Mòdul 6. Accés a dades

UF1. Persistència en fitxers

Serialització i deserialització d'objectes

Serialització

És el procés de convertir un objecte d'un llenguatge de programació en un format que pugui ser emmagatzemat o enviat.

Aquest format sol ser en binari o de text i permet que l'objecte es pugui reconstruir mitjançant la **deserialització**.

Serialització en Java

Es refereix específicament a la conversió d'un objecte en una seqüència de bytes que poden ser:

- ➔ Emmagatzemats en un fitxer
- ➔ Enviats per la xarxa
- ➔ Emmagatzemats en una base de dades o altres mitjans

Exemples d'ús

Persistència: guardar l'estat d'un objecte en un fitxer o base de dades per a què pugui ser recuperat més tard.

Comunicació: enviar objectes a través de la xarxa, per exemple en aplicacions distribuides o web service.

Clonació: creació d'una còpia exacta de l'objecte serialitzant-lo i deserialitzant-lo.

Avantatges

Persistència d'objectes: es pot guardar l'estat complet d'un objecte per reconstruir-lo posteriorment.

Intercanvi d'informació: permet l'intercanvi entre sistemes enviant l'objecte a través de la xarxa.

Facilitat per a clonar objectes: permet fer còpies d'objectes complexos.

Inconvenients o limitacions

- La serialització nativa a Java pot no ser adequada per a totes les aplicacions a causa de la dependència directa del llenguatge i de la versió.
- Problemes de rendiment en objectes molt grans o complexos.
- Depenen de la compatibilitat entre versions de les classes serialitzades.

Deserialització

És el procés invers a la serialització, per tant, permet convertir un flux de dades en un objecte de llenguatge de programació, permet reconstruir l'objecte original a partir de la seva representació **serialitzada**.

Exemples d'ús

Recuperació: permet recuperar l'estat d'objectes que s'hagin serialitzat prèviament a un fitxer o a una base de dades.

Comunicació: permet la comunicació entre aplicacions quan es reben objectes d'altres sistemes.

Lectura: permet llegir les configuracions o dades estructurades, de fitxers externs XML o JSON, per exemple.

Avantatges

Recuperació d'objectes: permet carregar i restaurar objectes emmagatzemats anteriorment en format serialitzat.

Intercanvi d'informació: permet l'intercanvi deserialitzant a partir de dades provinents d'altres sistemes o xarxes.

Eficiencia en l'emmagatzematge: permet emmagatzemar dades complexes en format binari o de text i posteriorment carregar-les quan sigui necessari.

Inconvenients o limitacions

- **Compatibilitat entre classes:** cal que sigui compatible amb la classe que es va utilitzar per a serialitzar l'objecte. Per tant, depèn de la versió de la classe.
- **Seguretat:** és important assegurar-se que la deserialització es realitza de manera segura, ja que al provenir les dades d'un fitxer o font externa pot ser vulnerable a atacs.

Exemples de codi

Classe serialitzable: Llibre.java

Serialització d'un fitxer: SerialitzarObjecte.java

Deserialització d'un fitxer: DesserialitzarObjecte.java

Xstream

És una biblioteca de Java que permet la serialització i deserialització d'objectes en format XML.

La funcionalitat principal és la de convertir objectes de Java en format XML de manera senzilla i llegible i viceversa, permetent el seu emmagatzematge o l'intercanvi d'informació.

Simplifica el procés de treballar amb dades en format XML amb Java.

Xstream

Característiques:

- Facilita la conversió d'objectes Java a XML.
- Es poden utilitzar anotacions per personalitzar la manera com els camps es mapegen a XML.
- Es poden crear convertidors per gestionar tipus de dades específics de manera personalitzada.
- És compatible amb col·leccions, herència i estructures complexes.

Xstream

Exemple de transformació d'un objecte:

```
XStream xstream = new XStream();

String xml = xstream.toXML(objecte);
```

Exemple de recuperació d'un objecte:

```
Object objecte = xstream.fromXML(xml);
```