

Angular 2 Meetup

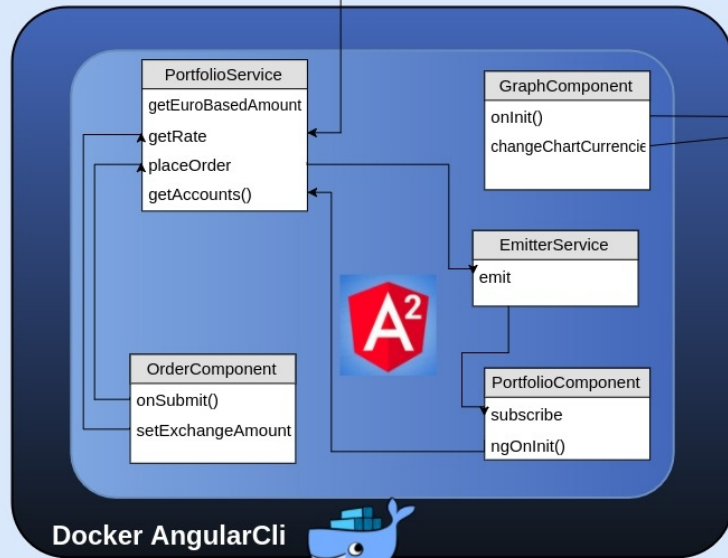


par Michel Doucerain

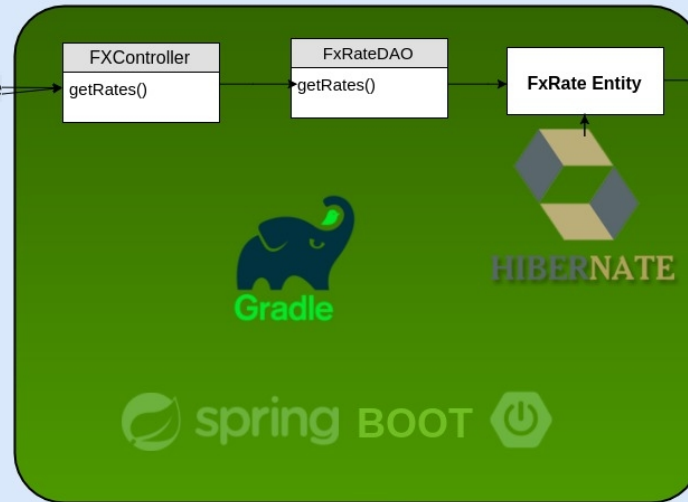
Angular Demo



Rest Service



Rest Service



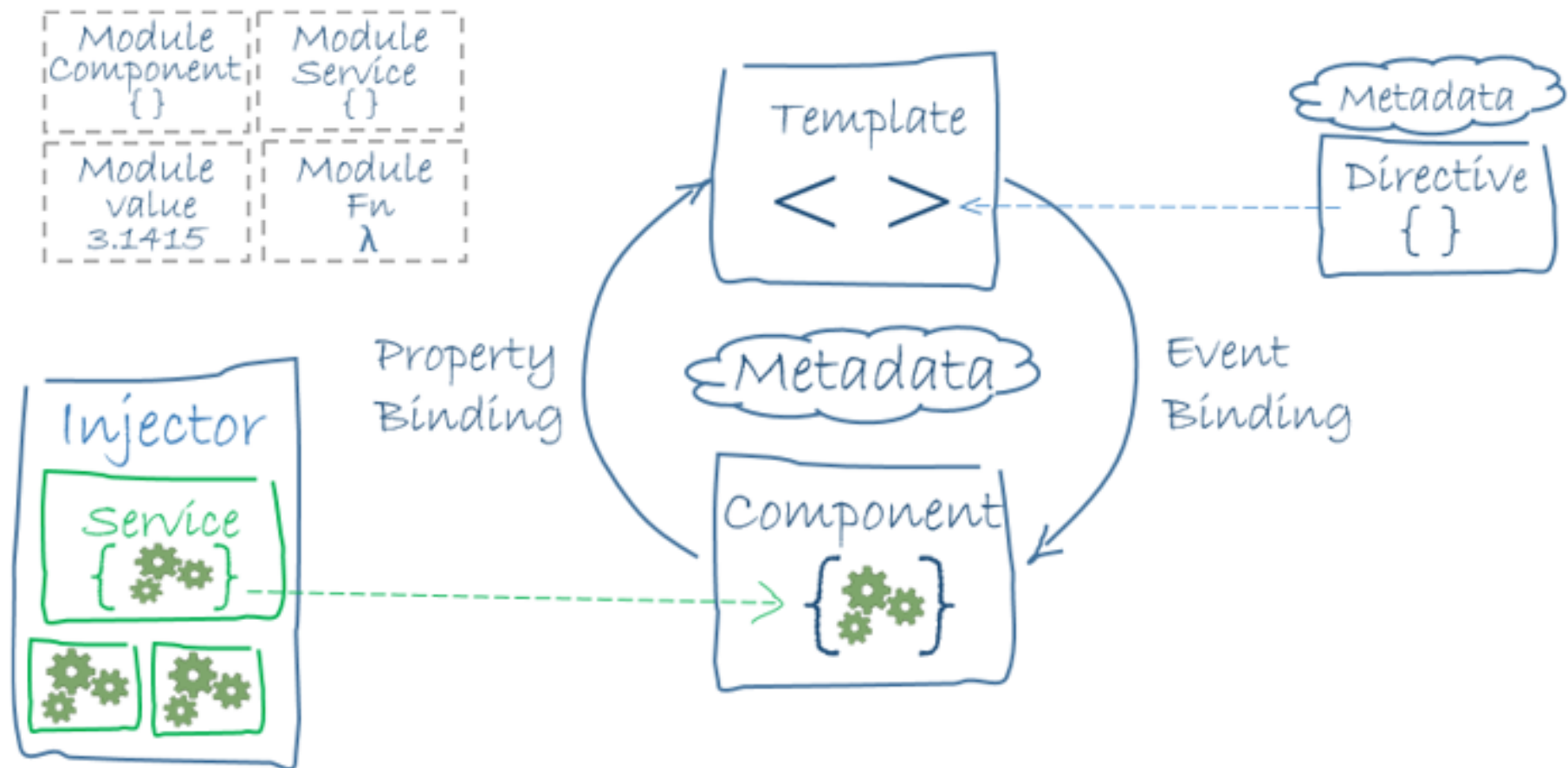
Google Cloud Compute Engine



Pourquoi utiliser Angular 2 ?

- Architecture MVC, inheritance (extends, implements), interface, OOP
- Modularization, separation of concerns, lazy loading of modules
- TypeScript, catch errors on compilation, ES6, Auto change detection, faster development
- Dependency Injection, clean loose coupling entre components et service
- Ui compatible nativement avec iOS et Android, supporté par **Google**
- Routing pour single page app
- Unit test et integration test bien supporté (Karma, Protractor)
- Rx.js librairie, données asynchrones et basé sur mécanisme de subscription, observable – observer
- Form Builder, Custom validation

Architecture Angular 2



Modules

- Toutes les apps ont un root module
- Décorateur @NgModule, bloc fonctionnellement cohésif
- Ces propriétés sont
 - Déclarations : les classes de vues
 - Components, Directives, Pipes
 - Providers:
 - Services
 - Imports
 - Autres modules importés dans ce module
 - Exports
 - Déclarations disponibles à d'autres module
 - Bootstrap
 - Root component

angular-demo > src > app > app.module.ts

component.html x order.component.ts x portfolio.component.ts x order.component.css x graph.component

```

23
24 import { RouterModule, Routes } from '@angular/router';
25 import { AppComponent, AppService } from './app.component';
26 import { PortfolioComponent } from './portfolio/portfolio.component';
27 import { OrderComponent } from './order/order.component';
28 import { PortfolioService } from './service/portfolio.service';
29 import 'hammerjs';
30 import { GraphComponent } from './graph/graph.component';
31 import { MyPortfolioComponent } from './my-portfolio/my-portfolio.component';
32 import { ChartModule } from 'angular2-highcharts';
33
34 declare var require: any;
35 export function highchartsFactory() {
36   const hc = require('highcharts/highstock');
37   const dd = require('highcharts/modules/exporting');
38   dd(hc);
39   return hc;
40 }
41
42
43 const appRoutes: Routes = [
44   { path: 'portfolio', component: MyPortfolioComponent },
45   { path: 'chart', component: GraphComponent },
46   { path: '**', component: MyPortfolioComponent }
47 ];
48
49
50
51
52 @NgModule({
53   declarations: [
54     AppComponent,
55     PortfolioComponent,
56     OrderComponent,
57     GraphComponent,
58     MyPortfolioComponent
59   ],
60   imports: [
61     RouterModule.forRoot(
62       appRoutes,
63       { enableTracing: true } // <-- debugging purposes only
64     ),
65     BrowserModule,
66     HttpClientModule,
67     MaterialModule,
68     DataTableModule,
69     // MdtFooter, MdtHeader, MdtColumns, MdtRows, MdtCellAlign, MdtTable,
70     BrowserAnimationsModule,
71     CurrencyMaskModule,
72     MdRippleModule,
73     MdButtonModule,
74     MdCheckboxModule,
75     ReactiveFormsModule,
76     FormsModule,
77     ChartModule
78   ],
79   providers: [PortfolioService, AppService,
80     { provide: HighchartsStatic, useFactory: highchartsFactory }
81   ],
82   bootstrap: [AppComponent],
83   schemas: []
84 })
85
86 export class AppModule { }

```

Push successful: Pushed 2 commits to origin/master // View files updated during the push (today 11:44 AM)

app-module.ts

main.ts

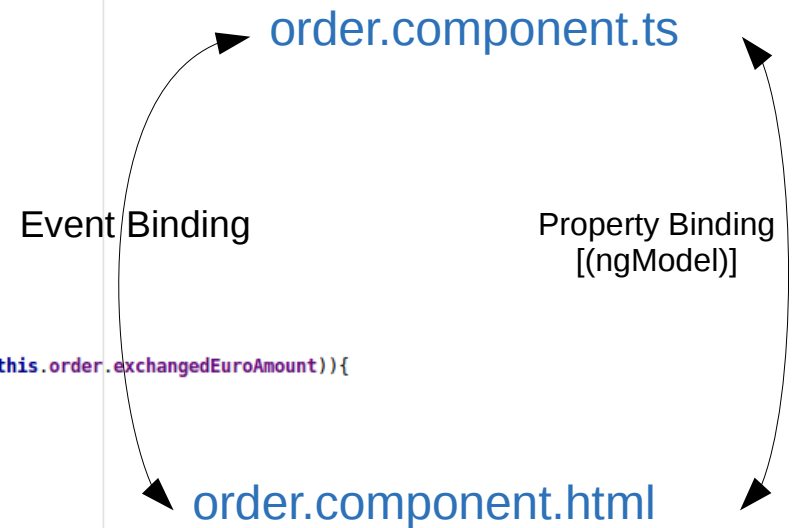
@Component()

- Classes en TypeScript, communique avec le template html avec des évènements ou des 'property binding', définit le comportement logique
- Dependency Injection avec des services, loose coupling, passe services dans constructor
- Besoin de définir providers et directives dans le component
- Lifecycle hooks: ngOnChanges, ngOnInit, ngDoCheck (change detection), ngAfterContentInit, ngAfterContentChecked, ngAfterViewInit, ngAfterViewChecked

```

1 import { Component, OnInit, OnDestroy, EventEmitter, Output } from '@angular/core';
2 import { PortfolioService } from '../service/portfolio.service';
3 import { Order } from '../model/order';
4 import { Account } from '../portfolio/model/account';
5 import { ExchangeRate } from '../model/exchange-rate';
6
7
8 @Component({
9   selector: 'app-order',
10  templateUrl: './order.component.html',
11  styleUrls: ['./order.component.css'],
12  providers: [
13    PortfolioService
14  ]
15 })
16 export class OrderComponent implements OnInit {
17
18
19   portfolio;
20   public order = new Order();
21
22
23   constructor(private portfolioService: PortfolioService) {}
24
25
26   ngOnInit() {
27     this.portfolio = this.portfolioService.getAccounts();
28   }
29
30
31   onSubmit() {
32     setTimeout(() => {
33       console.log(this.order);
34
35       if(!isNaN(this.order.amount)&&!isNaN(this.order.euroAmount)&&!isNaN(this.order.exchangedAmount)&&!isNaN(this.order.exchangedEuroAmount)){
36         this.portfolioService.placeOrder(this.order);
37       }
38     }, 1000);
39   }
40
41   resetOrder(order: Order){
42     order = new Order();
43     this.order = new Order();
44   }
45
46   setExchangeAmount(order: Order) {
47
48     let account = new Account();
49     account = this.portfolioService.getAccountById(order.fromAccountId);
50
51     order.fromCurrency = account.currency;
52
53     if(order.toCurrency.valueOf() != order.fromCurrency.valueOf()){
54
55       this.portfolioService.getRate(order.fromCurrency, order.toCurrency)
56         .subscribe(
57           data => {
58             let exchangeRate = new ExchangeRate();
59             exchangeRate = data;
60             this.order.exchangeRate = exchangeRate.rates[order.toCurrency];
61             order.exchangedAmount = this.portfolioService.calculateAmount(order.exchangeRate, order.operation, order.amount);
62           }
63         );
64

```



Template

- View en HTML, présentation UI, avec de l'encapsulation en utilisant des 'customs tags' qui font appel à d'autres composants
- Utilise interpolation `{{}}` pour récupérer un objet du component, peut aussi utiliser des templates expressions operators (`|` pipe pour formattage, `?.` Elvis operator pour protéger d'une erreur null pointer)
- Data binding et two way binding

```
ent.html x portfolio.component.html x my-portfolio.component.ts x order.component.ts
md-card.portfolio-container table.mdl-data-table.mdl-js-data-table.mdl-shadow--2dptbody>
1 <md-card class="portfolio-container">
2
3
4
5
6 <table class="mdl-data-table mdl-js-data-table mdl-shadow--2dp">
7   <thead>
8     <tr>
9       <th class="mdl-data-table__cell--non-numeric">Currency</th>
10      <th>Account #</th>
11      <th>Balance</th>
12      <th>
13        Balance in Euro
14      </th>
15      <th>
16        Variation
17      </th>
18    </tr>
19  </thead>
20  <tbody>
21    <tr *ngFor="let account of portfolio">
22      <td>
23        <img class="icon-flags" src='../assets/images/{{account.currency}}.gif' />
24      </td>
25      <td>
26        {{account.accountId}}
27      </td>
28      <td>
29        {{account.balance | currency:account.currency:true:'1.2-2'}}
30      </td>
31      <td>
32        {{account.euroBalance | currency:'EUR':true:'1.2-2'}}
33      </td>
34      <td>
35        <span *ngIf="account.variation >= 0">
36          <span class="positive-variation">{{account.variation | percent}}</span>
37        </span>
38        <span *ngIf="account.variation < 0">
39          <span class="negative-variation">{{account.variation | percent}}</span>
40        </span>
41      </td>
42    </tr>
43  </tbody>
44 </table>
45
46
47
48
49
50 </md-card>
51
52
53
```

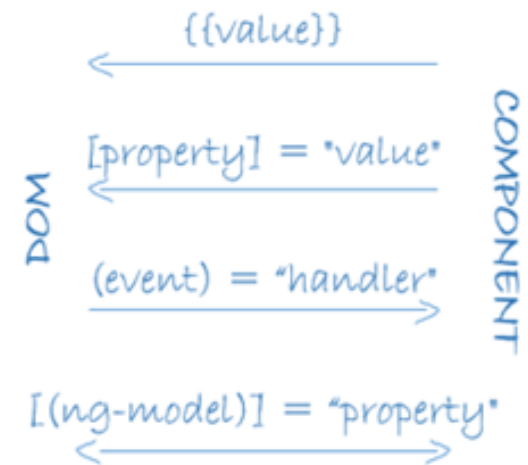
portfolio.component.html

order.component.html

Metadata

- Décorator qui spécifie type de classe et son comportement par défaut
- Exemples de décorator:
 - @Component
 - @Directive
 - @Pipe
 - @Injectable
 - @Input()
 - @Output()
 - @NgModule

Data binding



- Communique data entre template et component
- Property Binding, `@Input() myProperty;` est mise à jour dans le template `[myProperty]="someExpression">`
- Interpolation `{{account.accountId}}` réfère à l'objet `account` dans le component depuis le template
- Event binding `(change)="setExchangeAmount(order);"`, sur le changement du dropdown menu, on calcule le montant échangé dans une autre devise dans le component
- `[(ngModel)]="order.toCurrency"` lie l'objet `order` dans le template formulaire et le component

md-card.order-card form table.order-table tr td md-input-container input

```
4
5
6 <form (ngSubmit)="onSubmit(order)">
7
8   <table class="order-table">
9
10    <tr>
11      <td colspan="2">
12        <md-select required placeholder="Operation" name="operation" [(ngModel)]="order.operation">
13          <md-option value="buy">BUY</md-option>
14          <md-option value="sell">SELL</md-option>
15        </md-select>
16      </td>
17    </tr>
18    <tr>
19      <td colspan="2">
20        <md-select [(ngModel)]="order.toCurrency" required placeholder="Currency" name="toCurrency">
21          <md-option *ngFor="let account of portfolio" value="{{account.currency}}">
22            <img class="small-icon-flags" src='../assets/images/{{account.currency}}.gif' /> {{account.currency}}
23          </md-option>
24        </md-select>
25      </td>
26    </tr>
27    <tr>
28      <td colspan="2">
29        <md-input-container>
30          <input mdInput placeholder="Purchase or Sale Amount" currencyMask [options]="{ prefix: ' ' }"
31            name="amount"
32            required
33            pattern="[0-9.]*"
34            minlength="1"
35            maxlength="16"
36            [(ngModel)]="order.amount"
37          >
38        </md-input-container>
39      </td>
40    </tr>
41    <tr>
42      <td colspan="2">
43        <md-select (change)="setExchangeAmount(order);" required name="fromAccountId" [(ngModel)]="order.fromAccountId" placeholder="From Account#">
44          <md-option *ngFor="let account of portfolio" value="{{account.accountId}}">
45            <img class="small-icon-flags" src='../assets/images/{{account.currency}}.gif' /> {{account.accountId}}
46          </md-option>
47        </md-select>
48      </td>
49    </tr>
50    <tr>
51      <td colspan="2">
52        <md-input-container>
53          <input [(ngModel)]="order.exchangedAmount" name="exchangedAmount" mdInput disabled
54            placeholder="Cost or Proceeds" currencyMask [options]="{ prefix: ' ' }" value="{{order.exchangedAmount}}"
55          >
56          {{order.fromCurrency}}
57        </md-input-container>
58      </td>
59    </tr>
60    <tr>
61      <td>
62        <button type="submit" md-raised-button color="primary">Confirm</button>
63      </td>
64      <td>
65        <button (click)="resetOrder(order);" md-raised-button class="reset-button" color="danger">Reset</button>
66      </td>
67    </tr>
68  </table>
69</form>
```

order.component.html

@Directive

- Transforme template, altère l'apparence ou le comportement d'un élément du DOM.
 - Structural directives (rajoute, retire element du DOM)
 - `<md-option *ngFor="let account of portfolio"`
 - ```
= 0">
 {{account.variation | percent }}

<span *ngIf="account.variation < 0">
 {{account.variation | percent }}

```
  - Attribute directives (altère apparence ou comportement d'un élément)
    - ngModel directive, qui implémente le 'two-way data binding'

```
<md-select required placeholder="Operation" name="operation" [(ngModel)]="order.operation">
```

# Services

- Classe qui accomplit une fonction bien précise dans l'application
- Communique avec le serveur avec des services Rest
- Components font appel aux services pour communiquer avec serveurs ou autres components, (définit dans la propriété providers du component)
  - Publish/subscribe ou event emitter/listener
- Examples:
  - logging service
  - data service
  - message bus
  - Foreign exchange rate service

```
import ...

// Observable class extensions
import ...

// Observable operators
import ...

@Injectable()
export class PortfolioService {

 @Input()
 public portfolio: Account[] = [
 {accountId: 4526647, balance: 57060, euroBalance: 50000, currency: 'USD', variation: 0},
 {accountId: 8523495, balance: 50000, euroBalance: 50000, currency: 'EUR', variation: 0},
 {accountId: 7320665, balance: 6387500, euroBalance: 50000, currency: 'JPY', variation: 0},
 {accountId: 4451277, balance: 43966.5, euroBalance: 50000, currency: 'GBP', variation: 0},
 {accountId: 4135811, balance: 74255, euroBalance: 50000, currency: 'AUD', variation: 0},
 {accountId: 8220199, balance: 73925, euroBalance: 50000, currency: 'CAD', variation: 0},
 {accountId: 3352455, balance: 54650, euroBalance: 50000, currency: 'CHF', variation: 0}
];
```

## portfolio.service.ts

```
private _http = null;

constructor(private http: Http) {
 this._http = http;
}

public getAccounts(): Account[] {
 return this.portfolio;
}

public getRate(baseCurrency: string, currency: string) {
 return this._http.get('http://api.fixer.io/latest?symbols='+currency+'&base='+baseCurrency).map((res: Response) => res.json());
}
```

```
public calculateAmount(rate: number, operation: string, amount: number): number {
```

```
 var exchangeAmount = amount / rate;
 exchangeAmount = parseFloat(exchangeAmount.toFixed(2));
```

```
 if(operation.toLowerCase()=="buy"){
 return -1*exchangeAmount;
 } else {
 return exchangeAmount;
 }
}
```

```
public getEuroBasedAmount(rate: number, amount: number): number{
```

```
 var euroBasedAmount = amount;
 if(rate!=1){
 console.log("euro based rate",rate);
 euroBasedAmount = amount / rate;

 console.log("euroBasedAmount1",euroBasedAmount);

 euroBasedAmount = parseFloat(euroBasedAmount.toFixed(2));

 console.log("euroBasedAmount2",euroBasedAmount);
 return euroBasedAmount;
 } else {
```

## emitter-service.ts

```
import {EventEmitter} from '@angular/core';
```

```
export class EmitterService {
 private static _emitters: { [channel: string]: EventEmitter<any> } = {};
 static get(channel: string): EventEmitter<any> {
 if (!this._emitters[channel])
 this._emitters[channel] = new EventEmitter();
 return this._emitters[channel];
 }
}
```





# Dependency injection

- Principe de Loose Coupling pour définir dépendences de services dans le constructor du component
- Utilise décorator @Injectable() pour définir que le service est disponible à être injecté par le component
- Facilite la communication entre component, service à travers l'application

```
@Injectable()
```

```
export class PortfolioService {}
```

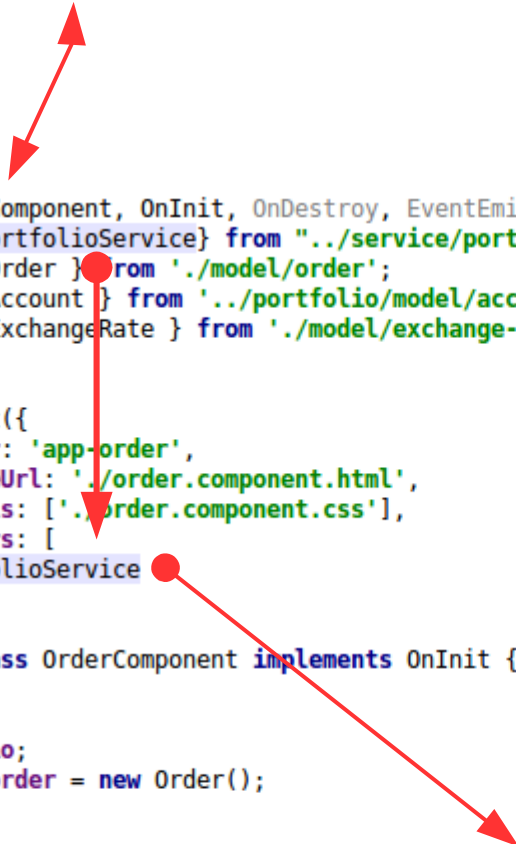
```
import { Component, OnInit, OnDestroy, EventEmitter, Output } from '@angular/core';
import {PortfolioService} from "../service/portfolio.service";
import { Order } from '../model/order';
import { Account } from '../portfolio/model/account';
import { ExchangeRate } from '../model/exchange-rate';
```

```
@Component({
 selector: 'app-order',
 templateUrl: './order.component.html',
 styleUrls: ['./order.component.css'],
 providers: [
 PortfolioService
]
})
export class OrderComponent implements OnInit {
```

```
 portfolio;
 public order = new Order();
```

```
 constructor(private portfolioService: PortfolioService) {}
```

```
 ngOnInit() {
 this.portfolio = this.portfolioService.getAccounts();
 }
}
```



# Routing

- Navigation dans l'application
- Single page app
- Navigation à travers plusieurs composants sur la même page
- ```
import { RouterModule, Routes } from '@angular/router';
```
- Router outlet permet de définir une navigation dans une tag et garder un menu commun

body.demo-body.mdc-typography

<!-- The whole content below can be removed with the new code.-->

```
<body class="demo-body mdc-typography">
<aside class="mdc-persistent-drawer">
  <nav class="mdc-persistent-drawer__drawer">
    <div class="mdc-persistent-drawer__toolbar-spacer"></div>
    <div class="mdc-list-group">
      <nav class="mdc-list">
        <a class="mdc-list-item mdc-persistent-drawer--selected" routerLink="/portfolio" routerLinkActive="active">
          <i class="material-icons mdc-list-item__start-detail" aria-hidden="true">account_balance</i>Portfolio
        </a>
        <a class="mdc-list-item" routerLink="/chart" routerLinkActive="active">
          <i class="material-icons mdc-list-item__start-detail" aria-hidden="true">trending_up</i>Chart
        </a>
      </nav>
    </div>
  </nav>
</aside>
<div class="demo-content">
  <header class="mdc-toolbar mdc-elevation--z4">
    <div class="mdc-toolbar__row">
      <section class="mdc-toolbar__section mdc-toolbar__section--align-start">
        <button class="demo-menu material-icons mdc-toolbar__icon--menu">menu</button>
        <span class="mdc-toolbar__title catalog-title">Forex Trading Platform</span>
      </section>
    </div>
  </header>

  <main class="demo-main">
    <div style="...">
      <i><!-- Left aligned menu below button --></i>

      <router-outlet></router-outlet>

    </div>
  </main>
</div>
```

app.component.html