Ext.js architecture

Ext JS suit l'architecture MVC / MVVM.

## Structure du projet Avec Ext JS App

----------src

----------resources

-------------------CSS files

-------------------Images

----------JavaScript

--------------------App Folder

-------------------------------Controller

------------------------------------Contoller.js

-------------------------------Model

------------------------------------Model.js

-------------------------------Store

------------------------------------Store.js

-------------------------------View

------------------------------------View.js

-------------------------------Utils

------------------------------------Utils.js

--------------------------------app.js

-----------HTML files

Le dossier app Ext JS résidera dans le dossier JavaScript de votre projet.

L'application contiendra contrôleur, vue, modèle, magasin, fichiers utilitaires avec app.js.

**app.js: le fichier principal de l'endroit où le flux de programme démarre, devra être inclus dans le fichier HTML principal utilisant la balise <script>.** App appelle le contrôleur d'application pour le repos de la fonctionnalité.

**Controller.js: Il est le fichier du contrôleur de l'architecture MVC Ext JS.** Il contient tout le contrôle de l'application, les events listeners, la plus grande fonctionnalité du code. Il a le chemin défini pour tous les autres fichiers utilisés dans cette application comme magasin, vue, modèle, exiger, mixins.

**View.js: Il contient la partie interface de l'application qui apparaît à l'utilisateur.**

**Store.js: Il contient les données localement mises en cache qui doivent être rendues en vue à l'aide d'objets modèle. Store récupère les données à l'aide de proxies dont le chemin est défini pour que les services puissent extraire les données du serveur.**

**Model.js: Il contient les objets qui relient les données du magasin à la vue. Il a mappage des objets de données backend à la vue dataIndex. Les données sont extraites avec l'aide de store.**

Dans MVVM le contrôleur de l'architecture est remplacé par ViewModel.

**ViewModel: Il traite les changements entre la vue et le modèle. Il lie les données du modèle à la vue. Dans le même temps, il n'a pas d'interaction directe avec la vue, il n'a que la connaissance du modèle.**

### Comment cela fonctionne

Par exemple, si nous utilisons un objet modèle à deux - trois endroits dans l'interface utilisateur, si nous changeons la valeur à un endroit de l'interface utilisateur, nous pouvons voir sans même enregistrer ce changement de la valeur des changements de modèle et ainsi se reflèter dans tous les endroits dans UI où que ce soit utilisé le modèle.

Il réduit les efforts des développeurs et les rend plus facile car aucun codage supplémentaire n'est nécessaire pour les données de liaison