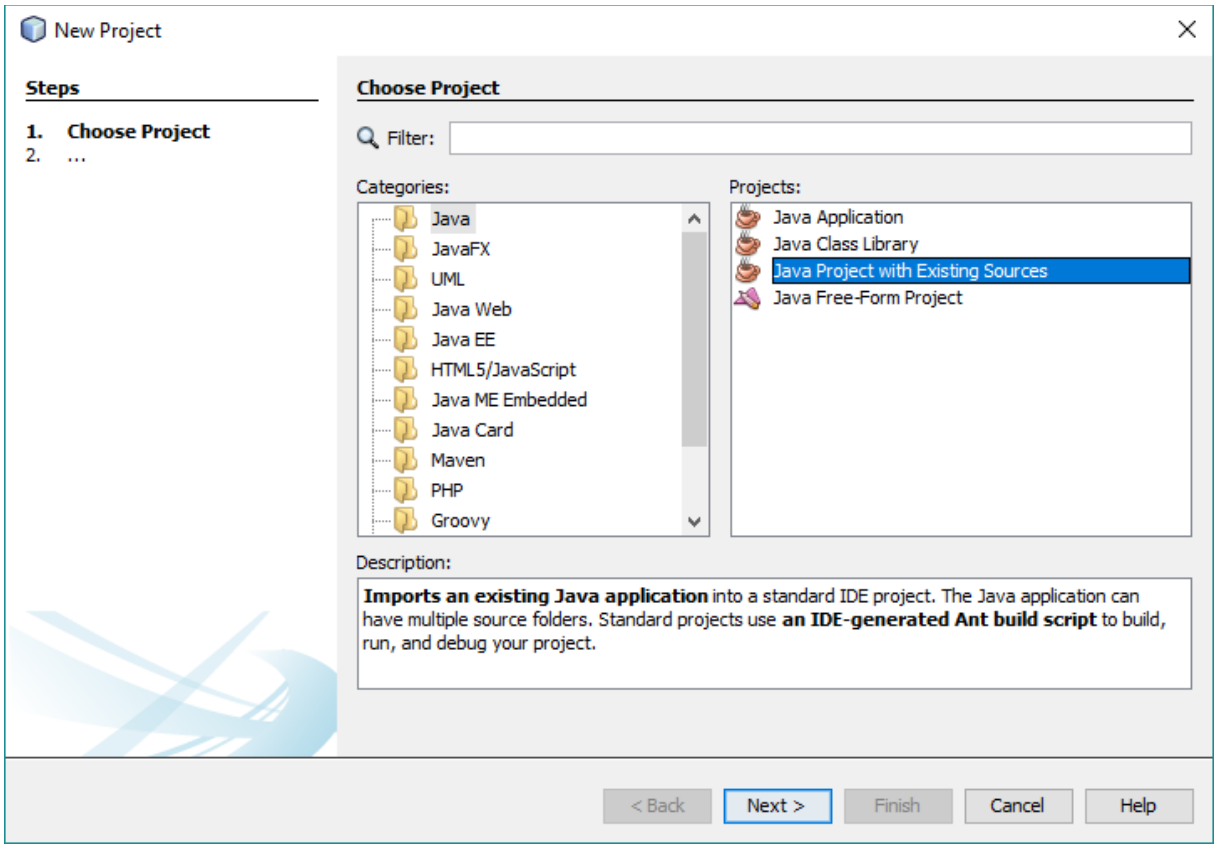
Documentation à l'intention des développeurs

Table des matières

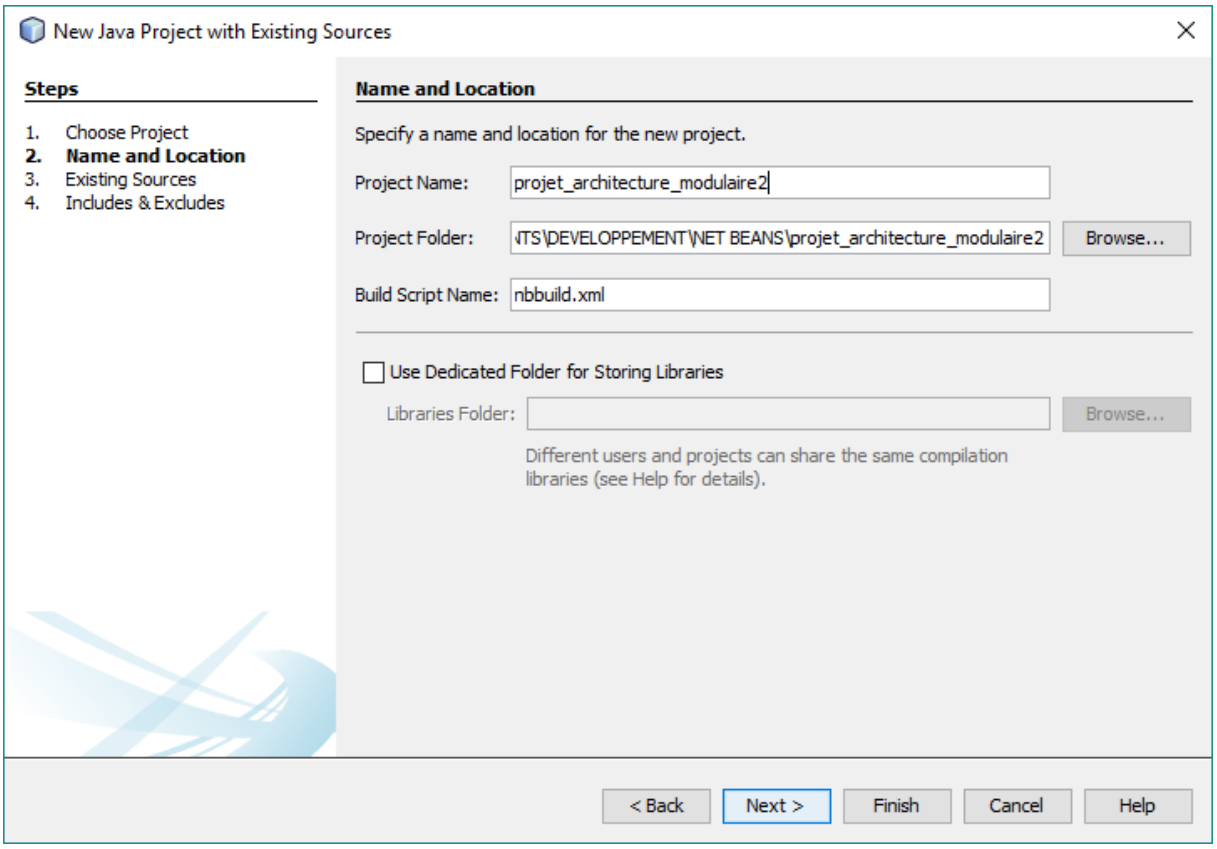
Préparation de l'environnement de développement

IDE Net Beans

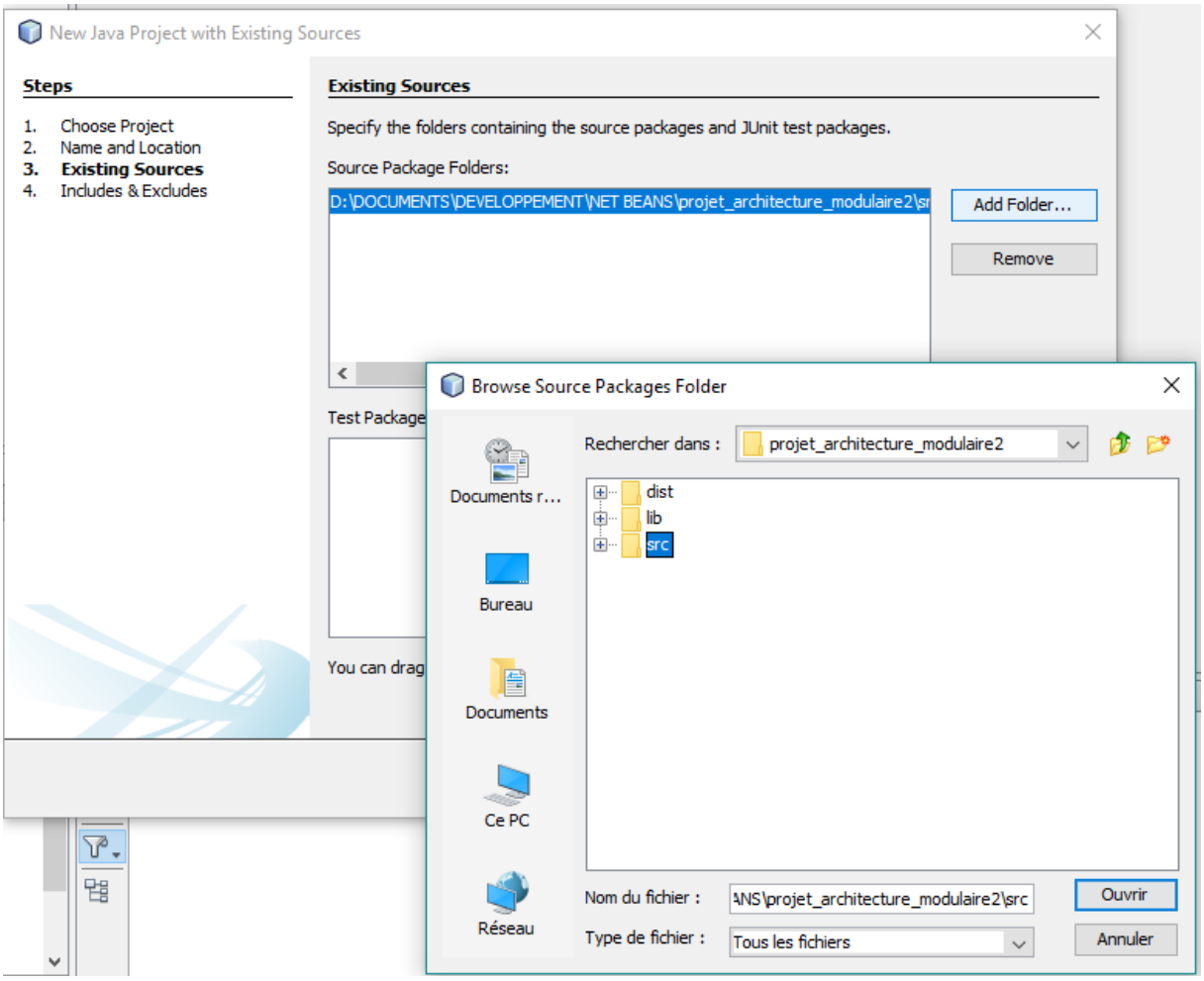
1- Extraire l'archive contenant le code source
2- Créer un nouveau projet java à partir des sources existantes :



3- Sélectionner l'emplacement des sources et donner un nom à votre projet :
Vous pouvez placer directement les sources dans le répertoire que vous indiquez à l'emplacement « project folder » :

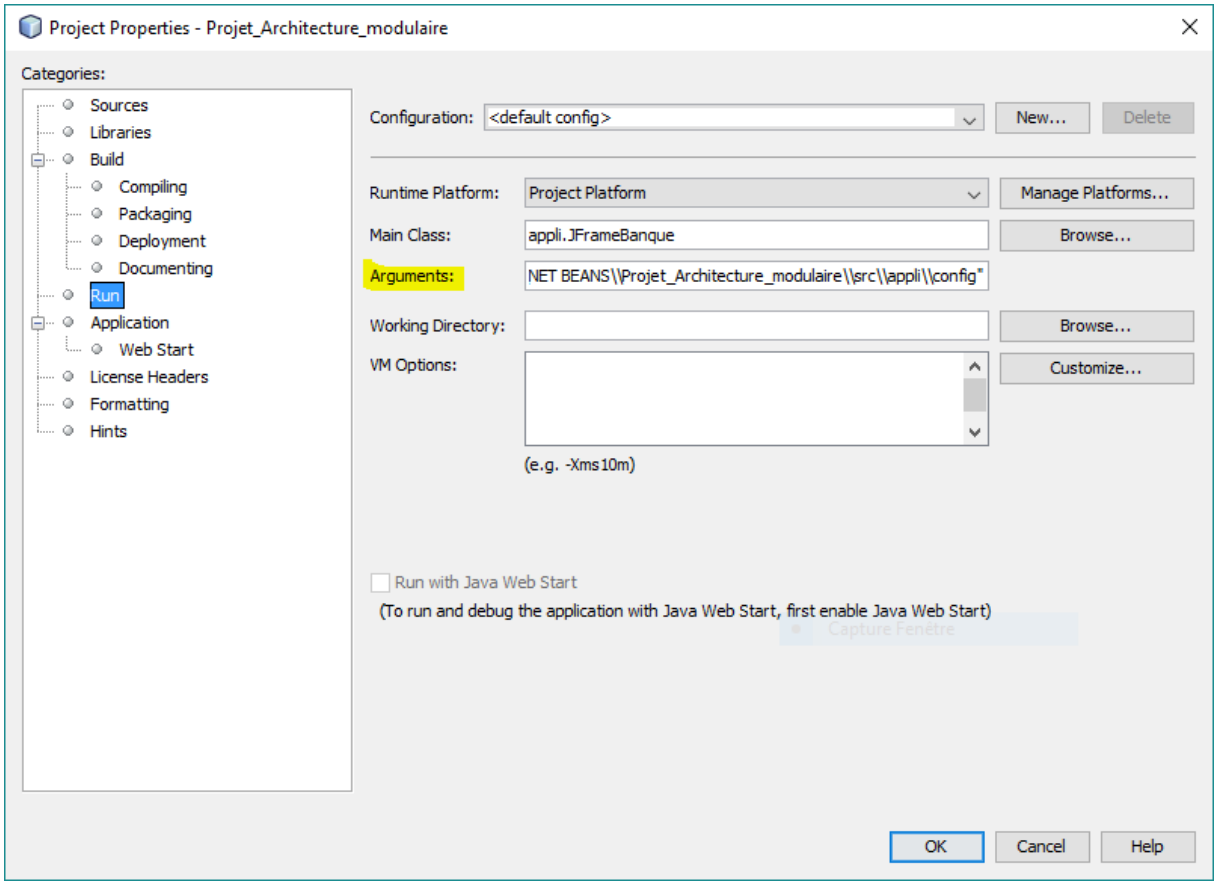


4- Sélectionner le répertoire correspondant aux sources :



5- Cliquer sur « finish », vous pouvez maintenant commencer votre développement

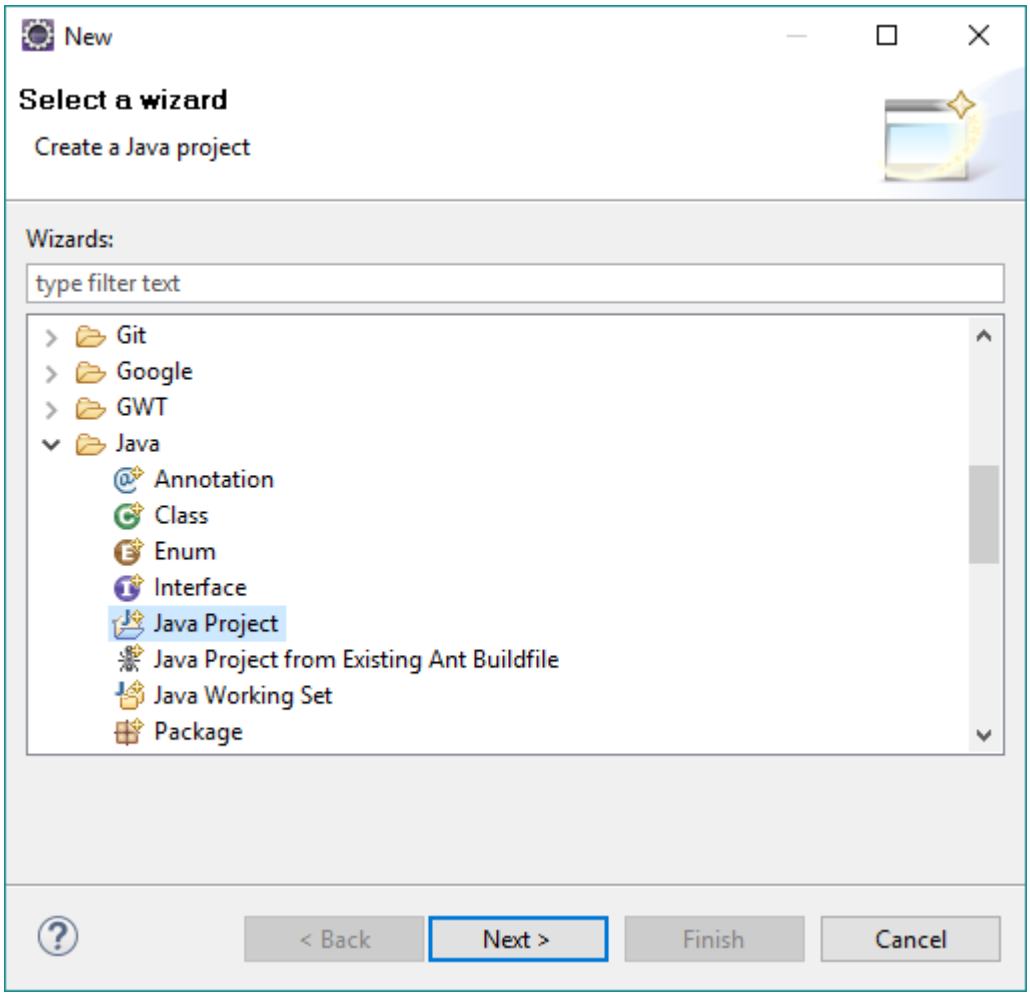
Pour exécuter le programme, il est nécessaire de lui passer un argument. Celui-ci correspond au chemin absolu du répertoire contenant les fichiers de configuration des plugins :



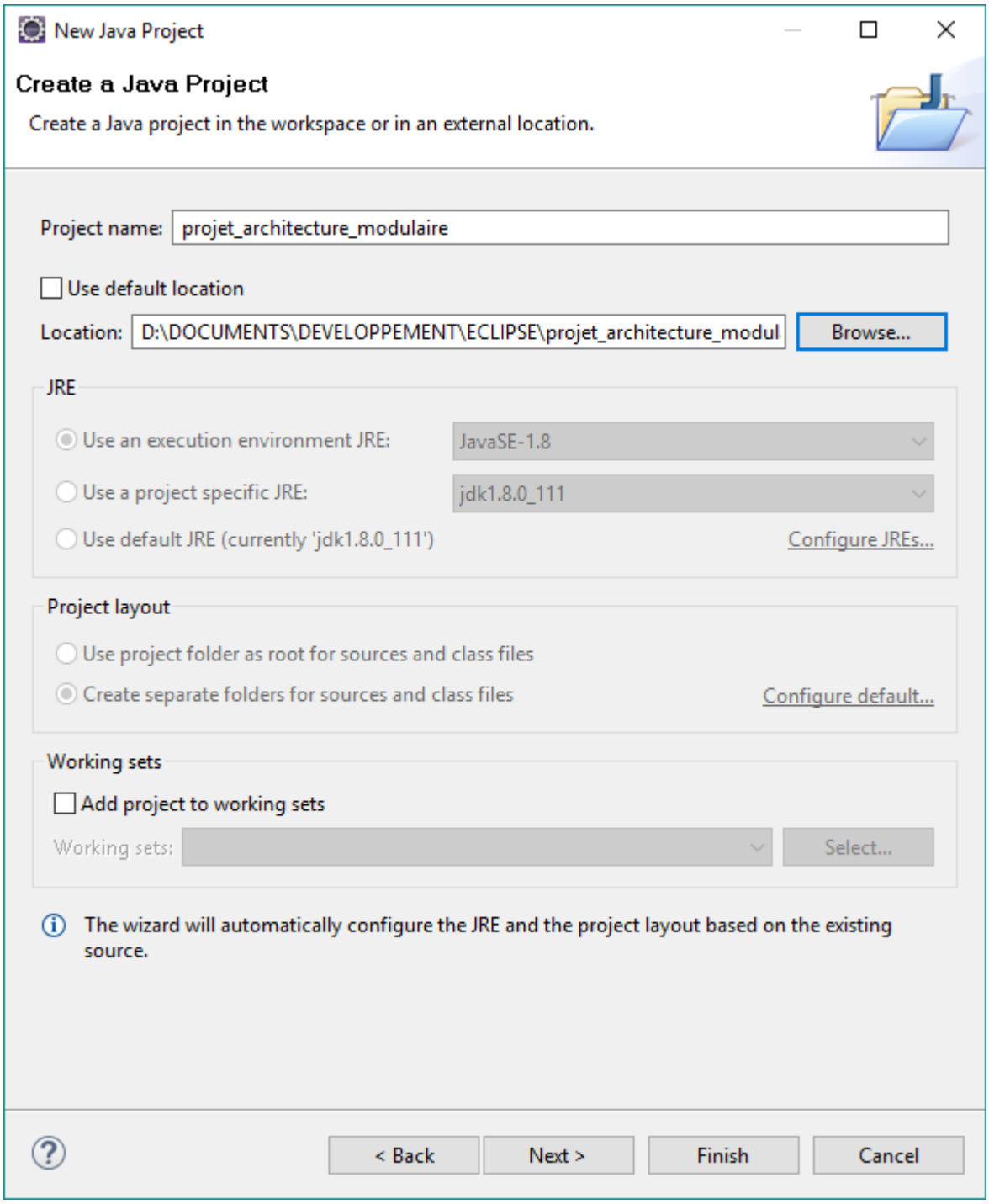
IDE Eclipse

Vous devez créer un nouveau projet à partir des sources de code présente dans l'archive, pour cela :

1- Désarchiver les sources dans un répertoire
2- Créer un nouveau projet JAVA :

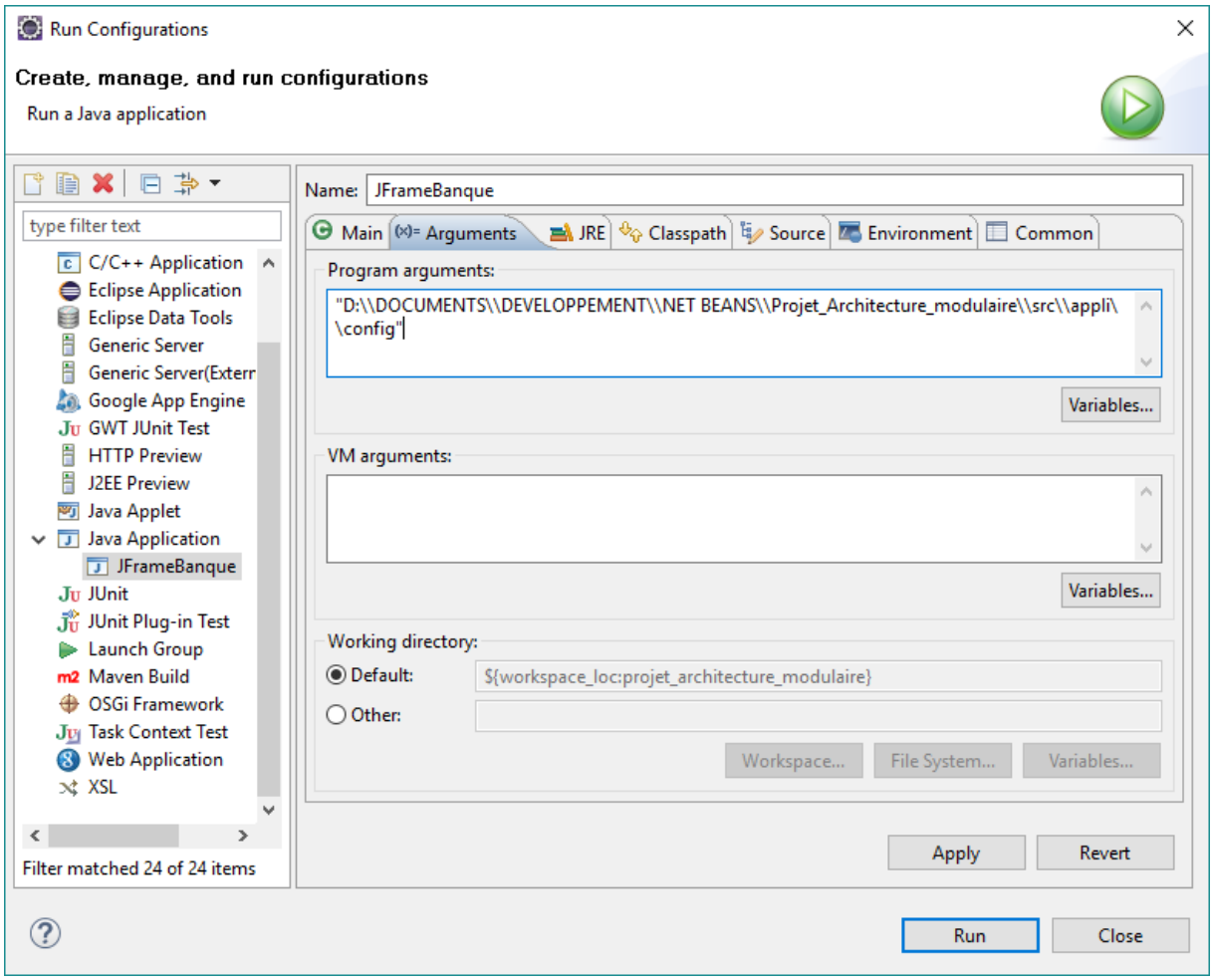


3- Décocher la case « use default location » pour pouvoir sélectionner l'emplacement où se trouve les sources du projet



4- Dans la fenêtre suivante, contrôler que le package des sources est bien sélectionné, puis valider. Vous pourrez alors développer.

Pour exécuter le programme, il est nécessaire de lui passer un argument. Celui-ci correspond au chemin absolu du répertoire contenant les fichiers de configuration des plugins :



Attention : s'il est nécessaire de modifier l'interface graphique, cela est plus facile en utilisant l'IDE NetBeans car la première version de l'interface graphique a été généré en utilisant l'outils de design de NetBeans. Il est cependant possible d'apporter des modifications à l'interface en modifiant directement de code.

Structure du projet

Notre projet est divisé en trois packages :

- Appli
- Plateforme
- Plugins

Nous avons ainsi souhaité séparer les différentes « vues ».

La partie client se situe dans le package appli. Nous y trouvons le modèle de données, les différentes interfaces qui permettent d'avoir des types de plugins, l'IHM ainsi que les fichier de configurations des plugins

La partie plateforme contient les différentes fonctions permettant de préparer un plugin à l'utilisation cliente.

Enfin, la partie plugins contient tous les plugins disponibles. Aujourd'hui nous trouvons deux types de plugins : les plugins permettant un affichage des données et les plugins permettant une modification. Chaque plugin respecte une description. Cette description est visible par le client, elle correspond aux fichiers de configurations.

Ajout d'un plugin

Un plugin correspond à une classe dans le package « plugins ». Il faut tout d'abord créer une classe et les différentes fonctions qui seront disponibles au travers de ce plugin.

Une fois la classe créée, si elle ne correspond à aucun des interfaces déjà présentes, nous pouvons extraire l'interface correspondante. Cette interface sera ajoutée dans le package « appli.data », contrairement au plugin, elle est connue de l'IHM.

Afin de charger le plugin, il faut lui associer un fichier de configuration. Le fichier de configuration devra respecter le format suivant :

```
nom= NomDuNouveauPlugin
contrainte=appli.data.INomDeLInterfaceAssociee
class=plugins.NomDuNouveauPlugin
```

Enfin, le plugin peut être utiliser dans l'IHM. Pour cela, il faut faire appel au Loader et récupérer la description du plugin parmi la liste des plugins chargés dans l'IHM. Si c'est une nouvelle interface, il faudra également charger une nouvelle liste. Deux lignes de codes sont importantes :

`loader.getPluginsDescriptions(NomInterface.class)` → permet de charger la description de toutes les descriptions des plugins correspondant à l'interface fournie.

Exemple de chargement d'un plugin à partir de son nom :

```
IAfficheur afficheur = (IAfficheur)
Loader.getInstance().getPlugin(this.getDescPlugins("afficheroperatio
n"));
```