

Savoirs-faire intéressants

- Sniffer (= regarder) les packets échangés
- Créer automatiquement des diagrammes pour tes rendus
- Compiler un projet avec Gradle et JavaFX

Sniffer (= regarder) les packets échangés

Pour les 3MIC et 4IR et pour tous les autres qui doivent à un moment faire un programme qui échange des données en réseau, déboguer peut être pénible. C'est là qu'on peut vérifier que les packets s'envoient bien (ou non) pour savoir où rechercher l'erreur.

Pour ce faire, on va utiliser Wireshark qui est disponible sur n'importe quelle distribution digne de ce nom (*insérer ici une blague sur Haiku*) qu'on peut installer par exemple sous Ubuntu avec :

```
sudo apt install wireshark
```

Une fois ouvert, il vous faudra sélectionner une interface (ou tous) sur lequel écouter.

Vous pourrez dans la barre en haut filtrer les packets avec des règles (et c'est recommandé au vu du nombre de packet qui passent ?).

Exemple de filtres:

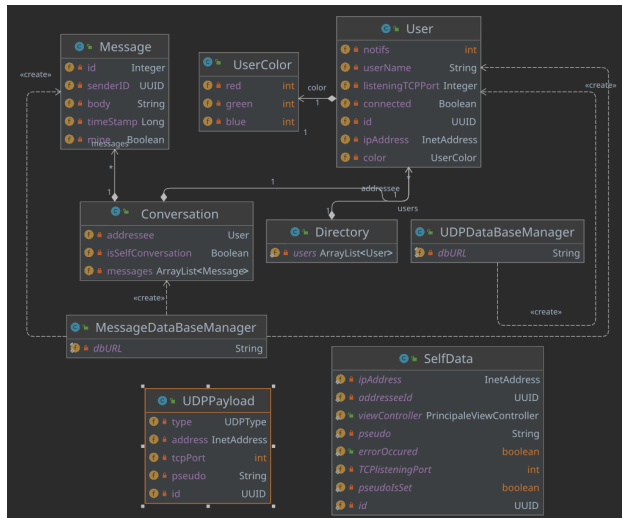
```
// ne laisse passer que les packets TCP
```

```
tcp
```

```
// ne laisse passer que les packets TCP entre 192.168.1.33 et 192.168.1.65
```

```
tcp && ((ip.src == 192.168.1.33 && ip.dst == 192.168.1.65)|| (ip.dst == 192.168.1.33 && ip.src == 192.168.1.65))
```

Permet de faire des diagrammes comme ça en 2 clics:



Générer un diagramme de Classe ... avec classe

Tuto pour utiliser le générateur installé de base

Générer un diagramme de séquence sans galérer à positionner les acteurs

Installez ce plugin et suivez ce tuto

Compiler un projet avec Gradle et JavaFX

Les exemples fournis ici sont codés avec Gradle KTS, si tu codes avec Groovy, voici un lien vers le projet OpenJFX qui explique la mise en place de JavaFX sous Gradle :

<https://github.com/openjfx/javafx-gradle-plugin>

Tu es actuellement en plein projet Java de 4A-IR (ou en plein projet personnel) et tu cherches à build ton projet JavaFX (car Swing, c'est quand même pas ouf) et Gradle ? Tu es au bon endroit !

Conseils de logiciels (IDE)

Pour ce faire, je te conseille tout d'abord d'utiliser l'IDE IntelliJ IDEA, qui est gratuit avec ton adresse mail étudiant (petite aide pour l'installer : <https://wiki.etud.insa-toulouse.fr/books/licenses-logiciels/page/jetbrains>), et qui te permettra bien des galères par rapport à Eclipse. Néanmoins, le tutoriel est censé fonctionner quelque soit l'IDE ou l'éditeur utilisé (même emacs pour les plus téméraires !)

Configuration

Une fois cela effectué, ouvre ton fichier `build.gradle.kts`, et procède aux modifications suivantes :

- Dans la partie plugin (à créer si nécessaire) :

```
plugins {  
    id("org.openjfx.javafxplugin") version "0.0.13"  
}
```

Liste des versions supportées par Gradle : <https://mvnrepository.com/artifact/org.openjfx/javafx-plugin?repo=gradle-plugins>

- Créer une partie `javaFx` avec les informations suivantes :

```
javaFx {  
    version = "17.0.2" // Ici ta version de JavaFX qui correspond à ta version de Java
```

```
modules("javafx.controls", "javafx.fxml")
}
```

Dans ce code d'exemple, seules les fonctions de base sont ajoutées afin de ne pas allonger le temps de compilation. Pour obtenir l'ensemble des modules disponibles (pour le support du BootStrap par exemple) : <https://openjfx.io/javadoc/11/>

- Dans des `dependencies`, ajouter :

```
dependencies {
    implementation("org.controlsfx:controlsfx:11.1.1")
    // Dans le cas d'ajout de bootstrap dans les modules
    implementation("org.kordamp.bootstrapfx:bootstrapfx-core:0.4.0")
}
```

Et... c'est tout ! C'était pas si compliqué ? ?

Build !

Pour build ton projet, lance la commande `gradle build` sur ton terminal, et normalement ton application devrait se lancer ! :)

Problèmes fréquents

L'application ne trouve pas de main

-> N'oublie pas de déclarer ta classe contenant le main que tu veux exécuter dans ton `build.gradle.kts`

```
application {
    // org.example.Main à remplacer par le chemin vers la classe qui contient ton main
    mainClass.set("org.example.Main")
}
```

J'utilise un Mac avec processeur ARM (M1/M2) et le build m'indique que JavaFX ne peut pas s'installer

-> Cela se produit car la version ARM (aarch64) n'existe pas par défaut lors de l'installation par gestionnaire Gradle. Pour ce faire, exécutes une première fois ton build sur les machines de l'INSA, push ton projet sur [Git](#) et relance-le sur ton Mac. Une fois cela effectué tout devrait fonctionner :)

Attention : il ne faut pas que le paramètre compileOnly soit activé, auquel cas cette manipulation ne fonctionnera pas !