SimplonBoulogne AngularJS

Design Pattern MVC avec AngularJS 1.5

Sommaire

- Introduction - Installation
- Expressions - Vues
- Directives
- Controllers / Scopes
- Modèles

Introduction

Angular.JS vous aide à mieux organiser votre code Javascript pour créer des sites web dynamiques.

C’est un framework Javascript côté client pour créer de l’HTML interactif. Parfait pour les SPAs (Single Page Applications)

Il est gratuit, open-source, créé par Google depuis 2009.

Une énorme communauté s’est créé autour de ce langage.

Installation Angular.JS

```
  <script src="https://ajax.googleapis.com/ajax/libs/
angularjs/1.5.0/angular.min.js"></script>
```

C’est tout!

Notion d’API

À l’avenir, vous pourrez créer côté serveur une API qui permettra d’émettre des données et que les développeurs externes ou applications tierces puissent puiser leurs ressources à partir de votre API

Par exemple, une web-app sous Angular pourra récupérer les données d’une API et s’exécuter.

Directives

Les directives permettent de booster votre vue : votre HTML.

C’est un tag HTML qui va dire d’exécuter du code Javascript

Vous pouvez également créer vos propres directives.

Quelques directives (built-in)

ng-repeat	ng-init (pour initialiser des valeurs par défaut)
ng-app	Définit l'encapsulation d'un module Angular
ng-controller	Définit un contrôleur, spécification d'un alias possible
ng-model	Bind les valeurs HTML à l'application (Nous allons voir plus en détail après)

Modules

Portion de code de notre application Angular.

```
<!doctype html>
<html ng-app="myModule">
<head>
   ...
```

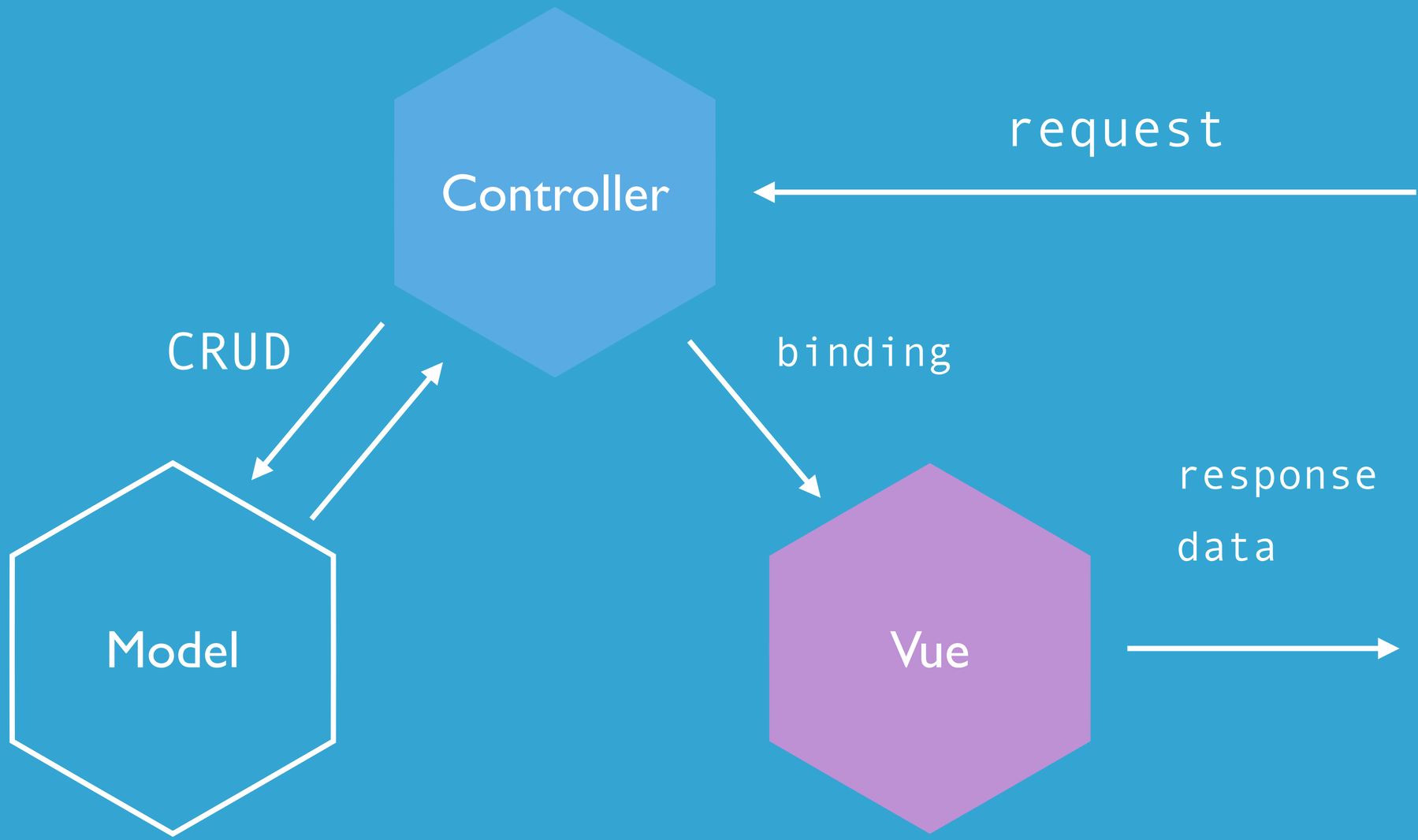
Modules

Portions de code de notre application Angular.

Permet aussi de définir les dépendances de notre code (2ème paramètre)

```
angular.module("myModule", ["firebase"])
```

Rappel : Modèle MVC



Controller

Là où l'on définit nos fonctionnalités avec des fonctions et valeurs.

app.js

```
myModule.controller("PostsCtrl", function($scope) {
  // Les choses sérieuses commencent ici
});
```

Controller - $scope

La variable scope sert à attacher des variables pour les utiliser dans la vue qui utilise le controller.

```
myModule.controller("PostsCtrl", function($scope) {
  $scope.posts = [
    { name: "Le welsh", desc: "C'est délicieux" },
    { name: "Les Macs", desc: "C'est la vie" }
  ];
});
```

Controllers - fonctions

Les controllers, comme pour Symfony, ont pour rôle d'exécuter du code de traitement en tout genre.

Chaque controller peut donc avoir plusieurs fonctions qui lui sont propres.

```
$scope.addTodo = function(){
    console.log("Un nouvel article a été posté");
    var post = {name: "Nouvel article"};
    //Code pour ajout du nouvel article… (push?)
```

Expressions et vue

Côté vue, c'est similaire à Twig avec Symfony, vous pouvez bindé des paramètres, faire des calculs, des conditions etc...

```
<div ng-controller="PostsCtrl">
  {{ posts[0].name }}
</div>
```

Penser à bien vous situez dans une directive qui a chargé le contrôleur correspondant à vos données.

Toujours plus de directives...

```
<ul>
  <li ng-repeat="post in posts">
    <strong>{{ post.name }}</strong>:
    <span>{{ post.desc }}</span>
  </li>
</ul>
```

Boucle

```
<div ng-if="posts.length > 2"> <!-- $scope -->
  <span>Il y a plus de deux articles</span>
</div>
```

Conditions

Modèles

Vous pouvez créer vos entités dans Angular pour ensuite les instancier dans vos contrôleurs.

Penser à bien séparer pour une architecture MVC correcte vos contrôleurs, vues et models (des noms de dossiers explicites suffiront)

Modèles

models.js

```
function Post(name, description) {
  this.name = name;
  this.desc = description;
};
```

controller.js

```
$scope.posts = [
  new Post("Le welsh", "C'est délicieux"),
  new Post("Another entity", "C'est la vie")
];
```

Modèles

Exemple d'utilisation d'une entité avec ng-model

Permet de binder une valeur HTML dans le $scope du contrôleur et l'actualisation se fait en direct !

```
<div ng-repeat="post in posts">
  Titre de l'article:
  <input type="text" ng-model="post.name"/>
  <br/><span>{{ post.name }}</span>
</div>
```

Affichage de la valeur du $scope

Liste d'événements

ng-blur	Quand un élément perd son focus
ng-change	Changement de valeur dans un input
ng-mouseleave	Détecte lorsque la sourie sort d'un élément
ng-mouseover	Détecte lorsque la sourie entre dans un élément

Appeler événement

Généralement, vous définissez une fonction dans votre contrôleur et l'appelez dans votre vue avec une directive

```
$scope.onMyButtonClick = function() { // app.js
  // Some code
};
```

```
<button ng-click="onMyButtonClick">Lorem ipsum</button>
```

Un p’tit récap’

- Les contrôleurs et les vues sont liés grâce au …
- Les … sont utilisés pour afficher des données côté client
- Un … peut contenir des … (2 réponses possibles)

Challenge 1

Créer un contrôleur qui bind des paramètres au $scope

Créer un modèle avec une entité Produit qui prend un titre, une description et un prix.

Penser à bien séparer pour une architecture MVC correcte vos contrôleurs, vues et models (des noms de dossiers explicites suffiront)

À partir de la vue, vous pourrez facilement voir, ajouter, modifier, supprimer un produit (penser donc à créer les fonctions nécessaires). Utilisation d'une BDD non demandé donc données non sauvegardées.

Ressources

- https://www.codeschool.com/courses/shaping-up-with-angular-js/
- https://angularjs.org
- https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Objets_globaux/Array/unshift
- http://www.w3schools.com/angular/tryit.asp?filename=try_ng_model_two-way
- Plug-in Angular Batarang pour Chrome (debug Angular)
- https://docs.angularjs.org/api/ng/filter (pour ng-repeat)