La programmation client/serveur

Christophe Borelly

IUT Béziers Dépt. R&T

http://www.borelly.net

Christophe@Borelly.net

Applications client/serveur

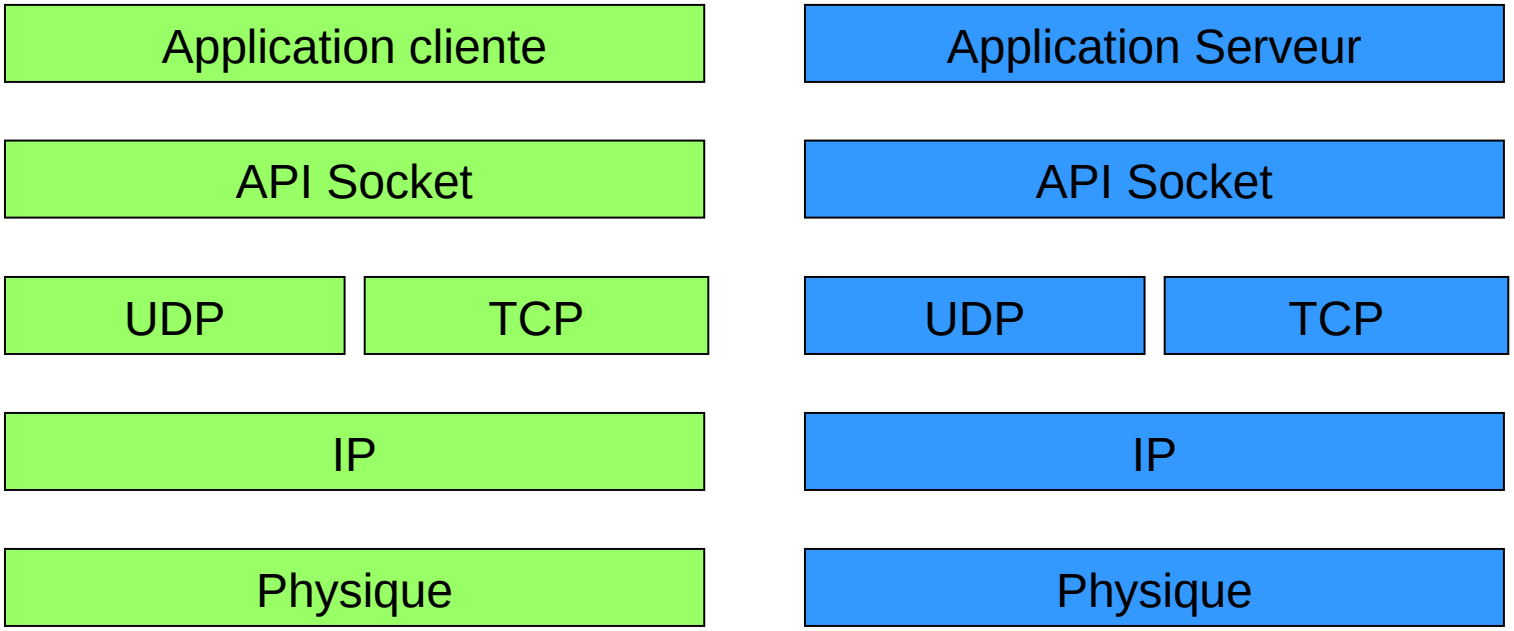
Communication réalisée par un dialogue entre processus, deux à deux (socket).

Un processus est le client, l'autre le serveur.

Le client initie l'échange (requête).

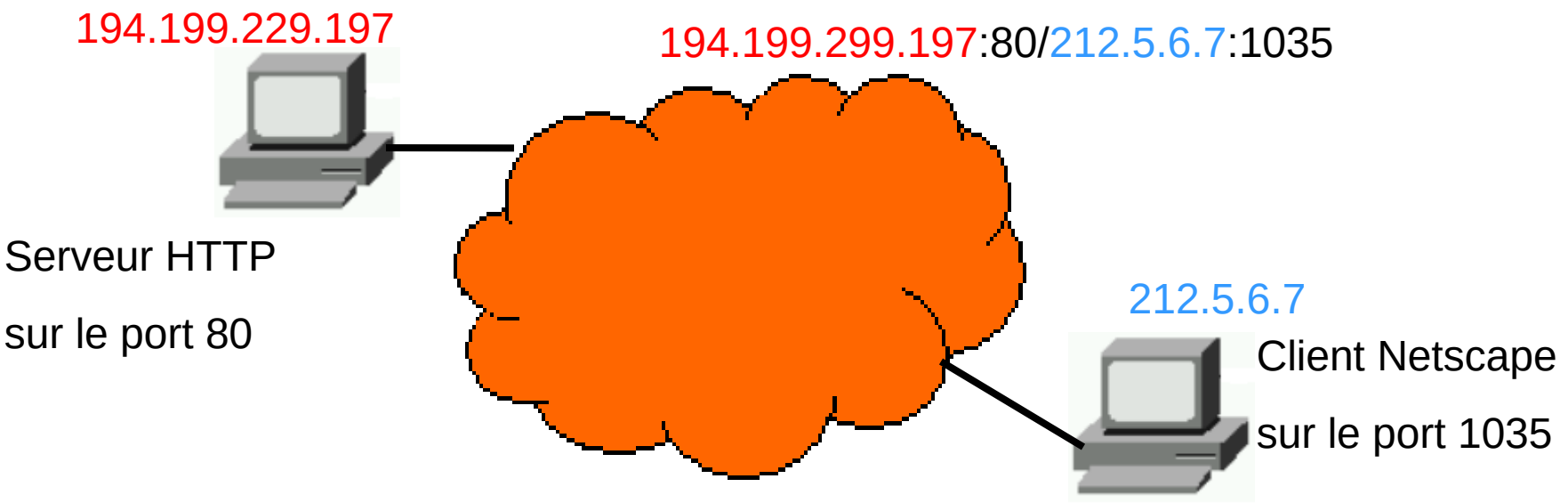
Le serveur répond à la requête du client (Il rend un service).

Modèle en couches



Les sockets

Une socket représente une association @IP source et port source avec @IP destination et port destination.



Affichage des connexions sous Windows NT 4.0

\winnt\system32\netstat

```
Connexions actives
Proto  Adresse locale Adresse extérieure     Etat
TCP    gtr212-8:1025  localhost:1028         ETABLIE
TCP    gtr212-8:1028  localhost:1025         ETABLIE
TCP    gtr212-8:1046  GTR-Server2:nbsession  ETABLIE
TCP    gtr212-8:1062  GTR-Server3:80         ETABLIE
TCP    gtr212-8:1063  GTR-Server3:80         ETABLIE
```

Programmation en C

Utilisation de l'API socket.

Fonctions :

- socket
- connect, bind, listen, accept
- read, write
- close

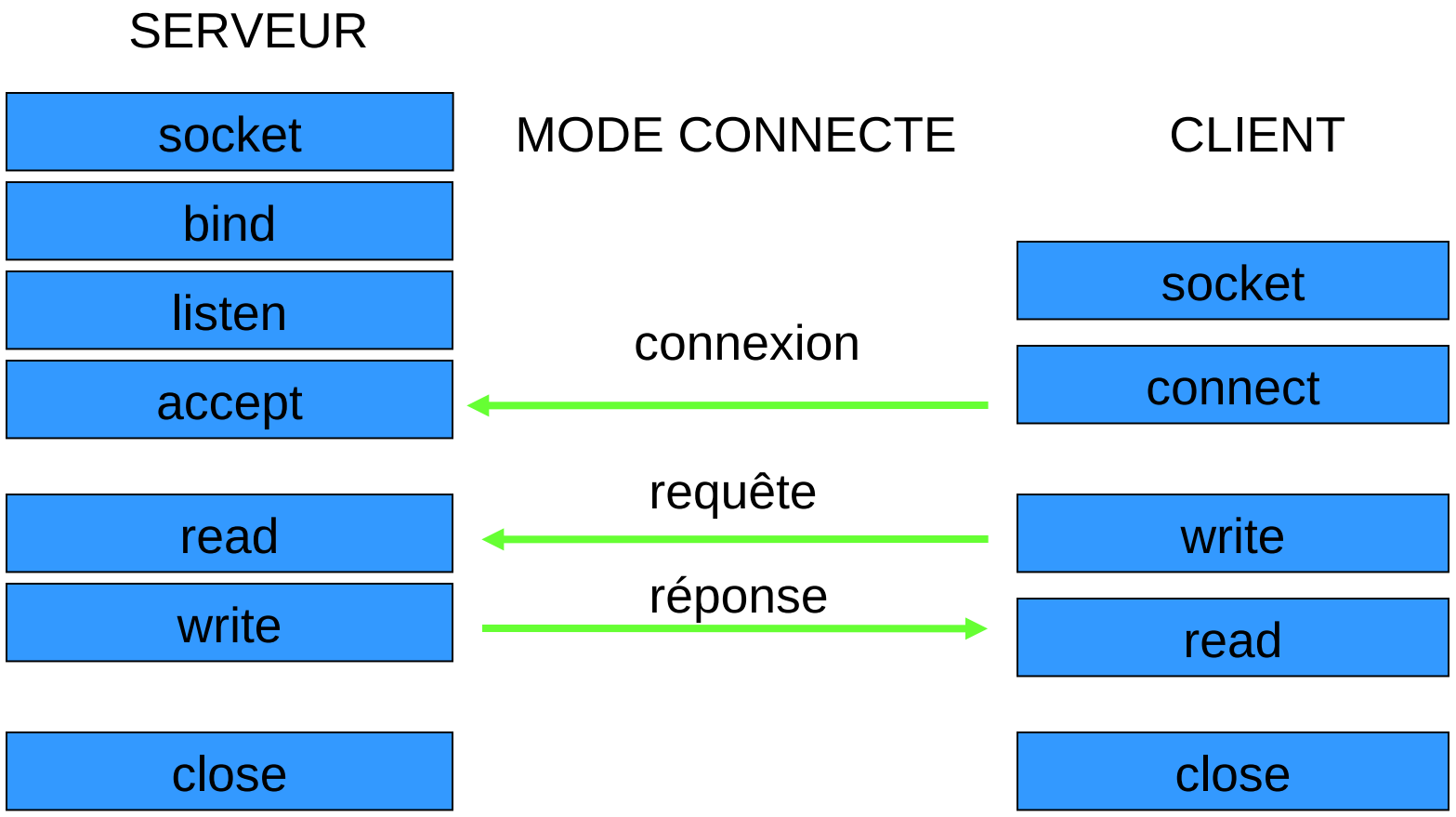
Utilisation des sockets

Equivalent à un descripteur de fichier pour le système d'exploitation.

Ecriture/lecture sur le réseau avec les mêmes fonctions que pour les fichiers

- `int **read**(int handle,void *buffer, unsigned int count);`
- `int **write**(int handle,void *buffer, unsigned int count);`

Séquencement des appels



Exemple IPv4 en C (1)

```
#define N 50
int r,sock,portServer=80,lenSockAddr=sizeof(struct sockaddr_in);
struct sockaddr_in servAddr;
char adrIPServer="www.yahoo.fr";
char buffer[N+1];
// Recherche de l'adresse IP...
host=gethostbyname(adrIPServer);
memcpy(&servAddr.sin_addr.s_addr,host->h_addr_list[0],host->h_length);
servAddr.sin_port=htons(portServer);
servAddr.sin_family=AF_INET;
sock=socket(PF_INET,SOCK_STREAM,IPPROTO_TCP);
if (connect(sock,(struct sockaddr*)&servAddr,lenSockAddr)<0){
   fprintf(stderr,"Erreur de connexion !!!\n");
   fprintf(stderr,"ERRNO : %d [%s]\n",errno,strerror(errno));
   exit(EXIT_FAILURE);
}
```

Exemple IPv4 en C (2)

```
strcpy(buffer,"Hello world !!!");
r=strlen(buffer); // Taille de la chaine
buffer[r]='\n';
r=write(sock,buffer,r+1); // On envoi un caractere de plus !!!
// Mise a zero des donnees
memset(buffer,0,N+1);
// Reception de la chaine
r=read(sock,buffer,N);
buffer[r-1]=0; // On remplace \n par fin de chaine...
printf("Message de %d octets recu...\n\t<=[%s]\n",r,buffer);
close(sock);
exit(EXIT_SUCCESS);
```

Exemple IPv6 en C (1)

```
int r,sock,portServer=80,lenSockAddr=sizeof(struct sockaddr_in6);
struct sockaddr_in6 servAddr;
char *adrIPServer="localhost";
servAddr.sin6_family=AF_INET6;
servAddr.sin6_port=htons(portServer);
servAddr.sin6_scope_id=0;
servAddr.sin6_flowinfo=0;
if (inet_pton(AF_INET6,adrIPServer,&(servAddr.sin6_addr.s6_addr))<0){
   fprintf(stderr,"Erreur de conversion d'adresse IP !!!\n");
   fprintf(stderr,"ERRNO : %d [%s]\n",errno,strerror(errno));
   exit(EXIT_FAILURE);
}
sock=socket(PF_INET6,SOCK_STREAM,IPPROTO_TCP);
```

Client-serveur SSL-TLS

Utilisation de la librairie openssl.

Compilation :

- `gcc gclient.c -o gclient -lssl -lcrypto`

Après avoir ouvert une connexion réseau classique, il suffit de créer un *contexte SSL*, d'initier la connexion SSL avec la fonction SSL_connect() et ensuite utiliser les fonctions SSL_read() et SSL_write().

Contexte SSL simple (1)

```
#include <openssl/ssl.h>
#include <openssl/err.h>
……
SSL_METHOD* meth;
SSL_CTX* ctx;
BIO* MyBIOErr=NULL; // un flot d'erreur BIO
static int SSL_ID_CTX; // SSL id context
char *cafile="ca.crt";
……
```

Contexte SSL simple (2)

```
SSL_library_init(); // Initialisation de OpenSSL
SSL_load_error_strings();
MyBIOErr=BIO_new_fp(stderr,BIO_NOCLOSE);
meth=SSLv23_method(); // TLSv1_method()
ctx=SSL_CTX_new(meth); // Creation du contexte SSL
if(!(SSL_CTX_load_verify_locations(ctx,cafile,0))) {
   BIO_printf(MyBIOErr,"BIO ERR : chargement du CA !!!\n");
   ERR_print_errors(MyBIOErr);
   exit(EXIT_FAILURE);
}
SSL_CTX_set_session_id_context(ctx,(void*)&SSL_ID_CTX,
   sizeof(SSL_ID_CTX));
```

Contexte SSL avec authentification (1)

```
static char* MyPass="PASSUSER";
// Mot de passe du fichier de cle privee (variable globale)

/** Recopie MyPass (variable globale) dans buffer */
int getPasswordCallBack(char *buffer,int num,int rwflag,
      void *userdata)
{
   printf("getPasswordCallBack...\n");
   if (num<strlen(MyPass)+1) return 0;
   strcpy(buffer,MyPass);
   return strlen(MyPass);
}
```

Contexte SSL avec authentification (2)

```
printf("Reading PrivateKey [%s]...\n",keyfile);
SSL_CTX_set_default_passwd_cb(ctx,getPasswordCallBack);
if(!(SSL_CTX_use_PrivateKey_file(ctx,keyfile,SSL_FILETYPE_PEM))) {
   BIO_printf(MyBIOErr,"BIO ERR : chargement cle privee !!!\n");
   ERR_print_errors(MyBIOErr);
   exit(EXIT_FAILURE);
}
printf("Reading PublicKey [%s]...\n",certfile);
if(!(SSL_CTX_use_certificate_chain_file(ctx,certfile))) {
   BIO_printf(MyBIOErr,"BIO ERR : chargement du certificat !!!\n");
   ERR_print_errors(MyBIOErr);
   exit(EXIT_FAILURE);
}
```

Connexion SSL

```
SSL* ssl;
BIO* sbio;
……
ssl=SSL_new(ctx);
sbio=BIO_new_socket(sock,BIO_NOCLOSE);
SSL_set_bio(ssl,sbio,sbio);
printf("SSL_connect...\n");
SSL_connect(ssl);
```

Affichage des algorithmes de chiffrement

```
STACK_OF(SSL_CIPHER)* cipherList;
SSL_CIPHER* cipher;
int index;
char *chipherName;
……
cipherList=SSL_get_ciphers(ssl);
for (index=0;index<sk_SSL_CIPHER_num(cipherList);index++)
{
chipherName=SSL_CIPHER_get_name(sk_SSL_CIPHER_value(cipherList,index));
printf("SSL_CIPHER_LIST : %s\n",chipherName);
}
cipher=SSL_get_current_cipher(ssl);
printf("SSL_CURRENT_CIPHER : %s\n",SSL_CIPHER_get_name(cipher));
```

Ecriture et lecture de données sur le flot SSL

```
r=SSL_write(ssl,buffer,10);
printf("Envoi de %d octets...\n",r);

// Mise a zéro des données
memset(buffer,0,N+1);
r=SSL_read(ssl,buffer,100);
printf("Lecture de %d octets...\n",r);
```

Programmation en JAVA

Utilisation du package java.net.

La classe des adresses IP :

- InetAdress

Les classes pour UDP :

- DatagramPacket, DatagramSocket, MulticastSocket

Les classes pour TCP :

- Socket, ServerSocket

java.net.URL

```
URL url=new URL("http://www.inria.fr/welcome.html");

InputStreamReader i=new InputStreamReader(url.openStream());
BufferedReader in=new BufferedReader(i);
while (true)
{
    line=in.readLine();
    if (line==null) break;
    System.out.println(line);
}
```

La classe InetAddress

Gestion des adresses internet.

```
String host="falconet.inria.fr";
InetAddress ip=InetAddress.getLocalHost();
System.out.println("IP locale : "+ip);
InetAddress adr=InetAddress.getByName(host);
System.out.println("IP : "+adr);
```

IP locale : gtr212-8/194.199.229.118

IP : falconet.inria.fr/128.93.25.46

java.net.Socket

Cette classe implémente une socket TCP coté client.

```
String line;
Socket s=new Socket("www.inria.fr",80);
PrintStream out=new PrintStream(s.getOutputStream());
out.println("GET / HTTP/1.0\r\n");

InputStreamReader i=new InputStreamReader(s.getInputStream());
BufferedReader in=new BufferedReader(i);
while (true)
{
    line=in.readLine();
    if (line==null) break;
    System.out.println(line);
}
```

java.net.ServerSocket

Cette classe implémente une socket TCP coté serveur.

```
int port=1234;
ServerSocket serveur=new ServerSocket(port);

while(true)
{
  Socket client=serveur.accept();
  new ClasseQuiFaitLeTraitement(client);
}
```

java.net.DatagramSocket

Cette classe implémente une socket UDP

```
// Ecriture UDP
byte[] data="un message".getBytes();

InetAddress addr=InetAddress.getByName("falconet.inria.fr");

DatagramPacket pkt=new DatagramPacket(data,data.length,addr,1234);
DatagramSocket ds=new DatagramSocket();

ds.send(pkt);
ds.close();
```

java.net.DatagramSocket

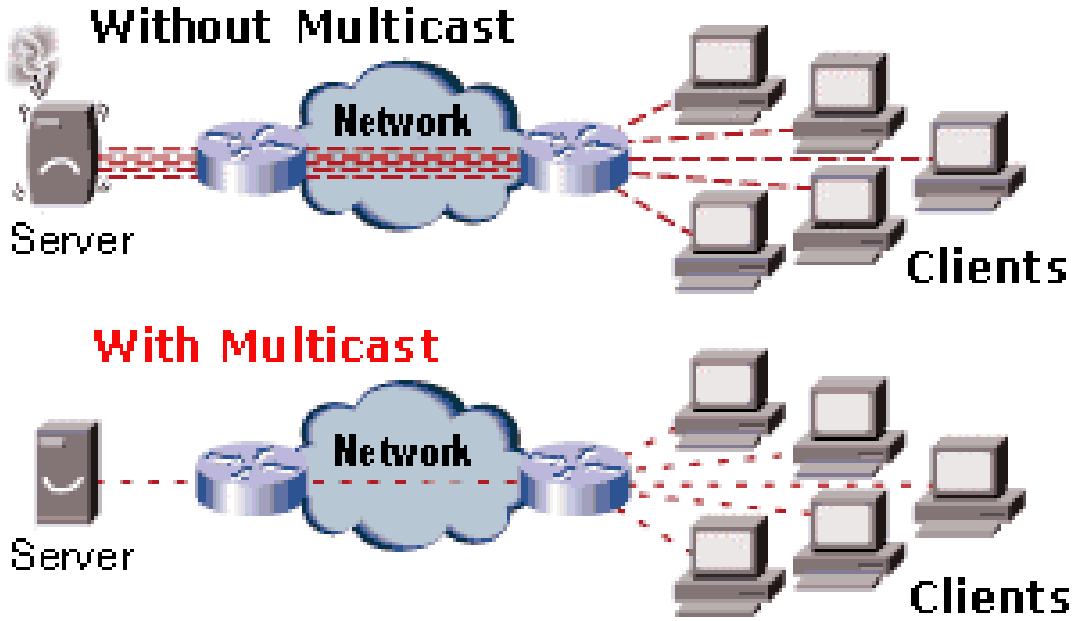
```
// Lecture UDP
DatagramSocket ds=new DatagramSocket(1234);

while(true)
{
   byte tab[]=new byte[1024];
   DatagramPacket pkt=new DatagramPacket(tab,tab.length);
   s.receive(pkt);
   String msg=new String(pkt.getData(),0,pkt.getLength());
   System.out.println("Message : "+msg);
}
```

Multicast UDP

Mode de diffusion ciblé.

Utilisation de la classe D d'adresse IP.



MulticastSocket (1)

Cette classe implémente une socket multicast (UDP)

```
// Ecriture
byte[] data="un message".getBytes();
InetAddress ip=InetAddress.getByName("experiment.mcast.net");

DatagramPacket pkt=new DatagramPacket(data,data.length,ip,1);

MulticastSocket s=new MulticastSocket();
s.joinGroup(ip);

s.send(pkt,(byte)1);

s.leaveGroup(ip);
s.close();
```

MulticastSocket (2)

```
// Lecture
InetAddress ip=InetAddress.getByName("experiment.mcast.net");
DatagramPacket packet=new DatagramPacket(new byte[1024],1024);

MulticastSocket s=new MulticastSocket(1234);
System.out.println("I listen on port "+s.getLocalPort());
s.joinGroup(ip);

s.receive(packet);

System.out.println("From : "+packet.getAddress());
String msg= new String(packet.getData(),0,packet.getLength()));
System.out.println("Message : "+msg);


s.leaveGroup(ip);
s.close();
```

SSL-TLS en JAVA

L'API JSSE contient deux packages javax.net et javax.net.ssl. Pour la programmation JAVA des sockets SSL, il suffira en grande partie d'utiliser les classes *SSLSocket* et *SSLServerSocket*.

Mais il n'existe pas de constructeurs utilisable directement. Il faut donc passer par une "usine" spécialisée : les objets *SSLSocketFactory* et *SSLServerSocketFactory*.

Par la suite les méthodes createSocket() et createServerSocket() permettent d'obtenir les socketSSL désirées.

SSLSocketFactory et SSLServerSocketFactory

Ces objets sont créés :

- Soit par la méthode de classe `getDefault()` qui renvoie l'implémentation fixée par les valeurs "ssl.SocketFactory.provider" et "ssl.ServerSocketFactory.provider" du fichier de propriétés de sécurité (<jre>\lib\security\ java.security).
- Soit en utilisant un contexte SSL, *SSLContext* (la création d'un contexte SSL sera développée ultérieurement).

Keytool (1)

Keytool est un outil de génération et de gestion des certificats pour JAVA. Les certificats sont stockés dans des fichiers appelés “ keystore ”.

Une liste par défaut de CA de confiance se trouve dans le fichier *\Program Files\Java\jx.y.z\lib\security\cacerts* ou *jre\lib\security\cacerts*.

Keytool (2)

Pour afficher le contenu d'un certificat, taper la commande

- `keytool -printcert -file ca.cer -v`

Pour générer un certificat d'autorité personnel, taper par exemple la commande :

- `keytool -genkey -validity 180 -alias cb -keyalg RSA -keysize 1024 -keypass cbkeypass -keystore cbstore -storepass cbpass`

Keytool (3)

Pour vérifier le keystore personnel :

- `keytool -list –alias cb -storetype jks -keystore cbstore -storepass cbpass -v`

Pour créer un fichier de certificat :

- `keytool -export -alias cb -file cb.cer -storetype jks -keystore cbstore -storepass cbpass -v`

Keytool (4)

Pour ajouter un certificat dans un keystore personnel.

- `keytool -import –alias ca -file ca.cer -storetype jks -keystore cbstore -storepass cbpass`

Exemple SSL

```
SSLSocketFactory ssf =(SSLSocketFactory)
      SSLSocketFactory.getDefault();
SSLSocket s=(SSLSocket)ssf.createSocket(host,port);
s.startHandshake();
String msg="GET / HTTP/1.0\r\n\r\n";
PrintStream out=new PrintStream(s.getOutputStream());
out.println(msg);
InputStream is=s.getInputStream();
InputStreamReader isr=new InputStreamReader(is);
BufferedReader in=new BufferedReader(isr);
String recu;
while((recu=in.readLine())!=null) System.out.println(recu);
s.close();
```

Création d'un contexte SSL

```
SSLContext ctx=SSLContext.getInstance("TLS");
String keyFile="cbstore";
String passPhrase="cbpass";
char pp[]=passPhrase.toCharArray();
java.security.KeyStore ks=
     java.security.KeyStore.getInstance("JKS");
ks.load(new FileInputStream(keyFile),pp);
KeyManagerFactory kmf=
     KeyManagerFactory.getInstance("SunX509");
kmf.init(ks,pp);
ctx.init(kmf.getKeyManagers(),null,null);
SSLSocketFactory ssf=ctx.getSocketFactory();
```

Authentification du client

```
...
((SSLServerSocket)ss).setNeedClientAuth(true);

// Affichage du certificat client
java.security.cert.Certificate[] cert=
   ((SSLSocket)s).getSession().getPeerCertificates();
for (int i=0;i<cert.length;i++){
   System.out.println(cert[i]);
}
```

Affichage des suites de chiffrements disponibles

```
import java.net.*;
import javax.net.ssl.*;
public class CipherSuite
{
   public static void main(String[] args) throws Exception {
       SSLServerSocketFactory factory=(SSLServerSocketFactory)
           SSLServerSocketFactory.getDefault();
       SSLServerSocket sslSocket=(SSLServerSocket)
           factory.createServerSocket(5757);
       String []cipherSuites=sslSocket.getEnabledCipherSuites();
       for (int i=0;i<cipherSuites.length;i++)
       {
       System.out.println("Cipher Suite "+i+" = "+cipherSuites[i]);
       }
   }
}
```

Références

Le Langage JAVA

- Ken Arnold et James Gosling. International THOMSON publishing.

Java in a nutshell

- D. Flanagan. O'Reilly.

JAVA Client-serveur

- C. Nicolas, C. Avare, F. Najman. Eyrolles.