Séminaire Angular

Pour l'ensemble des exercices, **se reporter aux slides** pour des exemples de solutions / d'implémentation.

Exercice #1

Le but de cet exercice sera de comprendre le coeur du fonctionnement d'Angular.js. A l'issue de cet exercice, l'objectif sera d'avoir une bonne connaissance de :

- La phase de digest d'Angular.js.
- La notion de watchers.
- A quoi servent les directives.

Ce TP est découpé en plusieurs TODO détaillés par la suite.

TODO #1

La première étape consiste à implémenter la fonction $apply. Cette fonction est primordiale dans Angular.js car elle permet de déclencher la phase de digest au coeur du data binding du framework.

L'implémentation de cette fonction se fait en deux étapes :

- Exécution de la fonction donnée en paramètre.
- Déclenchement de la phase de digest.

Se reporter aux slides pour un exemple d'implémentation.

TODO #2

La deuxième étape consiste à implémenter la phase de digest. L'algorithme à implémenter consiste à :

- Parcourir tous les watchers présents sur le scope.
- Vérifier la différence entre la précédente valeur et la nouvelle valeur.
- En cas de changement, exécuter la fonction listener du watcher.

Se reporter aux slides pour un exemple d'implémentation.

TODO #3

La troisième étape consiste à implémenter la directive ng-bind. Cette directive permet de mettre à jour le DOM lors d'un changement de variable (les changements étant détectés via la fonction $watch déjà implémentée).

Lors de l'exécution du watcher :

- Récupérer la nouvelle valeur (premier paramètre de la fonction listener).
- Mettre à jour le DOM avec cette nouvelle valeur.

Pour rappel, la mise à jour du DOM peut se faire en écrivant sur la propriété innerHTML :

```
element.innerHTML = 'Hello World';
```

TODO #4

Enfin, la dernière étape consiste à implémenter la directive ng-model :

- Dans l'événement keyup, récupérer la valeur du champ de saisie.
- Mettre à jour le scope avec la valeur du champ de saisie.

Pour rappel, la récupération de la valeur d'un champ de saisie peut se faire via le premier paramètre au déclenchement de l'événement :

```
element.addEventListener('keyup', function(e) {
  console.log('Value = ', e.target.value);
});
```

Exercice #2

Le deuxième exercice va consister à mettre en place un "tweeter like".

Cet exercice va nécessiter le lancement d'un serveur Web (celui-ci est déjà implémenté). Le lancement de ce serveur se fera via Node.js :

```
cd exercice_2
node.exe server/app.js
```

Le message suivant devrait apparaître dans la console : "App listening on port 4000"

TODO #1

La première étape consiste à ajouter Angular.js dans l'application :

- Dans le répertoire public, ouvrir le fichier index.html.
- Avant l'import du fichier app.js, ajouter l'import du fichier angular.js.
- Rafraichir l'écran, et vérifier qu'aucune erreur n'apparait dans la console.

TODO #2

La deuxième étape consiste à récupérer les tweets via un appel http et à les afficher à l'écran :

- Dans le répertoire public/js, ouvrir le fichier app.js.
- Dans le contrôleur TweetController, effectuer un appel HTTP en GET sur l'url tweets
    - Dans le callback de succès, affecter les données au contrôleur.

Ensuite, il faut les afficher à l'écran :

- Dans le répertoire public, ouvrir le fichier index.html.
- Dans le bloc html, afficher les tweets en utilisant le binding angular (utilisation de {{ }}).
    - La propriété login est disponibles via tweet.login.
    - La propriété message est disponibles via tweet.message.

Rafraichir l'écran pour vérifier que le tweet s'affiche correctement.

TODO #3

Nous allons maintenant permettre l'enregistrement d'un tweet :

- Dans le répertoire public, ouvrir le fichier index.html :
    - Rajouter les directives ng-model sur les champs de saisie.
    - Rajouter les attributs de validation sur les champs de saisie (les deux champs sont required, le message du tweet ne doit pas excéder 140 caractères).
- Dans le répertoire public/js, ouvrir le fichier app.js.
- Dans le contrôleur TweetController, implémenter la fonction submit :
    - Cette fonction doit un effectuer un appel HTTP POST sur l'url tweets
    - Implémenter la fonction de success afin de rafraichir l'écran lorsqu'un tweet à été sauvegardé.

Rafraichir l'écran et essayer de tweeter !

TODO #4

La troisième étape consiste à afficher le login utilisateur au format tweeter, c'est à dire préfixé par @ :

- Dans le répertoire public/js, ouvrir le fichier app.js.
- Créer un filtre tweeterLogin dont le rôle sera de renvoyer le login préfixé par le caractère @.

Ensuite, il faut utiliser ce filtre :

- Dans le répertoire public, ouvrir le fichier index.html.

- Lors de l'affichage du login, utiliser le filtre tweeterLogin.

Rafraichir l'écran pour vérifier que le login est bien préfixé par le caractère @.

TODO #5

Afin d'apporter plus d'interaction, nous allons utiliser l'API WebSocket afin de rafraichir automatiquement l'écran lorsqu'un tweet a été sauvegardé par un utilisateur :

- Dans le répertoire public/js, ouvrir le fichier app.js.
- Dans le contrôleur TweetController, capter l'événement tweet:new afin de rafraichir l'écran.

Pour rappel, il est possible de capter un événement via la fonction $on :

```
$scope.$on('tweet:new', function(event, tweet) {
  console.log('new tweet is coming: ', tweet);
});
```

Comme les créations de tweet sont captées via l'API WebSocket, la fonction de success associée à la sauvegarde d'un tweet devient inutile.

TODO #6

Afin de centraliser l'affichage d'un tweet, nous allons utiliser une directive customisée pour l'afficher à l'écran :

- Dans le répertoire public/js, ouvrir le fichier app.js.
- Créer une directive tweet : cette directive sera assez simple et affichera juste le contenu d'un tweet avec un template.

Par exemple, la directive pourrait ressembler à :

```
angular.module('app')
  .directive('tweet', function() {
    return {
      template: '<div><span class="label label-primary">{{ data.login | tweetLogin }}</span> <span>{{ data.message }}</span></div>',
      scope: {
        data: '='
      }
    };
  });
```

Enfin, utiliser la directive dans le fichier public/index.html !