

AngularJS - TP fil rouge

Partie 1

[Prérequis](#)

[Création d'un projet depuis zéro](#)

[Création d'une application AngularJS de base](#)

[Gestion d'utilisateur](#)

[Bonus](#)

Partie 2

[Implementer un router \(ui-router ou angular-router\)](#)

[Ranger le code en composant](#)

[Aller plus loin avec ui-router](#)

Partie 3

[Ecrire un service AngularJS](#)

[Mettre à jour l'application](#)

[Utiliser les données d'un serveur](#)

[Architecture composant](#)

[Bonus](#)

Partie 1

Prérequis

1. Node v4 & NPM v2
2. Installer live-server: ***npm install -g live-server***

Pour le lancer:

- a. Se rendre dans un dossier
- b. Executer la commande *live-server*

Création d'un projet depuis zéro

1. Vérifier les version de node & npm
2. Créer un nouveau dossier du nom de votre projet et vous rendre à l'intérieur
3. Initialiser le package.json (*npm init*)
4. Installer les dépendances (angular, angular-mocks?)
5. Dans votre dossier
 - a. Créer un dossier **src/**
 - b. Créer deux fichiers dans ce dossier **src/**: *index.html* et *app.js*
 - c. Créer un dossier **test/**
 - d. Créer un fichier de spécification dans **test/**: *app.spec.js*

Création d'une application AngularJS de base

1. Créer une application angular nommée "awesomeApp"
2. Créer un controller nommé "awesomeAppCtrl" avec:
 - a. Une variable string avec le titre de la page
 - b. Une variable avec la date du jour

3. Créer le code HTML pour afficher les valeurs

Gestion d'utilisateur

1. Créer une liste d'utilisateur (tableau d'objet User)
User: firstname, lastname, email, birthday_day, birthday_month, birthday_year, avatar_url
2. Créer une méthode pour ajouter un User à la liste
3. Créer une méthode pour supprimer un User par indice dans le tableau
4. Afficher un formulaire pour ajouter un utilisateur (avec affichage des erreurs)
5. Afficher la liste des utilisateurs

Bonus

1. Utiliser des filtres natif
2. Créer des filtres personnalisés
3. Créer des watchers
4. Utiliser du code asynchrone (timeout, requête AJAX)
5. Utiliser l'API <http://randomuser.me/>

Partie 2

Implementer un router (ui-router ou angular-router)

- Ajouter la dépendance dans le projet
- Utiliser cette dépendance dans l'application
 - Dépendance module
 - Directive de vue
 - Directive de liens
- Configurer à minima les routes suivantes:
 - / ou /home ⇒ Accueil
 - /users ⇒ Page listant les utilisateurs
 - /users/new ⇒ Page avec le formulaire
 - /users/[ID] ⇒ Page avec le détail d'un utilisateur

Ranger le code en composant

/src

-- /components

-- -- /user

/ controller, filter, directives directement reliés à un utilisateur */*

-- -- /home

/ controller, filter, directives directement reliés à l'accueil */*

-- -- /misc or /common

/ controller, filter, directives de type tooling/common */*

-- index.html

-- app.js

Aller plus loin avec ui-router

Utiliser les fonctionnalités avancées de ui-router (<https://github.com/angular-ui/ui-router>).

- Vues imbriquées

- Vues multiples
- Etats parent/enfant

Partie 3

Ecrire un service AngularJS

- Ce service UserService à pour rôle de gérer les users
- Ce service doit comporter à minima:
 - Une méthode pour récupérer la collection d'utilisateur
 - Une méthode pour récupérer un utilisateur par son ID
 - Une méthode pour ajouter un utilisateur
 - Une méthode pour supprimer un utilisateur
 - Une méthode pour connaître le nombre d'utilisateur

Mettre à jour l'application

- Mettre à jour les contrôleurs pour utiliser les données venant du service
 - Route /user/:id
 - Route /user

Utiliser les données d'un serveur

- Créer une constante contenant l'URL de l'api
https://randomuser.me
- Mettre à jour le service *UserService* pour utiliser **\$http** et l'API à la place de la collection d'utilisateur existantes.
- Faire un appel à la route <http://api.randomuser.me/?results=2000> en **GET**

Architecture composant

Le but de cette architecture est de ranger tout les fichiers par fonctionnalités en composant. Il faut donc ranger les différents composants dans différents modules

projet/

- index.html

- node_modules/

--- angular/

----- angular.js

--- jquery/

----- jquery.js

--- etc.

- app/

--- app.js ⇒ *angular.module('myApp', ['myApp.home', 'myApp.user'])*

--- modules/

----- home/

----- home.js (controller) ⇒ *angular.module('myApp.home', [])*

----- home.html (template)

----- homeFilter.js

----- etc.

----- user/

----- user.js (controller) ⇒ *angular.module('myApp.user', [])*

----- user.html (template)

----- userService.js (récupération données serveur)
----- userComponent.js
----- **common/**
----- **userCard/**
----- userCard.html
----- userCardController.js
----- userCardComponent.js

Bonus

- Utiliser un système de stockage local
 - localStorage
 - <https://github.com/grevory/angular-local-storage>
 - <https://github.com/gsklee/ngStorage>
 - IndexedDB : <https://github.com/bramski/angular-indexedDB>