Programmation Réseau

Serialisation

Dessins: Jean-Baptiste Yunès
Jean-Baptiste.Yunes@liafa.jussieu.fr

Coloriages: François Armand
armand@informatique.univ-paris-diderot.fr

UFR Informatique

2011-2012

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 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?

- Peut s’appliquer à d’autres environnements / langages que Java
  - Framework .NET de Microsoft
  - C++ (pas de manière native)
  - Ocaml
  - Python, etc
- Autres noms:
  - Marshalling / unmarshalling

Quels objets Java?

- Tous ceux qui implémentent l'interface Serializable
  - une interface « vide », i.e. sans méthode
- Choix explicite
  - Certains objets n'auraient que peu de bénéfices d'une sérialisation
  - Ex: Thread, ou InputStream

Comment?

- En général on décompose l’objet en éléments les plus petits (les 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
- Etc.

Exemple (from http://www.java2s.com/Tutorial/Java/0180__File/StoringObjectsinaFile.htm)

```
import java.io.BufferedOutputStream;
import java.io.FileOutputStream;
import java.io.ObjectOutputStream;
import java.io.Serializable;

public class MainClass {
  public static void main(String[] args) throws Exception {
    Junk obj1 = new Junk("A");
    Junk obj2 = new Junk("B");
    Junk obj3 = new Junk("V");
    ObjectOutputStream objectOut = new ObjectOutputStream(new BufferedOutputStream(
        new FileOutputStream(“/home/fa/JunkObjects.bin")));

    objectOut.writeObject(obj1); // Write object
    objectOut.writeObject(obj2); // Write object
    objectOut.writeObject(obj3); // Write object
    System.out.println("\n\nobj1:\n" + obj1 + "\n\nobj2:\n" + obj2 + "\n\nobj3:\n" + obj3);
    objectOut.close(); // Close the output stream
  }
}
class Junk implements Serializable{
  String str;
  public Junk(String s) {
    str = s;
  }
}
```

Exemple (from http://www.java2s.com/Tutorial/Java/0180__File/StoringObjectsinaFile.htm)

```
import java.io.BufferedInputStream;
import java.io.FileInputStream;
import java.io.ObjectInputStream;
import java.io.Serializable;

public class MainClass {
  public static void main(String[] args) throws Exception {
    ObjectInputStream objectIn = null;
    int objectCount = 0;
    Junk object = null;

    objectIn = new ObjectInputStream(new BufferedInputStream(new FileInputStream(
        « /home/fa/JunkObjects.bin")));

    // Read from the stream until we hit the end
    while (objectCount < 3) {
      object = (Junk) objectIn.readObject();
      objectCount++;
      System.out.println(object);
    }
    objectIn.close();
  }
}
class Junk implements Serializable {
  String str;
  public Junk(String s) {
    str = s;
  }
}
```

plus loin…

- on peut contrôler plus finement le processus de (dé)sérialisation
  - pour modifier le processus, par exemple ajouter des initialisations après la désérialisation
    - read/writeObject(…)
- pour modifier le codage de la sérialisation
  - Externalizable et read/writeExternal(…)
- contrôler des versions de classes
  - serialVersionUID

Quels problèmes?

- De manière plus large que JAVA
  - Une copie « brutale » de l’objet en mémoire ne peut pas fonctionner.
- Endianness
- Alignement
  - Machines 32 bits versus 64 bits
- Codage