La classe Policy

- Permet de spécifier des règles de sécurité à l'aide de fichiers de configuration
- Activée par défaut

Extrait du fichier $JAVA_HOME/jre/lib/security/java.security

```
#
# Class to instantiate as the system Policy. This is the name of the class
# that will be used as the Policy object.
#
policy.provider=sun.security.provider.PolicyFile

# The default is to have a single system-wide policy file,
# and a policy file in the user's home directory.
policy.url.1=file:${java.home}/lib/security/java.policy
policy.url.2=file:${user.home}/.java.policy
```

- Il est possible de fournir une classe qui remplace la classe Policy définie par défaut mais il est beaucoup plus fréquent de la conserver
- Permet d'utiliser différents fichiers de configurations
  - au niveau de la machine virtuelle
  - par utilisateur
  - par application

La classe Policy

Notes

Les permissions définies par les différents fichiers (JVM, utilisateur, application) sont combinées.

${user.home} a pour valeur:

- $HOME sous Unix,
- le chemin d’accès au profil de l’utilisateur sous Windows.

Utiliser un fichier de règles spécifique

Ajouter des règles à celles définies par défaut

- Application

```
java -Djava.security.policy=MesRegles.policy MonApplication
```

- Applet

```
appletviewer -J-Djava.security.policy=MesRegles.policy MonApplet.html
```

Pour utiliser seulement les règles définies par un fichier

```
java -Djava.security.policy==MesRegles.policy MonApplication
```

Utiliser un gestionnaire de sécurité

- Activé par défaut avec certains types d'applications : applet, RMI dans certains cas.
- A activer avec les applications autonomes
  - soit

```
System.setSecurityManager (new SecurityManager());
```

  - soit

```
java -Djava.security.manager -Djava.security.policy=MyApp.policy MyApp
```

Utiliser un fichier de règles spécifique

Notes

Structure d'un fichier de règles

grant

```
grant sourcecode
{
   permission1;
   permission2;
   ...
}
```

sourcecode

```
codeBase "url"
```

- omis si les permissions sont accordées à tout code quelle que soit son origine
- url peut être distante (http) ou locale (file)

permission

```
permission ClassePermission Valeur , ListeActions
```

- Valeur = valeur spécifique de permission : nom de fichier, numéro de port, ...
- ListeActions = actions spécifiques à une permission : read, connect, execute, delete, ...

Structure d’un fichier de règles

Notes

Structure d'un fichier de règles

Exemple (fichier livré avec le JDK)

```
// Standard extensions get all permissions by default

grant codeBase "file:${{java.ext.dirs}}/*" {
   permission java.security.AllPermission;
};

grant {
   permission java.lang.RuntimePermission "stopThread";
   permission java.net.SocketPermission "localhost:1024-", "listen";
   permission java.util.PropertyPermission "java.version", "read";
   permission java.util.PropertyPermission "java.vendor", "read";
   permission java.util.PropertyPermission "java.vendor.url", "read";
   permission java.util.PropertyPermission "java.class.version", "read";
   permission java.util.PropertyPermission "os.name", "read";
   permission java.util.PropertyPermission "os.version", "read";
   permission java.util.PropertyPermission "os.arch", "read";
   permission java.util.PropertyPermission "file.separator", "read";
   permission java.util.PropertyPermission "path.separator", "read";
   permission java.util.PropertyPermission "line.separator", "read";

...
```

Structure d’un fichier de règles

Notes