouchers" routerLinkActive="active">Vouchers</a>
    <a routerLink="/assets" routerLinkActive="active">Assets</a>
</nav>
```

Router Outlet

- Provides a container where the current view (Component) of the router is injected

```
<div style="border: 1px solid">
<router-outlet>

</router-outlet>
</div>
```

```
const appRoutes: Routes = [
{ path: '',
component: HomeComponent
},
{ path: 'vouchers',
component: VouchersComponent
},
{ path: 'voucher/:id',
component: EditVoucherComponent
},
```



Named Router Outlet

- Named Router Outlets enable us to specify the target outlet of a rout (if more than one outlet exist)
- Make use of the "name" attr of the outlet and use "outlet" in route const

```
{ path: 'persons',
  component: PersonsComponent,
  outlet: 'persons'
},
```

```
<div>
<router-outlet name='persons'></router-outlet>
</div>
```

Programtic Routing

- Archieved by injecting the Router into the Component
- Use the navigate or navigateByUrl method
- Can use NavigationExtras for granular configuration

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

…
export class HomeComponent implements OnInit {

constructor(private router: Router) { }

onShowAssets() {
  this.router.navigate(['/assets'])
 }
}
```

home.component.html

```
1  <p>
2    home works!
3  </p>
4
5  <button (click)="onShowAssets()">Show Assets</button>
```

Parameterized Routes

Parameterized Routes

- The colon (:) in the path indicates that :id is a placeholder for a specific id
- Import ActivatedRoute from @angular/router
- ActiveRoute.snapshot.params is used to gain access to param in detail component

```
<tbody>
<tr *ngFor="let v of vouchers" (click)="showVoucher(v.ID)">
<td>{{v.ID}}</td>
<td>{{v.Text}}</td>
<td><a [routerLink]="[ '/voucher', v.ID ]">Edit</a></td>
</tr>
```

```
export class EditVoucherComponent implements OnInit {
voucher: MinVoucher = new MinVoucher();
constructor(private route: ActivatedRoute) { }
ngOnInit() {
this.voucher.ID = this.route.snapshot.params['id'];
}
}
```

Query Params & Fragments

- Possible to pass Query Params and Fragments (#) with a link
- Existing Params can be preserved or merged using
  - queryParamsHandling merge | preserve

```
<a [routerLink]="['/voucher', 2]" [queryParams]="{readonly: true}" fragment="stophere" >Show Readonly Voucher by id</a>
```

```
ngOnInit() {
this.voucher.ID = this.route.snapshot.params['id'];
var qryParam = this.route.snapshot.queryParams;
var fragment = this.route.snapshot.fragment;
}
```

Outsourced Routes

- Good advice to keep routes in its own module
- Steps:
  - Create a RouteModule i.e. app-routing.module.ts
  - Move Route constants there
  - Import & export routes to / from RouterModule
  - Register in app.module.ts

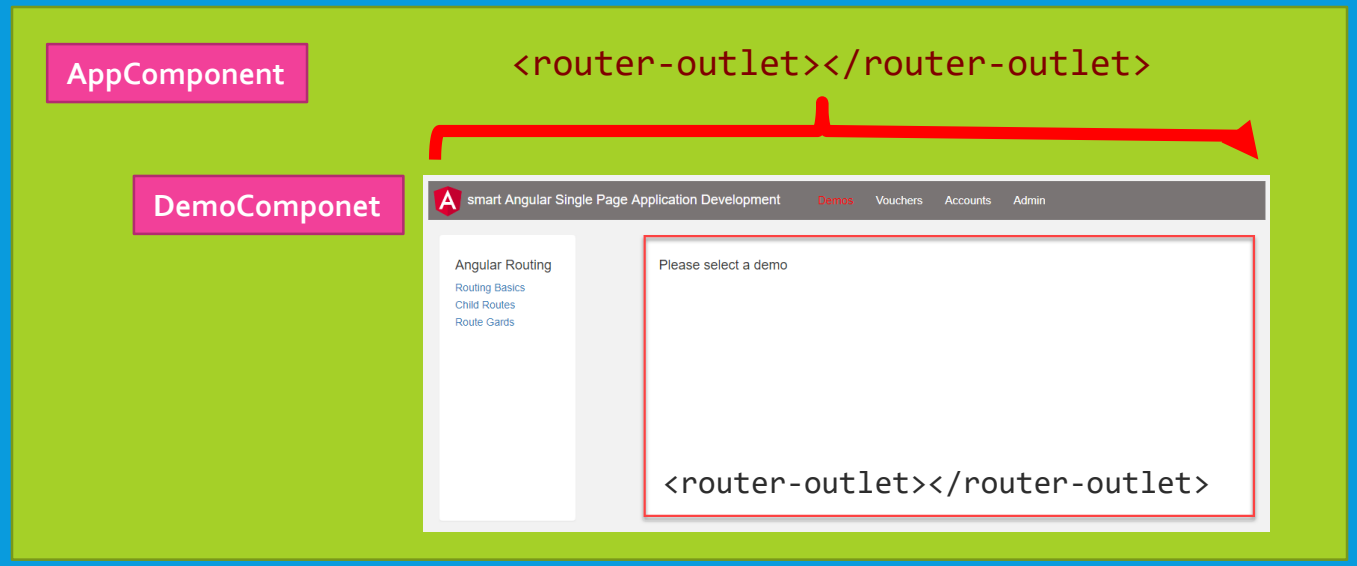
```
@NgModule({
imports: [RouterModule.forRoot(appRoutes, { enableTracing: false})],
exports: [RouterModule]
})
export class AppRoutingModule{

}
```

Child Routes

Child Routes

- Enambles us to have "child" or "nested" routes

```
const appRoutes: Routes = [
{ path: '',
 component: HomeComponent,
 children: [
 { path: 'routingbasics', component: RoutingBasicsComponent },
 { path: 'childroutes', component: ChildRoutesComponent },
 { path: 'routegards', component: RouteGardsComponent }
]
```



Route Gards

Route Gards

- Route Gards allow execution before a Routing event takes place
- Implement custom class
- Implement one of the following Interfaces:
  - CanActivate
  - CanActivateChild
  - CanDeactivate
  - Resolve – prefetching data

Route Gards implementation

- Use ActivatedRouteSnapshot & RouterStateSnapshot

```
{
 path: 'assets',
 component: AssetsComponent,
 data: { title: 'Assets' },
 canActivate: [RouteGuard]
},
```

app.routing.module.ts

```
@Injectable()
export class RouteGuard implements CanActivate, CanActivateChild {

allow: boolean = true;
constructor(private router: Router) {}
canActivate(route: ActivatedRouteSnapshot, state: RouterStateSnapshot): Observable<boolean> | Promise<boolean> | boolean {
 if(this.allow){
  return true;
 }
 else{
  this.router.navigate(['/']);
 }
```

Module Lazy Loading

Module Lazy Loading

- Lazy Loading loads Modules only when they are used
- Configured in the Router Module

```
{ path: 'demos', loadChildren: 'app/user/demo.module#DemoModule'}
```

```
@NgModule({
 imports: [
 CommonModule,
 FormsModule,
 ReactiveFormsModule,
 RouterModule.forChild(demoRoutes)
],
declarations: [
 DemoHomeComponent,
 DemoABCComponent
],
providers: []
})
export class DemoModule { }
```