Chapitre 1

Introduction

Chapitre 2

Le contexte du travail

2.1 Le cahier des charges ?

2.2 L'architecture de la plateforme cible

2.3 Le framework Angularjs ?

2.4 Les ressources REST

2.5 Le contrôle d'accès en utilisant des rôles (RBAC)

Chapitre 3

La Conception de notre générateur de module

3.1 Le concept général

3.1.1 Définition d'un module

Un module est une application métier [1] basée sur l'architecture client-serveur. Nous nous référons au terme module pour indiquer le caractère modulaire que peut prendre une application. Par exemple,la gestion des resources humaines est un assemblage de plusieurs sous-applications telles que la gestion des employés, la gestion des carrières, la gestion de la paie et la gestion des embaûches [2].

L'architecture client-serveur, implique qu'un module est composé d'une application *client* et d'une application *serveur*. L'application client est une interface graphique permettant à aux utilisateur d'interagir avec un système d'information. L'application serveur est celle qui gère les données métiers dans un système d'information. La gestion des données est faite essentiellement via une base de données et les processus métiers.

3.1.2 Le portail applicatif des modules

Le concept du portail applicatif est de fournir l'environnement d'installation et d'utilisation [3] des applications générées à partir de notre générateur. Il représente le moule qui donne au développeur, un apperçu d'à quoi va ressembler son application à l'installation.

L'interface du portail applicatif se décline dans la figure suivante. Elle est subdivisée en 6 parties à savoir :

— 1- Cette partie indique le nom du module installé

1. A expliquer
2. donner la liste complète si possible
3. Donner l'exemple des application PAE

— 2-

— 3- `Menus`, il s'agit de la liste des menus du module

— 4- `Linked modules`, liste des modules liés au module installé [4]

— 5- `Workspace`,

— 2-

Installation du module

Le protail applicatif est en quelques sorte le squelette des applications. Ceci se décline dans la figure ...

Après s'être connecter, l'utilisateur installe le module/l'application qu'il veut utiliser. L'installation consiste à remplir les différentes parties du portail. Ainsi, on a :

— 1- Le nom du module ;

— 2- Lien d'accès au dashboad du module ;

— 3- Emplacement des menus du module ;

— 4- Les liens vers les modules liés au module installé ;

— 5- Barre d'opérations, où se placent les actions de la page courante du module ;

— 6- L'espace de travail, où s'interchangent les différentes pages du module

3.1.3 Le principe générale

L'idée du générateur est d'utiliser un environnement graphique permettant de spécifier les différents éléments nécessaires à l'implémentation des applications client et serveur, en vue d'augmenter la productivité générale [5] d'un développeur. En effet, une fois les éléments spécifiés, nous passons à la génération automatique des codes des applications client serveur. Ceci permettra aux développeurs de se focaliser sur la conception de leurs modules.

Nous regroupons les éléments du module en deux catégories : ceux *côté client* et ceux *côté serveur*. Les éléments côté client sont ceux qui sont nécessaires pour la mise en place d'une architecture (MVC). Ils permettent de définir les différentes pages ou vues html ainsi que les services qui permettent de recupérer les données du serveur. Les éléments côté serveur permettent de définir les bases de données, les processus métiers et ressources permettant d'interagir avec le client.

4. expliquer la liaison de module
5. listing des productivités

3.2 La présentation des éléments graphiques côté client

3.2.1 Overview

Présentation de la vue générale

La vue générale se concentre sur les éléments côté client d'un module. Elle est l'étape principale dans le processus de réalisation d'un module. Aussi simple qu'elle parait, elle rassemble tous les outils nécessaires pour concevoir chacun des éléments de la partie cliente d'un module. Elle se résume à l'interface de la figure suivante. Sous les onglets `components` et `treeview`, nous disposons respectivement de la palette de composants html et de leur organisation sur la vue en forme d'arborescence. sous les onglets `variables` et `functions`, on définit les variables et les fonctions qui sont utilisées pour écrire le contrôleur de la vue. Sous les onglets `actions` et `configs` on définit respectivement les actions du module et les paramètres de configuration de la vue.

Les composants

Ils constituent un outil graphique permettant de manipuler les éléments qui entrent dans l'élaboration d'une interface web. Les manipulations sont basées sur un certains nombres de propriétés que nous leur avons associé afin de controller leur mise en forme, leurs attributs html ainsi que les données qu'ils manipulent.

Nous avons resencé les composants les plus couremment utilisés dans les applications web et nous les avons regroupé en trois grandes catégories :

— *Les composants simples* ; ce sont des éléments HTML dont la définition ne nécéssite pas l'imbrication d'un autre composant. exemple :

— *Les composants conteneurs* Ce sont des éléments qui peuvent contenir des composants de n'importe quelle catégorie, par exemple l'élément html `form` permet de regrouper les composants simples tels que les `input` et des `button` pour la réalisation d'un formulaire.

— Les composants widgets Ce sont des éléments non standards que nous avons entièrement écris, en proposant leur design et leur conception.

L'arbre des composants

Le concept de l'arbre des composants est d'augmenter la productivité des dévéloppeurs en limitant le perte de temps due aux changements récurents dans le développement des interfaces [6].

6. explication détaillée

L'arbre des composants est un outil permettant de représenter les composants d'une interface. Ses composants sont organisés sous forme d'arbre afin de mieux schématiser la hiérarchie induite par les composants conteneurs. La figure **??** représente l'arbre des composants de l'interface de la figure **??**.

Notre concept se décline via l'inteface de la figure **??** dans laquelle on constate qu'un certains nombres d'actions sont associées à chaque composant. En effet pour permettre la réutilisation de codes nous proposons les cinq actions suivantes :

— Monter - descendre : elles permettent les déplacements de composants vers le haut ou vers le bas, dans le même conteneur. Ceci permet de réorganiser les composants d'un même conteneur.

— couper : elle permet les déplacements de composants d'un conteneur à un autre. Ceci permet la réorganisation globale des composants.

— copier : permet de cloner un composant, soit dans un même conteneur ou non. Ceci permet de générer facilement une interface avec des composants semblables.

— Supprimer : elle permet d'enlever un composant de l'arborescence.

Les fonctions

Le concept ici est d'améliorer la productivité du dévloppeur en :

— évitant les *erreurs d'écriture* dans les noms des fonctions dans les vues et dans les contrôleurs.

— proposant une *vue d'ensemble* du travail à réaliser (isolation ?).

Le concept se décline via l'inteface de la figure **??**

Erreur d'écriture Le lien « generate functions », génère les noms des fonctions, à partir de l'arbre des composants. Les composants tels que button, ... et ... font appel à des fonctions qu'ils définissent dans leurs propriétés. à partir des propriétés des composants cités plus haut, nous extrayons le nom des variables à déclarer.

Vue d'ensemble Nous pouvons dinstinguer parmi les variables à définir, celles qui sont dejà définies ou non. La définition d'une variable se fait en l'isolant des autres variables. Ceci permet au développeur de se focaliser sur le reste des variables et surtout de ne pas perdre de temps à chercher une information dans les codes qui peuvent rapideent devenir illisibles.

Les variables

Le concept ici est d'améliorer la productivité du dévloppeur en :

— évitant les *erreurs d'écriture* dans les noms des fonctions dans les vues et dans les contrôleurs.

— proposant une *vue d'ensemble* du travail à réaliser.

— simplifiant la déclaration des variables.

Le concept se décline via l'inteface de la figure **??**

Erreur d'écriture. Le lien « generate vars », génère les noms des variables, à partir de l'arbre des composants. Les composants manipulent des variables. à partir des propriétés, nous extrayons le nom des variables à implémenter.

Vue d'ensemble. Nous pouvons dinstinguer parmi les fonctions à implémenter, celles qui sont dejà définies ou non. La définition d'une fonction se fait en l'isolant [7] des autres fonctions. Ceci permet au développeur de se focaliser sur le reste des fonctions et surtout de ne pas perdre de temps à chercher une information dans les codes qui peuvent rapideent devenir illisibles.

Déclaration. Les variables peuvent être de type *resolve*, *async* et *cache*. les variables de type *resolve* sont celles dont les valeurs sont présentes avant le chargement de la page qui les utilise. les variables de type *async* sont celles dont les valeurs proviennent d'un serveur de manière asynchrone ; dans ce cas, le service permettant d'obtenir la donnée doit être préciser. Les variables de type *cache* sont celles dont les valeurs sont mises ou proviennent d'un cache lorsque l'attribut cache a pour valeur `to` ou `from`.

Les actions

Ils constituent un outil graphique permettant d'associer des actions à une vue donnée.

représentent une partie des fonctionnalités du module faisables à partir de la vue encours d'édition. Pour ajouter une fonction, il faut :

— Préciser le nom de l'action ;

— Préciser le rôle de l'utilisateur qui peut y avoir accès ;

— Préciser l'icône de l'action, c'est sur cette dernière qu'il faut cliquer pour exécuter l'action ;

— Préciser les dépendances ; awesome file input

— Enfin écrire le corps de la fonction de l'action

7. collapsible

La configuration

Le concept ici est d'améliorer la productivité du développeur en simplifiant la mise en place des menus, des droits et de la navigation. Ce concept se décline via l'interface de la figure suivante **??**.

Navigation. Elle consiste à construire l'état/route auquel(le) la vue est associée tout en précisant le nom de l'état, l'url associé et le chemin static ou dynamic du template. Ainsi, en parcourant la liste des vues, on peu les connaître les différents états du module.

Menus. La case à cocher « module menu »permet de spécifier si une vue a une entrée ou non dans la liste des menus d'un module. De ce fait, il est possible en parcourant la liste des vues de la figure **??**, d'obtenir les éléments du menu. On pourra non seulement identifier les icônes associées aux menus mais aussi changer l'ordre d'apparition des éléments dans le menu par un simple déplacement dans la liste des vues.

Droits d'accès

3.3 La présentation des éléments graphiques côté serveur

Les resources

Le concept ici est de faciliter l'implémentation des APIs REST. Ceci se décline dans l'interface de la figure suivante. Elle est divisée en trois parties à savoir : la liste des ressources, les détails d'une ressource et les paramètres des ressources.

La liste des ressources . Elle permet d'avoir une vue d'ensemble sur les ressources d'un module. Son organisation en arborescence facilite la détection d'incohérences liées aux URIs

Les Détails d'une ressource. Au delà de l'URI visible dans l'arborescence, on distingue dans cette partie les informations liées à une ressource. Ces dernières peuvent être mise à jour en cliquant sur l'action `update`.

Les paramètres d'une ressource Cette partie donne une vue d'ensemble des `méthodes` d'une ressources et permet aussi d'en créer de nouvelles. Une méthode est la définition d'une opération sur la ressource.

Les opérations

Les opérations sont étroitement liées aux ressources

Les données Externes

Chapitre 4

L'implémentation du générateur

Chapitre 5

conclusion