Manuel de Travaux Pratiques

Formation AngularJS

Version : 1.1.1

Objectifs du TP

Dans ce TP, nous allons utiliser AngularJS pour développer une application de crowfunding (ou finance participative). Ce TP nous permettra de mettre en pratique au fur et à mesure ce qui a été vu pendant le cours.

Les notions suivantes seront abordées : Expressions, Formulaires, Scopes et Controllers, Vues et Route, Services, Resources http, Directives, Filtres Custom, les Tests Unitaires et End to End, l'Authentification et enfin, l'Internationalisation.

Nous nous intéresserons dans ce TP à la partie client de l'application, implémentée en AngularJS. La partie serveur sera hostée sur un serveur Express basé sur Node.js, et sur la base de données NoSQL MongoDB.

Mise en place du TP

Pré-requis

Vérifiez que Node.js et MongoDB sont installés.
Vérifiez que les package grunt-cli et karma-cli sont installés au niveau global via la commande :

```
npm list -g --depth=0
```

Si besoin, les installer via la commande :

```
npm install -g grunt-cli karma-cli
```

Mise en place

Récupérer le zip tp1-provided.

Pour gérer les données de notre application, nous allons utiliser mongoDB. Il faut donc créer un dossier qui contiendra vos données et démarrer la base de données en lançant la commande suivante :

```
mongod --dbpath chemin_du_dossier_qui_contiendra_les_données
```

Node étant installé, nous allons lancer, depuis le répertoire du tp unzippé, la commande :

```
npm install
```

Cette commande s'appuie sur le fichier package.json pour télécharger l'ensemble des modules node.js nécessaires à ce TP.

Grunt, nous permettra de lancer le serveur en utilisant la commande suivante.

```
grunt serve
```

Nous reviendrons sur cet outil plus tard dans cette formation.

Le répertoire « server » contient la configuration du serveur qui va porter notre application web Angular, et fournir des web services avec lesquels interagir.
Le répertoire « app » sera notre répertoire de travail. Il contient d'ores et déjà des bases pour notre TP : feuilles de styles, polices, images, librairies, …etc.
La page index.html (qui est appelé par défaut) contient déjà le squelette html, et l'import des css et fonts.

TP1 : Les Expressions

Lancer le serveur avec la commande suivante :

```
grunt serve
```

Dans le fichier principal : index.html.

1) Ajouter la dépendance vers la librairie AngularJS
2) Définir le body comme application AngularJS (en utilisant l'attribut ng-app).
3) Créer des expressions correspondantes aux intitulés :
    a. Calcul : 1+1
    b. Concatenation : « Hello » et « World »
    c. Accéder à une variable non définie.
4) Utiliser des filtres
    a. Filtrer la chaine de caractères en majuscules.
    b. Filtrer le nombre pour avoir une décimale avec 2 chiffres après la virgule
    c. Filtrer la date pour la mettre au format : dd/MM/yyyy

TP2 : Les formulaires

Créer un formulaire permettant de créer un projet, possédant les attributs suivants :

- nom du projet,
- Email du propriétaire du projet,
- Description du projet,
- Montant demandé pour le projet.

Utiliser la validation fournie par AngularJS en implémentant les contraintes suivantes :

- Nom du projet (20 caractères maximum)
- Email, doit être au format email
- Montant : doit être inférieur à 1000
- Le bouton de validation doit contenir le nom du projet (ex : Launch your project <ProjectName>)
- Le bouton de validation du projet doit être activé si et seulement si le formulaire est valide
- Changer la couleur du border pour que l'input soit vert si le remplissage du champ est correct, rouge sinon.

TP3 : Controller et Scope

Etape 1 : Utiliser le scope pour partager des données entre la vue et le controleur

Dans le dossier `js` créer un fichier app.js. Créer un module principal nommé `myApp`.

Créer un second module `controllers`, et ajouter le en tant que dépendance dans le module principal.
Dans le module `controllers`, créer un contrôleur `ProjectsCtrl` qui définit un objet projet :

```
{id:2, name:"Project 2", user:"pierre@mail.com", description:"Ceci
est une description de mon projet", goal:1000};
```

Dans la vue index.html, créer un div qui affiche les informations du projet.

Etape2 : Aller plus loin avec ng-repeat

Modifier le contrôleur pour renvoyer une liste de projets :

```
[{id:3, name:"Project 3", user:"pierre@mail.com", description:"Ceci est une description de mon projet", goal:700}, {id:4, name:"Project 4", user:"pierre@mail.com", description: "Ceci est une description de mon projet", goal:400}];
```

Modifier la vue html pour afficher la liste des projets, en utilisant l'attribut `ng-repeat` et le tableau html :

```
<table class="table table-striped">
```

Etape3 : Utiliser le scope pour proposer des méthodes

Créer dans le scope une méthode `display()` qui affichera une alerte contenant le nom du projet. Dans la vue, associer cette méthode au clic sur le nom du projet (en utilisant l'attribut `ng-click`).

Etape4 : Contrôleurs imbriqués

Créer un 2ème contrôleur `NewProjectCtrl`, l'associer au formulaire de création de projet créé au TP2 en l'imbriquant sous le contrôleur `ProjectsCtrl`.
Créer une méthode `addProject()` permettant d'ajouter un projet à la liste.

Etape5 : Communication entre contrôleurs imbriqués

Ajouter à la méthode d'ajout de projet, un événement `Created`. Catcher cet événement dans le contrôleur parent, et afficher une alerte.

TP4 : Route et vues

Ajouter la librairie angular-route.min.js à votre application.

Etape1 : Créer des vues et le router

Créer un répertoire partials, et déplacer :

- le formulaire de création d'un projet dans form.html,
- la liste des projets dans list.html

Dans index.html, définir une `ng-view`.

Dans le module principal (`myApp`) ajouter une configuration de module, pour définir les routes via le `$routeProvider`. Créer 2 routes `/projects` et `/newproject`, en leur associant vues et contrôleurs.
Information : avec cette nouvelle configuration, les deux contrôleurs ne sont plus imbriqués, il est donc normal que l'ajout de projet ne fonctionne plus. L'ajout de projet sera « réparé » dans le TP5.

Etape2 : Créer une barre de navigation

Inclure le fichier `navbar.html` (fourni dans le dossier tp04_provided) dans la vue principale en utilisant la directive `ng-include`.

Modifier `navbar.html` pour faire fonctionner correctement les liens.

Etape3 : Créer une vue pour afficher les détails d'un projet

Créer vue, contrôleur, et route pour afficher les détails d'un projet.

Dans la liste des projets, créer un lien pour accéder à la page de détail d'un projet à partir de son nom.
Attention : pour le moment, la liste des projets est définie dans le scope du contrôleur `ProjectsCtrl`. Il n'est donc pas possible d'y accéder depuis le nouveau contrôleur que vous allez créer. Pour le moment, nous nous contenterons d'afficher l'id du projet récupéré dans la route.

Au-dessus de la liste de projets, créer un lien permettant d'accéder directement à la page de création d'un nouveau projet.

TP5 : Services

Etape 1 : Gérer les projets via un service

Créer un nouveau module pour contenir les services, et l'ajouter dans l'application Angular.

Créer un service qui contient la liste des projets, et qui propose plusieurs méthodes :
- `getProjects()` : récupère la liste des projets,
- `getProjectById(id)` : récupère un projet via son id,
- `addProject(project)` : ajoute un projet à la liste.

Utiliser ce service dans les contrôleurs.

Modifier ensuite la vue détails du projet, pour afficher les informations du projet : nom, description, montant demandé.

Ajouter un champ don aux projets et créer un input et un bouton permettant de faire un don au projet.

Etape2 : Catégories

Créer un service qui décrit les catégories suivantes : Web, Games, Design.

Ajouter la catégorie au projet. Mettre à jour les différentes vues pour prendre en compte ce nouveau champ.
Dans la page de création du projet, utiliser la directive ng-options pour la création du select permettant de choisir la catégorie du projet.

TP6 : Ressources HTTP

Le serveur possède un webservice REST qui permet de gérer les projets.
Ainsi :
- `/api/projects` appelé en GET renvoie la liste des projets,
- `/api/projects/<id>` appelé en GET renvoie le projet identifié par l'id,

`/api/projects/` appelé en POST créé un projet,
`/api/projects/<id>` appelé en PUT permet la modification d'un projet,
`/api/projects/<id>` appelé en DELETE permet la suppression d'un projet.

Créer une factory se basant sur le service `$resource`.
Adapter les contrôleurs pour utiliser cette factory plutôt que le service créé précédemment pour accéder aux projets.

TP7 : Directives

Créer un nouveau module pour les directives.

Créer une directive `card` qui permet d'afficher les informations de détails d'un projet.
Utiliser cette directive dans la page de liste des projets.
Trier les `card` de la plus récente à la plus vieille.

Créer une page d'accueil, `main.html`, qui affichera les derniers projets en réutilisant la directive créée.

TP8 : Custom Filters (facultatif)

Etape1 : Filtre par état d'avancement du projet

Dans la page « list.html », ajouter l'élément html suivant :

```
<div class="col-md-3 discover-categories">
       <p class="lead">Projects</p>
       <div class="list-group">
           <a class="list-group-item">All</a>
           <a class="list-group-item">Finished</a>
           <a class="list-group-item">To finish</a>
       </div>
</div>
```

Créer un nouveau module `filters` pour les filtres, et créer un filtre custom qui permet de n'afficher dans la liste que les projets correspondants aux contraintes suivantes :

- Tous : afficher tous les projets
- Finished : afficher les projets dont le montant de don est supérieur ou égal au montant demandé.
- To Finish : afficher les projets dont le montant de don est strictement inférieur au montant demandé.

Etape2 : Filtre par catégorie

Créer un filtre par catégorie, et permettre de filtrer par catégorie dans la vue.

TP9 : Tests unitaires et Tests End to End

Ajouter à votre projet le dossier test fourni dans `tp09-provided`. Dans votre projet, ce dossier doit se situer au même niveau que les dossiers `app` et `server`.

Etape1 : Tests Unitaires

Utiliser Karma pour mettre en place les tests unitaires. Ecrire les tests unitaires en se basant sur Jasmine.
Créer des TU pour valider le fonctionnement des contrôleurs.
Créer des TU pour valider le fonctionnement des filtres.
Créer des TU pour valider le fonctionnement des directives.

Etape2: Tests End to End

Utiliser protractor pour mettre en place les tests End to End sur un ou plusieurs navigateurs. Ecrire les tests en se basant sur Jasmine.
Valider que lorsqu'on affiche la page principale, on a bien 4 projets affichés par défaut.
Valider que lorsqu'on affiche la liste des projets, on a bien une liste de projets.

TP10 : Authentification

Le serveur fournit un web service REST :
`/api/session/` appelé en POST avec email et password, renvoie le user si l'utilisateur est reconnu.
L'utilisateur test@test.com avec pour mot de passe « test » est défini par défaut en base.

Créer une page `login.html` et son contrôleur associé.
Créer un service pour utiliser la méthode d'authentification fournie par le serveur.
Restreindre l'accès aux pages pour que seul un utilisateur connecté puisse accéder aux pages : de détails d'un projet, de création d'un projet.

TP11 : Internationalisation

Ajouter dans votre application Angular les langages fournis dans le dossier `tp11-provided` et positionner une langue par défaut.
Modifier la page `main` et la barre de navigation pour utiliser ces traductions.
Ajouter dans la barre de navigation, un bouton permettant de choisir le langage.