TD n°3

- Réalisation du TD n°2
- Directives
- Services dont http_service



-- Exercice : Convertisseur de devises

-- Objectifs

- Utiliser des services Angular existants
- Mettre en oeuvre l'injection de dépendance

-- Fonctionnalités

- Saisir un montant dans une devise source
- Sélectionner la devise cible
- Calculer la conversion à partir des taux courants
- Effectuer l'opération inverse (échange des devises)

-- Ressources à utiliser

- Télécharger le fichier currencymap.json sur le site Locale Planet
- Enregistrer le dans le dossier **app/data** (à créer) de votre projet

Ce fichier JSON va permettre l'initialisation des listes de monnaies affichées dans l'application

Le service $http devra être injecté dans le contrôleur :

```
angular.module("currencyApp").controller("currencyController",['$http',
function($http) {
...
```

L'initialisation de la variable **currencies** grâce au fichier JSON se fera dans le contrôleur par une requête Ajax :

```
$http.get('app/data/currencymap.json').
    success(function(data, status, headers, config) {
        self.currencies = data;
    }).
    error(function(data, status, headers, config) {
```

```
        console.log("Erreur avec le statut Http : "+status);
    });
```

Pour l'obtention des taux de change, on utilisera l'API Currency sur http://free.currencyconverterapi.com/ via Ajax et JSONP

Exemple d'interrogation :
http://free.currencyconverterapi.com/api/v3/convert?compact=y&q=USD_EUR&callback=JSON_CALLBACK

Résultat JSON :

```
{"to": "EUR", "rate": 0.83329500000000001, "from": "USD", "v": 0.83329500000000001}
```

La récupération des données sur **free.currencyconverterapi.com** doit se faire par l'intermédiaire de **JSONP**, et non de **JSON**, ce qui impose l'appel de la méthode jsonp sur l'objet $http, et le passage du paramètre JSON_CALLBACK :

```
$http.jsonp('http://free.currencyconverterapi.com/api/v3/convert?compact=y&q='+from
.code+'_'+to.code+'&callback=JSON_CALLBACK').
        success(function(data, status, headers, config) {
            self.result=data[self.from.code+'_'+self.to.code].val;
...
```

-- Application/Contrôleurs

Application	**CurrencyApp** (app/currencyApp.js)
Contrôleur	**CurrencyController** (app/currencyController.js)
	currencies variable Objet stockant la liste des monnaies alimentée par le fichier JSON currencymap.json au démarrage du contrôleur
	from variable stockant la monnaie source (à initialiser en euro)
	to variable stockant la monnaie cible (à initialiser en dollars US)
	what variable stockant le montant à convertir (à initialiser à 1)
	result variable donnant le résultat de la conversion
	getResult() Retourne le résultat en effectuant une requête JSONP vers l'adresse free.currencyconverterapi.com Modifie en retour la variable result
	swap() Echange la monnaie source et la monnaie cible

-- Vue

-- Directives utilisées

- ng-app
- ng-controller
- ng-click
- ng-model
- ng-keypress

-- Interface



-- Historisation des conversions

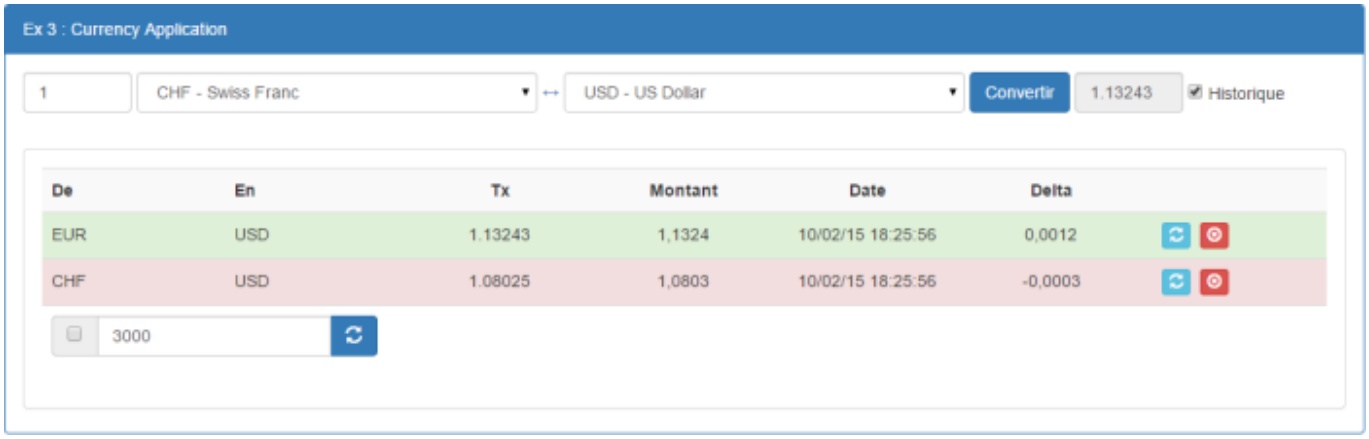
-- Fonctionnalités

L'historisation se fait selon les principes suivants :

- Toute nouvelle conversion ajoute une ligne dans l'historique des conversions (identifiée par le couple monnaie From + To)
- Toute requête vers une conversion déjà présente dans l'historique met à jour la ligne de l'historique concernée (rate & delta)
- L'utilisateur pourra afficher/masquer la liste des conversions déjà effectuées (les 2 monnaies, le sens , le tx et la date/heure de conversion, l'écart de taux avec une conversion précédente)
  - Vider cette liste
  - Supprimer un élément de la liste
  - Mettre à jour automatiquement les conversions en effectuant à nouveau une requête à intervalle donné
  - Calculer l'écart de tx ou de montant et faire apparaître en rouge les baisses, en vert les hausses

On utilisera pour mémoriser les conversions déjà effectuées un objet du type :

```
var conversion={from : //monnaie1,
        to : //monnaie2,
        amount : function(){ //retourne le montant (tx* somme) },
        initialAmount : function(){ //Retoune le montant avec tx initial * somme},
        delta : //écart Avec première requête (tx actuel - tx initial) * somme
        rate : //tx actuel,
        what : //Somme, date : //date & heure de la requête,
        update: //Flag pour "en cours de mise à jour" (requête ajax),
        initialRate : //tx initial  : invariant depuis la première requête
        };
```



-- Service

Intégrer la requête vers le serveur et l'historisation dans un service **CurrencyService**

Service	**CurrencyService** (app/currencyService.js)
	historique Tableau de conversions
	update(from,to,what,hasHisto,result) Effectue une requête vers le serveur, met à jour l'historique et le paramètre **result**

Tester les Progressbars

-- Exercice : Calculatrice

-- Objectifs

1. Créer un service (moteur de la calculatrice)
2. Utiliser un service (Injection de dépendance)
3. Créer des directives

-- Fonctionnalités

1. Faire des calculs simples
2. Mémoriser un résultat
3. Eteindre/allumer calculatrice

-- Vue



-- Consignes de réalisation

Directives :

- Créer une directive permettant de créer un bouton de calculatrice (placer ensuite les valeurs possibles prises par les boutons dans un tableau, initialisé dans le code :

```
    this.ops=[
                {value:'('},
                {value:')'},
                {value:'M'},
                {value:' '},
                {value:'RM'},
                {value:'Off',cssClass:"btn-danger",title:"Eteindre la
calculatrice"},
                {value:'\n'},
                {value:"←"},
                {value:"CE"},
                {value:"C"},
                {value:" "},
                {value:"±"},
                {value:"√x"},
                {value:"\n"},
                {value:7},
                {value:8},
```

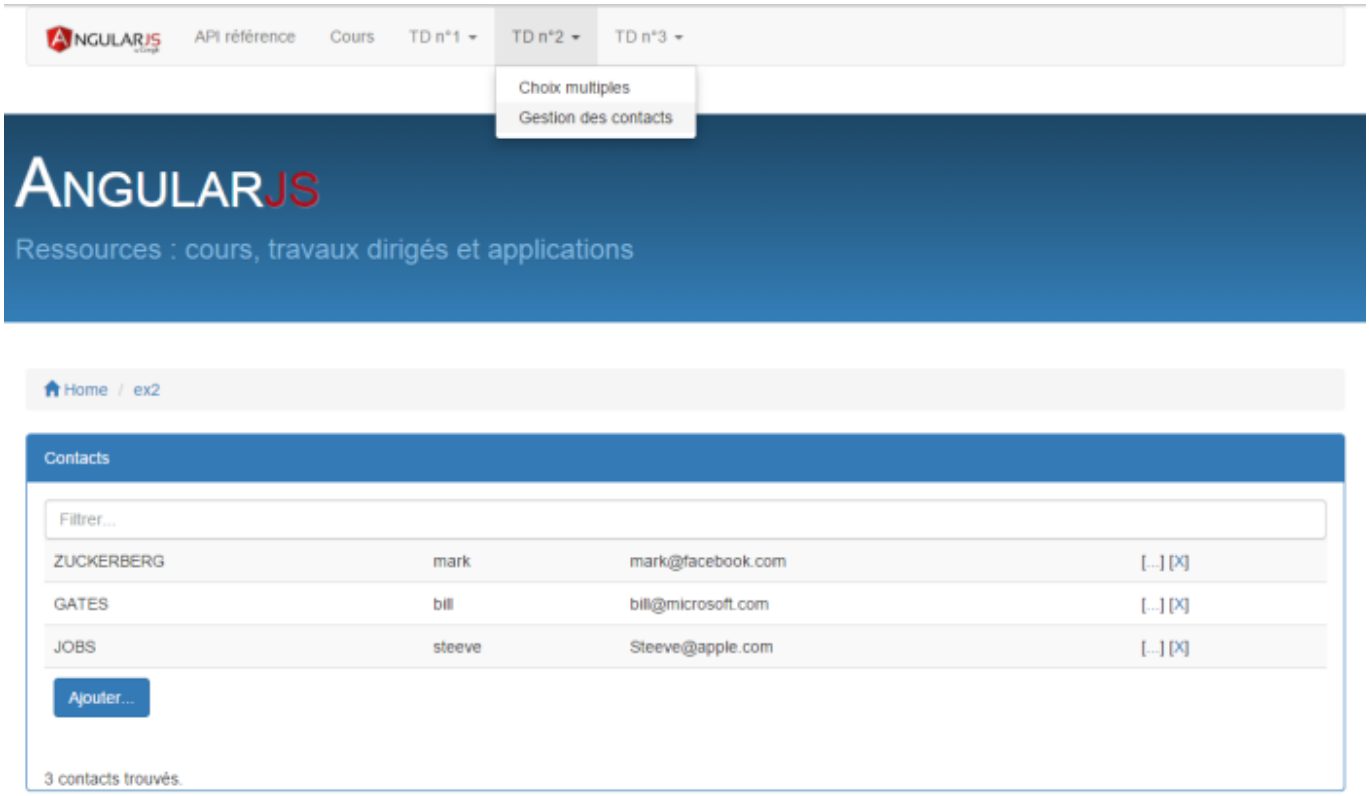
```
                {value:9},
                {value:' '},
                {value:'/'},
                {value:'%'},
                {value:'\n'},
                {value:4},
                {value:5},
                {value:6},
                {value:' '},
                {value:'*'},
                {value:'1/x'},
                {value:'\n'},
                {value:1},
                {value:2},
                {value:3},
                {value:' '},
                {value:'-'},
                {value:'\n'},
                {value:0,cssClass:"colspan btn-default"},
                {value:'.',cssClass:"btn-default"},
                {value:' '},
                {value:'+'},
                {value:'=',cssClass:"rowspan btn-success",title:"Calculer
([ENTREE]) !"}
                ];
```

- créer la partie calcul dans 1 service, et injecter ce service dans le contrôleur.

-- Routage

- Lecture de la section routage

Il s'agit de créer une application permettant de naviguer entre les différents exemples implémentés, en utilisant les fonctionnalités de routage d'AngularJS.



Tester le routage en ligne

From:
http://slamwiki2.kobject.net/ - **SlamWiki 2.1**

Permanent link:
http://slamwiki2.kobject.net/slam4/richclient/angularjs/td3?rev=1457422526

Last update: **2019/08/31 14:40**