

Document Technique

Automne 2014

Fonctionnement général de l’UI

Ce présent document explique le fonctionnement général haut niveau de l’UI, c'est-à-dire les fonctionnalités qu’elle gère et son interaction avec le backend.

Les fonctionnalités présentes dans l’application côté UI sont listées ci-dessous avec les explications nécessaires.

***Upload d’une image IRM vers le serveur distant :***

* Quand l’usager clic sur le bouton upload de l’interface usager, la séquence déclenchée est la suivante :
* Le contrôleur d’angularJS uploadController sous le répertoire upload sera invoqué. Ce dernier appelle la fonction de son modèle pour faire les traitements nécessaires de l’upload.
* La fonction s’assure que :
* L’usager a bien sélectionné une image à uploader
* L’image n’est pas déjà uplodée.
* L’image porte la bonne extension.
* L’usager a fait son premier upload d’images ou non. Dans le cas du premier upload, la fonction génère un token aléatoire qui sera le nom de son répertoire d’upload sur le serveur. Si ce n’est pas le cas, elle garde le token généré à son premier upload.
* Si les conditions ci-dessus sont remplies, la fonction fait un post vers le serveur distant en lui envoyant comme information le fichier à uploadé et le token.
* La fonction reçoit par après la réponse du serveur :
* Si la réponse est positive c'est-à-dire que l’upload s’est bien déroulé côté serveur, alors elle stocke les informations relatives au fichier uploadé tel que le path de son header data, le path de son raw data, le nom du fichier. Ces informations sont utiles par exemple pour faire un GET de l’image côté serveur ou supprimer l’image, etc … Ensuite la fonction ajoute le nom du fichier dans l’arborence des fichiers uploadé pour que l’usager puisse faire des manipulations dessus.
* Si la réponse est négative, un message d’erreur notifiera l’usager.

***Géneration dynamiques des formulaires des process :***

Quand l’usager sélectionne un process depuis la liste des process, la séquence déclenchée est la suivante:

* Le contrôleur d’angularJS commandsGeneratorController sous le répertoire commandGeneratorController sera invoqué.
* Ce contrôleur fait une requête HTTP post au serveur pour récupérer le fichier de configuration du process en question.
* Le contrôleur appelle une fonction de son modelé generateDynamicHTML(...) pour générer dynamiquement le formulaire HTML en fonction des paramètres du fichier JSON.

L’usager peut maintenant faire entrer les paramètres de commandes. Au moment où il appuie sur le bouton Run Process, la séquence est la suivante :

* Le contrôleur appelle la fonction verifEmptyMandatoryField(...) de son modelé pour vérifier s’il y’a des champs obligatoires non remplis par l’usager.
* Si tout est correct, le contrôleur fait appel à la fonction generateCommand(...) de son modèle pour associer chaque commande avec l’entrée de l’usager.
* S’il y’a des champs obligatoires non remplis, un message d’erreur notifiera le client.
* Si tout est correct, le contrôleur appelle la fonction sendCommandToServer(...) de son modèle, qui se chargera de faire une requête POST HTTP vers le serveur qui se chargera d’éxécuter la commande en question.
* La fonction reçoit par après les informations de l’exécution de la commande du serveur. S’il y’a des fichiers output genérés côté serveur elle les affiche côté UI dans l’arborescence pour que l’usager fait des traitements avec, en plus elle affiche les output du script exécuté sur le serveur dans l’onglet Output de l’UI. Ceci est utile pour superviser qu’est-ce qui est déroulé réellement comme traitement sur le serveur.
* Les deux modèls du contrôleur commandGeneratorController sont:
* commandOperations : contient les fonctions nécéssaires pour faire toutes sortes de manipulations sur les commandes (i.e envoi au serveur, génération de la commande ...)
* HTMLOperations : contient les fonctions nécessaires pour faire toutes sortes de manipulations HTML (i.e générer le formulaire dynamique, vérifier les champs obligatoires vides du formulaire, initialiser le formulaire ...)

***Gestion des fichiers/ dossiers/ volumes affichés :***

Ce module concerne :

* la suppression du nom de l’image uploadé de l’UI et du serveur si l’usager clic sur le bouton de suppression.
* La gestion des volumes d’affichage de brainbrowser en fonction de ce qui est coché comme images dans l’arborescence.

Chacun des deux points précédents est expliqué ci-dessous.

***Suppression fichier côté UI et côté serveur :***

Quand l’usager clic sur le bouton de suppression d’une image, la séquence déclenchée est la suivante :

* Le contrôleur d’angularJS filesManagerController sous le repertoire filesManager sera invoqué.
* Le contrôleur appelle la fonction deleteFile(...) de son modèle deletFileOperations qui se charge de:
* enlever le nom du fichier de l’arborescence.
* Appeler la fonction deleteFromServer(...) qui se charge d’envoyer une requête HTTP POST au serveur afin qu’il supprime l’image. La fonction reçoit par après la réponse du serveur pour notifier le client s’il y’a eu des erreurs lors de la suppression.

***Gestion des volumes affichables de brainbrowser :***

Ces fonctionnalités sont sous la direction du contrôlleur AngularJS filesManagerController.

Les fonctions du modèle volumesOperations sont invoquées par le contrôleur dans les cas suivants :

* À chaque fois que l’usager décoche une image dans l’arborescence des images téléversées, la fonction displayVolumes(...) du contrôleur appelle la fonction displayVolumes de son modèle pour qu’il mette à jour sa liste d’éléments sélectionnés.

Deux cas de figure se présentent :

* Il n’a pas d’images cochées dans la liste. On fait appel à la fonction generateStartupImageVolume(..) qui se charge de construire le paramètre d’affichage nécessaire à Brainbrowser pour faire sa requête GET côté serveur de l’image vide (fond noir) et l’afficher ensuite dans les panel de BrainBrowser.
* Il y’a des images cochées dans la liste : on fait appel à la fonction generateVolumes(...) qui se charge de créer le parametre de BrainBrowser contenant les informations nécessaires à ce dernier pour faire les requêtes GET au serveur et afficher notamment ces images par la suite dans les panel de BrainBrowser.

***Suppression du répertoire de l’upload sur le serveur :***

Cette action se déclenche au moment où l’usager essaie de fermer l’onglet et confirme sa fermeture.

La fonction du deleteUserRepositoryFromServer() est appelée à partir de la fonction deleteFilesSession() présente dans le fichier sessionClose.js. Ce dernier fichier comporte les fonctionnalités concernant la fermeture de la session de l’usager.

La fonction deleteUserRepositoryFromServer(...) envoi une requête POST au serveur qui se chargera de supprimer le répertoire de l’usager côté serveur.

**</ END OF DOCUMENT>**