IES CHAN DO MONTE
C.S. de Desarrollo de Aplicaciones Multiplataforma
Modulo Acceso a datos

ACTIVIDAD 5: Persistencia XML (StAX)

Esta actividad tiene como objetivo fundamental la **aplicación práctica de la Streaming API for XML (StAX)** dentro de la arquitectura por capas de la aplicación de gestión de corredores y equipos.

StAX, a diferencia del modelo DOM (orientado a memoria) y SAX (modelo *push*), opera bajo un modelo **"pull"** . Este enfoque permite al programador **controlar activamente el flujo de lectura**, solicitando el siguiente evento solo cuando es necesario, lo que resulta en una gestión más eficiente del consumo de memoria y un mejor rendimiento para el procesamiento de documentos XML de gran tamaño.

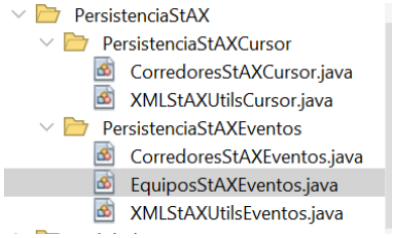
Objetivos

- Consolidar el manejo del modelo StAX Cursor (XMLStreamReader y XMLStreamWriter) para la lectura e escritura secuencial y eficiente de documentos XML.
- Aplicar el modelo StAX Eventos (XMLEventReader y XMLEventWriter) para la lectura y escritura orientada a objetos, mejorando la legibilidad del código.
- Comparar ambos modelos de StAX (Cursor vs. Eventos) en términos de complejidad, legibilidad y acceso a los datos.
- Integrar la lectura StAX con la modificación DOM para realizar la operación de actualización de datos, reforzando la separación de responsabilidades entre capas.

Esta práctica se plantea como una continuación del desarrollo de la aplicación de gestión de corredores, reutilizando las clases ya creadas en el paquete Clases (Corredor, Velocista, Fondista, Puntuación, Equipo, Patrocinador) y manteniendo la arquitectura por capas establecida.

Con el objetivo de aplicar el modelo **StAX** para el procesamiento eficiente de documentos XML, se creará un nuevo paquete llamado PersistenciaStAX. Para hacer ejercicios de los dos modelos que implementa, se crearán los paquetes:

1. PersistenciaStAXCursor: Contiene las clases que utilizan **el modelo Cursor** (XMLStreamReader y XMLStreamWriter):
   a. CorredoresStAXCursor.java
   b. XMLStAXUtilsCursor.java
2. PersistenciaStAXEventos: Contiene las clases que utilizan el modelo **Eventos** (XMLEventReader y XMLEventWriter):
   - CorredoresStAXEventos.java
   - EquiposStAXEventos.java
   - XMLStAXUtilsEventos.java



✓ **PersistenciaStAXCursor**

Este paquete contendrá las clases que implementan la lógica de StAX utilizando el modelo **Cursor**.

- **CorredoresStAXCursor.java:** Encapsula la lógica de acceso a corredores.xml utilizando XMLStreamReader para operaciones de lectura.
- **XMLStAXUtilsCursor.java:** Clase de utilidad central que manejará la **creación** y el **cierre** del XMLStreamReader y el XMLStreamWriter, además de contener métodos auxiliares específicos para la lectura/escritura del modelo Cursor.

✓ **PersistenciaStAXEventos**

Este paquete contendrá las clases que implementan la lógica de StAX utilizando el modelo **Eventos**.

- **CorredoresStAXEventos.java:** Encapsula la lógica de acceso a corredores.xml utilizando XMLEventReader para operaciones de lectura.
- **EquiposStAXEventos.java:** Clase encargada de la lógica de **actualización StAX a DOM** y la lógica en equipos.xml
- **XMLStAXUtilsEventos.java:** Clase de utilidad central que manejará la **creación** y el **cierre s** del XMLEventReader y el XMLEventWriter, además de contener métodos auxiliares específicos para la lectura/escritura del modelo Eventos.

✓ **Definición de la Capa de Utilidad (XMLStAXUtils)**

Las clases de utilidad son responsables de la configuración inicial, la validación, la gestión del *stream* de entrada (I/O) y el suministro de métodos auxiliares específicos para cada modelo StAX.

- **XMLStAXUtilsCursor.java (Modelo Cursor)**
  - **Rol Principal:** Actúa como utilidad central para crear y configurar el XMLStreamReader y el XMLStreamWriter. Se encarga de la validación de parámetros, la existencia del fichero y la validación estructural (XSD/DTD).
  - **Método public static XMLStreamReader cargarDocumentoStAX(String rutaFichero, TipoValidacion validacion):** Este método comprueba la validez de la ruta y del tipo de validación, verifica la existencia del archivo e invoca los métodos internos (validarConXSD o validarConDTD) para realizar la validación estructural. Finalmente, crea la XMLInputFactory y devuelve un XMLStreamReader listo para el procesamiento.
  - **Métodos Auxiliares:** Contiene métodos estáticos como obtenerNombreEtiqueta(lector), leerAtributo(lector, nombre) y leerTexto(lector) para simplificar la navegación y extracción de datos dentro del bucle Cursor.
- **XMLStAXUtilsEventos.java (Modelo Eventos)**
  - **Rol Principal:** Cumple el mismo rol que la utilidad Cursor, pero enfocado en el modelo Eventos. Gestiona la creación y configuración del XMLEventReader y el XMLEventWriter.
  - **Método public static XMLEventReader cargarDocumentoStAX(String rutaFichero, TipoValidacion validacion):** Este método reutiliza la lógica de validación de parámetros y existencia de fichero. Crea la XMLInputFactory y devuelve un **XMLEventReader** que emite objetos XMLEvent completos.
  - **Métodos Auxiliares:** Contiene métodos estáticos para simplificar la extracción de datos basada en objetos XMLEvent, como obtenerNombreEtiqueta(evento) o leerAtributo(startElement, nombre).

1.- **Operación de listar todos los corredores utilizando el procesador StAX** y con validacion con esquemas.

De las dos maneras: Stax por Cursor y StAX por Eventos

- ✓ Implementar **public List<Corredor> cargarTodosCorredoresCursor(String rutaXML, TipoValidacion validacion)** en **CorredoresStAXCursor.java.**
- ✓ Implementar **public List<Corredor> cargarTodosCorredoresEventos(String rutaXML, TipoValidacion validacion)** en **CorredoresStAXEventos.java.**
- ✓ En la capa Lógica **(GestorCorredores.java),** cree métodos para invocar ambas implementaciones y visualizar los resultados.

2.- Visualizar corredores de un equipo dado

Procesar el fichero corredores.xml y filtrar los corredores cuyo equipo coincida con el nombre de equipo pasado como parámetro.

Hacer de dos maneras: Stax por Cursor y StAX por Eventos. Los métodos deben devolver una lista de corredores perteneciente al equipo buscado.

- ✓ Implementar **public List<Corredor> leerCorredoresPorEquipo(XMLStreamReader reader, String equipoBuscado)**
- ✓ Implementar **public List<Corredor> leerCorredoresPorEquipo(XMLEventReader reader, String equipoBuscado)**

3.- **Actualización de equipos y patrocinadores mediante StAX → DOM**

Dado el documento XML `equipos.xml` que contiene información sobre equipos deportivos y sus patrocinadores. Y dado un segundo documento ActualizacionesEquipo`.xml` con nuevas donaciones, fechas y patrocinadores que deben incorporarse o actualizarse en el documento original.

Hay que actualizar el documento **DOM equipos.xml con las actualizaciones provenientes de ActualizacionesEquipo`.xml` leyendo con StAX por Eventos ,** aplicando los siguientes criterios:

Lectura StAX por Eventos:

- Procesa el documento `actualizaciones.xml` usando StAX.
- Extrae los datos de cada `<Patrocinador>`: nombre, donación, fecha y equipo asociado.

Actualización DOM:

- Si el equipo no existe, crea el equipo y añade el patrocinador.
- Si el patrocinador ya existe en el equipo, actualiza su donación y fecha.
- Si el patrocinador no existe, añádelo al equipo correspondiente.

Salida:

- Guarda el documento actualizado como `equiposUpdateStAX.xml`.

Estructura de ActualizacionesEquipo`.xml`

```
<?xml version="1.0" encoding="UTF-8"?>
<Actualizaciones>
    <!-- Caso 1: Actualizar donación y fecha de un patrocinador existente -->
    <Patrocinador idEquipo="E1" nombreEquipo="Rápidos del Norte">
        <nombre>Adidas</nombre>
        <Donacion fecha="2024-10-12">2500.0</Donacion>
    </Patrocinador>

    <!-- Caso 2: Añadir nuevo patrocinador a equipo existente -->
    <Patrocinador idEquipo="E2" nombreEquipo="Truenos del Oeste">
        <nombre>Columbia</nombre>
        <Donacion fecha="2025-03-01">1800.0</Donacion>
    </Patrocinador>
……..
    <!-- Caso 5: Añadir patrocinador a equipo nuevo -->
    <Patrocinador idEquipo="E9" nombreEquipo="Relámpagos del Este">
        <nombre>Joma</nombre>
        <Donacion fecha="2025-05-10">1200.0</Donacion>
    </Patrocinador>
</Actualizaciones>
```

4.- **Cálculo de Donación Total Global por Patrocinador**

A partir del fichero fuente equipos.xml, tenemos que crear un fichero XML llamado **DonacionesTotales.xml**.que cumpla con los siguientes criterios:

1. **Extracción y Cálculo:** Extraer el nombre y la donación de cada patrocinador. Debe calcular la **donación total global** de cada patrocinador único (agregación de todas sus donaciones a todos los equipos).
2. **Ordenamiento:** El xml debe estar **ordenado alfabéticamente por el nombre del patrocinador**.

Estructura XML Requerida

El documento debe seguir la siguiente estructura, con la donación total como un **atributo**:

- **Raíz:** <donaciones>
- **Hijo:** <Patrocinador totalDonado="[VALOR]">Nombre Patrocinador</Patrocinador>

Implementación de Utilidades (Helpers)

Añade en las clases correspondientes de utilidades StAX métodos generales para la escritura que contenga los métodos necesarios para:

- Añadir la **Instrucción de Procesamiento XML** (<?xml...?>).
- Añadir **Saltos de Línea e Identación** por nivel.
- Añadir **Atributos** a los elementos.
- Añadir **Elementos con Valor** y **sin Valor**.
- Etc..

Generación:

El fichero de salida debe generarse de **dos maneras distintas**, ambas utilizando la API StAX y haciendo uso de las utilidades generadas anteriormente

1. Generación mediante el **Modelo CURSOR**.
2. Generación mediante el **Modelo EVENTOS**

Ejemplo:

```
<?xml version="1.0" encoding="UTF-8"?>
<donaciones>
    <Patrocinador totalDonado="4900.0">Adidas</Patrocinador>
    <Patrocinador totalDonado="1450.0">ASICS</Patrocinador>
    <Patrocinador totalDonado="1450.0">Brooks</Patrocinador>
    <Patrocinador totalDonado="1700.0">Decathlon</Patrocinador>
    <Patrocinador totalDonado="1900.0">HOKA</Patrocinador>
    <Patrocinador totalDonado="2000.0">Lululemon</Patrocinador>
    <Patrocinador totalDonado="1700.0">Mizuno</Patrocinador>
    <Patrocinador totalDonado="1700.0">New Balance</Patrocinador>
    <Patrocinador totalDonado="6400.0">Nike</Patrocinador>
    <Patrocinador totalDonado="3600.0">Puma</Patrocinador>
    <Patrocinador totalDonado="1600.0">Reebok</Patrocinador>
    <Patrocinador totalDonado="1500.0">Salomon</Patrocinador>
    <Patrocinador totalDonado="950.0">Saucony</Patrocinador>
    <Patrocinador totalDonado="600.0">Skechers</Patrocinador>
    <Patrocinador totalDonado="2100.0">Under Armour</Patrocinador>
</donaciones>
```