**Projet 1 : Serveur de Vidéo – Web API 2 / AngularJS / TypeScript / TypeLite / Flux XML / ExpandoHelper – MongoDb**

Table des matières

[Introduction 2](#_Toc443469083)

[1. Présentation du Projet 2](#_Toc443469084)

[2. Définition et terminologie 2](#_Toc443469085)

[3. Architecture de développement 2](#_Toc443469086)

[Développement de la couche d’accès aux données 3](#_Toc443469087)

[1. Généricité de la couche d’accès aux données 3](#_Toc443469088)

[A. Qu’est-ce que Microsoft Unity 4 et qu’est-ce que cela nous apporte ? 3](#_Toc443469089)

[B. Utilité d’une architecture découplée 3](#_Toc443469090)

[C. Fonctionnement 3](#_Toc443469091)

[2. Installation et démarrage de Visual Studio 2015 4](#_Toc443469092)

[3. Développement de la couche d’accès aux données sous Visual Studio 2015 5](#_Toc443469093)

[A. Créez notre solution dans Visual Studio 5](#_Toc443469094)

[B. Ajoutez notre librairie de base STPVideoPlayer.DataLayer 5](#_Toc443469095)

[C. Ajoutez notre fournisseur MongoDB : STPVideoPlayer.Data.MongoProvider 6](#_Toc443469096)

[D. Ajoutez un projet de test unitaire 6](#_Toc443469097)

[Couche Service API 7](#_Toc443469098)

[1. Présentation de la couche service 7](#_Toc443469099)

[2. Développement sous Visual Studio 2015 7](#_Toc443469100)

[Couche présentation 9](#_Toc443469101)

[1. Présentation de la couche 9](#_Toc443469102)

[A. Les composants 9](#_Toc443469103)

[B. Projet Type 10](#_Toc443469104)

[C. Implémentation minimale 11](#_Toc443469105)

[2. Développer un lecteur vidéo guidé par un timeline XML 15](#_Toc443469106)

[A. Introduction et prérequis 15](#_Toc443469107)

[B. Dessiner l’interface globale 15](#_Toc443469108)

[C. En déduire les composants et les contrôles 15](#_Toc443469109)

# Introduction

## Présentation du Projet

Le projet Serveur de vidéo a pour objectif de permettre la mise en place d’une chaîne de diffusion vidéo contrôlé à travers d’un flux XML.

## Définition et terminologie

TimeLine : Un timeline est une programmation d’une vidéo. Tous les timelines sont paramétrés dans un fichier XML appelé MyTvLine.xml

Type de vidéo : Il existe différent type de vidéo

Le type Local qui permet d’afficher via le contrôle HTML 5 / Vidéo une vidéo en local, on peut définir des paramètres d’optimisation et de format différent ainsi qu’un poster fournit par le contrôle HMTL 5 Vidéo

Le type série : Voir série

Le type youtube : voir youtube

Générique

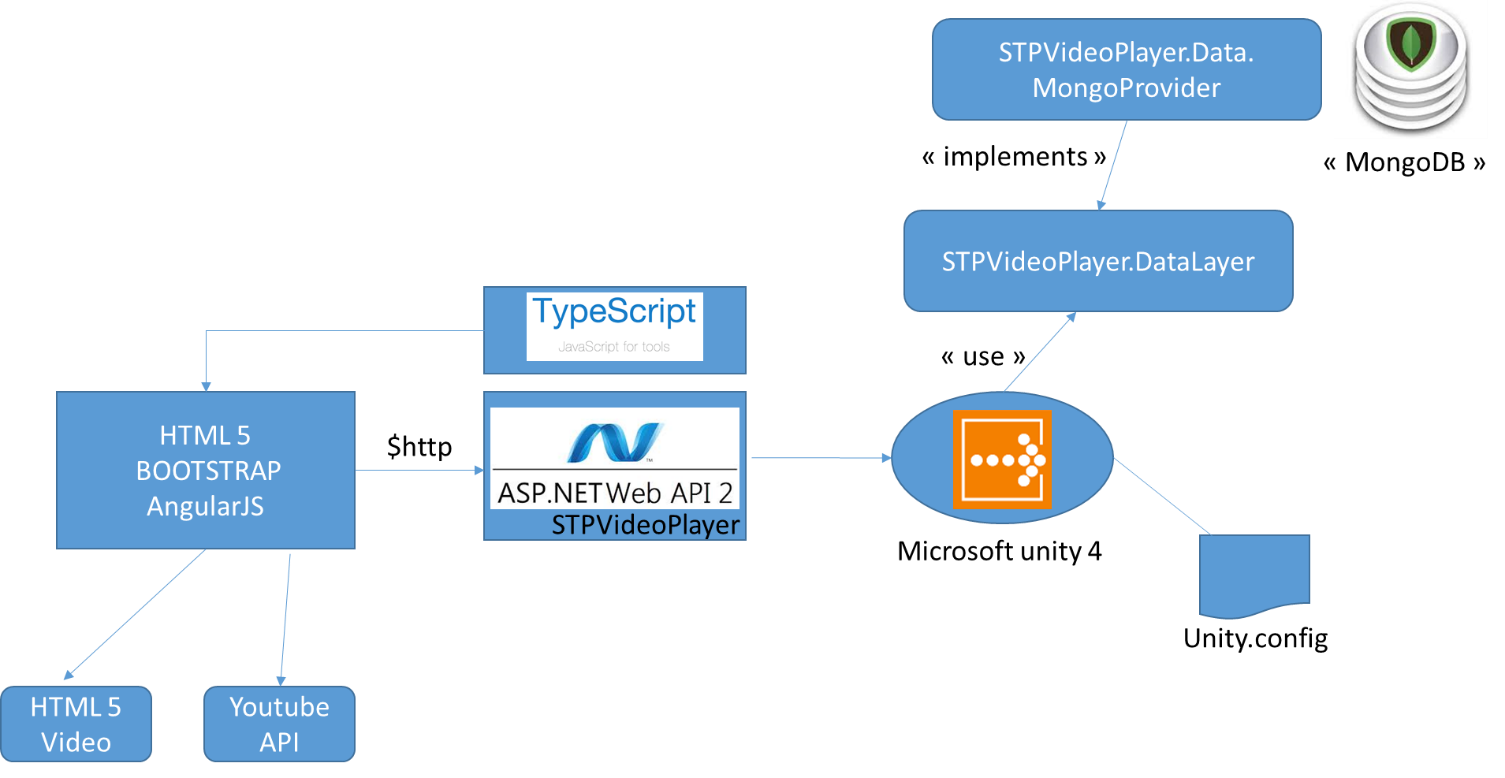
Il peut arriver que le timeline ne trouve aucune vidéo, alors une vidéo par défaut appelée générique est affichée

Série

Une série est un regroupement de vidéos. Les séries sont définies dans le fichier XML MySeries.xml

Chaque vidéo peut être de type local ou youtube

## Architecture de développement



Hébergement

L’hébergement pourra se faire sur tout serveur compatible ASP.Net. Mais pour notre exemple nous utiliserons une machine web role sur Windows Azure.

# Développement de la couche d’accès aux données

## Généricité de la couche d’accès aux données

Nous allons utiliser Microsoft Unity 4, pour permettre la généricité de la couche d’accès aux données.

### Qu’est-ce que Microsoft Unity 4 et qu’est-ce que cela nous apporte ?

Microsoft Unity Application Block est un framework qui fait partie des Patterns & Practices de Microsoft. C'est donc un outil suggéré par Microsoft pour un développement plus propre.

D'après le site officiel, Unity est un conteneur d'injection de dépendances, de poids léger, extensible et aidant à la construction d'architectures découplées

### Utilité d’une architecture découplée

Une architecture en couches est une architecture dont les responsabilités sont reparties sur plusieurs "briques". Elles ont donc un rôle bien défini.

Ici dans notre projet, l’utilité est simplement de répondre à la question suivante : comment pourrais-je faire si je veux permettre demain d’utiliser un autre système de persistance de données que MongoDb et cela sans bien sûr, avoir à tout redévelopper et sans développer tous les fournisseurs dès le départ ?

### Fonctionnement

#### Fabrique de classe

L’un des modèles de créations le plus utilisé est le modèle de fabrique de classe (Pattern Factory). Le nom de ce modèle est directement inspiré à la réalité de sa fonction, car il spécialise un objet unique à la fabrique (création) d’autres objets.

Ici la fabrique de classe s’appelle TvPlayerRepositoryFactory et continue une seul propriété Instance

#### Unity – Helper - container et configuration

Comme nous avons vu, nous allons utiliser la librairie Unity. Cette librairie permet d’injecter des dépendans.

##### Le Helper

Le Helper est une classe qui nous simplifie la vie avec unity Il fournit une propriété Instance qui renvoit l’instance du container en le liant à un fichier de configuration.

##### Le fichier de configuration

Le fichier de configuration permet d’ajouter des composants supplémentaires. On utilise la norme définit par Microsoft et documenté sur MSDN

##### Le container

Le container est l’objet renvoyé par le helper et permet de résoudre les indépendances des objets configuré dans le fichier de configuration.

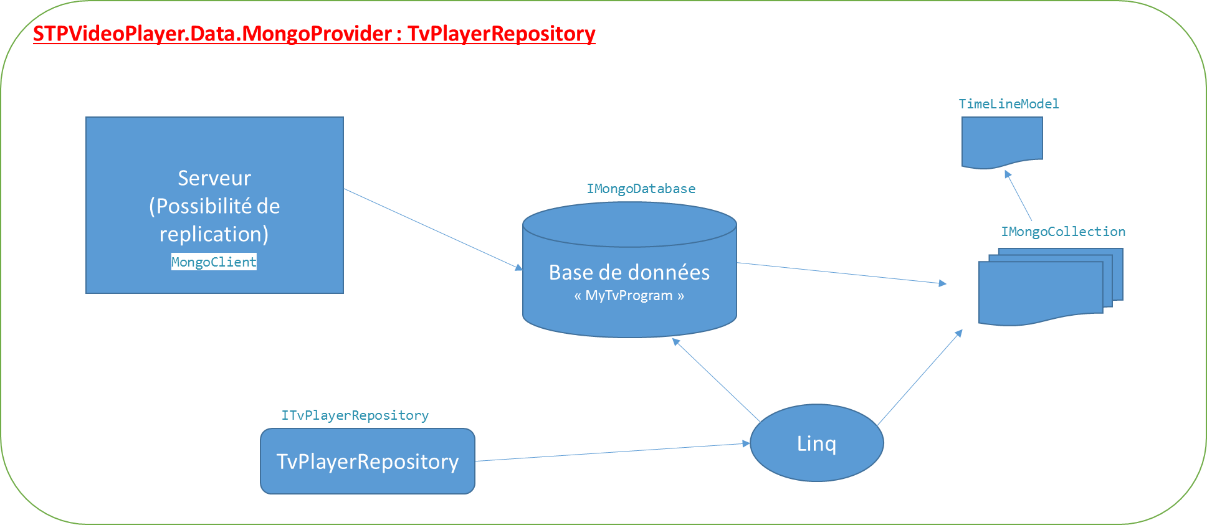
#### Fonctionnement MongoDB

Il existe différentes méthodes pour fonctionner avec MongoDB et .NET, nous utiliserons Linq pour le requêtage. Nous utiliserons le driver officiel fourni sur nuget pour accéder à MongoDB.

##### Installation du driver

Depuis la console nuget : **Install-Package MongoDB.Driver**

##### Architecture de la couche MongoDB



##### Paramétrage de la couche

Nous utiliserons le mécanisme classique des autres DAL. A savoir

* La connectionstring dans le web.config pour définir le serveur

<connectionStrings>

<add name="MongoDB" connectionString="mongodb://localhost"></add>

</connectionStrings>

* Le nom de la base de données en paramètre d’application

<appSettings>

…

<add key="MongoDBName" value="MyTvProgram"/>

</appSettings>

## Installation et démarrage de Visual Studio 2015

Commençons par Installer Visual Studio 2015, si vous ne l’avez pas déjà sur votre poste.

Si vous ne disposez pas de licence pour Visual studio 2015, vous pouvez utiliser la version community téléchargeable en ligne, depuis la page des téléchargements Visual Studio : <https://www.visualstudio.com/downloads/download-visual-studio-vs.aspx>

Pour installer Visual Studio, dans le répertoire racine du support d'installation de Visual Studio, exécutez le fichier d'installation de l'édition souhaitée :

|  |  |
| --- | --- |
| Édition | Fichier d'installation |
| Visual Studio Enterprise | vs\_enterprise.exe |
| Visual Studio Professional | vs\_professional.exe |
| Communauté Visual Studio | vs\_community.exe |

Pour plus d’informations : <https://msdn.microsoft.com/fr-fr/library/e2h7fzkw.aspx#BKMK_Media>

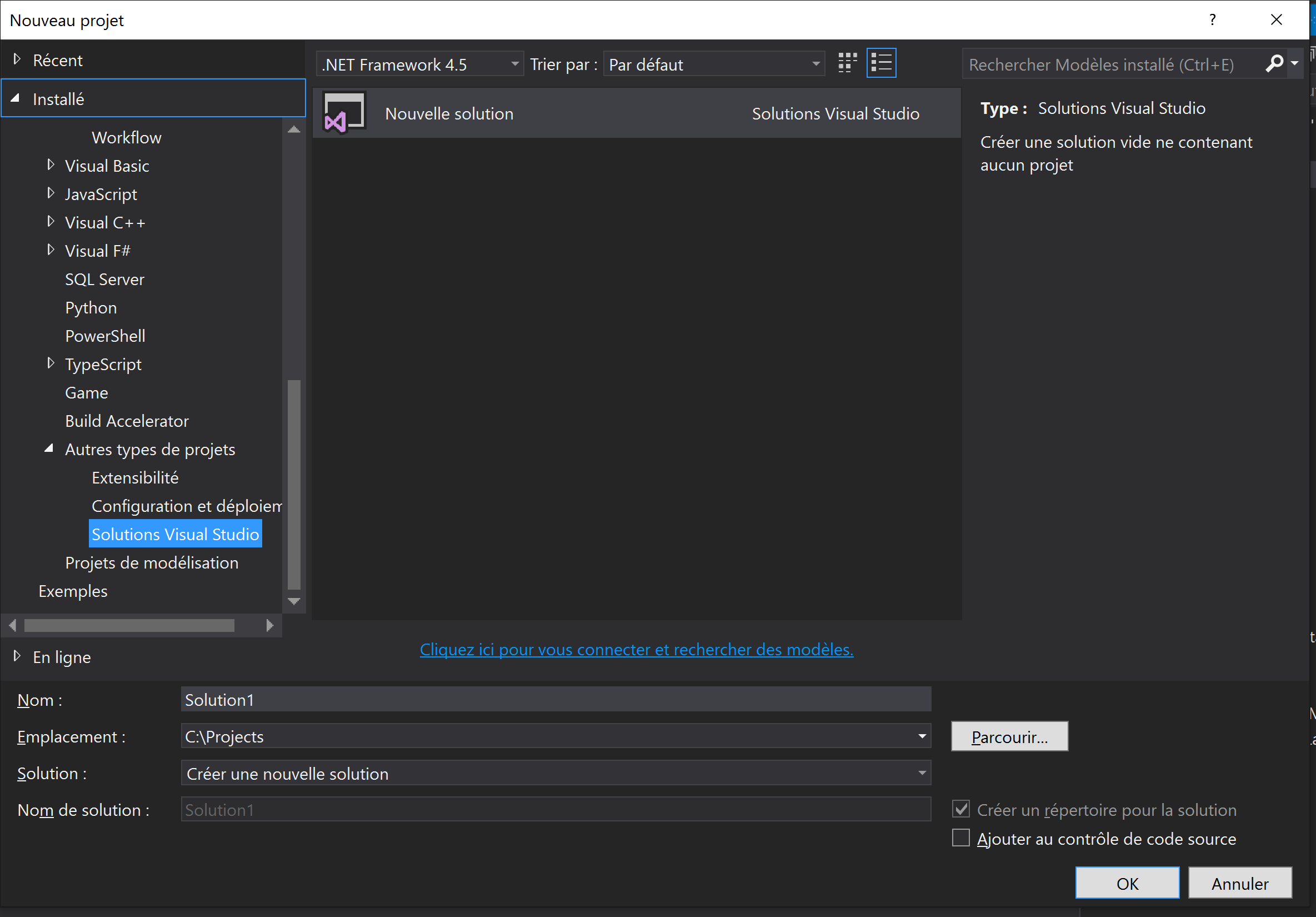
## Développement de la couche d’accès aux données sous Visual Studio 2015

### Créez notre solution dans Visual Studio

Une fois Visual Studio 2015 installé, ou si vous possédez déjà Visual Studio 2015 installé sur votre poste

1. Lancez Visual Studio 2015 en mode Admin

2. Créez une nouvelle solution, si vous possédez Team system, je vous conseille d’utiliser l’addins Nouveau Projet d’équipe

3. Choisissez « Autres types de projets » > « Solution Visual Studio »  


4. Choisissez l’emplacement de la solution

5. Nommez votre solution STPVideoPlayer.sln

### Ajoutez notre librairie de base STPVideoPlayer.DataLayer

1. Ajoutez un projet DataLayer de type « librairie de classes » pour contenir les entités et les interfaces : STPVideoPlayer.DataLayer

2. Ajoutez les classes entités pour décrire vos objets

* EpisodeModel: pour décrire un épisode
* LocalVideoModel : Pour décrire un fichier vidéo local
* YoutubeModel : Pour décrire une vidéo ou une playlist hébergée sur youtube
* TimeLineModel : Pour décrire une planification

3. Ajoutez une interface ITvPlayerRepository

Cette interface décrira le contrat à remplir par nos fournisseur d’accès aux données et injecter par le module Unity 4.

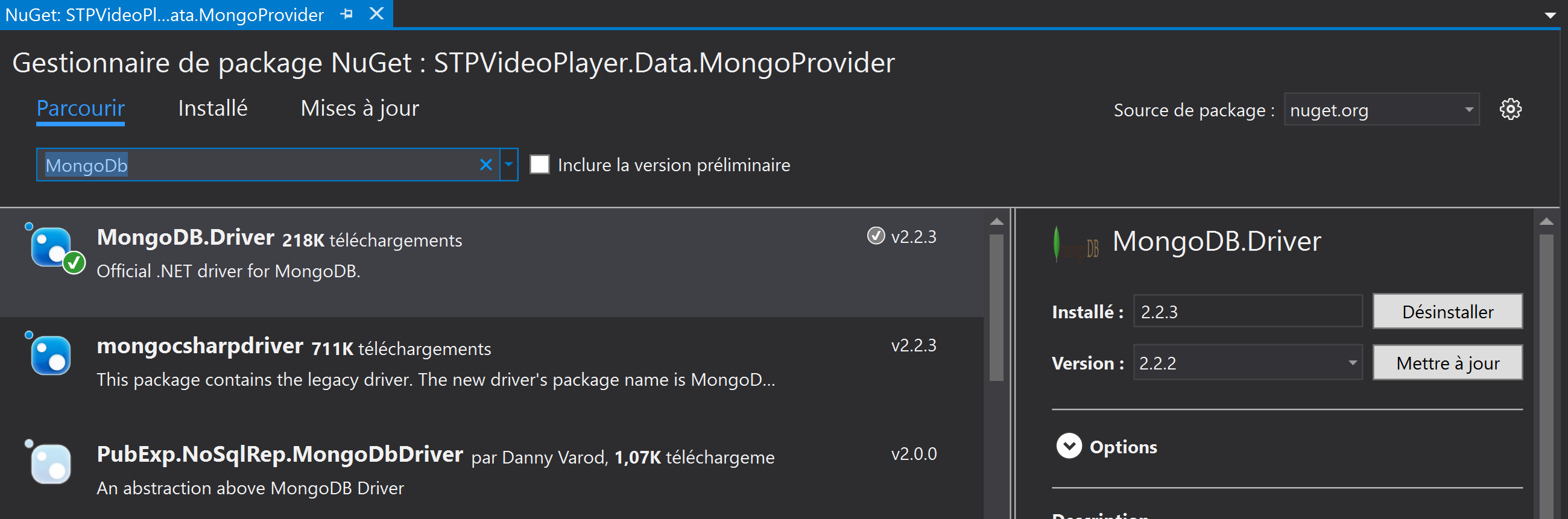
Ajoutez notre premier fournisseur d’accès aux données

Nous n’allons pas développer tous les types de fournisseurs possibles, mais nous utiliserons un seul pour l’instant c’est celui qui nous permettra de communiquer avec MongoDB.

### Ajoutez notre fournisseur MongoDB : STPVideoPlayer.Data.MongoProvider

1. Ajoutez un projet DataLayer de type « librairie de classes » pour contenir le fournisseur d’accès aux données

2. Installez via nuget le driver voir [chapitre fonctionnement MongoDB](#_Fonctionnement_MongoDB)



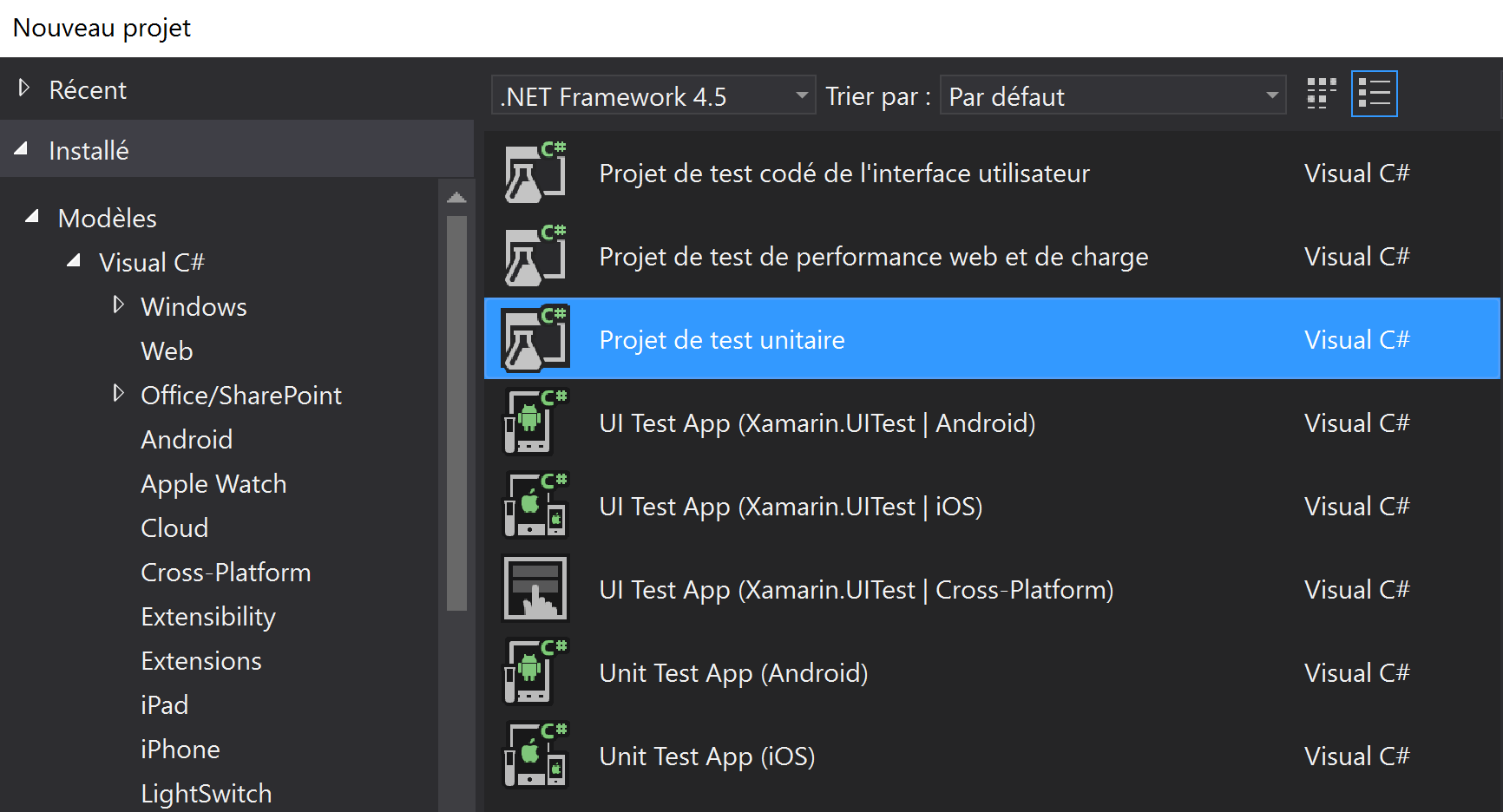
Référencez notre librairie de base STPVideoPlayer.DataLayer

3. Créez une classe ImportProcessor pour importer le flux XML. Nous utiliserons Expando pour mapper le flux XML en objet .NET

4. Créez une classe Repsitory TvPlayerRepository pour communiquer avec MongoDb et implémentant l’interface ITvPlayerRepository

### Ajoutez un projet de test unitaire

Il est important de pouvoir ajouter un projet de test unitaire STPVideoPlayer.Tests à notre solution pour permettre de tester le bon fonctionnement du fournisseur d’accès sans l’imbrication dans toute l’architecture qui sera mise en place.

Créez un projet de type Projet de test unitaire STPVideoPlayer.Tests  


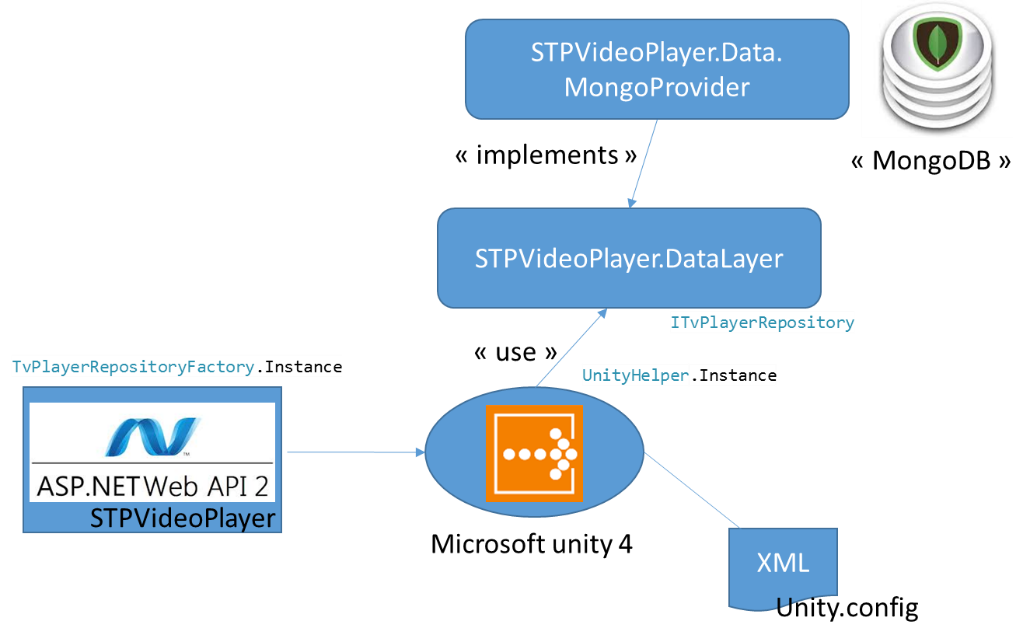
Ajoutez une classe TestClass fournissant une méthode capable d’instancier le fournisseur d’accès aux données et de requêter la collection des vidéos

# Couche Service API

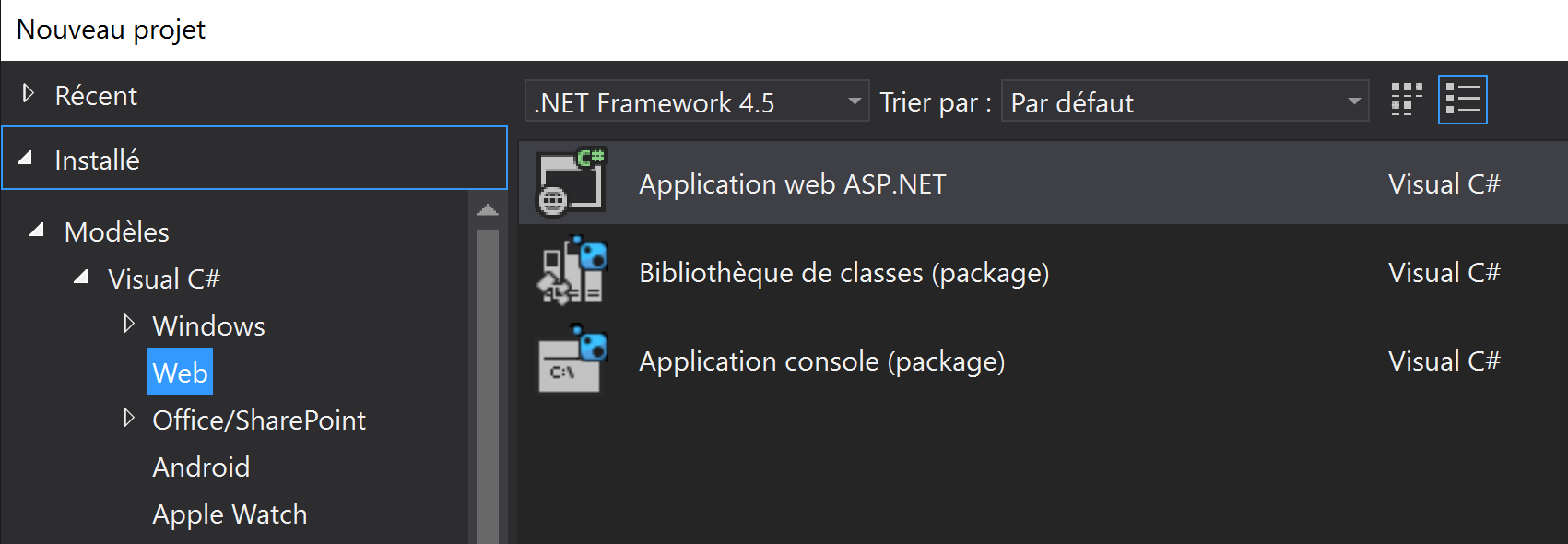
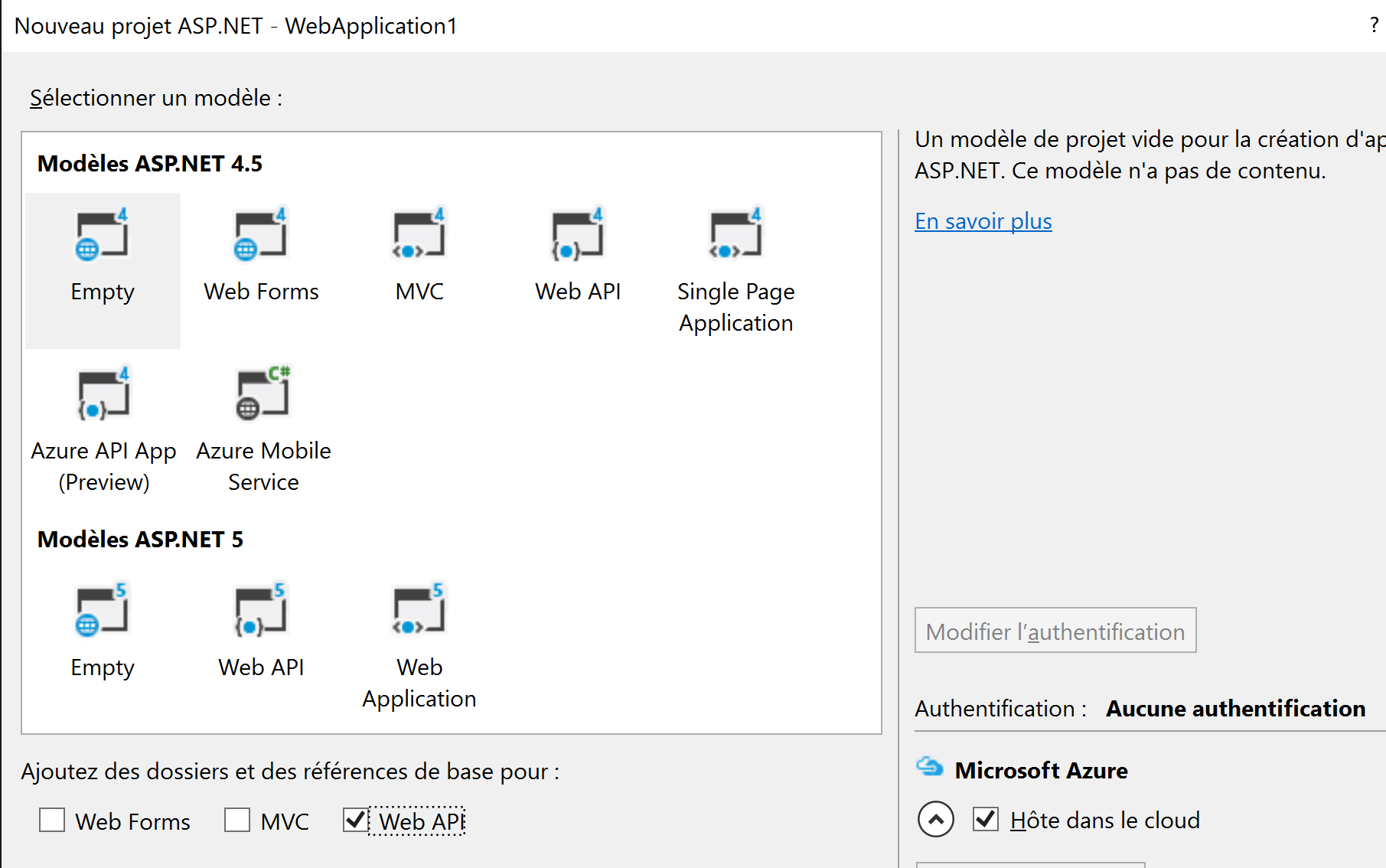
## Présentation de la couche service

La couche utilisateur accédera à la couche d’accès aux données, via une couche API. Cette couche doit

* Instancier la couche d’accès aux données
* Etre évolutive en permettant d’ajouter d’autres fournisseurs sans être recompilé et relivrer.
* Etre testable via un projet de test unitaire



## Développement sous Visual Studio 2015

1. Créez un projet de type Application Web ASP.NET  
   
2. Sélectionnez Le modèle Empty et cochez Web API pour ajouter les références pour WEB API  
   
3. Déclarez le routage pour votre contrôleur

Dans le dossier App\_Start, s’assurer de l’existence du fichier WebApiConfig

Si ce n’est pas le cas, le créer

1. Mappez la route avec votre service  
    // Itinéraires de l'API Web

config.MapHttpAttributeRoutes();

config.Routes.MapHttpRoute(

name: "DefaultApi",

routeTemplate: "api/{controller}/{action}/{id}",

defaults: new {action="Get", id = RouteParameter.Optional }

);

*Attention si le fichier existait et que l’itinéraire existait, par défaut l’api est configurer pour fonctionner en mode REST avec les commandes du http. Modifier le mapping pour pouvoir utiliser des méthodes personnaliser.*

1. Référencez le routage global.asax GlobalConfiguration.Configure(WebApiConfig.Register);
2. Définissez le format de transformation de retour de nos objets JSON

Comme nous retournerons des objets JSON nous pouvons définir le format et la transformation Camel par défaut pour nos objets, pour cela ajouter les lignes suivantes  
au global.asx

var jsonFormatter = GlobalConfiguration.Configuration.Formatters.OfType<JsonMediaTypeFormatter>().First();

jsonFormatter.SerializerSettings.ContractResolver = new CamelCasePropertyNamesContractResolver();

1. Installez le package Unity depuis nuget   
   Install-Package Unity

Ce package nous permet d’utiliser l’injection dynamique pour notre fournisseur d’accès aux données

1. Créez un fichier de configuration

Exemple de fichier : 

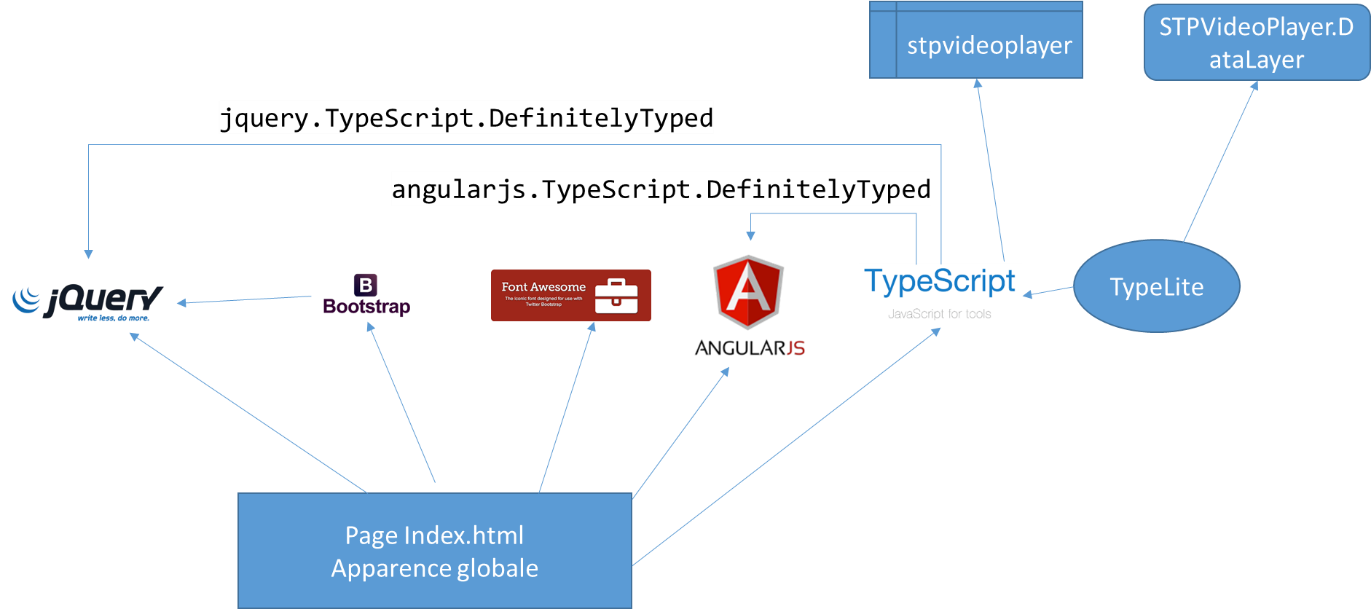
1. Créez un helper pour permettre l’injection dynamique UnityHelper
2. Créez une fabrique qui aura pour rôle de retourner une instance du fournisseur d’accès aux données TvPlayerRepositoryFactory le type à retourner ITvPlayerRepository
3. Créez un contrôleur api videoPlayerServerController capable d’accéder à TvPlayerRepositoryFactory
4. Fournissez les 2 web méthodes nécessaires à notre contrôleur

HttpResponseMessage GetGeneric() : capable de retourner le générique

HttpResponseMessage GetNextVideoTimeLine() capable de retourner la vidéo à diffuser

# Couche présentation

## Présentation de la couche



### Les composants

#### Jquery – version 2.2

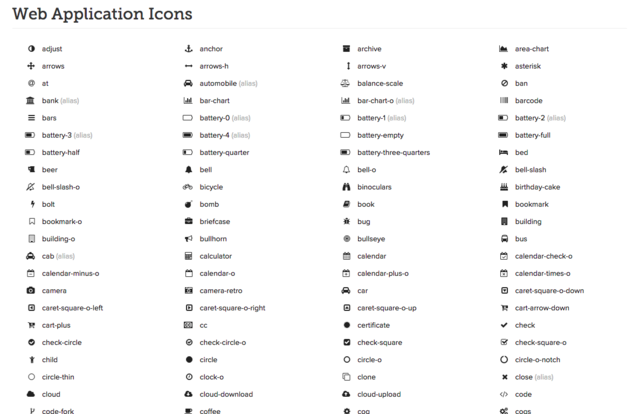
JQuery est une bibliothèque JavaScript libre et multiplateforme créée pour faciliter l'écriture de scripts côté client dans le code HTML des pages web

#### Bootstrap v3.3.6

Twitter Bootstrap est une collection d'outils utile à la création de sites et d'applications web. C'est un ensemble qui contient des codes HTML et CSS, des formulaires, boutons, outils de navigation et autres éléments interactifs, ainsi que des extensions JavaScript en option

#### Police font-awesome

Font Awesome est un toolkit d'icônes vectorielles disponible gratuitement sur internet. Il nous permet d'ajouter rapidement et simplement des icônes sur notre propre site internet. Voici quelques icônes présentes



#### AngularJS

AngularJS est construit autour de concepts clés. Comprenez bien que Shyam Seshadri, lorsqu'il créa Angular, n'a rien inventé. Il n'a fait qu'utiliser à bon escient des concepts et des bonnes pratiques incontournables dans le monde du développement web actuel.

Architecture MVC (Modèle-Vue-Contrôleur) : si vous connaissez le développement, vous avez sûrement entendu parler de ce type d'architecture incontournable qui consiste à avoir une stricte séparation entre les données (Modèle), la présentation des données (Vue), et les actions que l'on peut effectuer sur ces données (Contrôleur)

Data Binding : grâce à ce concept, les liens entre votre code HTML et JavaScript ne seront que plus forts.

Injection de dépendances : tout comme l'architecture MVC, lorsque l'on parle d'injection de dépendances, on parle d'un concept prépondérant dans tout développement. Grâce à cela, les modules que vous développerez n'auront plus à se soucier d'instancier leurs dépendances.

La manipulation du DOM au moyen de directives : la manipulation du DOM conduit souvent à la création de code difficilement maintenable et difficilement testable. Nous verrons qu'avec Angular, ce n'est plus le cas du tout !

#### TypeScript

Plateforme permettant le développement d’applications web de grande échelle. Elle permet

Typage statique en permettant de typer les valeurs de retour de fonctions.

Classes et fichiers de déclaration agissant comme des interfaces.

Support de fonctionnalités tirées d’ECMAScript 6.

Compatibilité entière avec JavaScript

#### TypeLite

TypeLite est un utilitaire qui permet de générer du script TypeScript à partir de classes .NET.

#### Librairie STPVideoPlayer.DataLayer

Fournit les classes entités pour permettre de communiquer avec la couche de communication Web API. Cette librairie est générée automatiquement via TypeLite

#### stpvideoplayer

Application TypeScript permettant de générer l’application angularJs nécessaire. L’application comprend notamment

Les directives (interface HTML)

Les contrôleurs nécessaires pour implémenter le comportement requis par les directives

Le service DataService permettant d’accéder aux données via la couche de communication Web API.

### Projet Type

#### Présentation

Vous pouvez utiliser le projet STPAngular qui permet de démarrer un nouveau projet avec l’ensemble des composants nécessaires pour une application TypeScript/angularJS

#### Arborescence du dossier js/app

Le dossier js/app permet de contenir tous les fichiers TypeScript nécessaire pour l’application. Il contient les sous dossiers suivants

* **Controllers**, recevra tous les contrôleurs des directives

Directives, au sens AngularJS, une directive permet de charger un template Html, de l’associer à un contrôleur et de fournir les comportements de base requis. Ce dossier contiendra toutes les directives requis pour l’application

* **Services**, classe permettant de fournir des services réutilisables par différents contrôleurs
* **Templates**, contient les templates html utilisés par nos directives

Interfaces, contient les interfaces exposées par les services et les contrôleurs

### Implémentation minimale

Bien que ce n’est pas obligatoire par AngularJS, mais nous vous conseillons d’avoir les composants suivants

#### L’objet Scope

Un objet qui permet de partager des données au niveau soit de l’application le $rootScope soit au niveau du contrôleur $scope. Scope n’impacte pas le DOM et on peut rattacher

* Des propriétés
* Des méthodes
* Des événements

Le scope applicatif $rootScope est unique par application.

Pour plus d’information : <https://docs.angularjs.org/guide/scope>

##### Interfacer un Scope

Bien que ce n’est pas obligatoire, car par défaut AngularJS type le Scope :any, il est conseillé de fortement typé le scope via une interface. Cette interface sera stockée dans le répertoire interfaces et s’appelera <application>\_ IScope.

##### Variables par défaut dans un rootScope

* Url : url du web api
* $nomduservice : instance du service d’accès aux données

#### Fichier références \_references.ts

Le fichier de références \_references.ts permet de déclarer les fichiers à importer dans l’ensemble de vos fichiers ts sans avoir à les déclarer à chaque fois.

Nous vous conseillons de segmenter votre fichier références en utilisant les régions.

**#region et #endregion**

//#region "global"

…

//#endregion "global"

#### Interface <application>\_ Idataservice.ts

Il est conseillé d’utiliser une interface pour décrire les méthodes exposées par le service. Cette interface sera stockée dans le dossier js/app/interfaces et portera le nom <application>\_I<service>.ts

#### Service <application>\_dataservice.ts

Le service dataService.ts permet d’accéder aux données. Il fournit le comportement CRUD à l’application. S’il existe plusieurs classes, nous les nommeront de la façon suivante

* ***<application>\_Dataservice\_<Object>.ts****exemple : app\_Dataservice\_order.ts pour la gestion des commandes et app\_Dataservice\_customer.ts pour la gestion des clients*

Dans le cas contraire

* ***<application>\_*dataservice.ts** exemple *stpvideoplayer\_dataservice.ts*

##### Evénements

Vous pouvez propager des événements en se rattachant au $rootScope et en utilisant la méthode $broadcast ou $emit voir ci-dessus

Nous utiliserons l’événement *videoReceived* pour permettre de notifier la réception d’une vidéo

##### $Apply et $digest

$apply permet de lancer manuellement le cycle $digest. Si vous modifiez une variable directement dans le scope, vous n'avez pas besoin de faire appel à cette méthode. AgularJS va directement détecter le changement, car il est programmé pour le faire. Si vous changez une variable en dehors du framework, ce ne sera pas le cas. Par exemple, un changement fait dans une requête Ajax avec JQuery ne sera pas détecté par le framework. C'est à ce moment qu'il faut utiliser $apply. $apply lance le cycle $digest du framework, et permet ainsi l'intégration des modifications faites par des méthodes non gérées par le framework

Emettre un event depuis le service

* $broadcast -- transmet l’événement à tous les scopes enfant
* $emit – transmet l’événement en respectant la hiérarchie du scope

##### Appelez un service web api

Pour appeler un service web api il faut utiliser le service $http: ng.IHttpService

Le service expose 2 méthodes

* Get pour l’appel en méthode http get
* Post pour le post de formulaire en méthode http post

Ces 2 méthodes sont appelées en asynchrone et retourne un IPromise<T> où T est le type attendu

On peut utiliser $q pour s’assurer que la synchronisation au niveau du binding avec l’interface DOM va bien se faire.

Voici comment l’utiliser

Dans le service

this.GetGeneric = (): ng.IPromise<any> => {

var wsurl = this.$rootScope.url + "/videoPlayerServer/GetGeneric";

var deferred = $q.defer();

return $http.get(wsurl).then(

function (response) {

deferred.resolve(response.data);

return deferred.promise;

}, function (response) {

deferred.reject(response);

return deferred.promise;

});

};

Dans le contrôleur

this.$stpvideoplayer\_dataservice.GetGeneric().then(function (result) {

$timeout(function (){

$scope.$apply(function () {

$scope.video = result;

$rootScope.$broadcast("videochanged", result);

});

}, 0);

}, function (error) {

$log.error(error.statusText);

});

##### Tester le service

Pour tester le service, il faut

* Créer un dossier <NomduService>Test sous le dossier directif
* Ajouter une classe directive <NomDuService>TestDirective
* Ajouter une page <NomDuService>Test.tpl.html
* Ajouter un contrôleur <NomDuService>TestCtrl.ts
* Déclarer le service, le contrôleur et la directive au niveau de l’application

stpvideoplayer.service('stpvideoplayer\_dataservice', ['$http', '$log', '$rootScope', Services.stpvideoplayer\_dataservice]);

stpvideoplayer.controller('stpvideoplayer\_dataserviceTestCtrl', stpvideoplayer\_dataserviceTestCtrl);

stpvideoplayer.directive('stpvideoplayer\_dataserviceTest', stpvideoplayer\_dataserviceTestDirective);

Référence

Je vous conseille de ne pas surcharger le fichier de référence global \_references.ts car celui-ci est partagé par tous vos fichiers ts. Créez donc pour chaque test un fichier <application>\_<test>\_references.

Créer une page de test

Ajouter un fichier html stpvideoplayer\_dataserviceTestPage.html

#### Directive <application>Container

La directive de base pour toute application, ce n’est pas une obligation de AngularJs mais je vous le conseille fortement. C’est cette directive qui sera utilisée dans le fichier index.html pour imbriquer votre application. Je vous conseille de l’associer avec un fichier html template main.tpl.html et un contrôleur <application>mainCtrl.ts

***Astuce***

Pour s’abonner à l’événement ready de l’objet document et exécuter une méthode lorsque le document est prêt, on peut : angular.element(document).ready(function () {});

Exemple : var ctrl = this;

angular.element(document).ready(function () {

ctrl.getnextvideo();

});

Pour calcul la durée, jquery n’offre pas de fonctionnalités, mais on peut implémenter une méthode au niveau de l’application et l’attacher au $rootscope

##### Associer une directive et un contrôleur

Lors de la déclaration d’une directive, on peut associer un contrôleur et une directive, pour cela nous utiliserons la propriété controller

##### Associer un contrôleur et un template

Pour nous permettre d’accéder depuis le template html à notre contrôleur nous devons lui donner un alias qui sera utilisé dans notre template. Pour cela nous utiliserons la propriété controllerAs

##### Gérer le scope pour le contrôleur et la directive

Il est possible d’indiquer à AngularJs s’il faut ou non créer un scope pour la directive. Pour cela nous utiliserons la propriété scope: true ou scope: false

Lorsqu’on demande à créer un nouveau Scope, AngularJs crée un nouveau en héritant du scope parent.

Pour plus d’information : <http://www.undefinednull.com/2014/02/11/mastering-the-scope-of-a-directive-in-angularjs/>

##### Etendre nos objets entités

Pour ajouter les propriétés nécessaires à notre template, il est nécessaire d’étendre notre objet entité. Pour cela AngularJS nous fournis la méthode extend. Cette méthode attend deux paramétre

* Nom de la propriété
* Objet à surcharger

###### Exemple d’utilisation

angular.extend(result, { duration: duration });

#### Application.ts

Chaque application doit avoir un fichier ts nommé ***Application.ts*** qui permettra de référencer les services, contrôleurs et les directives à utiliser

##### Déclaration des objets

Déclarer les objets dans l’ordre suivant

* Services
* Contrôleurs
* Directives

##### La méthode run()

Le bloc Run() – est exécuté après que l'injecteur soit créé et est utilisé pour démarrer l'application. Seules les instances et les constantes peuvent être injectés dans les blocs exploités. Ceci afin d'éviter en outre la configuration du système pendant le temps de l'application de l'exécution.

On peut comparer le block Run() à une méthode main() d’un programme console en .NET.

## Développer un lecteur vidéo guidé par un timeline XML

### Introduction et prérequis

Si vous prenez ce document en cours de route il est important d’avoir lu la première partie.

Pour rappel, l’objectif de cette application est de créer un lecteur vidéo guidé par un timeline XML. Le timeline sera hébergée sous MONGODB et accessible depuis un service web api 2.

Il est donc nécessaire d’y a voir implémenter la première partie de ce document.

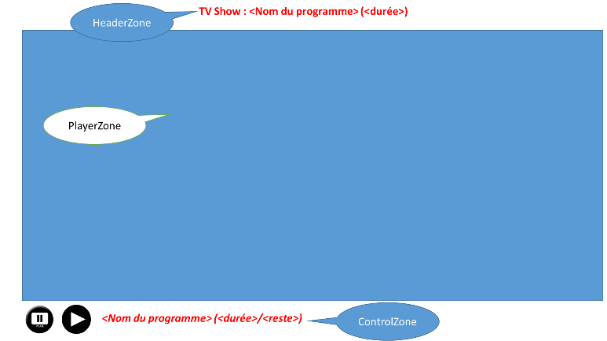
### Dessiner l’interface globale

Cette étape est souvent oubliée par les développeurs, cependant elle permet de garantir le bon fonctionnement de l’application.



### En déduire les composants et les contrôles

#### Déduire les composants ou directive au sens AngularJS



#### Pour chaque directive, en déduire les contrôles et fonctions

##### Directive headerZone

Cette directive permet d’afficher le nom de l’application ainsi que le nom du programme et sa durée. C’est une chaine de caractère qui peut être personnalisé en CSS. Cette chaîne est mise à jour lorsqu’une vidéo est chargée, pour afficher le nom de la vidéo et sa durée.

##### Directive controlZone

Cette directive permet d’afficher les contrôles de lecture, de pause, le nom du programme, la durée initiale et le temps restant.

Nous remarquons ici que nous aurons besoin d’un service timer, pour exécuter une fonction toutes les secondes qui affichera la durée restant pour la vidéo en cours de lecture.

AngularJS fournit la méthode **$interval,** celle-ci prend en paramètre la fonction qu’elle doit exécutée et l’intervalle.

Notre directive aura besoin donc d’un contrôleur pour fournir les fonctions suivantes

* Mettre en pause une vidéo
* Reprendre une vidéo
* Calculer le reste de temps nécessaire

##### Directive PlayerZone

La directive PlayerZone permet la diffusion proprement parlé de la vidéo. Il est basé sur un ng-switch pour charger la bonne directive en fonction du type de la vidéo à diffuser.

La directive PlayerZone communique également avec le Player pour arrêter ou reprendre une vidéo et pour démarrer la vidéo dans le bon moment.

***Astuce ng-switch***

Conditionner son template en fonction de la donnée à afficher

La directive ng-switch est une structure conditionnelle de type switch qui s’utilise directement dans le template. Elle permet d’afficher tel ou tel élément selon la valeur de l’expression qu’elle évalue.

**Pourquoi utiliser ng-switch ?**

* Proposer une interface différente selon le statut de l’utilisateur (administrateur / contributeur / visiteur / …). Mais cela n’empêche pas de vérifier ses droits côté serveur !
* Gérer les différents modes d’affichage de votre webapp, (plein écran / mode compact / …)
* De manière générale, présenter un visuel différent selon le contexte

**Utilisation**

En règle ng-switch s’utilise comment un attribut sur l’élément div.

*Exemple*

<div ng-switch="someVar">

    <div ng-switch-when="hey">

    </div>

    <div ng-switch-when="hi">

    </div>

    <div ng-switch-default>

    </div>

</div>

*Astuce complémentaire*

On peut avoir besoin d’utiliser un ng-switch sans avoir à ajouter au DOM un div, dans ce cas on peut l’utiliser directement comme un élément.

<ng-switch on="video.type">

<localplayer ng-switch-when="local"></localplayer>

</ng-switch>