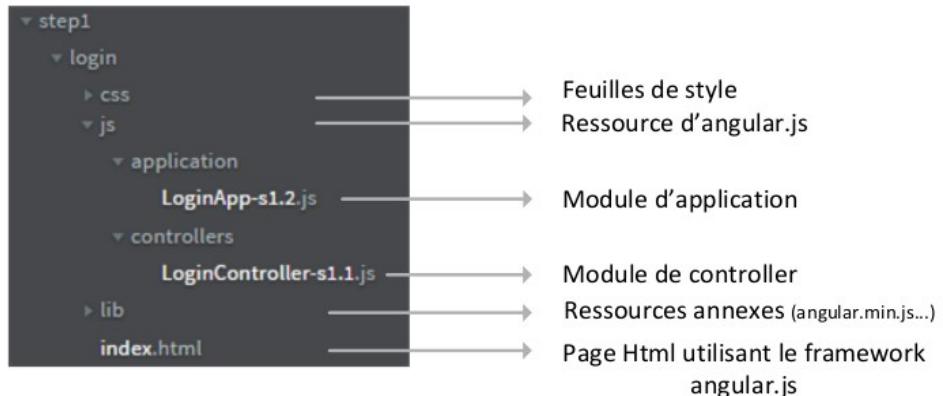


3. STEP 1 : Prise en main de concept Angular.js

3.1. Crédation d'un formulaire simple de login

3.1.1/ Crédation de la structure du projet

3.1.1. L'objectif est de créer une structure de programmation comme suit :



3.1.2/ L'organisation en package → rangement du projet

→ mieux s'y retrouver

3.1.3/ Le module Application permet d'accéder au module Controller.

Le module **Controller** permet d'alimenter le scope du module (initialisation du scope et association de comportement du scope).

3.2. Crédier index.html

3.2.1/ On crée index.html

```
<!DOCTYPE html>
<html lang="en" ng-app="loginApp" ng-controller="loginCtrl">
  <head>
    <title>Watcher Login</title>
    <!-- CSS -->
    <link rel="stylesheet" href="css/bootstrap.min.css">
  </head>
  <body>
    <!-- Top content -->
    <div class="top-content">
      <div class="inner-bg">
        <div class="container">
          <div>
            <div class="row">
              <div class="col-sm-6 col-sm-offset-3 form-box">
                <div class="form-top">
                  <div class="form-top-left">
                    <h3>Login to our site</h3>
                    <p>Enter your username and password to log on:</p>
                  </div>
                  <div class="form-top-right">
                    <i class="fa fa-key"></i>
                  </div>
                </div>
                <div class="form-bottom">
                  <form role="form" action="" method="post" class="login-form">
                    <div class="form-group">
                      <label class="sr-only" for="form-username">Username</label>
                      <input type="text" name="form-username" placeholder="Username..." class="form-username form-control" id="form-username" ng-model="user.login">
                    </div>
                    <div class="form-group">
                      <label class="sr-only" for="form-password">Password</label>
                      <input type="password" name="form-password" placeholder="Password..." class="form-password form-control" id="form-password" ng-model="user.pwd">
                    </div>
                    <button type="button" class="btn" ng-click="logAuth()">Let's go!</button>
                  </form>
                </div>
              </div>
            </div>
          </div>
        </div>
      </div>
    </div>
    <!-- Javascript -->
    <script src="lib/jquery-1.11.1.min.js"></script>
    <script src="lib/bootstrap.min.js"></script>
    <script src="lib/angular.min.js"></script>
    <script src="js/controllers/LoginController-s1.1.js"></script>
  </body></html>
```

Soient les directives suivantes :

- **ng-app** initialise une application AngularJS ;
- **ng-controller** lie la valeur entrée dans la balise HTML (input, select, textarea) aux données de l'application ;
- **ng-model** peut fournir le type de validation aux données d'application (nombre, email, required), le statut aux données d'application (invalid, dirty, touched, error), les classes CSS pour des éléments HTML et lier des éléments HTML à des formulaire (forms).

3.3. Créer les modules applications et contrôleur

On crée LoginApp-s1.2.js :

Fichier : LoginApp-s1.2.js

```
//Creation of an application not needed to bind it to a global variable
angular.module('loginApp', []);
```

On crée aussi LoginApp-s1.2.js :

Fichier : LoginApp-s1.2.js

```
angular.module('loginApp').controller('loginCtrl', loginCrtFnt);

loginCrtFnt.$inject=['$scope','$log'];

function loginCrtFnt($scope, $log){
    $scope.logAuth = function() {
        $log.info('user login', $scope.user.login);
        $log.info('user pwd', $scope.user.pwd);
    };
}
```

3.3.1/ La commande **loginCrtFnt.\$inject=['\$scope','\$log']**; sert à injecter le scope et le service log dans la fonction.

3.3.2/ **\$scope** est la zone mémoire du controller ainsi que de ses descendants : container d'objets et de données de l'application.

3.3.3/ On peut **tester cette application** en utilisant l'outil de développement du Web Browser. Il suffit de regarder dans la console pour vérifier le fonctionnement de l'application.

3.4. Modification du contrôleur

3.4.1/ On complète le controller afin d'avoir une méthode logAuthObject effectuant le même résultat que logAuth .

/ASI-2/step1/login/js/controllers/LoginController-s1.1.js (Angular) - Sublime Text 2 (UNREGISTERED)

```
1 angular.module('loginApp',[]).controller('loginCtrl',loginCrtFnt);
2 loginCrtFnt.$inject=['$scope','$log'];
3 function loginCrtFnt($scope, $log){
4     $scope.logAuth = function() { // Fonction sans paramètres qui s'appuie sur le scope
5         $log.info('user login:', $scope.user.login);
6         $log.info('user pwd:', $scope.user.pwd);
7     };
8     $scope.logAuthObject = function(user) { // Fonction avec paramètre qui lie ce dernier
9         $log.info('user object login:', user.login);
10        $log.info('user object pwd:', user.pwd);
11    };
12 }
13
```

3.4.2/ On modifie index.html de façon à déclencher la fonction logAuthObject du contrôleur.

```
roller-s1.1.js * index.html *
<div class="form-top-left">
  <h3>Login to our site</h3>
  <p>Enter your username and password to log on:</p>
</div>
<div class="form-top-right">
  <i class="fa fa-key"></i>
</div>
<div class="form-bottom">
  <form role="form" action="" method="post" class="login-form">
    <div class="form-group">
      <label class="sr-only" for="form-username">Username</label>
      <input type="text" name="form-username" placeholder="Username..." class="form-username form-control" id="form-username" ng-model="user.login">
    </div>
    <div class="form-group">
      <label class="sr-only" for="form-password">Password</label>
      <input type="password" name="form-password" placeholder="Password..." class="form-password form-control" id="form-password" ng-model="user.pwd">
    </div>
    <button type="button" class="btn" ng-click="logAuth();logAuthObject(user)">Let's go!</button>
    <div>Login :{{user.login}}</div>
    <div>password:{{user.pwd}}</div>
  </form>
</div>
</div>
</div>
<!-- Javascript -->
<script src="../../lib/jquery/jquery-1.11.3.min.js"></script>
<script src="../../lib/bootstrap/js/bootstrap.min.js"></script>
<script src="../../lib/angular/angular.min.js"></script>
<script src="js/controllers/LoginController-s1.1.js"></script>
```

3.4.3/ Quelle différence d'usage il y a-t-il entre logAuthObject et logAuth ?

logAuth est une fonction sans paramètre qui s'appuie sur le scope pour récupérer les données. Tandis que **logAuthObject** est une fonction avec un paramètre et qui affiche les propriétés de l'objet passé en paramètre.

3.5. Modification de index.html

3.5.1/ Modifier le fichier index.html de façon à afficher le login et le pwd saisie (binding)

```
<button type="button" class="btn" ng-click="listUser()>list user!</button>

</br>
Login : {{user.login}}
</br>
Pwd : {{user.pwd}}
</div>
</div>
</div>
```

4. STEP 2 : Réalisation du Canvas d'application, utilisation des services

4.1. Création d'un module de service

4.1.1/ L'objectif est de créer une structure de programmation comme suit :



4.1.2. Créer le module de service AuthService-s2.1.js comme suit :

The screenshot shows a Sublime Text 2 window with multiple tabs open. The active tab is 'AuthService-s2.1.js' (ASI-2). The code in the tab is as follows:

```
1 angular.module('authService', []).service('auth', ['$log', '$http', '$q', function authFnc($log,$http,$q) {
2     var userMap={};
3     userMap['jdoe']=['jdoepwd','watcher'];
4     //console.log(userMap['jdoe'][1]);
5     userMap['psmith']= ['psmithpwd','watcher'];
6     userMap['tp']= ['tp','admin'];
7     //console.log("bonjour");
8     var fncContainer={
9         checkUser: checkUser,
10        userList: userList
11    };
12
13     function checkUser(userlogin,userpwd){
14         var login = "false";
15         for (var name in userMap ) {
16             if (name==userlogin && userMap[name][0]==userpwd)
17             {
18                 login="success";
19                 return login;
20             }
21         }
22         login="false";
23         return login;
24     };
25
26     function userList(){
27         for (var name in userMap ) {
28             $log.info('user login',name);
29         }
30     };
31
32     return fncContainer;
33 }]);
```

4.1.3/ `angular.module('authService',[]).service('auth',authFnc);` : on crée un service auth qui dépend du module authService.

4.1.4/ Modifier le fichier LoginApp-s2.1.js de façon à ce que ces modules puissent accéder au service authService

The screenshot shows a Sublime Text 2 window with multiple tabs open. The active tab is 'AdminApp.js'. The code in the tab is as follows:

```
1 angular.module('loginApp', ['authService']);
```

4.1.5/ Modifier le fichier LoginController-s2.1 de façon à ce qu'il puisse utiliser les méthodes du service authService

```
EventController x Index-s3-1.html x AdminApp-s3.1 x index.html x AuthService-s2 x AuthService-s2 x LoginApi
1 angular.module('loginApp').controller('loginCtrl', ['$scope', '$log', '$window', 'auth',
2     function loginCrtFnt($scope, $log, $window, auth){
3         var login='index';
4         $scope.logAuth = function() {
5             $log.info('user login', $scope.user.login);
6             $log.info('user pwd', $scope.user.pwd);
7         };
8         $scope.logAuthObject = function(user) {
9             $log.info('user login', user.login);
10            $log.info('user pwd', user.pwd);
11            // document.write(document.getElementById('form-password').pwd);
12        };
13        $scope.listUser = function() {
14            auth.userList();
15            // document.write(document.getElementById('form-password').pwd);
16        };
17        $scope.checkUser= function(user) {
18            login=auth.checkUser(user.login, user.pwd);
19            if(login=="success") {
20                $window.location.href= "loginSuccess.html";
21            }
22            else {
23                $window.location.href= "index.html";
24            }
25        }
26    }
27 });
28 });
29 });
30 });
31 });
32 });
33 });
34 }
```

4.1.6/ Afficher la liste des utilisateurs à l'aide du service \$log :

Dans index.html :

```
34         </div>
35         <button type="submit" class="btn" ng-click="checkUser(user)">Lets go!</button>
36     </form>
37
38     <button type="button" class="btn" ng-click="listUser()">list user!</button>
39
40     <br>
41     Login : {{user.login}}
42     <br>
43     Pwd : {{user.pwd}}
44     </div>
45     </div>
46     </div>
47     </div>
48   </div>
49 }
```

Dans LoginController-s2.1.js :

```
EventC ✘ index-s ✘ Admin ✘ index.i ✘ AuthSe ✘ AuthSe ✘ LoginA ✘ LoginC ● LoginC ✘ admin
1 angular.module('loginApp').controller('loginCtrl', ['$scope', '$log', '$window', 'auth',
2   function loginCrtFnt($scope, $log, $window, auth){
3     var login='index';
4     $scope.logAuth = function() {
5       $log.info('user login', $scope.user.login);
6       $log.info('user pwd', $scope.user.pwd);
7     };
8
9     $scope.logAuthObject = function(user) {
10
11       $log.info('user login',user.login);
12       $log.info('user pwd', user.pwd);
13       // document.write(document.getElementById('form-password').pwd);
14
15     };
16     $scope.listUser = function() {
17       auth.userList();
18       // document.write(document.getElementById('form-password').pwd);
19     };
20     $scope.checkUser= function(user) {
21       login=auth.checkUser(user.login, user.pwd);
22       if(login=="success") {
23         $window.location.href= "loginSuccess.html";
24       }
25       else {
26         $window.location.href= "index.html";
27       }
28
29
30   }
31
32  }]);
33
```

4.1.7/ Modifier index.html afin de valider le couple login / Pwd entrée par l'utilisateur

```
<div class="form-bottom">
  <form role="form" method="post" class="login-form">
    <div class="form-group">
      <label class="sr-only" for="form-username">Username</label>
      <input type="text" name="form-username" placeholder="Username..." class="form-username form-control" id="form-username" ng-model="user.login">
    </div>
    <div class="form-group">
      <label class="sr-only" for="form-password">Password</label>
      <input type="password" name="form-password" placeholder="Password..." class="form-password form-control" id="form-password" ng-model="user.pwd">
    </div>
    <button type="submit" class="btn" ng-click="checkUser(user)">Lets go!</button>
  </form>

  <button type="button" class="btn" ng-click="listUser()">list user!</button>

  <br>
  Login : {{user.login}}
  <br>
  Pwd : {{user.pwd}}
</div>
</div>
```

4.1.8. En cas de success d'une authentification rediriger la page courante de l'utilisateur vers loginSuccess.html (utiliser le service \$window)

Dans LoginController-s2.1.js :

```
17      auth.authenticate();
18      // document.write(document.getElementById('form-password').pwd)
19  };
20  $scope.checkUser= function(user) {
21      login=auth.checkUser(user.login, user.pwd);
22      if(login=="success") {
23          $window.location.href= "loginSuccess.html";
24      }
25      else {
26          $window.location.href= "index.html";
27      }
28
29
30  }
31
32  }],
33  {});
```

4.2. Usage de renvoie d'information différée

4.2.1/ Copier le contenu de AuthService-s2.1. js dans AuthService-s2.2. js

4.2.2/ Modifier AuthService-s2.2.js afin de renvoyer une réponse dans une délais de 3s et des informations supplémentaires sur l'utilisateurs (login, validAuth, rôle) :

```
EventC x index-s x Admin x index.l x AuthSe x AuthSe x LoginA x LoginC ● LoginC x admin x loginSt x wa
1 angular.module('authService', []).service('auth', ['$log','$http','$q', function authFnc($log,$http,$q) {
2     var userMap={};
3     userMap['jdoe']=['jdoepwd','watcher'];
4     //console.log(userMap['jdoe'][1]);
5     userMap['psmith']= ['psmithpwd','watcher'];
6     userMap['tp']= ['tp','admin'];
7     //console.log("bonjour");
8     var fncContainer={
9         localAuthAsk:localAuthAsk,
10        checkUser: checkUser,
11        userList: userList,
12        authAsk: authAsk
13    };
14    ;
15
16    function localAuthAsk(login,pwd){
17        var deferred = $q.defer();
18
19        var interval = setInterval(function(login,pwd,deferred){
20            //console.log(userMap[login]);
21            if( userMap[login][0]==pwd){
22
23                if(userMap[login][1] == "watcher") {
24                    deferred.resolve('SuccessWatcher');
25                }
26                if(userMap[login][1] == "admin") {
27                    deferred.resolve('SuccessAdmin');
28                }
29            }
30            else {
31
32                deferred.reject('Error');
33            }
34
35            //console.log(this);
36
37            clearInterval(interval);
38        },3000,login,pwd,deferred);
39        console.log(deferred.promise);
40        return deferred.promise;
41    };
42    ;
43
44    //mettre tout le dossier dans wamp!!!
45    function authAsk(login,pwd){
```

4.2.3/ le service \$q permet de gérer des fonctionnalités de façon asynchrone. La méthode then() valide promise.

4.2.4/ Cela permettra de vérifier que les identifiant et mot de passe soient bons avant de rediriger vers la page.

→ Attend le serveur qui est aussi en asynchrone

4.2.5/ Copier le contrôleur LoginController-s2.1.js dans LoginController-s2.2.js

4.2.6/ Modifier LoginController-s2.2.js de façon à utiliser la fonction différée d'information et à rediriger l'utilisateur vers admin.html si son role est admin vers watcher sinon.

```
};

//mettre tout le dossier dans wamp!!!
function authAsk(login,pwd){
    var deferred = $q.defer();
    $http.post('http://localhost/fakeAuthWatcher/',[{'login':login,'pwd':pwd}].
    success(function(data, status, headers, config) {

        if(userMap[login][1] == "watcher") {
            deferred.resolve('SuccessWatcher');
        }
        if(userMap[login][1] == "admin") {
            deferred.resolve('SuccessAdmin');
        }

    }).
    error(function(data, status, headers, config) {
    //TODO

        deferred.reject('Error');

    });
    return deferred.promise;
};
```

4.3. Mise en œuvre d'une auth via http

4.3.1/ Copier le contrôleur LoginController-s2.2.js dans LoginController-s2.3.js

4.3.2/Modifier le service afin d'effectuer une requête http vers le serveur pour l'authentification.

```
//mettre tout le dossier dans wamp!!!
function authAsk(login,pwd){
    var deferred = $q.defer();
    $http.post('http://localhost/fakeAuthWatcher/',[{'login':login,'pwd':pwd}].
    success(function(data, status, headers, config) {

        if(userMap[login][1] == "watcher") {
            deferred.resolve('SuccessWatcher');
        }
        if(userMap[login][1] == "admin") {
            deferred.resolve('SuccessAdmin');
        }

    }).
    error(function(data, status, headers, config) {
    //TODO

        deferred.reject('Error');

    });
    return deferred.promise;
};
```

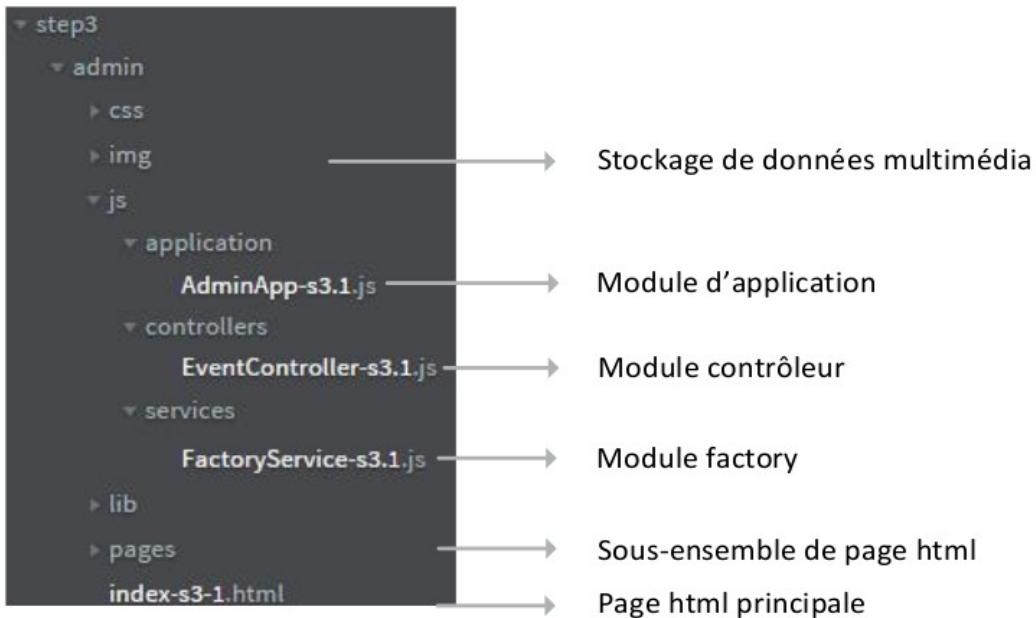
4.3.3/ Pourquoi utilise-t-on le service \$q avec cette requête http d'auth ?

- utilise des promises
- on attend qu'une valeur arrive avant d'effectuer ce qu'il y a avant le promises
- traitement asynchrone
- Utile pour attendre la réponse du serveur

5. STEP 3 : Edition et présentation des informations à l'aide du double binding

5.1. Création et affichage d'une liste de données

5.1.1/ L'objectif est de créer une structure de programmation comme suit :



5.1.2/ Créer le fichier `index-s3-1.html` comme suit :

```
EventC * index-s3-1 ● AdminC * IndexC * AuthSe * AuthSe * LoginA * LoginC * LoginC * adminC * loginSe * wa
1  <!DOCTYPE html>
2  <html lang="en" ng-app="adminApp" ng-controller="eventCtrl">
3  <head>
4  <title>Bootstrap 101 Template</title>
5  <!-- Bootstrap -->
6  <link href="css/bootstrap.min.css" rel="stylesheet">
7  <!-- <link href="css/js.css" rel="stylesheet"> -->
8  </head>
9  <body>
10 <div class="container-fluid fullScreenHeight zeroRef" >
11  <div class="row container-fluid fullScreenHeight ">          <!-- ***** LEFT PANEL ***** -->
12  <div class="col-xs-2 col-sm-2 col-md-2 col-lg-2 zeroRef fullScreenHeight divOverflow" >
13  <div class="col-xs-12 col-sm-12 col-md-12 col-lg-12">
14  <div class="btn-group" role="group" aria-label="...>
15  <button type="button" class="btn btn-default">
16  <span class="glyphicon glyphicon-edit" aria-hidden="true"></span>
17  </button>
18 </div>
19 <div class="btn-group" role="group" aria-label="...>
20  <button type="button" class="btn btn-default" ng-click="savePres()">
21  <span class="glyphicon glyphicon-floppy-disk" aria-hidden="true"></span>
22  </button>
23  <button type="button" class="btn btn-default" ng-click="newSlide()">
24  <span class="glyphicon glyphicon-plus" aria-hidden="true"></span>
25  </button>
26  <button type="button" class="btn btn-default">
27  <span class="glyphicon glyphicon-remove" aria-hidden="true"></span>
28  </button>
29 </div>
30 <div class="form-group">
31  <label for="currentPresTitle">Title</label>
32  <input type="text" ng-model="currentPresentation.title" class="form-control" id="currentPresTitle">
33 </div>
34 </div>
35 </div>
36 </div>
```

5.2. Création d'un factory

5.2.1/ Créer le fichier FactoryService-s3.1.js comme suit :

```
var contentType={}
contentType.IMG_URL="IMG_URL";
contentType.IMG_B64="IMG_B64";

angular.module('factoryServices', []).factory('factory',factoryFnc);

function factoryFnc(){
    var factory = {
        generateUUID: generateUUID,
        contentCreation: contentCreation,
        slidCreation: slidCreation,
        presentationCreation: presentationCreation,
        mapToArray: mapToArray

    };

    // http://stackoverflow.com/questions/105034/create-guid-uuid-in-javascript
    function generateUUID(){
        var d = new Date().getTime();
        var uuid = 'xxxxxxxx-xxxx-4xxx-yxxx-xxxxxxxxxxxx'.replace(/[xy]/g, function(c) {
            var r = (d + Math.random()*16)%16 | 0;
            d = Math.floor(d/16);
            return (c=='x' ? r : (r&0x3|0x8)).toString(16);
        });
        return uuid;
    }

    function contentCreation(title,type,src){
        // TODO
    };

    function slidCreation(title,txt){
        // TODO
    };

    function presentationCreation(title,description){
        // TODO
    };

    function mapToArray(map){
        contentArray=[];
        for(key in map){
            contentArray.push(map[key]);
        }
        return contentArray;
    };

    return factory;
};
```

```
function presentationCreation(title,description){
    // TODO
};

function mapToArray(map){
    contentArray=[];
    for(key in map){
        contentArray.push(map[key]);
    }
    return contentArray;
};

return factory;
};
```

Ainsi on a :

```
function contentCreation(title,type,src){
    var content={};
    content['id']=contentId;
    content['title']=title;
    content['type']=type;
    content['src']=src;
    contentId++;
    return content;
};

function slidCreation(title,txt){

    var slid={};
    slid['id']=slidId;
    slid['title']=title;
    slid['txt']=txt;
    slid.contentMap={};
    slidId++;
    return slid;
};

function presentationCreation(title,description){

    var presentation={};
    presentation['id']=presentationId;
    presentation['title']=title;
    presentation['description']=description;
    presentation.slidArray={};
    presentationId++;
    return presentation;
};
```

5.2.2/ Créer le fichier AdminApp-s3.1.js , module d'application de façon à ce qu'il puisse utiliser la factory.



```
EventController-s3.1.js * index-s3-1.html * AdminApp-s3.1
1 angular.module('adminApp', ['factoryServices']);
```

5.2.3/ Créer le fichier EventController-s3.1.js, module contrôleur, de façon à ce qu'il puisse utiliser la factory.

```
EventController-s3.1.js * index-s3-1.html * AdminApp-s3.1.js *
```

```
1 angular.module('adminApp').controller('eventCtrl', ['$scope','$log','$window', 'factory',
2   function loginCrtFnt($scope, $log, $window, factory){
3
4
5   $scope.currentPresentation=factory.presentationCreation("Star wars","presentation episode 7");
6
7   //console.log($scope.currentSlid);
8   $scope.newSlide=function(titleSlid,txtSlid,contentTitle,contentType,contentSrc){
9     currentSlid = factory.slidCreation(titleSlid,txtSlid);
10    currentSlid.contentMap[titleSlid] = factory.contentCreation(contentTitle,contentType,contentSrc);
11    $scope.currentPresentation.slidArray[titleSlid] = currentSlid;
12  };
13  $scope.newSlide("Dark Vador", "Bonjour je m'appelle Dark Vador","description dark vador",
14    "Dark Vador est une machine-humaine","../admin/img/Dark-Vador.jpg");
15  $scope.newSlide("Luke", "Bonjour je m'appelle Luke","description dark vador","Luke est un jedi",
16    "../admin/img/Luke.jpg");
17
18  // function displaySlid(currentPresentation){
19  //   for (var id in currentPresentation)
20  //   {
21  //     document.innerHTML
22  //   }
23  // }
24  console.log($scope.currentPresentation);
25
26 });
27
28
29
```

5.3. Création et affichages des objets

5.3.1/ Modifier EventController-s3.1.js de façon à créer un objet « présentation » associé à \$scope.currentPresentation

```
2
3
4
5   $scope.currentPresentation=factory.presentationCreation("Star wars","presentation episode 7");
6
```

5.3.2/ Modifier EventController-s3.1.js et créer une méthode rattachée au scope
\$scope.newSlide=function(){ ... }.

Cette méthode va ajouter un objet slid à l'objet \$scope.currentPresentation.
Le nouvel objet slid devra contenir un objet content alors créé.

```
//console.log($scope.currentSlid);
$scope.newSlide=function(titleSlid,txtSlid,contentTitle,contentType,contentSrc){
  currentSlid = factory.slidCreation(titleSlid,txtSlid);
  currentSlid.contentMap[titleSlid] = factory.contentCreation(contentTitle,contentType,contentSrc);
  $scope.currentPresentation.slidArray[titleSlid] = currentSlid;
};
```

5.3.3/ Modifier le fichier index-s3-1.js de façon à afficher la liste des objets slid de \$scope.currentPresentation

```
<input type="text" ng-model="currentPresentation.slidArray['firstSlid']" class="form-control"
      id="currentPresTitle" -->
</div>
<div ng-repeat="slid in currentPresentation.slidArray">
    {{slid.title}}
    <div ng-repeat="content in slid.contentMap">
        
    </div>
</div>
</div>
```

5.4. Synchronisation des zones d'édition et d'affichage

5.4.1/ Modifier index-s3-1.html et ajouter les éléments suivants :

```
EventC x index-s x Admin x index.i x AuthSe x AuthSe x LoginA x LoginC ● \ LoginC x admin. x loginSt x wa
27     | <span class="glyphicon glyphiconremove" aria-hidden="true"></span>
28     | </button>
29   </div>
30   <div class="form-group">
31     <label for="currentPresTitle">Title</label>
32     <input type="text" ng-model="currentPresentation.title" class="form-control" id="currentPresTitle">
33     <!-- <label for="currentPresTitle">list of slid</label>
34     <input type="text" ng-model="currentPresentation.slidArray['firstSlid']" class="form-control"
            id="currentPresTitle" -->
35   </div>
36   <div ng-repeat="slid in currentPresentation.slidArray">
37     {{slid.title}}
38     <div ng-repeat="content in slid.contentMap">
39       
40     </div>
41   </div>
42   </div>
43   </div>
44   </div>
45   | </div>
46   <script src="lib/jquery-1.11.1.min.js"></script>
47   <script src="lib/angular.min.js"></script>
48   <script src="js/services/FactoryService-s3.1.js"></script>
49   <script src="js/application/AdminApp-s3.1.js"></script>
50   <script src="js/controllers/EventController-s3.1.js"></script>
51   </body>
52 </html> |
```

5.4.2/ Modifier EventController-s3.1.js et les éléments comme suit :

```
$scope.selectCurrentSlide=function(slide){
    $scope.currentSlide=slide;
}

$scope.isSlidContentEmpty=function(slid){
    if(slid.contentMap[1]== undefined){
        return true;
    }
    return false
}
```

5.4.3/ Modifier index-s3-1.html de façon à déclencher la fonction selectCurrentSlid lors d'un clic sur l'image.

```
<!-- <label for="currentPresTitle">list of slid</label>
    <input type="text" ng-model="currentPresentation.slidArray['firstSlid']" class="form-control"
        id="currentPresTitle"> -->
</div>
<div ng-repeat="slid in currentPresentation.slidArray">
    {{slid.title}}
    <div ng-repeat="content in slid.contentMap">
        
    </div>
</div>
</div>
<div class="col-xs-7 col-sm-7 col-md-7 col-lg-7 zeroReffullScreenHeight" >
    <div class="col-xs-12 col-sm-12 col-md-12 col-lg-12">
    </div>
    <div class="col-xs-12 col-sm-12 col-md-12 col-lg-12 fullScreenHeight">
        <div class="form-group" >
            Title: <input type="text" ng-model="currentSlide.title" class="form-control"> <br>
            Description: <input type="text" ng-model="currentSlide.txt" style="width:80%; height:200px" class="form-control">
        </div>
        
    </div>
</div>
</div>
<script src="lib/jquery-1.11.1.min.js"></script>
<script src="lib/angular.min.js"></script>
<script src="js/services/FactoryService-s3.1.js"></script>
<script src="js/application/AdminApp-s3.1.js"></script>
<script src="js/controllers/EventController-s3.1.js"></script>
</body>
</html>
```

5.4.4/ Modifier index-s3-1.html de façon à afficher le titre et la description du slid courant et affichage l'image du premier contenu du slid courant.

```
36     <!-- currentPresTitle -->
37     </div>
38     <div ng-repeat="slid in currentPresentation.slidArray">
39         {{slid.title}}
40         <div ng-repeat="content in slid.contentMap">
41             
42         </div>
43     </div>
44     </div>
45     <div class="col-xs-7 col-sm-7 col-md-7 col-lg-7 zeroReffullScreenHeight" >
46         <div class="col-xs-12 col-sm-12 col-md-12 col-lg-12">
47         </div>
48         <div class="col-xs-12 col-sm-12 col-md-12 col-lg-12 fullScreenHeight">
49             <div class="form-group" >
50                 Title: <input type="text" ng-model="currentSlide.title" class="form-control"> <br>
51                 Description: <input type="text" ng-model="currentSlide.txt" style="width:80%; height:200px" class="form-control">
52             </div>
53
54             
56
57         </div>
58     </div>
59
```

5.4.5/ Modifier index-s3-1.html de façon à afficher un message si le slide courant n'a pas de contenu.

Dans EventController-s3.1.js :

```
1 angular.module('adminApp').controller('eventCtrl', ['$scope','$log','$window', 'factory',
2   function loginCrtFnt($scope, $log, $window, factory){
3
4
5
6   $scope.currentPresentation=factory.presentationCreation("Star wars","presentation episode 7");
7
8   //console.log($scope.currentSlide);
9   $scope.newSlide=function(titleSlid,txtSlid,contentTitle,contentType,contentSrc){
10     currentSlid = factory.slideCreation(titleSlid,txtSlid);
11     currentSlid.contentMap[titleSlid] = factory.contentCreation(contentTitle,contentType,contentSrc);
12     $scope.currentPresentation.slideArray[titleSlid] = currentSlid;
13   };
14   $scope.newSlide("Dark Vador", "Bonjour je m'appelle Dark Vador","description dark vador","Dark Vador est une machine");
15   $scope.newSlide("Luke", "Bonjour je m'appelle Luke","description Luke","Luke est un jedi","../admin/img/Luke.jpg");
16   $scope.newSlide("Clone", "Bonjour je suis un clone","description Clone","Clone est un soldat","../admin/img/Clone.jpg");
17   console.log($scope.currentPresentation.slideArray);
18
19
20
21   $scope.selectCurrentSlide=function(slide){
22     $scope.currentSlide=slide;
23     console.log($scope.currentSlide);
24   }
25
26   $scope.isSlideContentEmpty=function(slid){
27     if(slid.contentMap[1]== undefined){
28       return true;
29     }
30     return false
31   }
32
33 });
34
35 
```

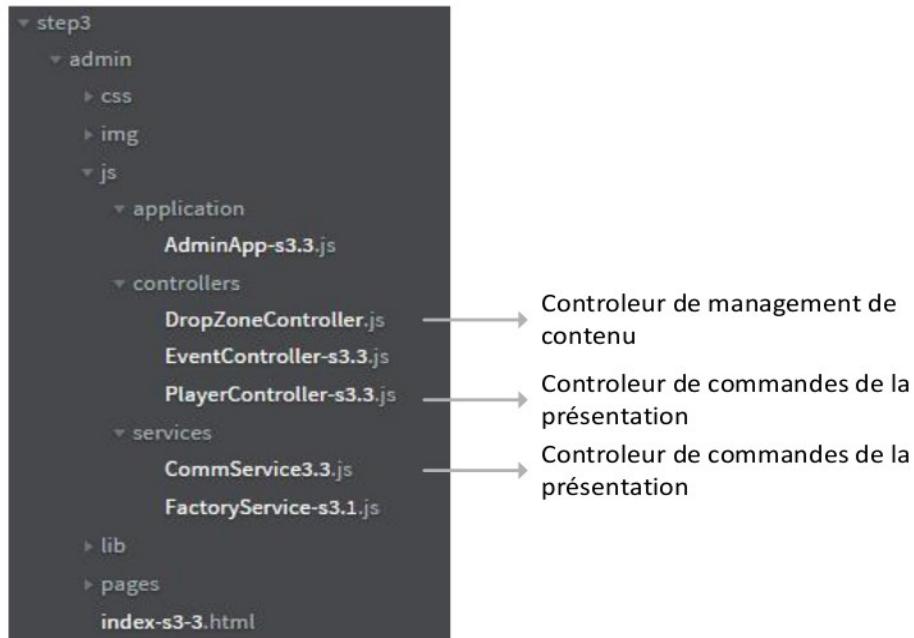
Dans index-s3.html :

A FAIRE !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

6. STEP 4 : Envoi et récupération de contenu extérieur

6.1. Récupération de la structure de programme

6.1.1/ Récupérer l'archive du projet qui se présente comme suit :



6.2. Mise en place du service CommService3-3.js

6.2.1/ Modifier le fichier CommService3.3.js afin de permettre de renvoyer une Map de plusieurs content au bout de 3s (idem pour le cas de la présentation)

```

function loadImages(presName,presID) {
  // TODO
}

function loadPres(presName,presID) {
  // TODO
}
  
```

6.2.2/ Expliquer le code suivant :

```

<div ng-drag="true" ng-drag-data="content" ng-drag-
success="onDragComplete($data,$event)">
  
</div>
  
```

Ce code permet de sélectionner le contenu persistant d'une image le temps d'un déplacement.

6.2.3/ Expliquer le code suivant :

6.2.3.Expliquer le code suivant

```

<div class="col-xs-12 col-sm-12 col-md-12 col-lg-12 contentBlock" ng-
repeat="content in contentMap.array | filter:searchContent | orderBy:
'title'" >
  
```

Ce code permet de trier par ordre alphabétique de titre dans un endroit bien défini sur la page les différentes slides.

6.2.4/ Modifier le code afin de filtre non plus pas titre mais pas type de content.

6.2.5/ Tester votre application

6.2.6/ Afin de lier notre application à un service web, modifier le fichier CommService3.3.js comme suit :

```
function loadImages(presName,presID) {
    var deferred = $q.defer();
    $http.get('/resources_list').
        success(function(data, status, headers, config) {
            deferred.resolve(data);
        }).
        error(function(data, status, headers, config) {
            deferred.reject(status);
            // or server returns response with an error status.
        });
    return deferred.promise;
};

function loadPres(presName,presID){
    var deferred = $q.defer();
    $http.get('/loadPres').
        success(function(data, status, headers, config) {
            deferred.resolve(data);
        }).
        error(function(data, status, headers, config) {
            deferred.reject(status);
            // or server returns response with an error status.
        });
    return deferred.promise;
}
```

6.2.7/ Pourquoi est-il indispensable d'utiliser un promise dans le cas de l'usage du service \$http ?

→ gérer facilement les enchaînements d'opérations asynchrones dans une application.

6.2.8/ A quoi sert la fonction .success ? .error ?

- La promise a une méthode then() qui dispose de deux méthodes .success() et de .error()
- Ces 2 méthodes sont plus pratiques que la méthode then(), car le callback que l'on envoie à ces méthodes reçoit directement 4 paramètres qui décomposent le contenu de la réponse HTTP.
- then() crée et renvoie une nouvelle promise chaînée au retour du callback ; tandis que les méthodes success() et error() ne créent pas de nouvelle promise et renvoient l'objet this sur lequel elles sont appelées, cad la promise créée par \$http().
- On peut faire suivre les appels à success() et error() dans l'ordre que l'on veut.
- success() et error() ne permettent pas d'enchaîner des requêtes HTTP sous la forme d'un chaînage de promises.

(Cf <http://www.frangular.com/2012/12/api-promise-angularjs.html>)

6.3. Mise des actions de contrôle de la présentation

6.3.1. Mise à jour du service Comm

6.3.2/ Modifier le fichier CommService3.3.js en ajoutant les fonctions suivantes comme suit :

```
// Order for watcher clients
comm.io={};
comm.io.socketConnection=function(scope,uuid){
    var socket = io.connect();

    comm.io.uuid=uuid;
    socket.on('connection', function () {
        socket.emit('data_comm',{'id':comm.io.uuid});
    });

    socket.on('newPres', function (socket) {

    });
    socket.on('slidEvent', function (socket) {

    });
    return socket;
}

comm.io.emitPrev=function(socket){
    socket.emit('slidEvent', {'CMD':'PREV'});
}

comm.io.emitNext=function(socket){
    socket.emit('slidEvent', {'CMD':'NEXT'});
}

comm.io.emitStart=function(socket,presUUID){
    socket.emit('slidEvent', {'CMD':'START','PRES_ID':presUUID});
}

comm.io.emitPause=function(socket){
    socket.emit('slidEvent', {'CMD':'PAUSE'});
}

comm.io.emitBegin=function(socket){
    socket.emit('slidEvent', {'CMD':'BEGIN'});
}

comm.io.emitEnd=function(socket){
    socket.emit('slidEvent', {'CMD':'END'});
}
```

6.3.3/ Qu'est ce que soket.io ?

- Une bibliothèque de développement de Node.js
- Permet communication bidirectionnelle en temps réel entre 2 entités
- Utilise WebSockets
- Ne gère pas WebSocket (comme tous les navigateurs)
- Compatible sur un très grand nombre de navigateurs
- Détermine pour chaque client quelle est la méthode de communication temps réel la plus adaptée pour le client (WebSocket, Adobe Flash Socket, AJAX log polling, AJAX multipart streaming, Forever Iframe, JSONP Polling)

6.3.4/ Quelles différences de communication il y a-t-il vis-à-vis de l'usage du service \$http ?

- \$http utilise des requêtes ; tandis que socket.io utilise le protocole Websocket
- Socket.io : API qui supporte plusieurs protocoles
- D'après la spécification HTTP, il est possible qu'il soit nécessaire d'encoder ou de décoder des messages

6.4. Mise en place du contrôleur Payer

6.4.1. Mettre en place le PlayerControlleur-s3.3.js permettant les fonctions suivantes :

- pause
- end
- begin
- backward
- forward
- play
- savePres

6.4.2/ Modifier le fichier index-s3-3.js de façon à lier les boutons de contrôleur de la présentation aux fonctions préalablement créées.