jump scripted plugin manager

Gestionnaire de scripts pour JUMP

Licence

Historique des versions

Version 0.1.0 (15 Jan 2005)

Résumé

Ce document décrit le mode d'installation et de fonctionnement du plug-in JUMP-SPIM.

Table des matières

1. Introduction

1.1. Qu'est ce que jump-spim ?

Jump-spim est un petit plugin permettant d'ajouter à JUMP des plugins sous forme de scripts beanshell (voir aussi le plugin BeanShellEditor).

1.2. Que puis-je faire avec jump-spim ?

Lier un élément de menu avec un script beanshell quelconque. Sachant que beanshell vous donne accès à toute l'API java ainsi qu'aux API de JUMP et de JTS, les possibilités d'ajouter des fonctionnalités à JUMP sont quasiment illimitées. De plus, une fois que vous avez établi un lien entre un élément de menu et votre script, vous pouvez modifier et améliorer ce dernier très simplement : il vous suffit d'éditer votre fichier de script (avec un éditeur de texte ou avec BeanShellEditor), de le modifier, et de le relancer à l'aide de l'élément de menu déjà défini.

1.3. Installation

Il vous suffit de placer le fichier jump-spim.jar dans le répertoire de jump dédié aux plugins (répertoire ext par défaut) et de lancer JUMP. Le plugin JUMP-spim est automatiquement

activé.

2. Comment ajouter un plugin écrit en script à JUMP

Pour qu'un script beanshell puisse être appelé à partir du menu de JUMP, il faut avoir :

- jump-spim.jar installé dans le répertoire des plug-in
- un script beanshell qui fonctionne (que vous pouvez avoir élaboré avec BeanShellEditor)
- un fichier de configuration appelé `ScriptedPlugInManager.txt` placé dans le répertoire de lancement de JUMP (voir § 2.1)
- optionnellement, vous pouvez placer un script d'initialisation `init.bsh` dans le répertoire de lancement de JUMP (voir § 2.2)

2.1. Le fichier de configuration

Vous devez créer, s'il n'existe pas, un fichier appelé `ScriptedPlugInManager.txt` à l'endroit d'où votre programme JUMP est lancé (à coté du `.exe` ou du `.bat` sous windows).

Ce fichier éditable, très simple, permet de créer et de faire correspondre des éléments de menu à vos scripts beanshell. Il contient une paire clé/valeur par ligne, la cle et la valeur étant séparées par le signe '='. La clé identifie l'élément de menu, tandis que la valeur contient l'adresse du fichier et de la fonction script à exécuter.

Exemple de fichier :

```
// Plugins

ScriptedPlugIns.MyPlugIns.LoadData = bsh/MyPlugIns/loaddata.bsh

ScriptedPlugIns.MyPlugIns.MergeRoads = bsh/MyPlugIns/Merge.bsh;mergeRoads

ScriptedPlugIns.MyPlugIns.MergeHighways = bsh/MyPlugIns/Merge.bsh;mergeRoads;Hi
```

Sur la première ligne, vous avez un commentaire, commençant par //. Vous pouvez insérer des commentaires ou des lignes blanches n'importe où dans le fichier `ScriptedPlugInManager.txt` file.

La deuxième ligne, associe deux éléments. Le premier, **ScriptedPlugIns.MyPlugIns.LoadData** désigne un élément de menu (menu ScriptedPlugIns, sous-menu MyPlugIns, commande LoadData). Le deuxième indique le chemin relatif du script à exécuter (le chemin est décrit par rapport au répertoire d'où JUMP a été lancé).

La troisième ligne ressemble à la deuxième, sauf que le chemin d'accès au script est suivi d'un point-virgule puis d'un nom de méthode. Ceci veut dire que lors de l'appel de l'élément de menu ScriptedPlugIns --> MyPlugIns --> MergeRoads, le script `bsh/MyPlugIns/Merge.bsh` sera entièrement exécuté, puis, la méthode sans argument **mergeRoads** sera appelée. Celà est utile, pour accéder à une des fonctions d'un script qui ne fait que définir des méthodes.

Dans le troisième plugin, la deuxième partie de la définition posséde à la suite du nom de méthode, la définition d'un argument. Les arguments sont toujours séparés de la méthode et entre eux par des points-virgules. On ne peut passer que des arguments textuels (comme par exemple ici le nom d'un layer).

2.2. Créer un script d'initialisation

Afin de faciliter la réutilisation de fonctions, de constantes et d'imports par de nombreux plugins, vous pouvez créer un script beanshell qui sera systématiquement exécuté avant un "scripted-plugin". Ce script doit s'appeler `init.bsh`, se situer aux cotés de `ScriptedPlugInManager.txt`, et pourrait par exemple ressembler au script suivant :

```
import com.vividsolutions.jump.feature.*;
import com.vividsolutions.jump.task.*;
import com.vividsolutions.jump.tools.*;   // OverlayEngine
import com.vividsolutions.jump.ui.*;
import com.vividsolutions.jump.util.*;
import com.vividsolutions.jump.util.java2xml.*;
import com.vividsolutions.jump.workbench.*;
import com.vividsolutions.jump.workbench.model.*;
import com.vividsolutions.jump.workbench.plugin.*;
import com.vividsolutions.jump.workbench.ui.*;
import com.vividsolutions.jump.workbench.ui.plugin.*;
import com.vividsolutions.jump.workbench.ui.wizard.*;

import com.vividsolutions.jts.algorithm.*;
import com.vividsolutions.jts.geom.*;
import com.vividsolutions.jts.geom.util.*;
import com.vividsolutions.jts.geomgraph.*;
import com.vividsolutions.jts.index.strtree.*;
import com.vividsolutions.jts.noding.*;
import com.vividsolutions.jts.operation.buffer.*;
import com.vividsolutions.jts.operation.distance.*;
import com.vividsolutions.jts.operation.linemerge.*;
import com.vividsolutions.jts.operation.overlay.*;
import com.vividsolutions.jts.operation.polygonize.*;
import com.vividsolutions.jts.operation.relate.*;
import com.vividsolutions.jts.operation.valid.*;
import com.vividsolutions.jts.planargraph.*;
import com.vividsolutions.jts.precision.*;
import com.vividsolutions.jts.util.*;

//************************************************************************
// USEFUL HANDLES AND BASIC FUNCTIONS TO ACCESS JUMP LAYER
//************************************************************************

LayerManager layerManager = wc.getLayerManager();      // wc est une variable d

Layer[] getSelectedLayers(){return wc.layerNamePanel.getSelectedLayers();}

Layer getSelectedLayer(){
    layers = wc.layerNamePanel.getSelectedLayers();
    if (layers.length>0) return layers[0];
    else print("One layer must be selected");
}

FeatureCollection getFeatureCollection(String layer) {return wc.getLayerManager

java.util.Collection getSelection(){return wc.getLayerViewPanel().getSelectionM

HTMLFrame outputFrame = wc.workbench.frame.outputFrame;
```

2.3. Exemple pas à pas

- Placer jump-spim.jar dans le répertoire ext de votre installation de jump.
- Créer le fichier `ScriptedPlugInManager.txt` avec un éditeur de texte et placez-le dans le répertoire de votre exécutable (.exe, .bat, .sh,...)

- Ecrire dans le fichier `ScriptedPlugInManager.txt` la ligne suivante :

  ```
  MesPlugIns.MonPlugIn.Comptage = bsh/comptage.bsh
  ```

- Créer le fichier script suivant avec un éditeur de texte ou avec le plugin BeanShellEditor

  ```
  layers = wc.layerManager.layers;

  for(layer:layers) print(layer + " : " + layer.featureCollectionWrapper.size +
  ```

- Appelez ce fichier `comptage.bsh` et placez-le dans le répertoire `bsh` (le répertoire `bsh` est au même niveau que l'exécutable et que le fichier `ScriptedPlugInManager.txt`)
- Lancez JUMP. Un élément de menu MesPlugIns --> MonPlugIn --> Comptage apparait. Lancez-le, et lisez dans la fenêtre sortie le résultat des opération d'écriture (print) du script.

A. Historique des changements

- **Version 0.1.0** (15 jan 2005)
  - Version initiale