Découverte du Framework Angular JS avec le contexte GSB

Partie 3 Gestion des rapports d'un médecin

Description du thème

|  |  |
| --- | --- |
| Propriétés | Description |
| **Intitulé long** | Découverte du Framework Angular JS avec le contexte GSB dans sa partie gestion des rapports de visite |
| **Formation concernée** | BTS SIO option SLAM |
| **Matière** | SLAM 4 |
| **Présentation** | Accompagnement dans la découverte d’Angular. Développement pas à pas d’une application à partir du contexte GSB |
| **Notions** | • D4.1 - Conception et réalisation d’une solution applicative  • D4.2 - Maintenance d’une solution applicative  Savoir-faire  • Programmer un composant logiciel  • Exploiter une bibliothèque de composants  • Adapter un composant logiciel  • Programmer au sein d’un Framework |
| **Prérequis** | Les principes du développement web, PHP, SQL, JavaScript |
| **Outils** | SGBD MySQL, un environnement de développement |
| **Mots-clés** | GSB, Angular JS, Ajax, MVVM |
| **Durée** | 6 heures |
| **Auteur(es)** | Patrice Grand. Relectures de Cécile Nivaggioni, Yann Barrot, Luc Frebourg et Gaëlle Castel. |
| **Version** | v 1.0 |
| **Date de publication** | novembre 2016 |

Présentation

Nous allons poursuivre l’application en réalisant les cas d’utilisation concernant la gestion des médecins ; ces cas sont décrits dans la présentation du contexte. Le code à utiliser est dans le répertoire gsbAJSV3.0.

Pour ceux qui souhaitent poursuivre à partir des sources obtenues dans les parties précédentes, vous devez :

* + remplacer votre vue *medecins.html* par celle fournie dans gsbAJSV3.0 ;
  + ajouter les vues *majMedecin.html* et *derniersRapports.html* (disponible dans gsbAJSV3.0) à vos sources ;
  + remplacer votre contrôleur *medecinsController* par celui fourni dans gsbAJSV3.0 ;
  + ajouter les fichiers *traiterrecherchemedecins.php, traitermajmedecins.php* et *traitergetlesrapports.php* (disponible dans gsbAJSV3.0) dans votre répertoire « ajax » ;
  + ajouter les contrôleurs *majMedecinController* et *derniersRapportsController* (disponible dans gsbAJSV3.0) dans votre fichier controllers.js ;
  + créer les routes :
    - /majmedecins faisant appel à la vue *majMedecin.html* et au contrôleur *majMedecinController* ;
    - /derniersrapports faisant appel à la vue *derniersRapports.html* et au contrôleur *derniersRapportsController* ;

##### La recherche du médecin

Ainsi, on désire ajouter dans la page *medecins.html*, une zone de recherche d’un médecin :

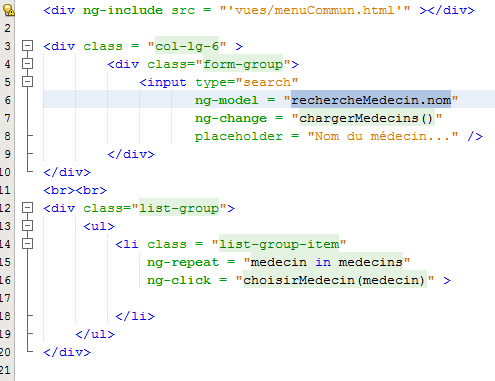
|  |  |
| --- | --- |
| Fig 1 | Fig 2 |
| Fig 3 |  |

* + Fig 1, le visiteur saisit le début du nom du médecin recherché.
  + Fig 2, la liste des médecins triée apparaît.
  + Fig 3, le visiteur choisit le médecin qui apparaît dans la zone de recherche.

Le médecin sélectionné est celui qui sera concerné par les options du sous-menu :



Nous allons commencer par commenter la vue disponible *medecins.html* :

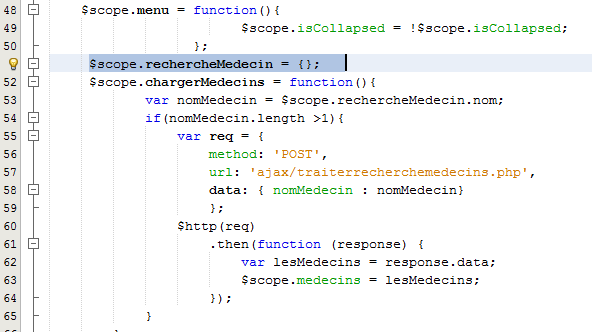


* + remarquez l’indentation d’usage proposée pour les attributs et directives des éléments HTML, sympathique, non !!
  + les lignes 3 à 10 créent la zone de recherche (fig 1) ; la donnée saisie sera disponible dans le contrôleur en tant que propriété *recherchemedecin.nom* du scope (voir remarque plus bas).
  + lors d’un événement ‘‘*onChange’’* la méthode *chargeMedecins* sera exécutée dans le contrôleur.
  + les lignes 12 à 20 créent la zone de liste présentant les médecins répondant au filtre sur les premières lettres du nom.
  + la ligne 15 utilise la directive *ng-repeat* (déjà vue dans la partie 2).
  + sur le clic d’une ligne *li* la fonction *choisirMedecin* sera exécutée. Notez le paramètre *medecin* qui réfère la variable *medecin* de la directive ng-repeat ; la portée de la variable *medecin* est l’élément html où elle est déclarée (l’élément HTML *li*). Ce paramètre sera récupéré et exploité dans le contrôleur.

**Remarque** :

La ligne 6 utilise la propriété *rechercheMedecin.nom* et non *rechercheMedecin*. Angular préconise de déclarer les propriétés qui sont *bindées* vers le contrôleur (vue🡪contrôleur) sous forme d’objet JavaScript. Ceci est lié au mécanisme d’héritage très particulier à JavaScript ; dans de très nombreux cas déclarer une propriété comme *valeur* ne pose pas de problème (voir partie 2) mais ceci peut poser des problèmes pour récupérer cette valeur dans le *scope parent*. Aussi, nous prendrons soin maintenant de définir les propriétés comme des objets (lorsqu’elles sont *bindées* vers le contrôleur). Acceptons cette règle sans y prêter une importance particulière.

Dans le contrôleur *medecinsController*, la méthode *$scope.chargerMedecins* est présente dans le code fourni; le code ressemble à ce que vous avez vu dans la partie 2, nous commenterons deux points :



* + la ligne 51 crée l’objet *rechercheMedecin*, voir remarque au-dessus,
  + la ligne 53 récupère ce qui a été saisi dans la vue,
  + la ligne 54 lance la requête Ajax uniquement si l’utilisateur saisit au moins deux lettres,
  + la ligne 63 valorise la propriété *medecins* dont a besoin la directive *ng-repeat*.

Pour rendre l’ensemble opérationnel, il faut écrire la méthode *choisirMedecin* dans le contrôleur :



**Travail à faire**

* Compléter la vue *medecins.html* afin que le nom, le prénom ainsi que le département (figure 2) s’affichent dans la liste générée par la directive *ng-repeat*.
* Ajouter dans le contrôleur associé la méthode *choisirMedecin* de l’objet $scope qui doit :- afficher le médecin sélectionné (son nom et son prénom) dans le champ de recherche (balise *input*) ;  
  - vider la liste.

##### Les options du sous-menu

Un peu de théorie…

Chaque option du sous-menu utilise le médecin sélectionné ; le modèle choisi par Angular lie très fortement la vue avec son contrôleur (modèle que l’on appelle modèle-vue-vue/modèle, ici la partie vue/modèle est gérée par le contrôleur, qui porte très mal son nom).

Cette architecture où la vue et sa vue/modèle -le contrôleur- communiquent très naturellement rend parfois assez délicate la communication entre les contrôleurs.

Une solution consiste à utiliser la classe mère de tous les objets *$scope* à savoir l’objet *$rootScope*. Consultez l’annexe 1 qui entre dans les détails.

**Travail à faire**

* Ajouter le médecin sélectionné dans le *rootScope*.
* Tester que le médecin est bien présent dans le *rootScope*, en utilisant l’extension pour le navigateur Chrome (voir annexe 1) ou en loguant :



Nous sommes presque prêts à décliner les différentes options du sous-menu. En effet, il nous faut d’abord régler le problème qui survient lorsque l’on sélectionne une option du sous-menu sans avoir au préalable choisi un médecin ; ceci occasionne un bug actuellement, le formulaire de mise à jour du médecin s’affiche (alors qu’il ne devrait pas) de cette façon :



###### Lever le bug dans le cas où on n’a pas choisi de médecin

Plusieurs solutions existent ; ne pas rendre visible le bouton ou afficher un message d’erreur ou simplement ramener le visiteur à la page *medecins*. C’est cette dernière solution que nous allons privilégier.

**Rappel :** la méthode *url* de $location permet de rediriger vers une page passée en paramètre.

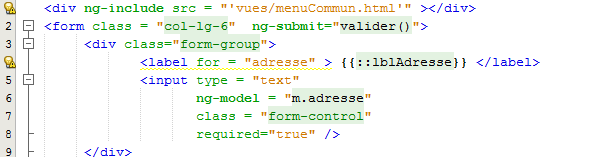
**Travail à faire**

* Ajouter le code qui va tester dans le contrôleur *majMedecinController* si le médecin existe dans le scope courant ; dans le cas contraire, on dirigera le visiteur vers la page *medecins*.
* Faire de même pour l’affichage des derniers rapports : s’assurer qu’un médecin a été sélectionné.

###### Mise à jour du médecin

Pour cette partie, le code est (presque) totalement fourni ; nous allons nous contenter de commenter quelques éléments.

La vue majMedecin.html



On retrouve ce que nous évoquions plus haut ; les propriétés *bindées* sont des objets, ligne 6.

Le contrôleur



Rien de particulier ici, non plus :

* + les lignes 71 à 81 gèrent l’affichage dans la vue,
  + la ligne 82 récupère par « héritage » le médecin stocké dans le *rootScope* (cf paragraphe 2),
  + la ligne 83 crée l’objet *m,*
  + les lignes 84 à 86 permettent d’afficher au chargement de la page les données actuelles du médecin,
  + la ligne 85 définit le code de la propriété *valider* (*ng-submit = valider()* dans la vue) ; lance une requête Ajax qui fournit les données saisies en entrée (plus l’*id* du médecin pour la requête *update* de *pdo*).

L’ensemble est opérationnel, néanmoins on désire qu’un message s’affiche dans la vue en cas de succès ou d’échec de la requête Ajax (valeur de retour) :

|  |  |
| --- | --- |
|  |  |

**Travail à faire**

Effectuer les modifications nécessaires afin de répondre au nouveau besoin.

*Vous pouvez utiliser les classes BootStrap : http://getbootstrap.com/components/#alerts*

###### Gestion des rapports du médecin

Vous êtes prêts à prendre en charge cette partie ; on vous a fourni dans gsbAJSV3.0 le code de la vue (*derniersRapports.html*) et le fichier *traitergetlesrapports.php* qui sera appelé par la requête Ajax.

On désire avoir les écrans suivants, selon les cas :

|  |  |
| --- | --- |
|  |  |

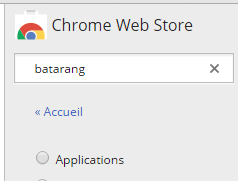
**Travail à faire**

Terminer cette partie en écrivant le code du contrôleur *derniersRapportsController* puis tester.

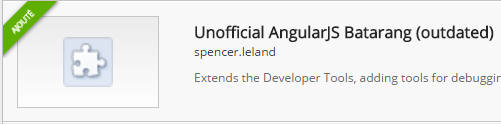
Ceci termine cette partie 3 dont le corrigé se trouve dans le répertoire **gsbAJSV3.1**.

Annexe 1 : outil de débogage et l’objet $rootScope

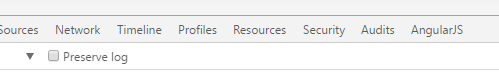
Chrome propose des extensions dédiées à Angular. Pour les installer, il faut aller dans Paramètres/Plus d’outils/extensions et chercher *batarang* :



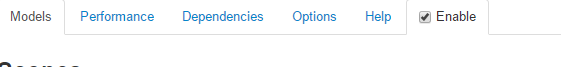
Plusieurs extensions sont proposées ; j’ai un petit faible pour :

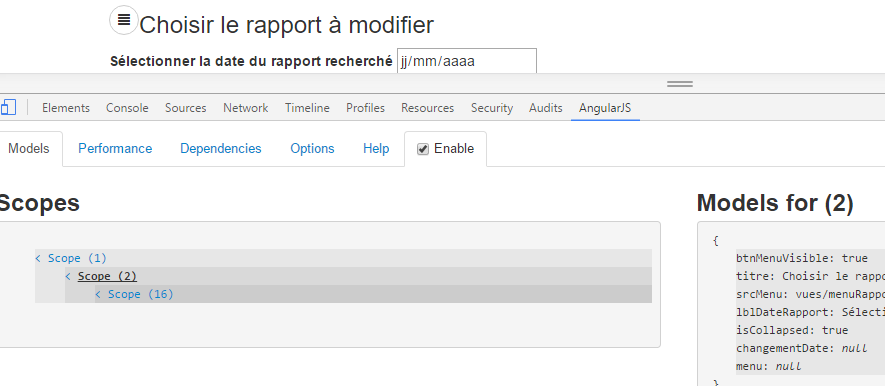


Après l’ajout de l’extension, votre environnement de débogage (touche F12) s’est enrichi d’un nouvel icone :



Testons cela, lançons l’application ; vérifier que vous avez bien coché la case :





La partie Scopes montre la hiérarchie des Scopes ; la partie Models indique les propriétés actuelles du Scope sélectionné. Le Scope le plus haut hiérarchiquement est le rootScope ; il est unique et chaque Scope en hérite (dans le sens JavaScript) : chaque Scope peut ainsi utiliser une propriété définie sur rootScope.

Par exemple, si vous écrivez n’importe où dans un contrôleur (***controller1*** par exemple) ou ailleurs :

*$rootScope.personne = {nom : ‘Durand’, prenom : ‘Jean’} ;*

*// il est fortement conseillé d’utiliser des objets et non seulement des valeurs*

Vous pourrez dans n’importe quel contrôleur (***controller2***, par exemple) appeler :

*$scope.personne.nom*

**Attention :** l’utilisation de $rootScope nécessite une injection de cet objet dans le contrôleur qui l’utilise, ***controller1*** dans notre exemple.