Deboguage

Debuggueur minimal

Il faut activer :

```
-Xdebug -Xrunjdwp:transport=dt_socket,address=5555,server=y,suspend=n
```

Puis on peut utiliser la commande `jdb` :

```
jdb -attach localhost:5555
# puis qque chose du genre
stop in org.esupportail.portal.services.ProlongationENT.doGet
```

ou mieux, installer rlwrap pour avoir un support “readline” (historique) puis faire :

```
rlwrap jdb -attach localhost:5555
```

> Sous Linux, il n'y a pas de “Shared Memory Attaching Connector”. Donc pour se connecter à un `dt_shmem`, il faut passer par un “Process Attaching Connector”. Exemple : `jdb -connect com.sun.jdi.ProcessAttach:pid=1358`. Conclusion : il est bien plus simple d'utiliser `dt_socket` !!

Debugger un test maven

```
mvn -Dmaven.surefire.debug test
```

les tests attende qu'un débuggeur se connecte

NB : il peut être très utile de fournir les xxx-sources.jar via `mvn dependency:sources`

Debugage distant eclipse

L'utilisation du debugage distant permet d'avoir un Tomcat sur un serveur et de faire du debugage (pas à pas) sur le poste local (poste du développeur).

Le principe du debugage distant est le suivant :

1. La JVM server ouvre sur socket (sur le server et un port spécifique)
2. Eclipse va se connecter sur cette socket afin d'intercepter les point d'arrêt et de communiquer avec la JVM distante.

Pour utiliser la JVM en mode debugage distant il faut lui paramètrer le port d'attente.

Pour ce faire plusieurs solutions (en voici 2) :

- on utilise un script sh afin de lancer le Tomcat.

```
#!/bin/sh
JAVA_HOME=/usr/java/j2sdk1.4;export JAVA_HOME
JAVA_OPTS="-server ";export JAVA_OPTS
# si debug : mettre MY_DEBUG a 1, et preciser le port avec JPDA_ADDRESS
MY_DEBUG=1
JPDA_ADDRESS=55555;export JPDA_ADDRESS
CATALINA_HOME=/home/cri/jmarchal/uPortal/Tomcat;export CATALINA_HOME
CATALINA_BASE=$CATALINA_HOME;export CATALINA_BASE
if [ "$MY_DEBUG" = "1" ]; then
        $CATALINA_HOME/bin/catalina.sh jpda start
else
        $CATALINA_HOME/bin/catalina.sh start
fi
```

Tomcat utilise des variables d'environement afin de choisir le port d'attente de la socket de debug (**JPDA_ADDRESS**).

- dans la deuxième solution, on positionne l'ensemble des options Java supplémentaires en définissant uniquement la variable JAVA_OPTS

Sous un UNIX:

```
JAVA_OPTS="$JAVA_OPTS -Xdebug -Xrunjdwp:transport=dt_socket,address=55555,server=y,suspend=n"
```

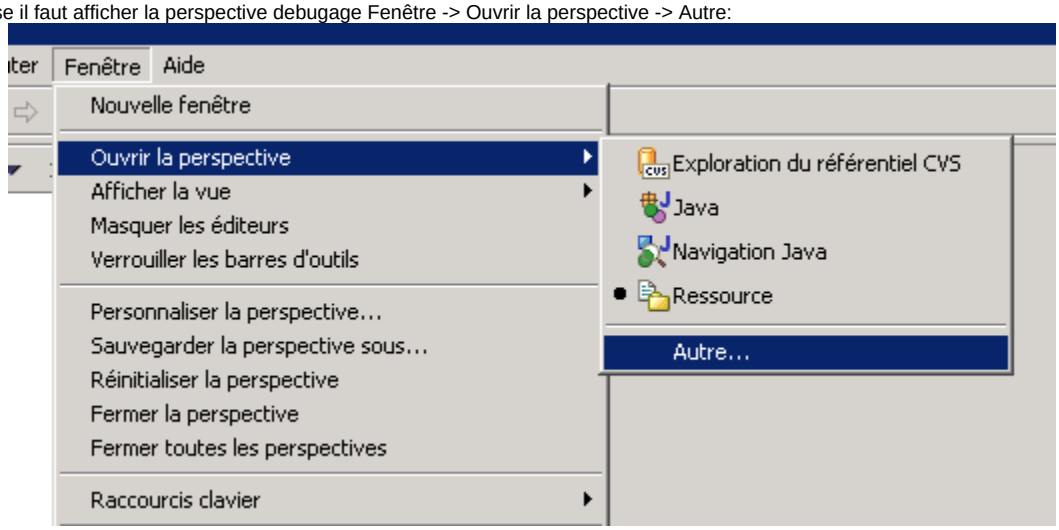
Sous un Windows :

```
rem set JAVA_OPTS=-Xdebug -Xrunjdwp:transport=dt_socket,address=55555,server=y,suspend=y %JAVA_OPTS%
```

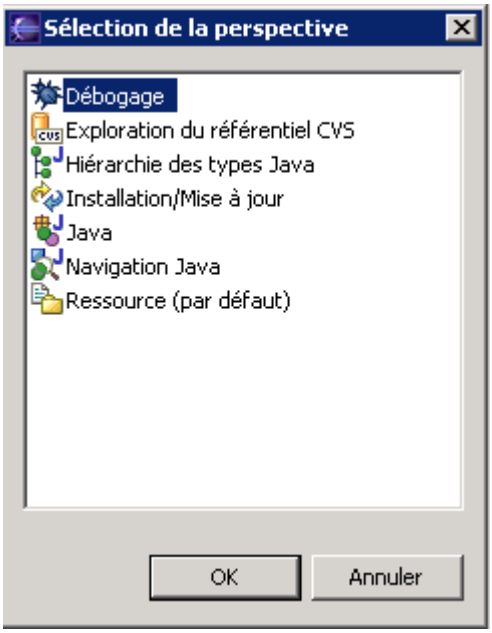
Remarquez ici l'option suspend : positionnée à y elle permet de faire temporiser le démarrage du Tomcat, celui-ci ne démarrant que si la communication socket est activée. Cela permet notamment de débuguer l'initialisation des applications, ce qui peut s'avérer bien pratique.

La seconde partie de cette manipulation va se faire du coté Eclipse.

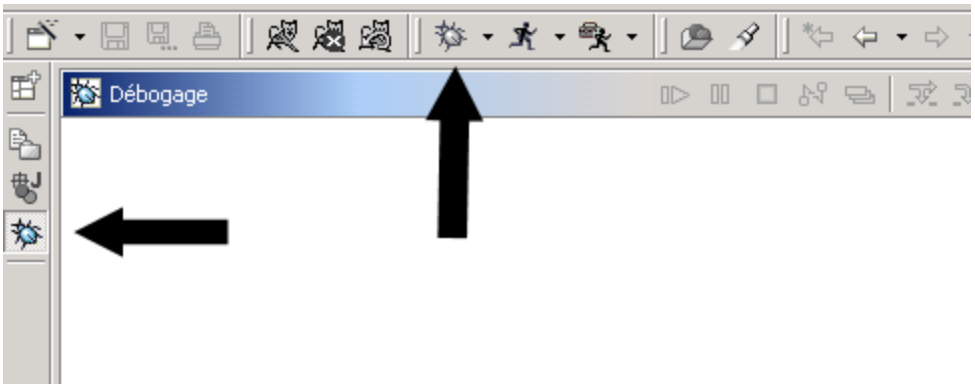
Avant toutes chose il faut afficher la perspective debugage Fenêtre -> Ouvrir la perspective -> Autre:



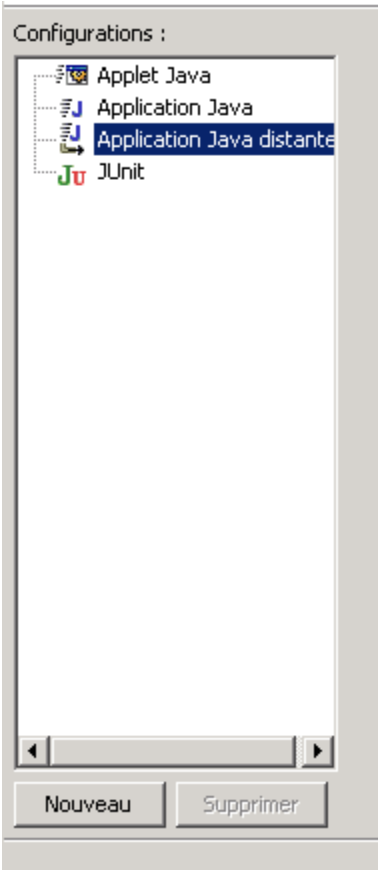
Choisir Debogage :



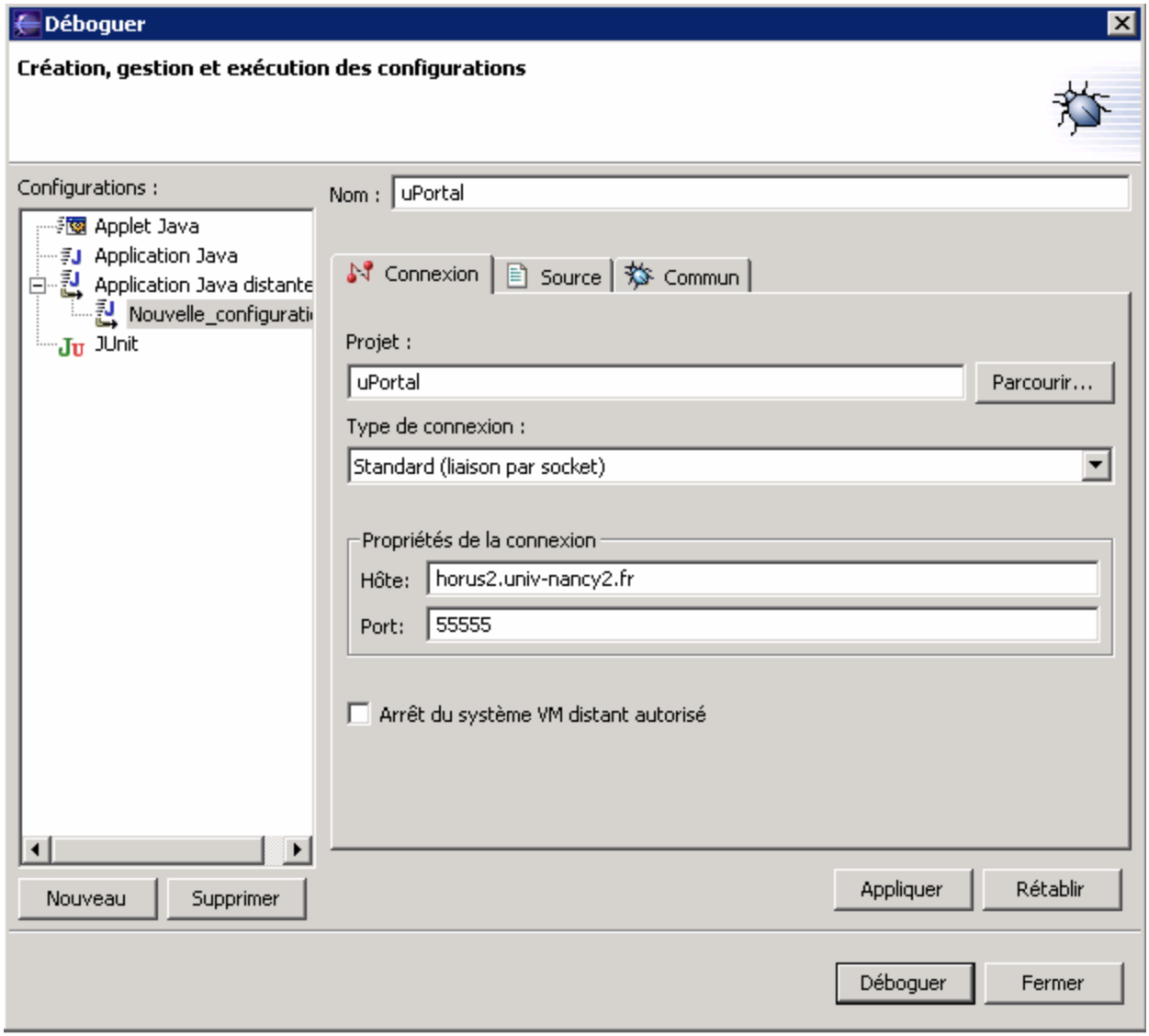
Votre affichage va changer comme suit :



Ensuite il faut créer un profil de debugage :



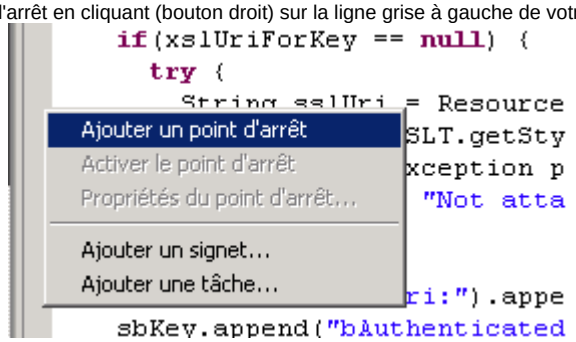
Chosissez Application Java distante puis cliquer sur Nouveau.



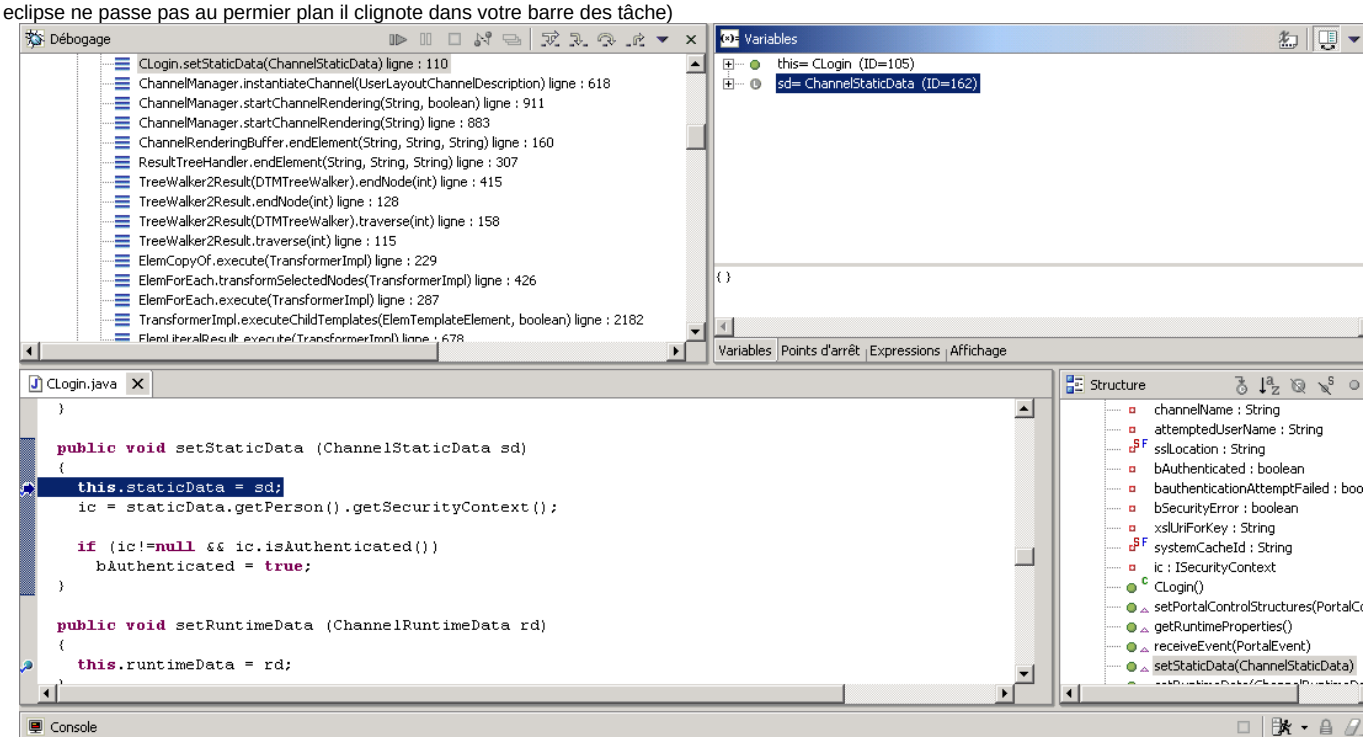
Donner un nom à cette configuration.
Choisissez le projet auquelle la configuration doit faire référence.
Valuez le nom du serveur distant (Hôte) et le port qui doit être le même que le **JPDA_ADDRESS** (cf script de lancement au dessus)

Cliquer sur Déboguer.

Par la suite vous pouvez appliquer despoint d'arrêt en cliquant (bouton droit) sur la ligne grise à gauche de votre source.



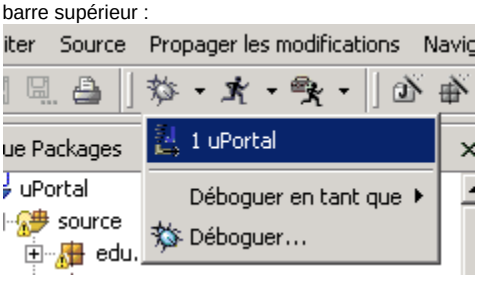
Utiliser un navigateur classique pour allez voir votre uPortal. Si vous passer sur le code où vous avez mis un point d'arrêt Eclispe s'activera (attention eclipse ne passe pas au permier plan il clignote dans votre barre des tâche)



Vous avez ainsi la vision des variables en cours d'exécution (et aussi de chaque Thread de votre JVM intéressant)

La touche lancement (>) en haut à gauche de la vue débugoge vous permet de continuer.

Par la suite vous pouvez lancer le debugage via la barre supérieur :



Debugguer avec emacs

Sinon on peut utiliser la commande `jdb` de emacs. Voici un script qui aide bien pour démarrer `jdb` :

/usr/local/bin/jdb-emacs

```
#!/bin/sh

attach=$1
sourcepath=$2
initial_breakpoint=$3

usage() {
    echo "usage example: jdbc-emacs localhost:5555 src/main/java org.esupportail.portal.services.
ProlongationENT.doGet"
    exit 1
}
[ $# = 3 -o $# = 2 ] || usage
[ -d "$sourcepath" ] || { echo "$sourcepath does not exist"; usage; }

cmd="(jdb \"jdb -attach $attach -sourcepath$sourcepath\")"
if [ -n "$initial_breakpoint" ]; then
    cmd2="(gud-call \"stop in $initial_breakpoint\")"
    cmd="(progn $cmd $cmd2)"
fi
exec emacs --eval "$cmd"
```