1 Introduction

2 blablabla

3 Conception ...

3.1 Le concept d'un module

Le concept de module dans une architecture client-serveur représente un ensemble d'éléments côté client interagissant avec d'autres éléments côté serveur.

3.2 Les éléments côté client

La partie cliente dun module est basée sur l'architecture Model Views Controllers et est constituée des différentes pages html du module, des services qui permettent de récupérer les données du serveur via les Contrôleurs.

3.2.1 Présentation de la vue générale

La conception de la partie cliente passe par quelques étapes que sont:

- La création de la vue, en cliquant sur l'icône + de la barre des actions;
- La configuration de la vue sous l'onglet configs;
- La construction de la vue en cliquant sur les composants sous l'onglet composants et l'alignement des composants avec l'arbre des composants
- La déclaration des fonctions et des variables du contrôleur de la vue
- Et enfin la déclaration des actions liées à la vue

3.2.2 Les composants

Nous avons classé les composants en trois catégories que sont :

A) *Les composants simples*; ce sont des éléments HTML classiques.

B) *Les composants conteneurs*; Les composants conteneurs sont des éléments HTML qui peuvent contenir d'autres éléments HTML. Ils regroupent des un ensemble de composants simples, conteneurs, ou widgets.

C) Les composants widgets

3.2.3 L'arbre des composants

L'arbre des composants est la vue globale de lensemble des composants qui constistuent une interface encours de création. Larborescence est obtenue au fur et à mesure que le développeur ajoute des composants à l'interface. Le choix de la structure en arbre ne s'est pas fait au hasard, en effet les composants conteneurs qui regroupent d'autres composants conteneurs font que la structure se construit tout seul. L'objectif visé est de permettre au développeur de perdre moins de temps, à l'indentation de code, à la recherche d'un composants html, à devoir gérer un fichier long fichier HTML et surtout à réécrire les mes composants plusieurs fois. toujours dans le but de faciliter la vie au développeur, on associe des actions à chaque composants de l'arbre. Nous avons mis en place 5 actions permettant la réutilisation de composants à savoir :

- Monter - descendre : elles permettent de faire monter ou de faire descendre localement un composant dans l'arborescence. Ceci permet de réorganiser les composants sur l'interface en terme de conteneur.
- couper : permet de déplacer les 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 dans un autre. par exemple sur une page contenant un formulaire nom, prénom, email et mot de passe, le développeur place un premier champ avec tous ses paramètres, puis clone ce dernier en prenant soins de changer seulement les variables liées aux clones. Ceci permet donc de générer facilement une interface avec des composants semblables.
- Supprimer : enlève tout simplement le composant de l'arborescence.

3.2.4 Les fonctions

Sans perdre de vue l'objectif premier qui est d'améliorer la productivité du développeur, le concept ici est de lui permettre d'avoir une idée du travail réalisé et le reste en terme de fonctionnalités de l'interface encours de création [1],

Erreur d'écriture Il nous est arrivé plus d'une fois de perdre du temps à chercher pourquoi telle ou telle fonction n'est pas appelée au clique avant de nous rendre compte que la fonction déclarée dans la vue est différente de

[1] exemple de phpstorm repliant des fonction.

celle écrite dans le contrôleur. Pour y remédier, nous générons les fonctions à implémenter à partir de l'arbre des composants. Ceci se fait en parcourant les attributs de chaque composant et en extrayant les valeurs des attributs correspondants à des appels de fonctions.

Recherche et vue d'ensemble Cette manière de gérer permet au développeur de voir la liste des fonctions, leurs définitions, de définir celles qui ne le sont pas, de pouvoir faire facilement des recherches dans une fonction précise.

3.2.5 Les actions

Les actions représentent 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;
- Enfin écrire le corps de la fonction de l'action

3.2.6 Les variables

Le cas des variables est semblables à celui des fonctions, en effet il y a plusieurs variables qui soit affiche leurs valeurs soit récupèrent des données (inputs, select, etc...). Elles sont soit extraites de l'arborescence des composants soit déclarées par le développeur.

Déclaration Les variables peuvent être de type resolves, c'est à dire que leurs valeurs sont récupérées bien avant le chargement de la vue. Asynchrones, leurs valeurs sont récupérées lorsque ces dernières sont disponibles, il faut dans cas préciser le service.En fonction de la valeur de l'attribut cache, les variables peuvent être récupérer ou être placer dans le cache.

3.2.7 La configuration

Non seulement cette étape est une étape clé dans la création de l'interface, mais aussi, une grande partie de l'implémentation est cachée au développeur qui ne se contente que de de fournir quelques paramètres. Ceci pour garder le même niveau de simplicité à tous les niveaux. A partir des paramètres fournis ici, nous allons générer l'état de l'application qui fait appel à cette interface ainsi que la liste des rôles du module.

Menus Défini si l'interface encours d'édition est accessible à partir du menu du module, si oui, préciser son rôle et son icône. A partir de là, nous déterminons le menu du module encours de création.

3.3 Les éléments côté serveur

3.3.1 Les resources

3.3.2 Les opérations

3.3.3 Les données Externes

3.4 Overview

3.5 Conception du portail applicatif

Le concept ici est d'implémenter une plateforme de de type PAE pour portail d'application d'entreprise par laquelle les utilisateurs d'une entreprise accèdent aux différentes applications de l'entreprise et aux données correspondantes en fonction de leur profil. l'idéale est donc de garantir la fluidité de la plateforme àfin d'améliorer l'expérience de navigation de l'utilisateur. A cet effet, elle est pensée single page application, application monopage, en gros, du lancement du portail jusq'à la fin de la session, une seule page charge dynamiquement les données en fonction de l'activité de l'utilisateur.Ceci lui de ne pas s'embrouiller et de ne pas confondre l'application installée et la plateforme d'accès.

3.5.1 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

4 Implémentation ...

5 conclusion