



Router Deep Dive

ANGULARarchitects.io

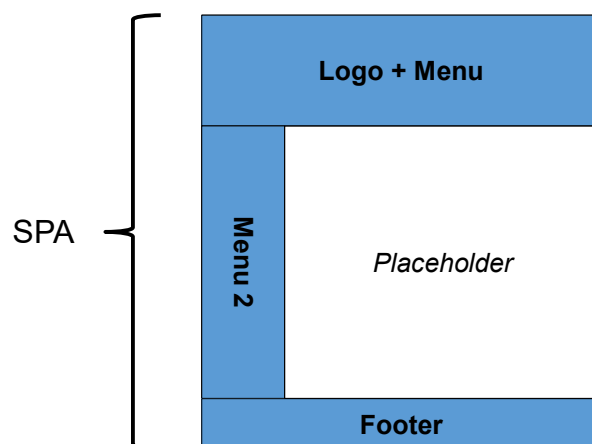
Contents

- Basics
- Child Routes
- Aux Routes
- Guards
- Resolver
- Lazy Loading

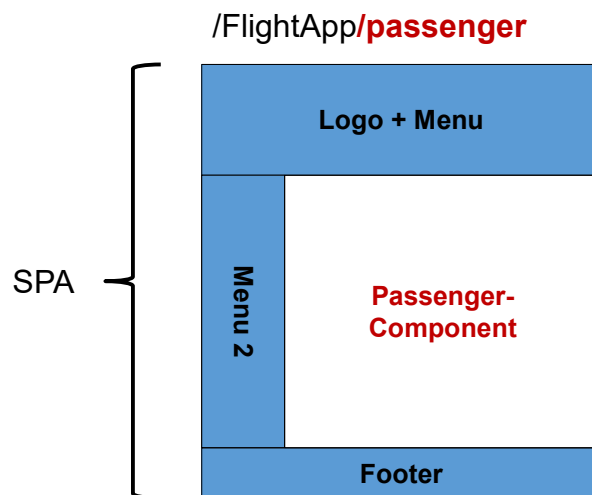


Angular Router

Routing in Angular



Routing in Angular



Configuration

```
const APP_ROUTES: Routes = [  
  {  
    path: 'home',  
    component: HomeComponent  
  },  
  {  
    path: 'flight-search',  
    component: FlightSearchComponent  
  },  
  {  
    path: '**',  
    redirectTo: 'home'  
  }  
]
```

Configuration

```
// app.module.ts
@NgModule({
  imports: [
    BrowserModule,
    HttpClientModule,
    FormsModule,
    RouterModule.forRoot(ROUTE_CONFIG)
  ],
  [...]
```

For Root-Module
For Feature-Module: forChild

AppComponent

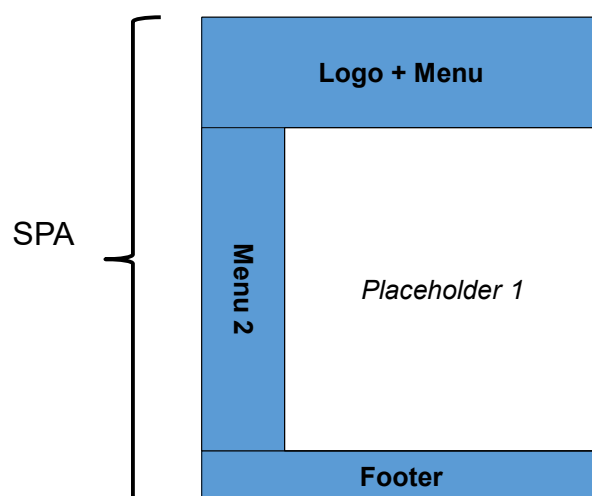
```
<a routerLink="/home">Home</a>
<a routerLink="/flight-search">Flight Search</a>

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

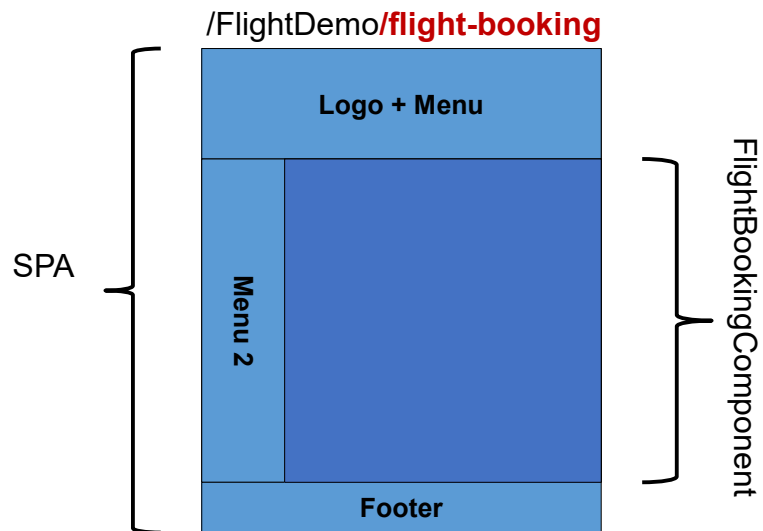


Hierarchical Routing

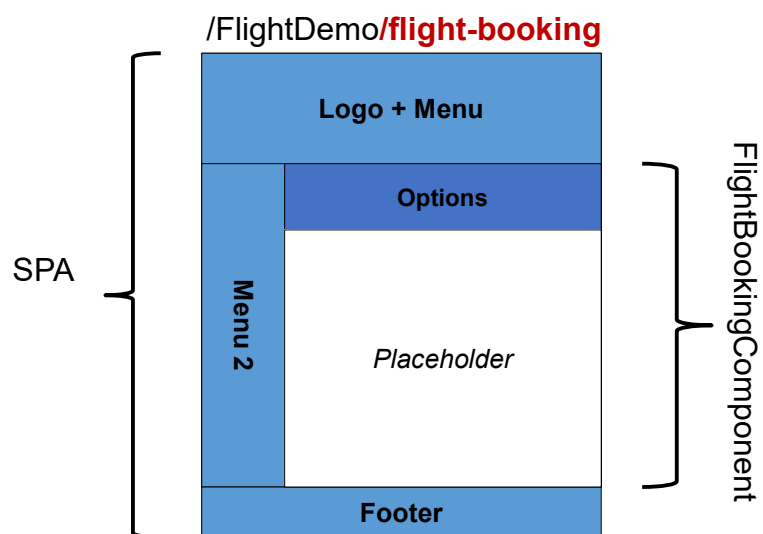
Hierarchical Routing



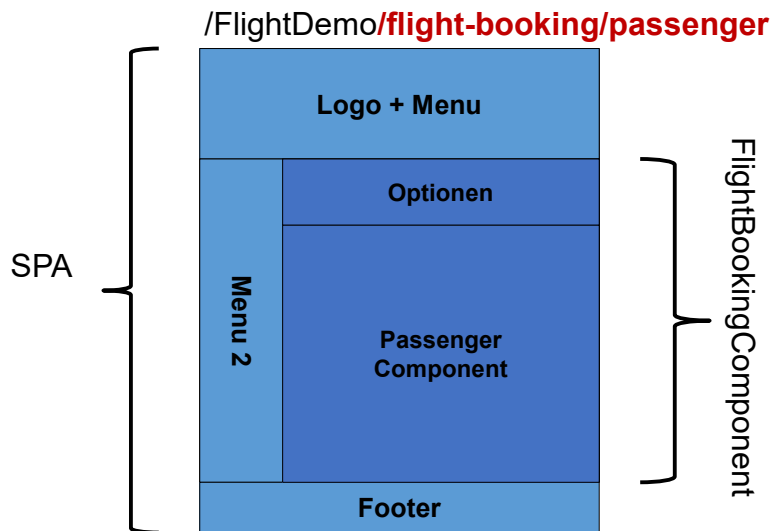
Hierarchical Routing



Hierarchical Routing



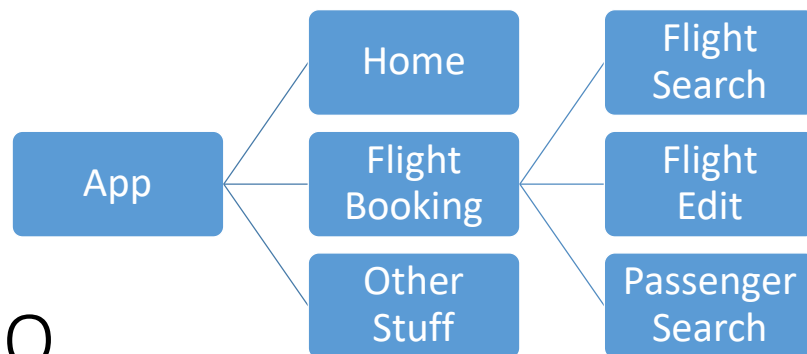
Hierarchical Routing



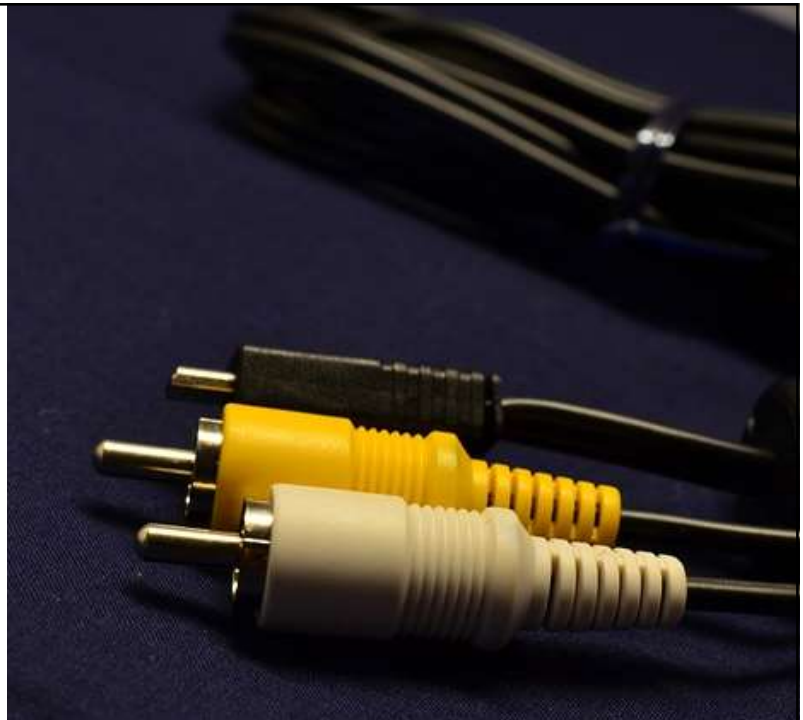
Configuration

```
const APP_ROUTES: Routes = [  
  {  
    path: '',  
    component: HomeComponent  
  },  
  {  
    path: 'flight-booking',  
    component: FlightBookingComponent,  
    children: [  
      {  
        path: 'flight-search',  
        component: FlightSearchComponent  
      },  
      [...]  
    ]  
  }  
];
```

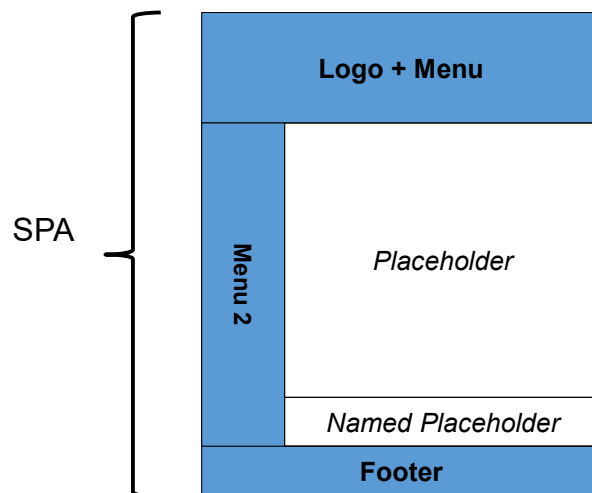
DEMO



Aux Routes

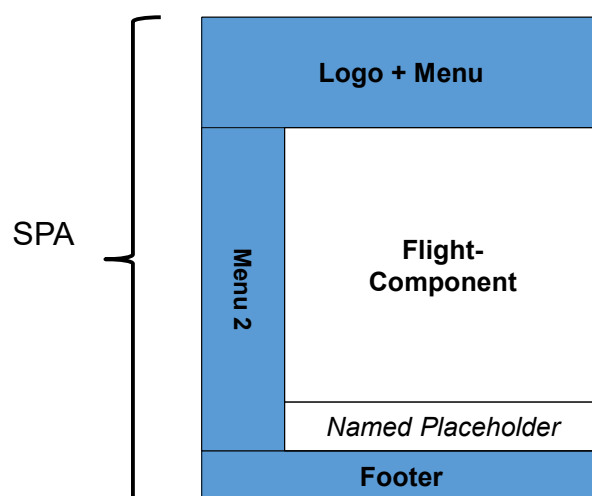


Aux-Routes



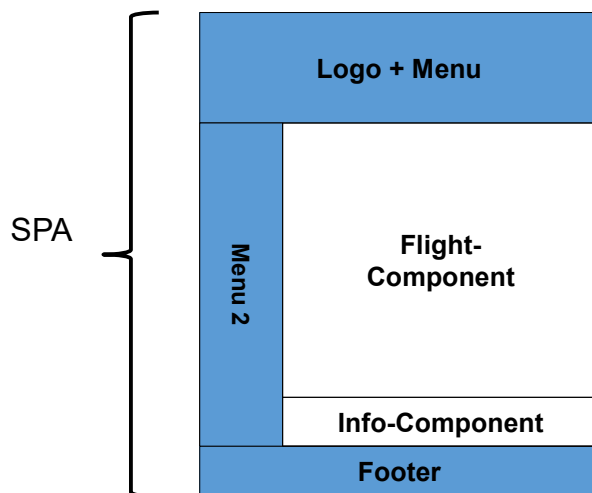
Aux-Routes

/FlightApp/**flights**



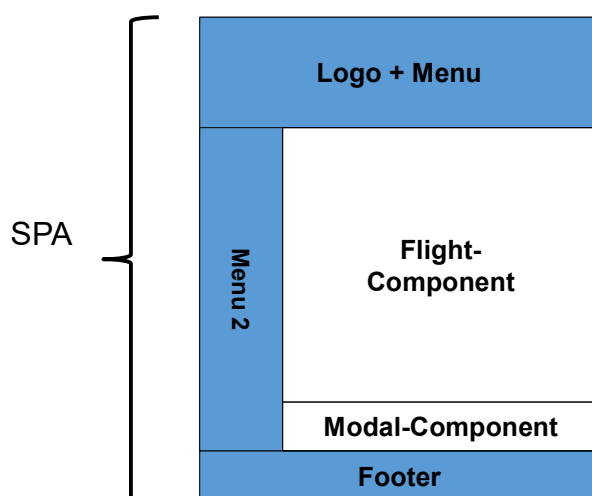
Aux-Routes

/FlightApp/**flights(aux:info)**



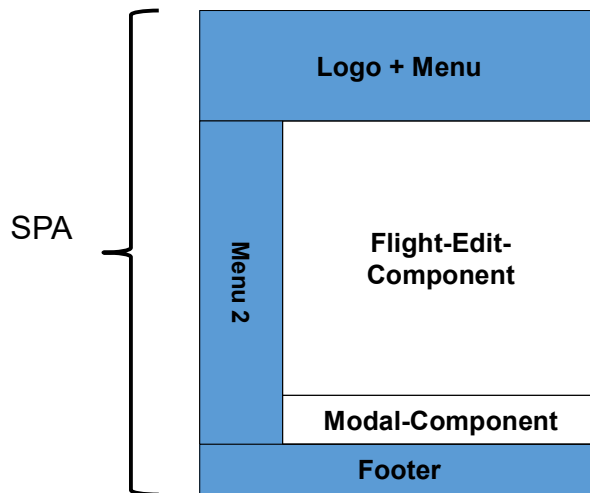
Aux-Routes

/FlightApp/**flights(aux:info/modal)**



Aux-Routes

/FlightApp/**flights(aux:info/modal)/edit/17**



Use Cases

- Partly autonomous parts of an application
- „Norton Commander Style“
- (CSS-based) Popups and Modals

Define Outlets

Default Name: primary

```
<router-outlet></router-outlet>  
  
<hr>  
  
<router-outlet name="aux"></router-outlet>
```

Configuration

```
export const ROUTE_CONFIG: Routes = [  
  {  
    path: 'home',  
    component: HomeComponent  
  },  
  {  
    path: 'info',  
    component: InfoComponent,  
    outlet: 'aux'  
  },  
  {  
    path: 'dashboard',  
    component: DashboardComponent,  
    outlet: 'aux'  
  }  
]
```

Activating Aux-Routes

```
<a [routerLink]="[{outlets: { aux: 'info' }}]">  
  Activate Info  
</a>  
  
<a [routerLink]="[{outlets: { aux: null }}]">  
  Deactivate Info  
</a>
```

Activating Several Aux Routes at Once

```
<a [routerLink]="[{outlets: {  
  aux: 'basket',  
  primary: 'flight-booking/flight-search' }}]"> ... </a>  
  
<a [routerLink]="[{outlets: { aux: 'basket',  
  primary: ['flight-booking', 'flight-search'] }}]"> ... </a>  
  
<a [routerLink]="[{outlets: { aux: 'basket',  
  primary: ['flight-booking', 'flight-edit', 17] }}]"> ... </a>
```

Code-based Routing

```
export class AppComponent {  
  constructor(private router: Router) {  
  }  
  
  activateInfo() {  
    this.router.navigate([{outlets: { aux: 'info' }}]);  
  }  
  
  deactivateInfo() {  
    this.router.navigate([{outlets: { aux: null }}]);  
  }  
}
```

DEMO

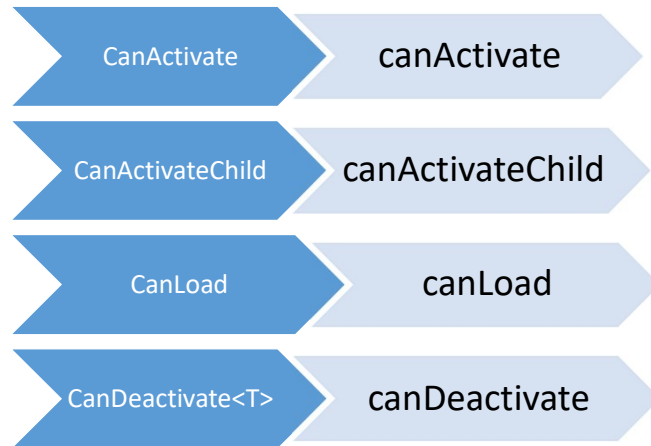


Guards

What are Guard?

- Services
- Can prevent the Activation or Deactivation of a Route

Guards



Result: boolean | Observable<boolean> | Promise<boolean>

Guards and the Router Configuration

```
const APP_ROUTES: Routes = [  
  {  
    path: '/flight-booking',  
    component: FlightBookingComponent,  
    canActivate: [AuthGuard],  
    children: [  
      {  
        path: 'flight-edit/:id',  
        component: FlightEditComponent,  
        canDeactivate: [FlightEditGuard]  
      },  
      [...]  
    ]  
  }  
]
```

Provider for Guards

```
// app.module.ts
@NgModule({
  providers: [
    FlightEditGuard,
    AuthGuard
  ],
  [...]
})
export class AppModule {
}
```

DEMO

Lab



Resolver

What are Resolver?

- Services
- Are activated when the Router switches over to another route
- Can load needed data
- Postpone activation of target route until data is loaded
- Meanwhile, a loading indicator can be shown

Resolver

```
@Injectable()
export class FlightResolver implements Resolve<Flight> {
  constructor(private flightService: FlightService) {
  }

  resolve(route, state):
    Observable<Flight> | Promise<Flight> | any {
    return [...]
  }
}
```

Register Resolver

```
const FLIGHT_BOOKING_ROUTES: Routes = [
  [...]
  {
    path: 'flight-edit/:id',
    component: FlightEditComponent,
    resolve: {
      flight: FlightResolver ←----- Token
    }
  }
];
```

Receive Data in Component

```
@Component({ ... })
export class FlightEditComponent {

  flight: Flight;

  constructor(private route: ActivatedRoute) { }

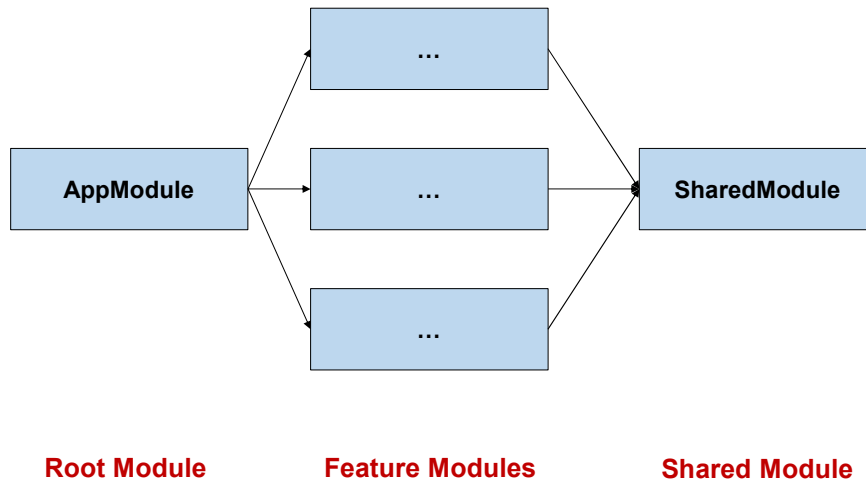
  ngOnInit() {
    this.route.data.subscribe(
      data => {
        this.flight = data['flight'];
      }
    );
  }
}
```

DEMO

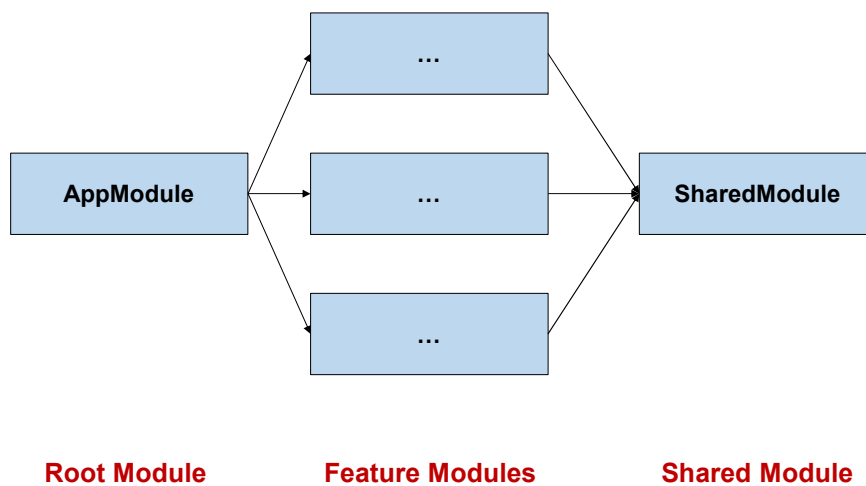
Lazy Loading



Module Structure



Lazy Loading



Root Module with Lazy Loading

```
const APP_ROUTE_CONFIG: Routes = [  
  {  
    path: 'home',  
    component: HomeComponent  
  },  
  {  
    path: 'flights',  
    loadChildren: () => import('./[...]/flight-booking.module')  
      .then(m => m.FlightBookingModule)  
  }  
];
```

Routes for Feature Module

```
const FLIGHT_ROUTES = [  
  {  
    path: '',  
    component: FlightBookingComponent,  
    [...]  
  },  
  [...]  
]
```

Routes for Feature Module

```
const FLIGHT_ROUTES = [  
  {  
    path: '/bookings',  
    component: FlightBookingComponent,  
    [...]  
  },  
  [...]  
]
```

Url: /flights/bookings

Triggers Lazy Loading

DEMO

Tree-Shakable Provider for Lazy Modules

Lazy Modules

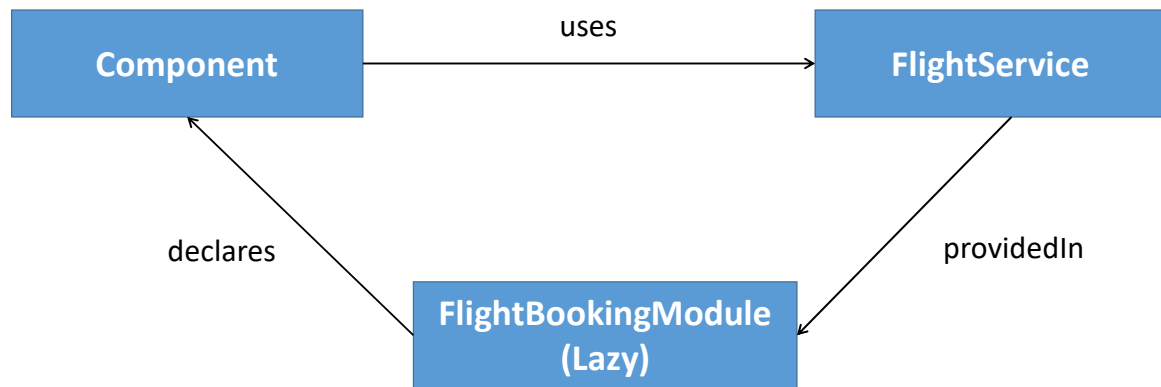
```
@Injectable({ providedIn: LazyApiModule })  
export class FlightService {  
  [...]  
}
```

Service is loaded alongside lazy module!

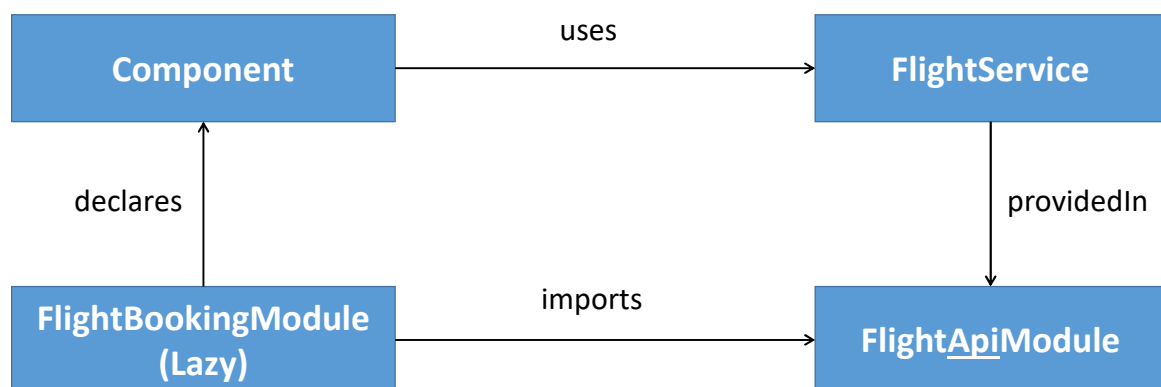
Only makes sense with lazy loading !!

All "classic" modules: root scope

Preventing Cycles



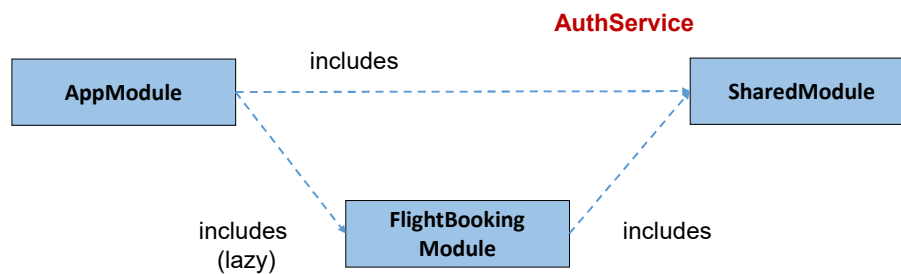
Preventing Cycles



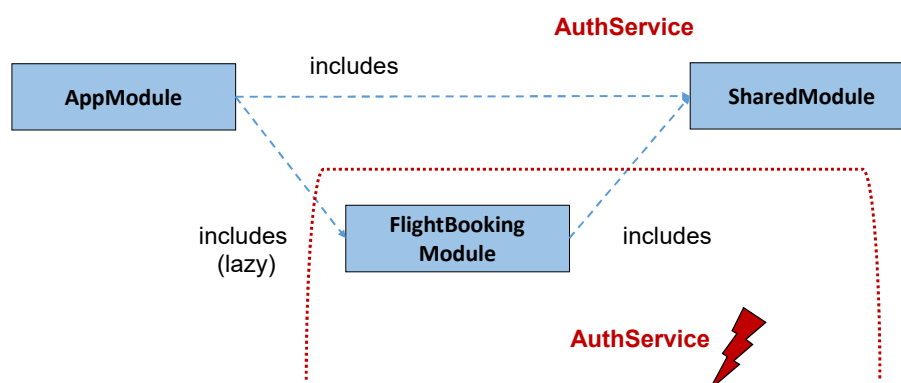
DEMO

Problem with Lazy Loading and
Classic Providers

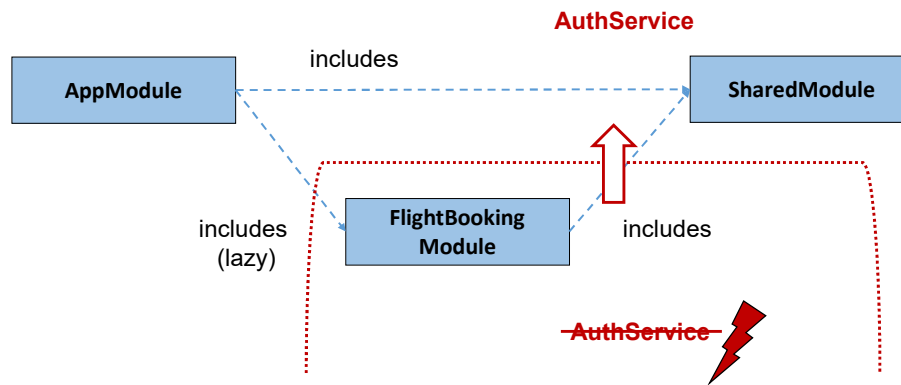
Lazy Loading and Shared Modules



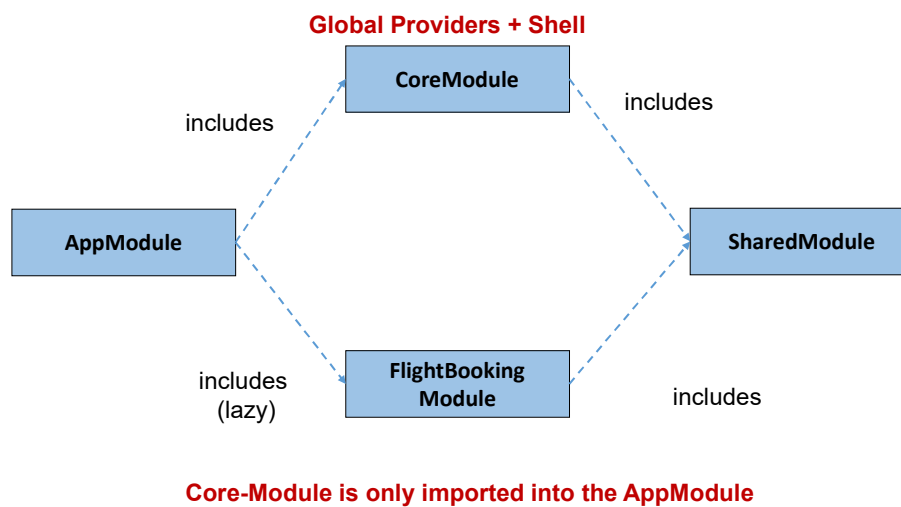
Lazy Loading and Shared Modules



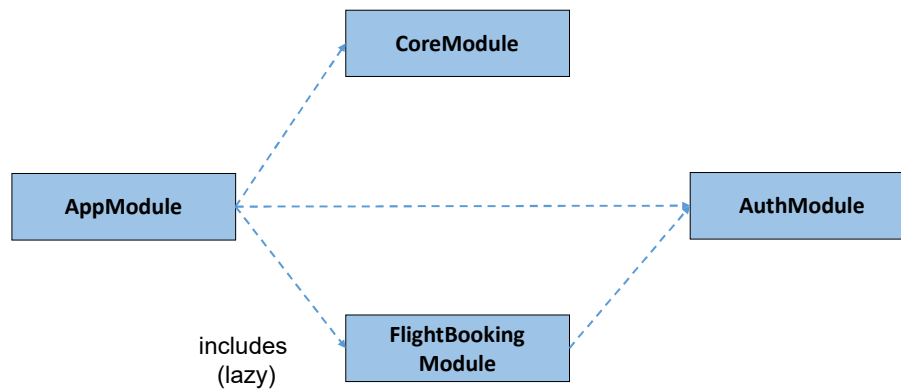
Lazy Loading and Shared Modules



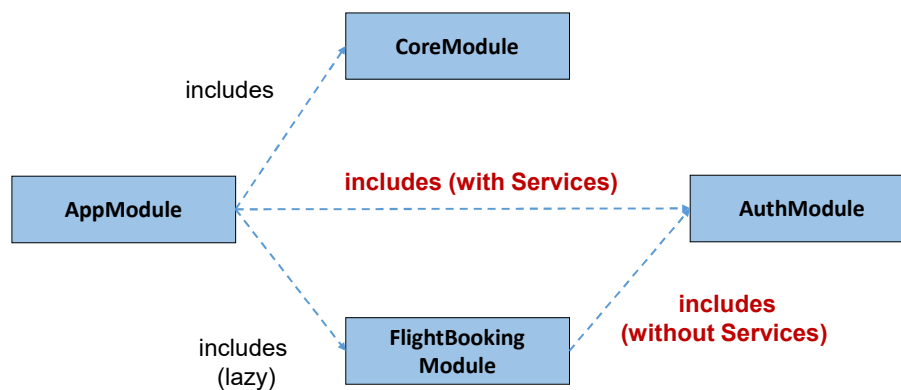
Solution 1: CoreModule



Solution 2: forRoot



Solution 2: forRoot



AuthModule

```
@NgModule({  
  [...],  
  providers: []  
})  
export class AuthModule {  
}
```

AuthModule

```
@NgModule({  
  [...],  
  providers: []  
})  
export class AuthModule {  
  
  static forRoot(): ModuleWithProviders<AuthModule> {  
    return {  
      ngModule: AuthModule,  
      providers: [AuthService, [...]]  
    }  
  }  
}
```

DEMO

Solution 3: Tree-shakable Provider

```
@Injectable({ providedIn: 'root' })  
export class AuthService {  
  [...]  
}
```

Preloading



Idea

- Modules that **might be needed** later are loaded after (!) the start of the application
- When the module is actually needed, it is available **immediately**

Activating Preloading

```
const AppRoutesModule =  
  RouterModule.forRoot(  
    ROUTE_CONFIG,  
    { preloadingStrategy: PreloadAllModules });
```

DEMO

Summary

- Child Routes
- Aux Routes
- Guards and Resolvers
- Lazy Loading and Preloading
- Lazy Loading and Providers