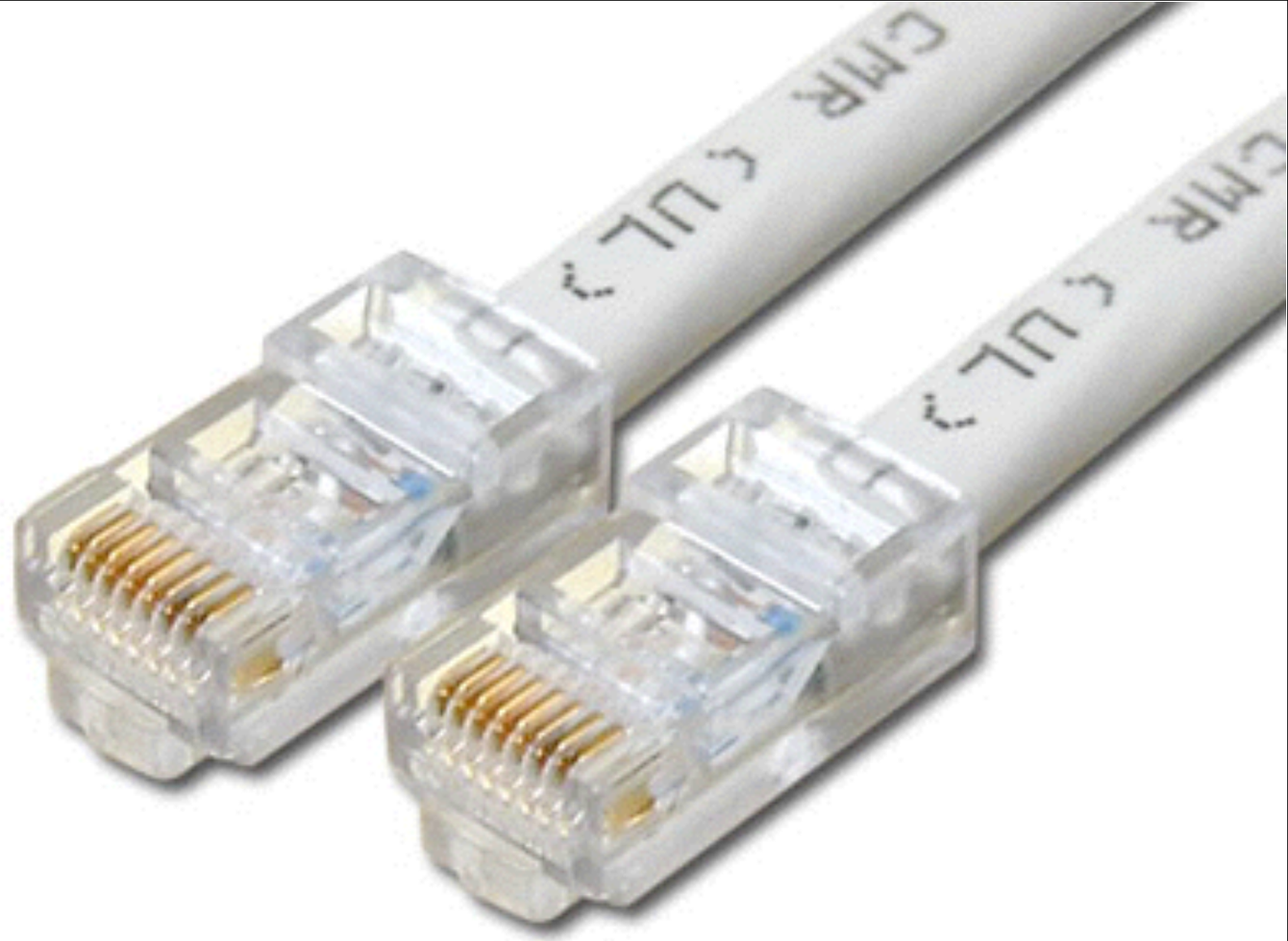
Programmation Réseau

La sérialisation



Jean-Baptiste.Yunes@univ-paris-diderot.fr
armand@informatique.univ-paris-diderot.fr

UFR Informatique

2013-2014

université PARIS DIDEROT PARIS 7

Pourquoi

- Encoder l'état mémoire d'un objet pour
  - Des besoins de persistance:
    - Pouvoir le stocker et le « re-créer » ultérieurement dans une autre instance de JVM
    - On peut stocker un objet sérialisé dans un fichier, une base de données…
  - De transmission:
    - Déplacer un objet via un appel de type RMI

Java seulement?

- C’est un procédé utile complètement indépendant des langages
  - Framework .NET de Microsoft
  - C++ (pas de manière native)
  - Ocaml
  - Python, etc
- Autres noms:
  - Marshalling / unmarshalling
  - Linéarisation

Quels objets Java?

- Tous ceux qui implémentent l'interface `Serializable`
  - une interface « vide », i.e. sans méthode
- Choix explicite
  - Il n'y a pas de sens à sérialiser certains objets
    - `InputStream`
  - Le bénéfice qui serait associé n'est pas évident
    - `Thread`

Comment?

- En général on décompose l'objet en éléments les plus petits (jusqu'à descendre à des éléments de types de base du langage), et on encode chacun de ces éléments
  - les éléments doivent être eux aussi sérialisables
    - pour déclarer un élément comme non sérialisable il faut le déclarer en tant que `transient`
- Il faut aussi conserver la structure de l'objet pour recomposer l'objet au moment de la « désérialisation ».
  - Objets composés, tableaux, listes,...

Quelle représentation?

- XML (SOAP)
  - « Lisible » ☺ mais Volumineux ☹
- XML binaire
- JSON (essentiellement lié à JavaScript)
- YAML
- XDR (historique C)
- Formats binaires spécifiques (ex: Java)
- etc.

Sérialisation

```
//  inspiré de http://www.java2s.com/Tutorial/Java/0180__File/StoringObjectsinaFile.htm)
import java.io.*;

public class Junk implements Serializable {
    private String str;
    public Junk(String s) {
        str = s;
    }
    public String toString() {
        return str;
    }
}
```

Sérialisation

```
// inspiré de http://www.java2s.com/Tutorial/Java/0180__File/StoringObjectsinaFile.htm)
import java.io.*;

public class SerOut {
  public static void main(String[] args) throws Exception {
    Junk obj1 = new Junk("A");
    Junk obj2 = new Junk("B");
    Junk obj3 = new Junk("V");
    FileOutputStream fos = new FileOutputStream("JunkObjects.bin");
    ObjectOutputStream objectOut = new ObjectOutputStream(fos);
    objectOut.writeObject(obj1);
    objectOut.writeObject(obj2);
    objectOut.writeObject(obj3);
    System.out.println("obj1: " + obj1);
    System.out.println("obj2: " + obj2);
    System.out.println("obj3: " + obj3);
    objectOut.close();
  }
}
```

Sérialisation

```
// inspiré de http://www.java2s.com/Tutorial/Java/0180__File/StoringObjectsinaFile.htm)
import java.io.*;

public class SerIn {
  public static void main(String[] args) throws Exception {
    int objectCount = 0;
    Junk object = null;
    FileInputStream fis = new FileInputStream("JunkObjects.bin");
    ObjectInputStream objectIn = new ObjectInputStream(fis);
    while (objectCount < 3) {
      object = (Junk)objectIn.readObject();
      objectCount++;
      System.out.println("Object n°"+objectCount+": "+object);
    }
    objectIn.close();
  }
}
```

Un contrôle plus fin

- La sérialisation par défaut pourrait ne pas convenir tout à fait…
- on peut vouloir faire quelque chose avec les `transients`
- les (ré)initialiser
- lire/écrire les choses dans un ordre différent
- etc.

Un contrôle plus fin

- lorsqu'un objet Serializable implémente les méthodes privées
  - `void readObject(ObjectInputStream)`
  - `void writeObject(ObjectOutputStream)`
- ces méthodes sont appelées lors de la (dé)sérialisation

Sérialisation

```
// inspiré de http://www.java2s.com/Tutorial/Java/0180__File/StoringObjectsinaFile.htm)
import java.io.*;

public class JunkBis implements Serializable {
    private String str;
    private int age;
    public JunkBis(String s,int age) {
        str = s;
        this.age = age;
    }
    public String toString() {
        return str+"("+age+")";
    }
    private void writeObject(ObjectOutputStream oos) throws Exception {
        System.out.println("Je contrôle la situation");
        oos.writeInt(age);
        oos.writeUTF(str);
    }
    private void readObject(ObjectInputStream ois) throws Exception {
        System.out.println("Je contrôle la situation");
        age = ois.readInt();
        str = ois.readUTF();
    }
}
```

Sérialisation

```
// inspiré de http://www.java2s.com/Tutorial/Java/0180__File/StoringObjectsinaFile.htm)
import java.io.*;

public class SerOutBis {
  public static void main(String[] args) throws Exception {
    JunkBis obj1 = new JunkBis("A",12);
    JunkBis obj2 = new JunkBis("B",13);
    JunkBis obj3 = new JunkBis("V",14);
    FileOutputStream fos = new FileOutputStream("JunkObjectsBis.bin");
    ObjectOutputStream objectOut = new ObjectOutputStream(fos);
    objectOut.writeObject(obj1);
    objectOut.writeObject(obj2);
    objectOut.writeObject(obj3);
    System.out.println("obj1: " + obj1);
    System.out.println("obj2: " + obj2);
    System.out.println("obj3: " + obj3);
    objectOut.close();
  }
}
```

Sérialisation

```
//  inspiré de http://www.java2s.com/Tutorial/Java/0180__File/StoringObjectsinaFile.htm)
import java.io.*;

public class SerInBis {
  public static void main(String[] args) throws Exception {
    int objectCount = 0;
    JunkBis object = null;
    FileInputStream fis = new FileInputStream("JunkObjectsBis.bin");
    ObjectInputStream objectIn = new ObjectInputStream(fis);
    while (objectCount < 3) {
      object = (JunkBis)objectIn.readObject();
      objectCount++;
      System.out.println("Object n°"+objectCount+": "+object);
    }
    objectIn.close();
  }
}
```

- dans le contrôle fin, on peut utiliser les méthodes :
  - `void defaultWriteObject()` de `ObjectOutputStream`
  - `void defaultReadObject()` de `ObjectInputStream`
- qui permettent d'obtenir la sérialisation par défaut de tous les attributs non-`transients`

Version de classe

- Les classes pouvant évoluer au cours du temps (pour un même concept), il est important de pouvoir définir une compatibilité temporelle

- La valeur du champ
  - `private static final long serialVersionUID;`
- est utilisé pour identifier la version de la classe
  - contrairement à ce qui est raconté partout, n'importe quelle valeur peut-être employée…

Sérialisation « externe »

- Il existe un dernier mécanisme de sérialisation Java à grain fin
  - la sérialisation externe
- La différence principale est que les objets doivent d'abord exister avant d'être désérialisés
  - Ils sont d'abrod créés via un constructeur sans arguments
- La (dé)sérialisation n'offre pas de mécanisme par défaut

Sérialisation

```
import java.io.*;

public class JunkTer implements Externalizable {
    private String str;
    private int age;
    public JunkTer() {
        System.out.println("Empty args ctor called");
        str = "";
        age = 0;
    }
    public JunkTer(String s,int age) {
    str = s;
        this.age = age;
    }
    public String toString() {
    return str+"("+age+")";
    }
    public void writeExternal(ObjectOutput oo) throws IOException {
        System.out.println("Je contrôle la situation");
        oo.writeInt(age);
        oo.writeUTF(str);
    }
    public void readExternal(ObjectInput oi) throws IOException {
        System.out.println("Je contrôle la situation");
        age = oi.readInt();
        str = oi.readUTF();
    }
}
```

Sérialisation

```
import java.io.*;

public class SerOutTer {
  public static void main(String[] args) throws Exception {
    JunkTer obj1 = new JunkTer("A",12);
    JunkTer obj2 = new JunkTer("B",13);
    JunkTer obj3 = new JunkTer("V",14);
    FileOutputStream fos = new FileOutputStream("JunkObjectsTer.bin");
    ObjectOutputStream objectOut = new ObjectOutputStream(fos);
    objectOut.writeObject(obj1);
    objectOut.writeObject(obj2);
    objectOut.writeObject(obj3);
    System.out.println("obj1: " + obj1);
    System.out.println("obj2: " + obj2);
    System.out.println("obj3: " + obj3);
    objectOut.close();
  }
}
```

Sérialisation

```
import java.io.*;

public class SerInTer {
  public static void main(String[] args) throws Exception {
    int objectCount = 0;
    JunkTer object = null;
    FileInputStream fis = new FileInputStream("JunkObjectsTer.bin");
    ObjectInputStream objectIn = new ObjectInputStream(fis);
    while (objectCount < 3) {
      object = (JunkTer)objectIn.readObject();
      objectCount++;
      System.out.println("Object n°"+objectCount+": "+object);
    }
    objectIn.close();
  }
}
```

XML

- XML (eXtensible Markup Language) est un langage de description de données par balisage

  - Il est très employé comme format pivot

- Java offre la sérialisation XML à travers ses beans (composants Java normalisés)

Sérialisation

```
import java.io.*;

public class JunkQuater {
    private String name;
    private int age;
    public JunkQuater() {
        System.out.println("Empty args ctor called");
        setName("");
        setAge(0);
    }
    public void setName(String n) {
        this.name = n;
    }
    public String getName() {
        return name;
    }
    public void setAge(int a) {
        this.age = a;
    }
    public int getAge() {
        return age;
    }
    public JunkQuater(String s,int age) {
        setName(s);
        setAge(age);
    }
    public String toString() {
     return getName()+"("+getAge()+")";
    }
}
```

pas de

un ctor sans args

des accesseurs

Sérialisation

```
import java.io.*;
import java.beans.*;

public class SerOutQuater {
  public static void main(String[] args) throws Exception {
    JunkQuater obj1 = new JunkQuater("A",12);
    JunkQuater obj2 = new JunkQuater("B",13);
    JunkQuater obj3 = new JunkQuater("V",14);
    FileOutputStream fos = new FileOutputStream("JunkObjects.xml");
    XMLEncoder encoder = new XMLEncoder(fos);
    encoder.writeObject(obj1);
    encoder.writeObject(obj2);
    encoder.writeObject(obj3);
    System.out.println("obj1: " + obj1);
    System.out.println("obj2: " + obj2);
    System.out.println("obj3: " + obj3);
    encoder.close();
    fos.close();
  }
}
```

Sérialisation

```
import java.io.*;
import java.beans.*;

public class SerInQuater {
  public static void main(String[] args) throws Exception {
    int objectCount = 0;
    JunkQuater object = null;
    FileInputStream fis = new FileInputStream("JunkObjects.xml");
    XMLDecoder decoder = new XMLDecoder(fis);
    while (objectCount < 3) {
      object = (JunkQuater)decoder.readObject();
      objectCount++;
      System.out.println("Object n°"+objectCount+": "+object);
    }
    decoder.close();
    fis.close();
  }
}
```

Sérialisation

```
<?xml version="1.0" encoding="UTF-8"?>
<java version="1.6.0_37" class="java.beans.XMLDecoder">
 <object class="JunkQuater">
  <void property="age">
   <int>12</int>
  </void>
  <void property="name">
   <string>A</string>
  </void>
 </object>
 <object class="JunkQuater">
  <void property="age">
   <int>13</int>
  </void>
  <void property="name">
   <string>B</string>
  </void>
 </object>
 <object class="JunkQuater">
  <void property="age">
   <int>14</int>
  </void>
  <void property="name">
   <string>V</string>
  </void>
 </object>
</java>
```

Transfert via TCP

- Un object sérialisable
- Un serveur de réception (qui doit impérativement connaître la classe de l’objet à désérialiser)
- Un client envoyant un objet au serveur...

Un transfert d'objet sur une liaison réseau

```
import java.io.*;

public class Junk implements Serializable {
  private String str;
  private int age;
  public Junk(String s,int a) {
    str = s;
    age = a;
  }
  public String toString() {
    return str+" "+age;
  }
}
```

Un transfert d'objet sur une liaison réseau

```
import java.io.*;
import java.net.*;

public class ObjectSender {
  public static void main(String[] args) throws Exception {
    if (args.length<2) {
      System.err.println("usage: java ObjectSender name host");
      System.exit(1);
    }
    Junk obj1 = new Junk(args[0],12);
    Socket s = new Socket(argv[1],61234);
    ObjectOutputStream objectOut =
            new ObjectOutputStream(s.getOutputStream());
    objectOut.writeObject(obj1);
    objectOut.flush();
  }
}
```

Un transfert d'objet sur une liaison réseau

```
import java.io.*;
import java.net.*;

public class ObjectReceiver {
  public static void main(String[] args) throws Exception {
    int objectCount = 0;
    ServerSocket ss = new ServerSocket(61234);
    while (true) {
      Socket service = ss.accept();
      ObjectInputStream objectIn =
        new ObjectInputStream(service.getInputStream());
      Junk object = (Junk)objectIn.readObject();
      objectCount++;
      System.out.println("Object n°"+objectCount+": "+object);
    }
  }
}
```