UNIVERSIDAD TECNOLÓGICA DEL PERÚ

Facultad de Ingeniería

Programación Orientada a Objetos

Sesión 15: Clases Persistentes. Serialización en archivos XML

Recordando:

¿Que vimos la clase pasada?

Universidad Tecnológica del Perú

Logro de aprendizaje

Al finalizar la sesión, el estudiante soluciona problemas Clases Persistentes. Serialización en archivos XML usando Java en la resolución de ejercicios.

Universidad Tecnológica del Perú

Utilidad

Las clases persistentes son aquellas cuyas instancias pueden ser guardadas y recuperadas de un almacenamiento permanente, como una base de datos o un archivo.
Esto es útil para aplicaciones que necesitan mantener datos entre diferentes sesiones de ejecución.

Saberes Previos

- Conceptos básicos de programación orientada a objetos.
- Familiaridad con el lenguaje de programación (preferiblemente Java).
- Comprender la importancia de la reutilización de código y la mantenibilidad del software.

Agenda

- Conceptos fundamentales de persistencia de datos.
- Serialización de datos.
- Archivos XML como formato para la serialización.

Universidad Tecnológica del Perú

¿Qué es la Persistencia de Datos?

La persistencia de datos se refiere a la capacidad de los datos para mantenerse a lo largo del tiempo, independientemente de la ejecución de un programa o la vida útil de una aplicación.

Importancia de la Persistencia en Programación

Estado de los Datos:

La persistencia permite que los datos mantengan su estado incluso después de que el programa que los generó haya terminado.

Almacenamiento Permanente:

Los datos persisten en almacenamiento permanente, como discos duros, bases de datos o archivos, en contraste con la memoria volátil que se borra cuando se apaga un sistema.

Tipos de Persistencia

- **Persistencia de Objeto:** Mantener el estado de objetos y sus atributos a lo largo del tiempo.
- **Persistencia de Archivos:** Almacenar y recuperar datos a través de archivos.

Mecanismos de Persistencia

- **Serialización:** Proceso de convertir datos en un formato que puede ser almacenado o transmitido, como en archivos XML, JSON o binarios.
- **Base de Datos:** Utilización de sistemas de gestión de bases de datos para persistir información de manera estructurada.

Beneficios de la Persistencia de Datos

- **Conservación de Información:** Permite conservar y recuperar datos esenciales para la aplicación en diferentes ejecuciones.
- **Interoperabilidad:** Facilita la transferencia de datos entre diferentes plataformas y sistemas.

Ejemplo Simple

Imagina una aplicación de notas que guarda tus escritos. La persistencia de datos asegura que, cuando cierres la aplicación y la vuelvas a abrir, tus notas sigan estando disponibles, ya que se han guardado en algún tipo de almacenamiento persistente.

Serialización de datos

¿Qué es la Serialización?

La serialización es el proceso de convertir datos en un formato específico que permite su almacenamiento o transmisión. Esto facilita la reconstrucción de los datos originales al deserializarlos.

Serialización de datos

https://www.youtube.com/watch?v=EX0_plIjumM

Universidad Tecnológica del Perú

Serialización de datos

Objetivo de la Serialización

- **Almacenamiento y Transmisión:** Facilita la persistencia de datos al convertirlos en una forma que puede ser guardada en archivos, bases de datos u otros medios de almacenamiento, así como transmitida a través de redes.

Mecanismos de Serialización

- **Formatos de Serialización:** Pueden ser en formato binario, texto plano (como JSON, XML) o en otros formatos específicos de acuerdo con las necesidades del sistema.

Ventajas de la Serialización

- **Facilidad de Almacenamiento:** Permite almacenar datos de manera estructurada y organizada.
- **Transferencia de Datos:** Simplifica la transmisión de información entre sistemas o aplicaciones.
- **Implementación en Programación**
  En la mayoría de los lenguajes de programación, existen librerías o herramientas que ofrecen funciones para serializar y deserializar datos.

Ejemplo Simple

Imagina un objeto en un programa que tiene varios atributos (nombre, edad, dirección). La serialización tomaría este objeto y lo convertiría en un formato legible y estructurado, listo para ser almacenado en un archivo o base de datos.

Archivos XML como Formato de Serialización

¿Qué es XML?

XML (Lenguaje de Marcado Extensible) es un lenguaje de marcado que define reglas para codificar documentos de manera legible tanto para humanos como para máquinas.

Archivos XML como Formato de Serialización

Características Principales de XML

- **Estructura Jerárquica:** Utiliza etiquetas anidadas que definen la estructura de los datos.
- **Legibilidad y Flexibilidad:** Su formato legible hace que sea fácil de entender y mantener.

Uso de XML en Serialización

- **Formato de Intercambio de Datos:** XML se usa comúnmente para estructurar datos de manera que puedan ser almacenados, transmitidos y procesados por diferentes sistemas de manera consistente.

Estructura de un Archivo XML

Tags (Etiquetas): Define elementos y sus atributos.
Elementos y Atributos: Representan la estructura de los datos a ser almacenados.
Aplicaciones de XML en Serialización

- **Configuración de Aplicaciones:** Muchas aplicaciones utilizan archivos XML para almacenar configuraciones debido a su estructura flexible y legible.
- **Intercambio de Datos:** Se emplea en intercambios de datos entre sistemas heterogéneos.

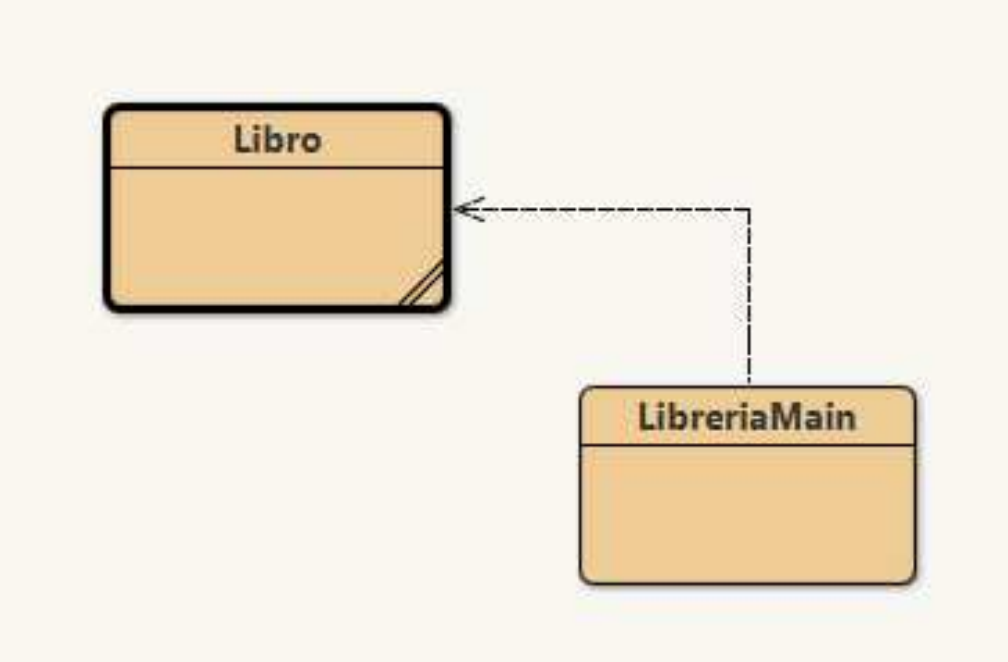
Estructura de un Archivo XML

Ejemplo Simple de Archivo XML

```
<Libro>
<Titulo>El Señor de los
Anillos</Titulo>
<Autor>J.R.R. Tolkien</Autor>
<Año>1954</Año>
</Libro>
```

En este ejemplo, se puede observar cómo se estructuran los datos usando etiquetas XML. Cada etiqueta define un elemento dentro del documento, permitiendo una clara representación de la información.

Ejemplo de Clase Persistente con Serialización en NetBeans: Librería



Ejemplo de Clase Persistente con Serialización en NetBeans: Librería

```
import java.io.Serializable;
public class Libro implements Serializable {
    private String titulo;
    private String autor;
    private int añoPublicacion;
    public Libro(String titulo, String autor, int añoPublicacion) {
        this.titulo = titulo;
        this.autor = autor;
        this.añoPublicacion = añoPublicacion;
    }
    // Getters y Setters
    public String getTitulo() {
        return titulo;
    }
    public void setTitulo(String titulo) {
        this.titulo = titulo;
    }
```

Clase Libro

```
// Getters y Setters
    public String getTitulo() {
        return titulo;
    }
    public void setTitulo(String titulo) {
        this.titulo = titulo;
    }
    public String getAutor() {
        return autor;
    }
    public void setAutor(String autor) {
        this.autor = autor;
    }
    public int getAñoPublicacion() {
        return añoPublicacion;
    }
    public void setAñoPublicacion(int añoPublicacion) {
        this.añoPublicacion = añoPublicacion;
    }
    // Método para mostrar información del libro
    public void mostrarInformacion() {
        System.out.println("Título: " + titulo);
        System.out.println("Autor: " + autor);
        System.out.println("Año de Publicación: " + añoPublicacion);
    }
}
```

Clase Libro

Ejemplo de Clase Persistente con Serialización en NetBeans: Librería

```
import java.io.*;
import java.util.ArrayList;
public class LibreriaMain {
    private static final String ARCHIVO_XML = "libros.xml";
    public static void main(String[] args) {
        ArrayList<Libro> libros = new ArrayList<>();
        // Crear algunos libros
        Libro libro1 = new Libro("Cien años de soledad", "Gabriel García Márquez", 1967);
        Libro libro2 = new Libro("El señor de los anillos", "J.R.R. Tolkien", 1954);
        Libro libro3 = new Libro("Harry Potter y la piedra filosofal", "J.K. Rowling", 1997);
        // Agregar libros a la lista
        libros.add(libro1);
        libros.add(libro2);
        libros.add(libro3);
        // Serializar la lista de libros a un archivo XML
        serializarLibros(libros);
        // Deserializar y mostrar la información de los libros desde el archivo XML
        ArrayList<Libro> librosRecuperados = deserializarLibros();
        if (librosRecuperados != null) {
            for (Libro libro : librosRecuperados) {
                libro.mostrarInformacion();
                System.out.println("--------------------------");
            }
        }
    }
```

Clase Main

Clase principal para manejar la Serialización y Deserialización

```
// Método para serializar libros a un archivo XML
   private static void serializarLibros(ArrayList<Libro> libros) {
      try {
         FileOutputStream fileOutputStream = new FileOutputStream(ARCHIVO_XML);
         ObjectOutputStream objectOutputStream = new
ObjectOutputStream(fileOutputStream);
         objectOutputStream.writeObject(libros);
         objectOutputStream.close();
         fileOutputStream.close();
         System.out.println("Libros serializados correctamente.");
      } catch (IOException e) {
         e.printStackTrace();
      }
   }
   // Método para deserializar libros desde un archivo XML
   private static ArrayList<Libro> deserializarLibros() {
      ArrayList<Libro> librosRecuperados = null;
      try {
         FileInputStream fileInputStream = new FileInputStream(ARCHIVO_XML);
         ObjectInputStream objectInputStream = new ObjectInputStream(fileInputStream);
         librosRecuperados = (ArrayList<Libro>) objectInputStream.readObject();
         objectInputStream.close();
         fileInputStream.close();
      } catch (IOException | ClassNotFoundException e) {
         e.printStackTrace();
      }
      return librosRecuperados;
   }
}
```

Clase Main

Clase principal para manejar la Serialización y Deserialización

Universidad Tecnológica del Perú

Explicación del codigo

Clase libro

Descripción:

- **Implementación de Serializable:** La clase Libro implementa la interfaz Serializable, lo que permite que sus objetos sean convertidos a un flujo de bytes para ser almacenados en un archivo o transmitidos.
- **Atributos:** La clase tiene tres atributos: titulo, autor y añoPublicacion.
- **Constructor:** El constructor inicializa los atributos con valores proporcionados.
- **Getters y Setters:** Métodos para acceder y modificar los atributos privados.
- **Mostrar información:** Un método mostrar Informacion que imprime la información del libro.

Explicación del codigo

Clase LibreriaMain

Descripción:

Constante ARCHIVO_XML: Define el nombre del archivo donde se almacenarán los libros serializados.

Método main:

- Crea una lista de libros (ArrayList<Libro>).
- Añade varios libros a la lista.
- Serializa la lista de libros al archivo XML utilizando serializarLibros.
- Deserializa la lista de libros desde el archivo XML utilizando deserializarLibros y muestra su información.

Explicación del codigo

Clase LibreriaMain

Método serializarLibros:

- Abre un FileOutputStream para escribir en el archivo.
- Crea un ObjectOutputStream para serializar el objeto.
- Escribe la lista de libros en el archivo.
- Cierra los streams.

Método deserializarLibros:

- Abre un FileInputStream para leer desde el archivo.
- Crea un ObjectInputStream para deserializar el objeto.
- Lee la lista de libros del archivo.
- Cierra los streams.

Explicación del codigo

- Este código crea una **clase Libro** con atributos de título, autor y año de publicación.
- La clase **LibreriaMain** utiliza la serialización para guardar una lista de libros en un **archivo XML** y luego deserializa esa lista, mostrando la información de los libros recuperados.
- Este código demuestra cómo crear una lista de objetos, serializarla a un archivo y luego deserializarla para recuperar los objetos y utilizarlos.

Practica

Implementa los ejercicios vistos.

Conclusiones:

- La serialización es fundamental para la persistencia de datos, ya que permite que la información sea guardada en un formato comprensible y recuperable, lo que facilita su utilización a lo largo del tiempo y en diferentes contextos.
- El uso de archivos XML como formato de serialización ofrece una forma estructurada y legible de almacenar datos, siendo especialmente útil para el intercambio de información entre diferentes sistemas o para la persistencia de datos en aplicaciones.

Universidad Tecnológica del Perú

Cierre

¿Qué hemos aprendido hoy?

Bibliografía

- MORENO PÉREZ, J. “Programación orientada a objetos”. RA-MA Editorial. https://tubiblioteca.utp.edu.pe/cgi-bin/koha/opac-detail.pl?biblionumber=31933
- Vélez Serrano, José. “Diseñar y programar, todo es empezar: una introducción a la Programación Orientada a Objetos usando UML y Java”. Dykinson. https://tubiblioteca.utp.edu.pe/cgi-bin/koha/opac-detail.pl?biblionumber=36368

Universidad Tecnológica del Perú