Manejador completo XML-JSON-SQL

David Gómez Rivas

12 de enero de 2024

Índice

1. Introducción

Partiendo de una tabla reservasçon los siguientes campos:

- nombre (Varchar)
- telefono (Varchar)
- fecha_evento (Varchar)
- tipo_evento (Varchar)
- n_personas (Varchar)
- tipo_cocina (Varchar)
- n_jornadas (Varchar)
- n_habitaciones (Varchar)
- tipo_mesa (Varchar)
- n_comensales (Varchar)

Crea una función en tu manejador universal que permita obtener reservas de la base de datos, para exportarlos a un XML o un JSON.

2. Análisis del Problema

El desafío central de este ejercicio es el desarrollo de una función que pueda interactuar con la base de datos de reservas. La tabla de reservas consta de diversos campos, cada uno almacenando información clave sobre las reservas, como nombre, teléfono, fecha del evento, tipo de evento, número de personas, tipo de cocina, número de jornadas, número de habitaciones, tipo de mesa y número de comensales, todos almacenados como cadenas de caracteres (Varchar).

En resumen, la solución propuesta debe abordar los siguientes puntos clave: 1. Interacción segura y eficiente con la base de datos. 2. Procesamiento y validación adecuados de los datos. 3. Capacidad para exportar datos en formatos XML y JSON manteniendo sus estructuras. 4. Flexibilidad y escalabilidad para adaptarse a posibles cambios en los requisitos de la base de datos.

3. Solución Propuesta

La solución consistirá en hacer el método ‘exportReservas‘, que se encargará de exportar las reservas a los formatos XML o JSON, según se especifique. Este método aceptará una lista de objetos ‘Reserva‘, un string ‘formato‘ que determinará si la salida será en XML o JSON, y un string ‘salida‘ que definirá el nombre del archivo de salida.

Para el caso de XML, se utilizará ‘DocumentBuilderFactory‘ para crear un nuevo documento XML. Por cada objeto ‘Reserva‘ en la lista, se creará un elemento ‘reserva‘ dentro del documento, donde cada campo de la reserva se añadirá como un subelemento. Una vez construido el documento XML, se utilizará ‘Transformer‘ para escribir el XML en un archivo.

En cuanto a la exportación a JSON, se utilizará un ‘JSONArray‘ para almacenar los objetos JSON correspondientes a cada reserva. Cada reserva se convertirá en un ‘JSONObject‘ con sus respectivos campos. Este array se escribirá luego en un archivo JSON, utilizando una indentación para facilitar su lectura.

El método también incluirá manejo de excepciones para garantizar que se notifiquen los errores, como un formato no soportado, y asegurar que el archivo se escriba correctamente.

Este enfoque garantiza una solución flexible y adaptable, permitiendo la exportación de datos en dos de los formatos más comunes para el intercambio de datos, y proporcionando una base sólida para futuras expansiones o modificaciones del método.

Con esta solución, se ofrece una herramienta eficiente y versátil para la exportación de datos, fundamental en un contexto donde la interoperabilidad y la flexibilidad de formatos son clave para la integración de sistemas.

4. Analisis, diseño, pseudocódigo y diagrama

4.1. Análisis

4.1.1. Definición del problema

El problema consiste en desarrollar un método eficiente y versátil, 'exportReservas', capaz de extraer datos de una base de datos de reservas y exportarlos en dos formatos distintos: XML y JSON. Este proceso debe manejar datos variados relacionados con reservas, como nombre, teléfono, fecha del evento, entre otros, y debe ser capaz de adaptarse a diferentes estructuras de datos, manteniendo la integridad de los datos.

4.1.2. Datos de Entrada

Los datos de entrada incluyen:

- Una lista de objetos 'Reserva', donde cada 'Reserva' contiene los campos: nombre, teléfono, fecha del evento, tipo de evento, número de personas, tipo de cocina, número de jornadas, número de habitaciones, tipo de mesa, y número de comensales.
- Un string 'formato' que especifica el formato de salida deseado (XML o JSON).
- Un string 'salida' que define el nombre del archivo de salida.

4.1.3. Información de salida

La información de salida será un archivo en el formato especificado (XML o JSON), que contendrá la representación de los datos de las reservas. Cada reserva estará detallada con todos sus campos relevantes.

4.1.4. Variables

Las variables involucradas en el método 'exportReservas' incluyen:

- **reservas**: Lista de objetos 'Reserva' (Tipo: Lista de Objetos).
- **formato**: Tipo de formato de salida (Tipo: String).
- **salida**: Nombre del archivo de salida (Tipo: String).
- Variables internas para la construcción de los formatos XML y JSON, como 'document', 'elemento', 'jsonArray', 'jsonObjeto', entre otros (Tipo: Diversos, dependiendo de su uso específico).

4.2. Diseño

Podemos dividir el problema de la exportación de reservas en problemas más pequeños:

- 1. Leer la lista de objetos 'Reserva'.
- 2. Determinar el formato de salida (XML o JSON).
- 3. Construir la estructura del archivo de salida según el formato elegido.
- 4. Escribir los datos en el archivo.

4.3. Refinamiento del algoritmo

4.3.1. Leer la lista de reservas

Recibimos una lista de objetos 'Reserva', cada uno con campos como nombre, teléfono, fecha del evento, etc.

4.3.2. Determinar el formato de salida

Basándonos en el string 'formato' proporcionado, decidimos si la salida será en formato XML o JSON.

4.3.3. Construir la estructura del archivo de salida

Para XML:

- Crear undocumento XML.
- Añadir elementos para cada reserva y sus detalles.
- Utilizar la clase DocumentBuilder y sus métodos para construir el documento.

Para JSON:

- Inicializar un JSONArray.
- Convertir cada objeto Reserva en un JSONObject y añadirlo al array.
- Utilizar las clases de manejo de JSON para estructurar los datos.

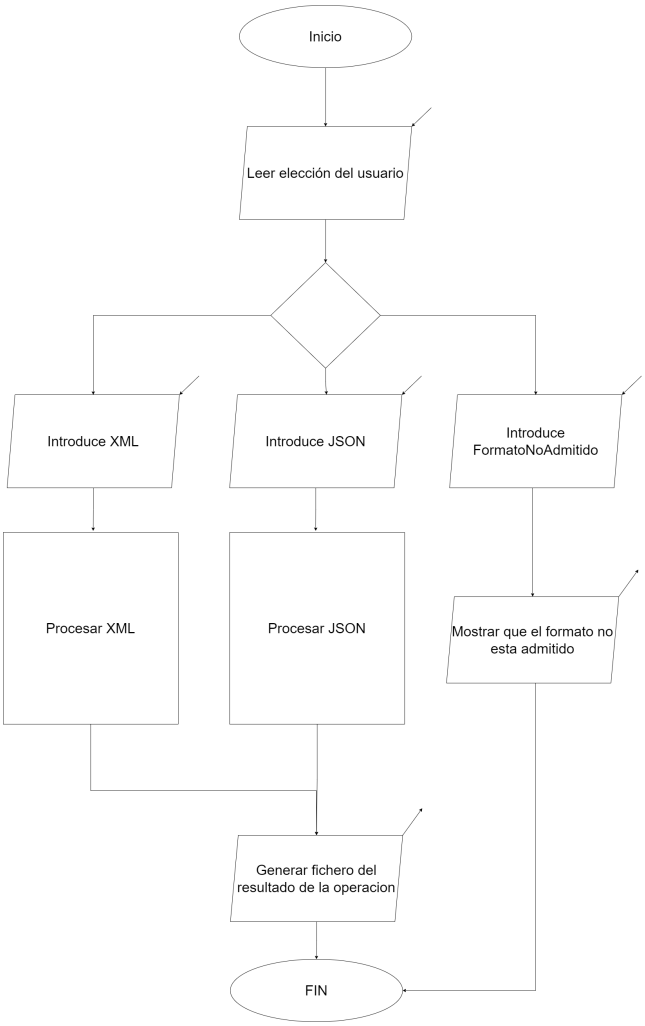
4.3.4. Escribir los datos en el archivo

- Para XML: Utilizar Transformer para escribir el documento XML en un archivo.
- Para JSON: Utilizar un FileWriter para escribir el JSONArray en un archivo JSON.

4.4. Pseudocodigo

```
Proceso exportarReservas
    Definir reservas Como Lista de Objetos Reserva;
    Definir formato, salida Como Cadena;
    Definir documento, raiz, reservaElemento, transformador Como XML;
    Definir jsonArray, jsonObjeto Como JSON;
    Definir archivo Como Archivo;

    Leer reservas, formato, salida;

    Si formato es igual a "XML" Entonces
        documento <- CrearDocumentoXML();
        raiz <- documento.CrearElemento("reservas");
        documento.Agregar(raiz);

        Para Cada reserva en reservas Hacer
            reservaElemento <- documento.CrearElemento("reserva");
            raiz.Agregar(reservaElemento);
            reservaElemento.Agregar(CrearElemento(documento, "nombre", reserva.n
            // Repetir para cada campo de Reserva
        FinPara

        transformador <- CrearTransformadorXML();
        archivo <- CrearArchivo(salida);
        transformador.Transformar(documento, archivo);

    Sino Si formato es igual a "JSON" Entonces
        jsonArray <- CrearJSONArray();

        Para Cada reserva en reservas Hacer
            jsonObjeto <- CrearJSONObject();
            jsonObjeto.Agregar("nombre", reserva.nombre);
            // Repetir para cada campo de Reserva
            jsonArray.Agregar(jsonObjeto);
    FinPara

    archivo <- CrearArchivo(salida);
    EscribirEnArchivo(archivo, jsonArray.FormatearComoString());
    Sino
        Lanzar Excepcion("Formato no soportado: " + formato);
    FinSi
FinProceso
```

4.5. Diagrama de flujo



5. Conclusión

En general creo que es una solucion al problema bastante buena, creo que es bastante optimo que resuelve el problema, como todo en esta vida seguro que se puede mejorar