

Document Technique

Automne 2014

Configuration des fichiers JSON de l’application

* L’application comporte deux types de fichiers json à configurer au besoin:

1- Un fichier contenant le nom de tous les process de la toolbox.

2- Les fichiers de configurations de chaque process.

Nous détaillerons la configuration de ces deux types de fichiers dans les parties ci-dessous.

**List\_scripts.JSON**

* C’est le fichier qui contient les informations sur les scripts de la ToolBox.
* Son path sous Git est: inf4990-02\spinal\src\main\resources\viewRessouces\js\commandsInputGenerator\data\liste\_scripts.json
* Son path sous le répertoire de déploiement du serveur est :

ROOT\WEB-INF\classes\commandsInputGenerator\data\liste\_scripts.json

La structure de ce fichier est la suivante:

* Il s’agit d’un tableau d’éléments ( car présence des [ ] au début et la fin du fichier) qui sont les informations relatives à chaque process.
* Ci-dessous un exemple d’élément contenant les informations d’un script en particulier de la ToolBox.

{

**"title"**: "sct\_apply\_transfo",

**"pathConfig"**:"/resources/viewRessouces/js/commandsInputGenerator/data/config/sct\_apply\_transfo\_config.json",

**"activateScript"** : "true"

}

**title** : c’est le nom du script qui va être affiché dans la liste déroulante des process dans le UI.

**pathConfig** : c’est le path physique du fichier de configuration du script en question. Il contient les informations nécessaires pour la generation du formulaire côté UI relative à la commande que l’usager voudra exécuter. La structure de ce fichier de configuration sera expliquée dans les prochaines parties de ce document.

**activateScript** : sa valeur est “true” si on veut exposer le script dans la liste déroulante des process côté UI. Sa valeur est à “false” dans le cas contraire.

* Si vous ajoutez un nouveau script dans la ToolBox, ajouter ses informations associées dans le fichier List\_scripts.json comme expliqué en haut.
* Pas besoin de classer les process par ordre alphabétique au niveau du champ **title** dans List\_scripts.json car c’est géré dans l’application.

**Les fichiers de configurations des process**

* Chaque process a son fichier de configuration propre à lui.
* Ce fichier contient les informations nécessaires pour générer dynamiquement dans le UI le formulaire associé au script qui sera rempli par l’usager afin de générer sa commande.

La structure du fichier est la suivante:

* **commandName** : c’est le nom du process (i.e : sct\_proposeg)
* **scriptName** : doit être le même que **comandName**
* **additionalScriptGenerator** : comporte le nom des fonctions spéciales additionnelles qui feront un certain traitement pour associer les bons arguments à la commande mentionnée.

Prenant comme exemple le script sct\_labels\_utils :

**"additionalScriptGenerator"** :

[

{

**"command"**: "-x",

**"specialScript"** : "labelsGenerator"

}

]

Comme l’argument -x requiert un argument du type x,x,x:y,y,y:...:.z,z,z qui provient des labels tagués par l’usager, le formulaire HTML dynamique ne peut faire le formatage de cet argument car il est générique et non spécifique à un script en particulier.

On fait donc appel à un script externe contenant la fonction labelsGenerator qui se charge de lire les labels tagués par l’usager, de les formatter, les concaténer avec le champ **command** qui est dans ce cas -x, puis concaténer le tout avec la commande générée à partir du formulaire dynamique générique.

Cela dit pour chaque nouvelle fonctionnalité qu’on désire ajouter, il faut crée un fichier script dédié à cette fonctionnalité et d’ajouter son nom dans la section **additionalScriptGenerator** tout en l'associationavec la commande en question.

* **arguments** : il s’agit d’un tableau d’éléments. Chaque élément contient les informations relatives aux composants HTML appartenant à une section en particulier (par exemple les éléments appartenant à la section mandatory et les éléments appartenant à la section optional).
* Les éléments du champ **arguments** est la suivante:

Ci-dessous un exemple d’élément:

{

**"argumentsSection"** : "MANDATORY ARGUMENTS",

**"parameters"** :

[

{

**"command"** : "-i",

**"id"** : "sct\_label\_utils\_i\_ID",

**"description"** : "labels or image to create labels on. Must be 3D",

**"HTMLRendering"** :

{

**"HTMLType"** : "file",

**"Title"** : "label or image"

},

**"mandatory"** : "true"

},

{

**"command"** : "-t",

**"id"** : "sct\_label\_utils\_t\_ID",

**"description"** : "process : cross -c/remove -r/display-voxel/create -l",

**"HTMLRendering"** :

{

**"HTMLType"** : "text",

**"Title"** : "warping field"

},

**"mandatory"** : "true"

}

]

}

**ArgumentSection** : c’est le nom de la section tel qu’elle va être afficher dans le formulaire generé.

**parameters** : c’est un tableau d’élements. Chaque element est contient les informations relatives à un composant HTML en particulier.

Ci- dessous un exemple de parametre :

{

**"command"** : "-i",

**"id"** : "sct\_label\_utils\_i\_ID",

**"description"** : "labels or image to create labels on. Must be 3D",

**"HTMLRendering"** :

{

**"HTMLType"** : "fileList",

**"Title"** : "label or image"

},

**"mandatory"** : "true"

}

**command** : le nom de la commande

**id** : identifiant de la commande. TRÈS IMPORTANT qu’il soit unique d’un élément à un autre.

**Description** : description de la commande.

**HTMLRendering** : c’est un element contenant les informations relatives à l’affichage HTML.

**HTMLType** : c’est le type du composant HTML.

L’application gère les types HTML ci-dessous:

* fileList : s’il s’agit un fichier en input.
* text : s’il s’agit d’un champ de texte.
* number : s’il s’agit d’un nombre
* checkbox : s’il s’agit des cases à cocher
* select : s’il s’agit d’une liste déroulante d’éléments.

**Title** : Titre de la commande tel qu’il va être affiché dans le formulaire généré.

**mandatory** : “true” si la commande doit être renseignée par l’usager avant de lancer le process. “false” dans le cas contraire.

**</ END OF DOCUMENT>**