ASI-2
Angular.js
Step By Step

2015

Ce Tp a pour objectif de concevoir une application cross technologie permettant d'appréhender la fonction et l'usage de chaque technologie.
Cette première partie vous permettra de mettre en œuvre l'application Web basé sur un socle Angular.js

Part I : Angular.js

Table des matières

1. Introduction

Ce Tp déroule en 3 étapes comme suit :

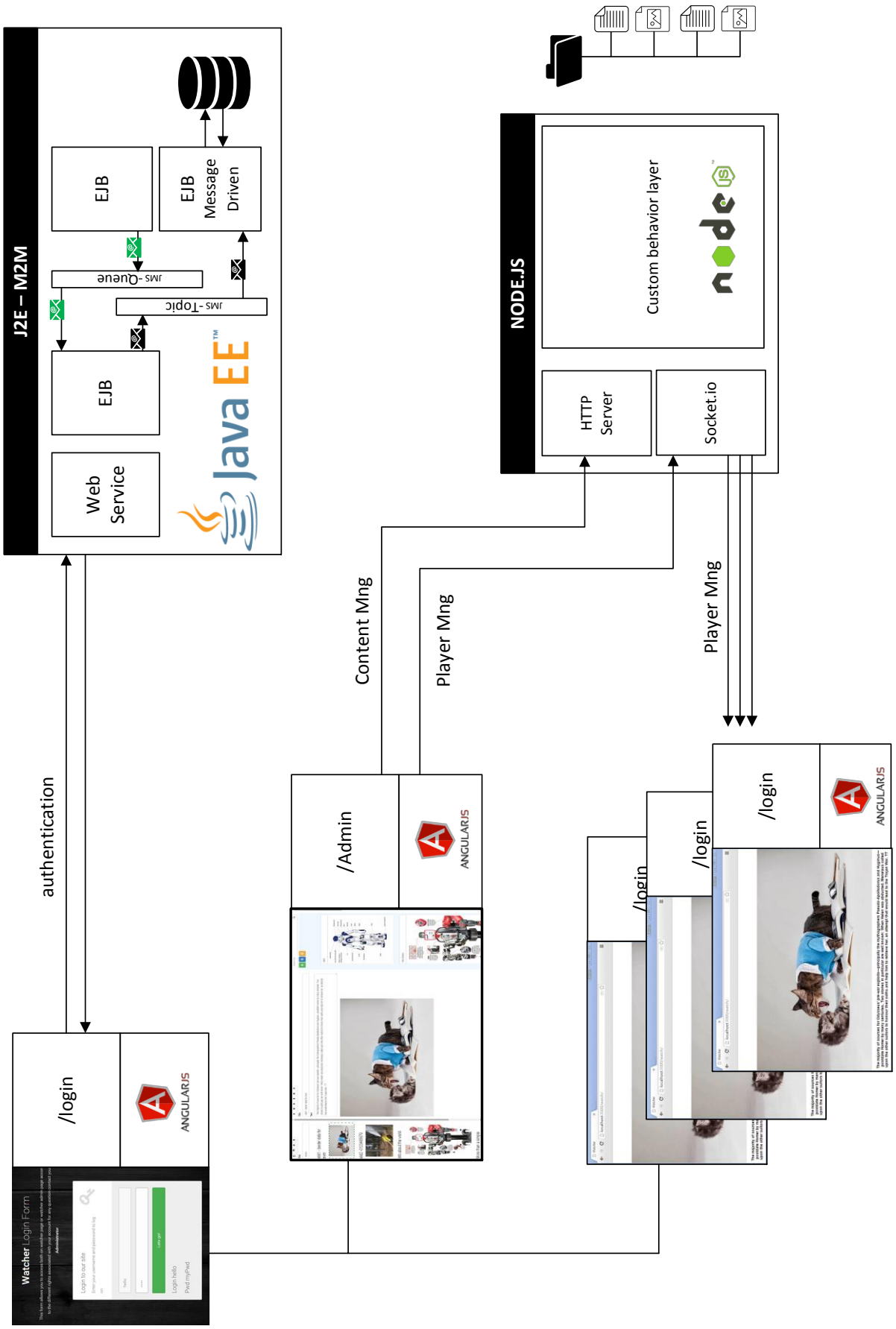
- **Step 1** : prise en main bases de concepts d'Angular.js, des bonnes pratiques, de l'usage des modules applications, contrôleurs
- **Step 2** : Mise en place de l'ensemble des éléments nécessaires à une application en ajoutant les modules services
- **Step 3** : Manipulation de formulaire, mise en œuvre du double binding
- **Step 4** : Finalisation de l'application avec mise en œuvre de plusieurs contrôleurs et services avec appel de services distants.

CHAQUE ETAPE et SOUS ETATE DEVRA ETRE VALIDEE PAR UN ENSEIGNANT AVANT DE PASSER A LA SUIVANTE.

L'évaluation prendra en compte votre compréhension de l'étape, la mise en œuvre et les bonnes pratiques de programmation.

2. Présentation générale

2.1.Présentation du projet dans sa globalité

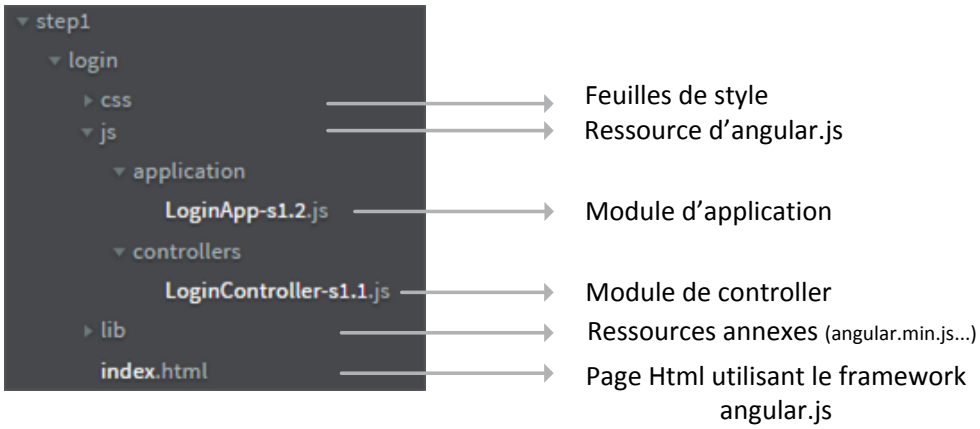


3. STEP 1 : Prise en main de concept Angular.js

Objectif : Prise en main bases de concepts d'Angular.js, des bonnes pratiques, de l'usage des modules applications, contrôleurs

3.1. Création d'un formulaire simple de login

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



3.1.2.A quoi sert l'organisation en Paquage d'un projet ?

3.1.3.Quelles sont les rôles respectifs des modules applications et contrôler ?

3.2. Créer 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()">Lets 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>
```

3.2.1. A quoi sert les directives suivantes:

- ng-app?
- ng-controller?
- ng-model?

3.3.Créer les modules applications et controlleurs :

Fichier : LoginApp-s1.2.js

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

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.A quoi sert la commande loginCrtFnt.$inject=['$scope','$log']; ?

3.3.2. Qu'est ce que $scope ?

3.3.3. Comment peut-on tester cette application ?

FAIRE VALIDER L'ETAT D'AVANCEMENT

3.4. Modification du controlleur:

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);
      };

     $scope.logAuthObject = function(user) {

              // TODO

       };

}
```

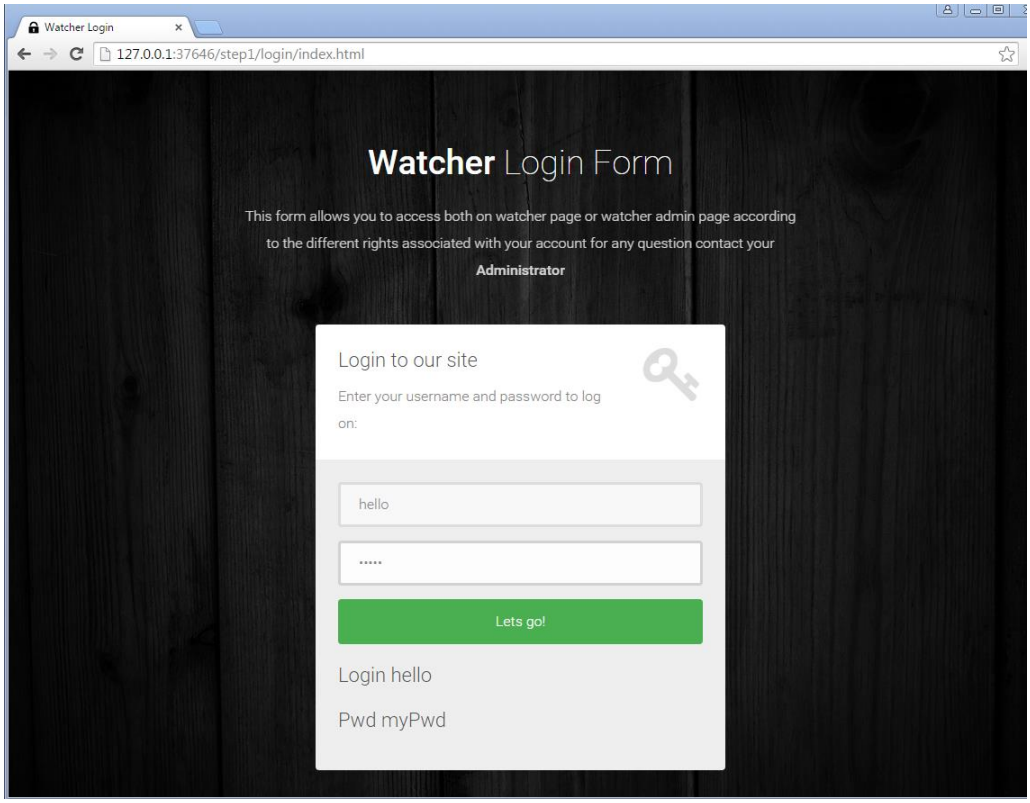
3.4.1. Compléter le controlleur afin d'avoir une méthode logAuthObject effectuant le même résultat que logAuth

3.4.2. Modifier index.html de façon à déclencher la fonction logAuthObject du controlleur.

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

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)



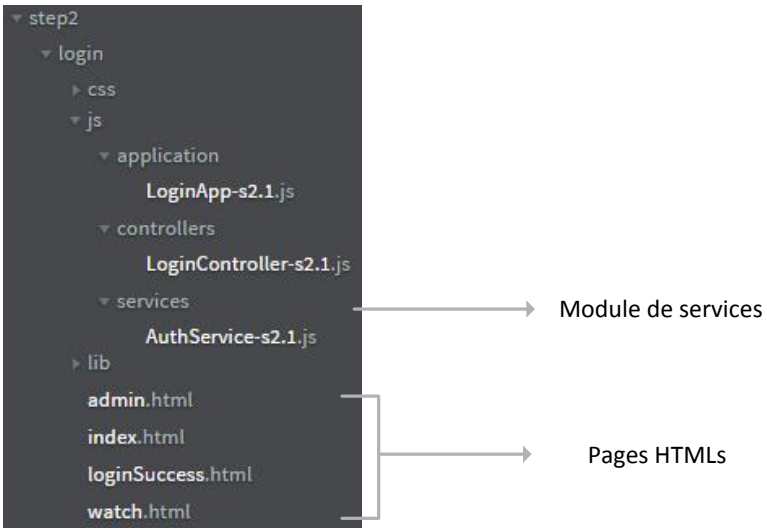
FAIRE VALIDER L'ETAT D'AVANCEMENT

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

Objectif : Mise en place de l'ensemble des éléments nécessaires à une application en ajoutant les modules 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 comme suit

```
angular.module('authService', []).service('auth',authFnc);

function  authFnc() {
    var userMap={};
    userMap['jdoe']='jdoepwd';
    userMap['psmith']='psmithpwd';
    userMap['tp']='tp';

    var fncContainer={
                        checkUser: checkUser,
                        userList: userList
                      };
    function checkUser(userlogin,userpwd){

       // TODO

    };
    function userList(){

        // TODO

    };

    return fncContainer;

}
```

4.1.3. Expliquer la ligne de code suivante

```
angular.module('authService',[]).service('auth',authFnc);
```

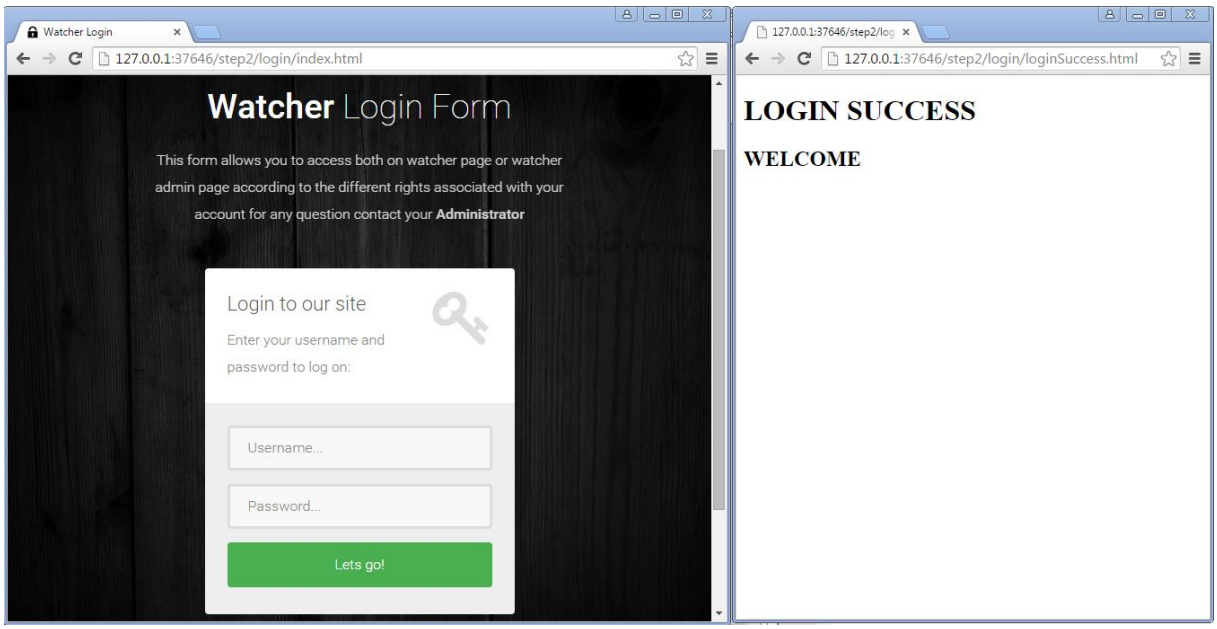
4.1.4. Modifier le fichier LoginApp-s2.1.js de façon à ces que ces modules puissent accèder au service authService

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

4.1.6. Afficher la liste des utilisateurs à l'aide du service $log.

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

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



FAIRE VALIDER L'ETAT D'AVANCEMENT

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, role)

```
angular.module('authService', []).service('auth',authFnc);
authFnc.$inject=['$http','$q'];
function  authFnc($http,$q) {
    var userMap={};
    userMap['jdoe']='jdoepwd';
    userMap['psmith']='psmithpwd';
    userMap['tp']='tp';

    var fncContainer={
                        localAuthAsk:localAuthAsk
                    };

    function localAuthAsk(login,pwd){
        var deferred = $q.defer();
        setInterval(function(login,pwd){
                if( userMap[login]==pwd){
                    //TODO
                }else{
                    //TODO
                }
                clearInterval(this);
            },3000,login,pwd);

        return //TODO;

    }

    return fncContainer;
}
```

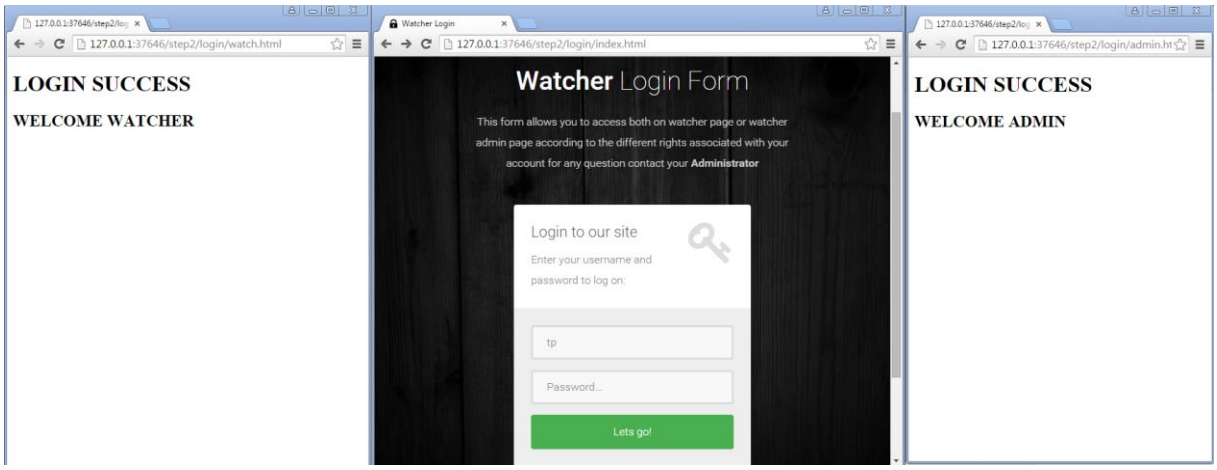
4.2.3. Que permet de faire le service $q ?

4.2.4.Quelles seront les conséquences sur le contrôleur qui voudra exploiter ces données ?

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.

Voir : https://docs.angularjs.org/api/ng/service/$q



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 requète http vers le serveur pour l'authentification.

```
function authAsk(login,pwd){
   var deferred = $q.defer();
     $http.post('/fakeauthwatcher',{'login':login,'pwd':pwd}).
        success(function(data, status, headers, config) {
        //TODO

     }).
     error(function(data, status, headers, config) {
        //TODO
        // or server returns response with an error status.
     });
     return //TODO;
 };
```

4.3.3.Pourquoi utilise-t-on le service $q avec cette requète http d'auth ?

4.3.4. BONUS : Créer un serveur web (J2E/PHP) permettant de renvoyer les informations demandées.

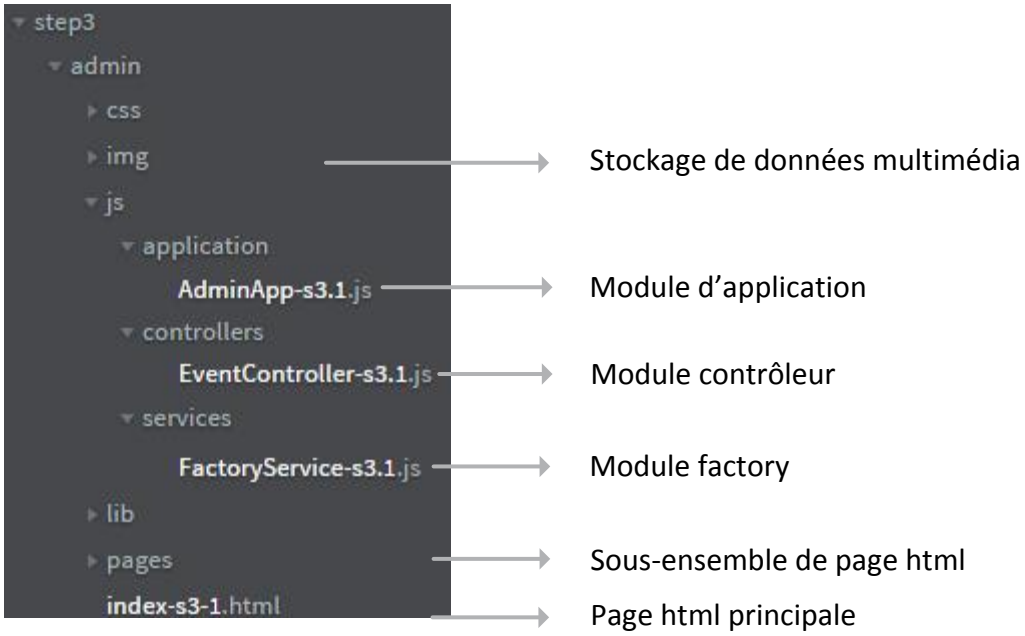
FAIRE VALIDER L'ETAT D'AVANCEMENT

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

Objectif : Manipulation des données, usage des directives de double binding, d'affichage de données multiples, mise en avant de la notion et des périmètres des $scope

5.1.Création et affichage d'une liste de données

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



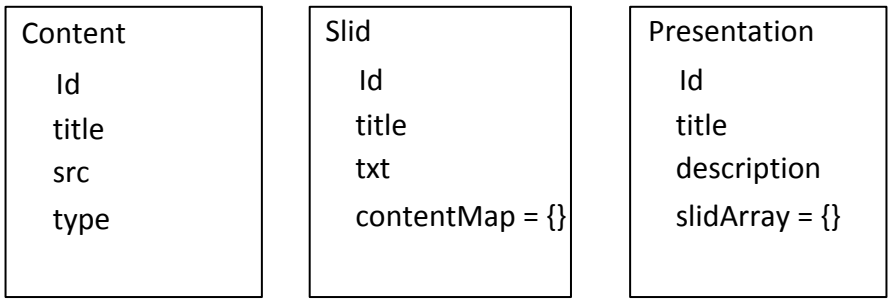
5.1.2.Créer le fichier index-s3-1.html comme suit :

```
<!DOCTYPE html>
<html lang="en" ng-app="adminApp" ng-controller="eventCtrl">
  <head>
    <title>Bootstrap 101 Template</title>

    <!-- Bootstrap -->
    <link href="lib/bootstrap-3.3.5-dist/css/bootstrap.min.css" rel="stylesheet">
    <link href="css/js.css" rel="stylesheet">

  </head>
  <body>
    <div class="container-fluid fullScreenHeight zeroRef" >
    <div class="row container-fluid fullScreenHeight ">

<!-- *************   LEFT PANEL  *************-->
    <div class="col-xs-2 col-sm-2 col-md-2 col-lg-2 zeroRef fullScreenHeight divOverflow"  >
      <div class="col-xs-12 col-sm-12 col-md-12 col-lg-12">
         <div class="btn-group" role="group" aria-label="...">
            <button type="button" class="btn btn-default"><span class="glyphicon glyphicon-
edit" aria-hidden="true"></span></button>
         </div>
         <div class="btn-group" role="group" aria-label="...">
            <button type="button" class="btn btn-default" ng-click="savePres()"><span
class="glyphicon glyphicon-floppy-disk" aria-hidden="true"></span></button>
            <button type="button" class="btn btn-default" ng-click="newSlide()"><span
class="glyphicon glyphicon-plus" aria-hidden="true"></span></button>
            <button type="button" class="btn btn-default"><span class="glyphicon glyphicon-
remove" aria-hidden="true"></span></button>
         </div>
         <div class="form-group">
            <label for="currentPresTitle">Title</label>
            <input type="text" ng-model="currentPresenation.title" class="form-control"
id="currentPresTitle">
         </div>
```

```
      </div>
        // TODO LATER
    </div>

  <script src="lib/js/jquery-1.11.3.min.js"></script>

  <!-- ANGULAR -->
  <script src="lib/js/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>
```

5.2.Création d'un factory

Afin de créer rapidement et d'une façon homogène des objets que nous allons manipuler, nous allons créer un module particulier, factory qui sera appelé à chaque création d'objet. Les objets à créer sont les suivants.



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;
};
```

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

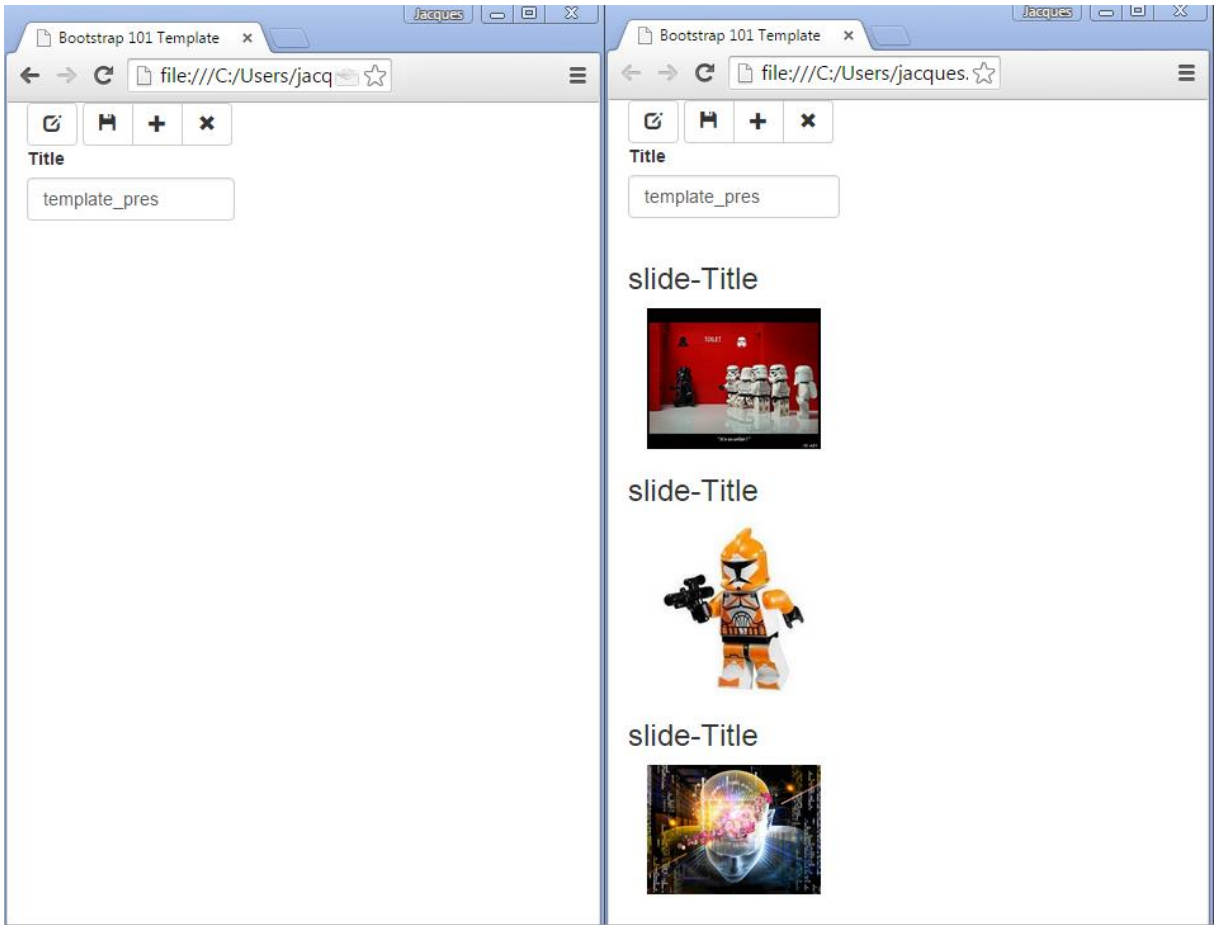
5.2.3. Créer le fichier EventController-s3.1.js, module contrôleur, de façon à ce qu'il puisse utiliser la factory

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.currentPresenation*

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.currentPresenation.* Le nouvel objet slid devra contenir un objet content alors créé.

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



FAIRE VALIDER L'ETAT D'AVANCEMENT

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

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

```
          ...
                      <!-- *************   CENTER PANEL  **************-->
                      <div class="col-xs-7 col-sm-7 col-md-7 col-lg-7 zeroRef
fullScreenHeight" >
                 <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" >

                       //TODO DISPLAY CURRENT SLID TITLE AND DESCRIPTION (EDITION)

                     </div>

                       //TODO DISPLAY CONTENT IMAGE (DISPLAY)

                 </div>
             </div>

    <script src="lib/js/jquery-1.11.3.min.js"></script>

      <!-- ANGULAR -->
  <script src="lib/js/angular.min.js"></script>
       ...
```

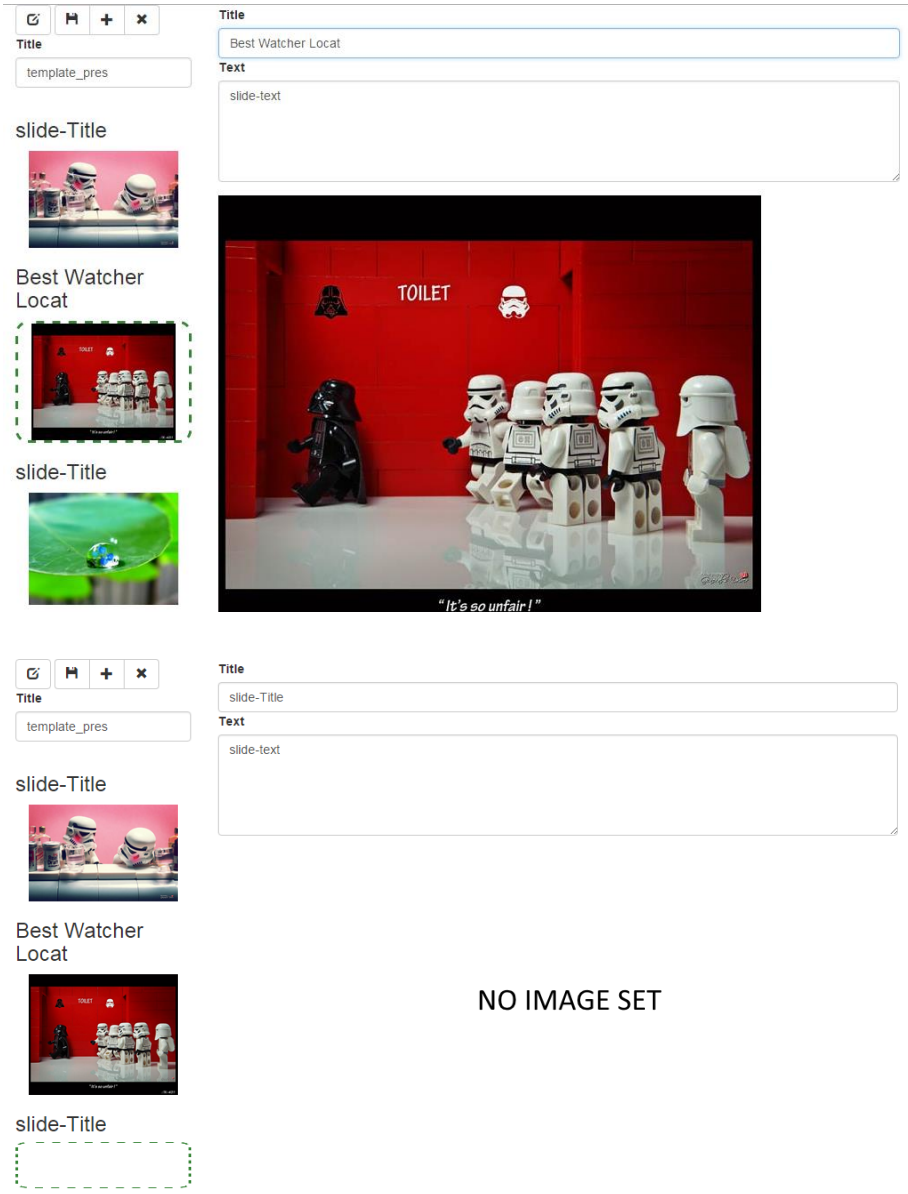
5.4.2. Modifier EventController-s3.1.js et les éléments comme suit :

```
$scope.selectCurrentSlid=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.

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.

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



NO IMAGE SET

FAIRE VALIDER L'ETAT D'AVANCEMENT

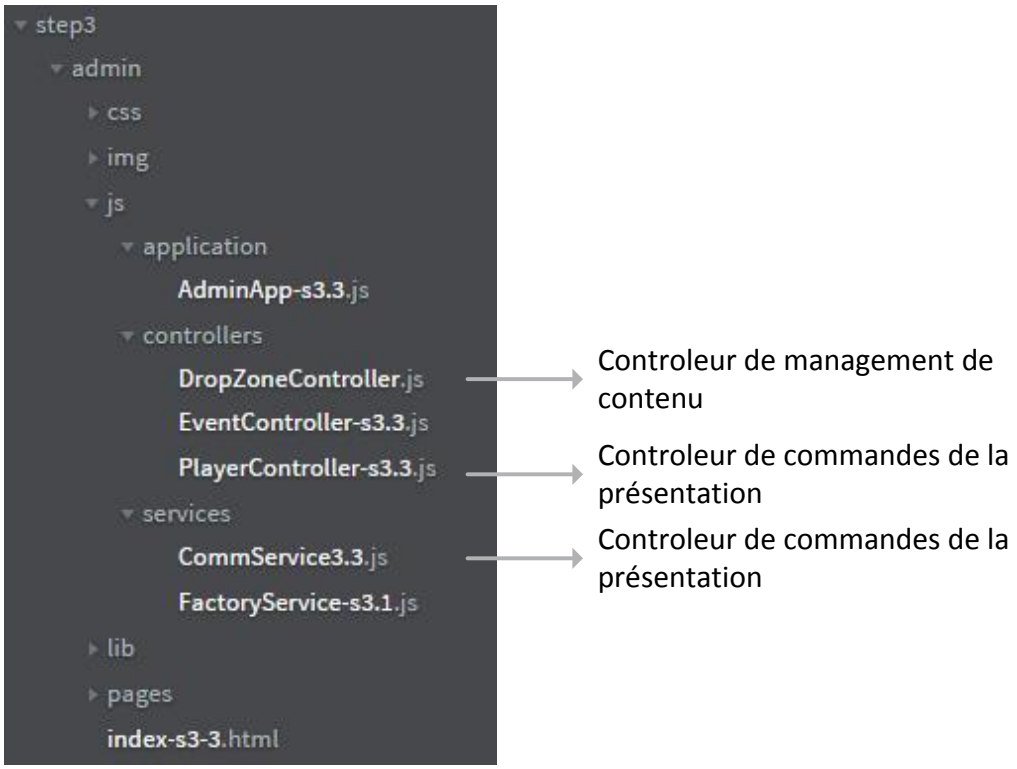
6. STEP 4 : Envoi et récupération de contenu extérieur

Objectif : Usage de service de communication pour recevoir des données, envoyer des données.

Synchronisation de la liste d' information en conséquence

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 :

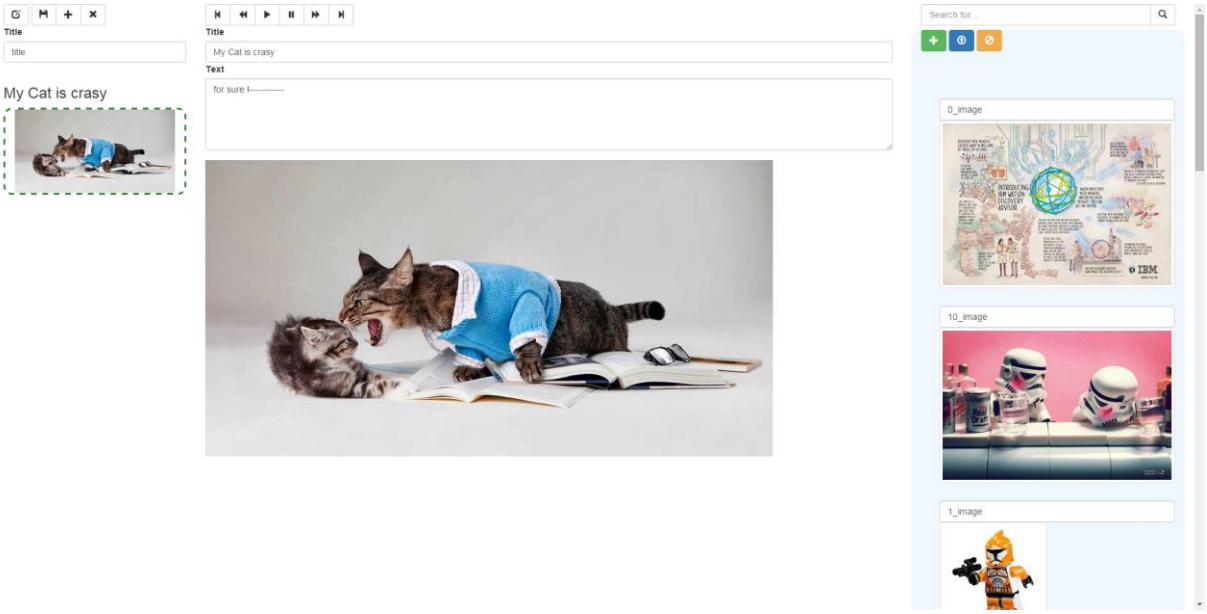
```
<div ng-drag="true" ng-drag-data="content" ng-drag-
success="onDragComplete($data,$event)">
      <img ng-src="{{content.src}}" alt="{{content.title}}" class="img-
thumbnail">
</div>
```

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'" >
```

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'utilisé un promise dans le cas de l'usage du service $http ?

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

FAIRE VALIDER L'ETAT D'AVANCEMENT

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 ?

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

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 crées.

FAIRE VALIDER L'ETAT D'AVANCEMENT