Formateur Moussa CAMARA

Angular

Angular reste sur ce concept de SPA (Oriente Single Page Application), mais il a amené un nouveau concept au framerwork : le component

TMA

Chaque composant a sa template et son style

@component

Ng-if=

\*Ngif=

L’utilisation de typescript dans l’application requière d’une transpilation

ECMAScript 6 prendre la code typescript et le transcrive à JS pour tous les navigateurs

Un transpilateur

La transpilation prendre notre code source et le transcrive à JS

TypeScript est un langage open source développé par Microsoft

TypeScript est une couche supplémentaire sur ES5 et ES6 ‘EcmaScript 2015)

TypeScript rajoute les annotations, les classes, imports et les types au langage.

Pour installer TypeScript

$ npm install –g typescript

Angular cli => comand ligne interface

Github

$ git clone « je copie le document » ==> enter

Git dd –A (ALL)

**Git Les commandes**

* Git clone »url repo »
* Git init
* Git commit
* Git push
* Git pull
* Git checkout
* Git status
* Git branch

**Pour installer angular**

Commandes à exécuter après le clone du projet

CD formation

Npm install

Npm start

Pourquoi angular a besoin de NodeJS ? Car angular à besoin de un serveur

Aller au navigateur

**Localhost :4200** ==> enter

Installer dans une ligne

Npm install –g angular /cli

Ng new ‘projetthem’

------------------

Ng serve

----------------------------

**ARCHITECTURE DE FICHIERS**

**node\_modules/** ===> important c’est ici qui on va chercher les fichier

crs/

e2e (and two and) pour test de bout en bout

index.html

styles.css

Angular2 – ES6-SHIM

Angular 2 - ANGULAR POLYFIL

Angular 2 – SYSTEMJS

Est un « module loader

Angular2 – RXJS

Quand on déclare un composant il y a sa Template et son css

Import { component } from ‘@angular/core’ ;

@Component({

Selector :’app-root’ ,

cd node\_modules/

cd app-root-path/

Ne jamais toucher dans le dossier de nodes modules !

Exercice

npm install -g @angular/cli

Ng generate component nomDeMonComponent

Il va créer les fichiers :

create src/app/testcomponent/testcomponent.component.html (32 bytes)

create src/app/testcomponent/testcomponent.component.spec.ts (677 bytes)

create src/app/testcomponent/testcomponent.component.ts (297 bytes)

create src/app/testcomponent/testcomponent.component.css (0 bytes)

update src/app/app.module.ts (568 bytes)

Dans mon dossier chercher le fichier html et le modifie

<h1>

{{title}}

<app-testcomponent></app-testcomponent>

</h1>

--------------------------------------------------------------------------

Exercice1

Reproduire l’écran ci-dessous

My firt ng2 App

Hello word

-----------------------------------------------------------------------------

Exercice2

My firt ng2 App

Et maintenant en rafraichissent notre navigateur, on peut admirer notre hello word

TP

1- TP – Afficher la liste de tous les étudiants dans un tableau

2- Afficher que les admis avec une moyenne supérieure ou égale à 10

Etudiant.moyenne > 9

Admis

< 10

Non admis

3- Afficher les non-admis

PIPE

Templete : ‘<p>{{now | date}}</p>

Voir les cours d’Angular

Corrections <https://github.com/iknsa-formation/angular/free/show-list-students>

------------------------------------------

Git clone site nom projet

Npm install ( tu attends)

Npm start

-------------------------------------------------------

**Installer Angular en partant de rien :**

Verifier si node existe.

npm install –g @angular/cli

ng new my-dream-app

cd my-dream-app

ng serve

------------------------------------------------------------

serveur ==== >localhost4200

--------------------------------------------------------

poec-symfony

jamais commit dans le master

git push tp-tierbrand

Télécharger : nodejs.org ; bash git ou git for window ou git in bash

-------------------------------------------------

Créer un brache en Git

git push « nom de la brache »

on doit créer un brache et après clone

git commit

git remote –g

git remote add symfony

git remote –vgit add src

git push symfony tp-mussa

angular ou poec symfony

-------------------------------------------------------------------------

Instalar Angular quand on n’a pas Gitpush

**Node –v**

**------------------------------------------------------------------------**

Instalar Angular quand on n’a pas Gitpush

Npm install –g @angular/cli

Ng new my-dream-app

Cd my-dream-app

Ng serve

--------------------------------------------------------------------------

Pour faire une branche en Git

<https://github.com/iknsa-formation/poec-symfony/branches>

Dans mon dépôt poec-symfony

Créer ma branche **Tp-mon nom** = tp-veronica

git remote -v

git remote add (origin) veronica [https://github.com/iknsa-formation/po HYPERLINK "https://github.com/iknsa-formation/poec-symfony.git"e HYPERLINK "https://github.com/iknsa-formation/poec-symfony.git"c-symfony.git](https://github.com/iknsa-formation/poec-symfony.git) [enter]

git remote –v (va générer un nouveau dépôt = l’origen et le nom que je l’ai donné

git status (voir le statuts de mon fichier ou dossier )

git add src (additionne le url de mon dossier)

git commit –m « commentaire » (donner un commentaire)

Créer ma nouvelle branche

*git checkout –b tp-veronica (mon nom)*

je dois faire un git dans mon dépôt principal

git push veronica tp-veronica (symfony est le depot )

git remove pour elever

git status (état de lieu, voir tous le fichiers modifies en rouge )

git add –A src/…. (Fichier ajouté)

git add scr src/ (tous le fichiers) on fait après git commit « commentaire »

git add –A src/…. (Fichier ajouter en ordre)

git commit –am ‘‘test’’ le même dans une seule ligne

git checkout –b

git brach (montrer où je suis ou quelle branche je suis en train d’utiliser)

git brach –a (toutes les branches)

git checkout nom de ma branche

git pull –r origin master

git commit –m’’test’’

git commit –am’’test’’

git push origin master

git stash (

git stash pop

git brach –a lister tous mes dossier

ANGULAR LES ROUTES

Importation dans app.module

Import { Routes, routerModule} fron ‘@angular/router’ ;

Const appRoutes : Routes =[

{

Path : ‘’,’’

Component :HomeComponent

] ;

@NgModule({

Imports :[ Router

}

---------------------------------------------------------

Angular-declarations dans template

Template :

<h1>Angular Router<h1>

<nav>

<a routerLink=’’/’’>Acueil<a/>

<a routerLink=’’/about’’>Abaut</a>

</nav>

**<router-outlet><router-outlet>** Où on va afficher notre contenu

----------------------------------------------------------------------

Dans mon fichier app on crée mon router

Il faut importe mon router

* Je copie la commande

import { RouterModule, [Routes](https://angular.io/api/router/Routes) } from '@angular/router';

Dans mon fichier app on crée mon router

**2.-**

import { RouterModule, Routes } from '@angular/router';

**3.-**

imports: [

RouterModule.forRoot(appRoutes),

BrowserModule,

FormsModule,

HttpModule

],

4.-

const appRoutes: Routes = [

{

path:'about',

component: Acueil

},

{

path:'about',

component: About

},

{

path:'about',

component: Contact

}

];

**5.-**

Il faut utiliser les menus avec le routerLink

<a routerLink=’’/’’ class=’’list –group-item’’>Accueil</a>

<a routerLink=’’/about’’ class=’’list –group-item’’>à propos de nous</a>

<a routerLink=’’/ contact’’ class=’’list –group-item’’>Contactl</a>

6.-

<div class = ‘’jombotton demo’’>

<router-outlet><router-outlet>

</div>

App module j’’import

Pass pour envoyer un dossier en Githup

Git status

git add src

git commit -m "mon premier TP"

----------------------------

git checkout -b tp-veronica (pour la branche)

-------------------------------

git push veronica tp-veronica

------------------------------------------

Pour créer les enfants

Dans mon fichier app.modules on modifie dans parents on met un table des enfants

Exemple

----------------------------------------------------

const appRoutes: Routes = [

{

path:'',

redirectTo:'acueil', pathMatch: 'full'

},

{

path:'acueil',

component: AcueilComponentComponent

},

{

path:'about',

component: AboutComponentComponent,

children: [

{

path: 'about-enfant1',

component: AboutEnfant1Component

},

{

path: 'about-enfant2',

component: AboutEnfant2Component

}

]

},

{

path:'contact',

component: ContactComponentComponent

}

];

------------------------------------------------------

Dans mon fichier enfant je fais un balise avec le router

--------------------------------------

<a [routerLink]=’’[‘child-one’]’’>Enfant 1 </a>

<a [routerLink]=’’[‘child-one’]’’>Enfant 1 </a>

-------------------------------------------------

Pour revenir à mon accueil

Je vais à mon fichier app modules

const appRoutes: Routes = [

{

path:'',

redirectTo:'Acueil', pathMach: 'full'

}

{

path:'acueil',

component: AcueilComponentComponent

},

{

path:'about',

component: AboutComponentComponent

},

{

path:'contact',

component: ContactComponentComponent

}

];

Lien relative à partir e un lieu

Absolu on doit tout le lieu

Qu’est ce que angular ?

Diseigne paterne

SPO

Npm

Pipe | serve permet de faire transformations dans notre template

Des Directifs ng fort, ngif, ng

Git status = permet de voir les fichiers modifier et le met en rouge

Git commit = va faire un référencement du fichier (il y a une clé générer pour git pour revenir en arrière)

Git commit –am ‘’test’’ et igale git add

Git checkout master = on switch une branche sur un autre branche

Git checkout –b Veroncia créer une brache

Git pull origen master = pousser et mettre en jour le local

Git stash = prendre tous les modifications et les met dans une boite

Git stash pop = remet les modifications (très importante ! ) il faut faire tout le temps quand on va sortir ou laisse de coder

-------------------------------------------------------------

**ANGULAR – LES SERVICES**

Dans un projet angular doit être Independent

Agular services are singletons

Singletons == évite de répéter notre code tout le temps === Pour appeler ma base de données dans une seule fois

sans le app. Component.ts

private user= new User() ;

Un service est considéré comme un simple module qui est chargé de retourne un objet.

La dclaration

**Ng generate service user** (nom de mon service)

N g s c/s user

Import {lnjectable} from’@angular/core ;

@lnjectable()

Export class UserService{

construction()}

getUser(){}

}

**Dans mon fichier app.components**

import { Component } from '@angular/core'; === import de tous les utilisateurs

@Component({

selector: 'app-root',

templateUrl: './app.component.html',

styleUrls: ['./app.component.css']

})

export class AppComponent {

title = '';

private user :User[] ;

constructor[private us : UserService]{}

}

ngOninit(){

this.user = this.us.getUsers();

}

-------------------------------------------------------

**INJECTER UN SERVICE**

Providers qui va effectivement instancier le service. En fait la notation :

**Providers : [usersService]**

Le service prêt à utiliser

----------------------------------------------------

JSON === fichier de stockage

Le service permet de non pas répéter notre, c’est un design pattern

First name : string

Last name : string

Email : string

-----------------------------------------------------------

On doit créer deux dossiers un component et un service

On va créer un component d’user

**Ng generate component user**

On va créer un service

**Ng generate service user**

import { user} from './user.model';

import {userService } from '../service /users.service';

On crée un fichier à la main pour recouper mon modèle

user.model.ts

export class User{

firstName: string;

lastName: string;

email:string;

}

----------------------------------------------------

Private user : User[] ;

Constructor (private us : UsersService) { }

ngOninit(){

this.user = this.us.getUsers();

}

Dans mon fichier user.service.ts on fait une table de mon fichier modele que j’ai créé à la main

import { Injectable } from '@angular/core';

Import {user} from '../user/user.model' ;

@Injectable()

export class UserService {

User : any ;

getUser(){

this.user = [

{

firstName:"Veronica",

lastName:"Ochoa",

email:"veronikot11@yahoo.fr"

},

{

firstName:"Moussa",

lastName:"Camara",

email:"moussa@iknsa.fr"

},

] ;

Return this.users ;

}

}

----------------------------------------------------------

Dans mon fichier user.component.html

html

<tbody>

<tr \*ngFor=’’let user of users’’>

<td

CRUD = Create read Update Delete

User.modele

ANGULAR – http –

Hypertext transfer protocol

Angula rets côte serveur

Dialoguer avec le serveur

Les **API**  proposant des architectures REST sont de plus en plus présentes ; Un de nos précédents billets traite de l’implémentation d’une telle architecture cote serveur (Zend Framework 2). Aujourd’hui nous allons voir comment il est possible, en utilisant Agular JS de communiquer avec des API.

**API REST :**

* Sans état
* Cachable
* Oriente client-serveur
* Mise en cache
* Avec une interface , etc.

**REST** Representational state transfer : est un style d’architecture pour les systèmes hypermédi distribués ….

**GET** envoyer de requêtes

**POST** récupérer

GIT créer branches :

Git remote –v

Git init

Git remote add veronica (url page Github)

Git remote –v

Git status

Git add –A

Git commit –m "tets"

*git checkout -b tp-php-veronica (pour la branche)*

git push tp-php-Veronica (peut être origin) tp-php-veronica

**(ou git push origin tp-php-veronica)**