

IES CHAN DO MONTE**C.S. de Desarrollo de Aplicaciones Multiplataforma****Modulo Acceso a datos****ACTIVIDAD 4: Persistencia XML (SAX)**

Esta práctica tiene como finalidad aplicar de forma integrada el manejo de ficheros XML mediante el modelo SAX (Simple API for XML), la validación estructural del documento utilizando DTD o esquemas XSD, y la construcción de objetos Java a partir de los datos procesados. Se mantendrá la estructura de capas de la aplicación y la jerarquía de clases modelo existente, reforzando el diseño modular y la separación de responsabilidades

Objetivos

- Consolidar el manejo del modelo **SAX** para la lectura secuencial y eficiente de documentos XML.
- Aplicar técnicas de **procesamiento por eventos** para extraer información y construir objetos Java.
- Garantizar la **integridad estructural** del documento XML mediante validación con **DTD o XSD**.
- Reforzar el principio de **arquitectura por capas**, delegando la persistencia XML a la capa Persistencia.
- Comparar el modelo SAX con DOM en términos de rendimiento, consumo de memoria y complejidad de implementación.

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 SAX (Simple API for XML) para el procesamiento eficiente de documentos XML, se **creará un nuevo paquete** llamado **PersistenciaSAX**.

Desde el punto de vista didáctico, se busca separar claramente el enfoque SAX del modelo DOM trabajado previamente, permitiendo comparar ambos paradigmas de procesamiento XML.

En este paquete se incluirán todas las clases necesarias para gestionar la persistencia de los datos contenidos en los ficheros corredores.xml y equipos.xml, utilizando el procesador SAX para construir los objetos correspondientes a partir de los datos XML.

1.- Operación de listar todos los corredores utilizando el procesador SAX y con validación con esquemas.

Hay que implementar **las siguientes clases**: (Estas nos servirá para

- La clase **XMLSAXUtils**:

Esta clase actuará como **utilidad central para el procesamiento de documentos XML mediante SAX**, y será utilizada por cualquier clase que necesite cargar y analizar un fichero XML, como por ejemplo CorredoresSAX o EquiposSAX.

Métodos:

- **Método principal para procesar un documento XML con SAX**

.Este método es el punto de entrada para procesar un documento XML. Se encarga de:

- ✓ Validar que la ruta del fichero no esté vacía.
- ✓ Validar que el tipo de validación no sea nulo.
- ✓ Comprobar que el fichero XML existe en disco.
- ✓ Configurar el parser SAX llamando un método que lo configure
- ✓ Crear el parser SAX.
- ✓ Ejecutar el análisis del documento con parser.parse(...), utilizando el manejador proporcionado.

Parámetros:

- rutaFichero: ruta al archivo XML que se desea procesar (por ejemplo, "corredores.xml").
- handler: Manejador de eventos instancia de una clase que extiende DefaultHandler, encargada de gestionar los eventos SAX.

- validacion: tipo de validación deseada (NO_VALIDAR, DTD, XSD), definido en la clase TipoValidacion

- Método auxiliar para configurar el parser SAX

Método se encarga de configurar la factoría de parsers SAX según el tipo de validación solicitado. Es llamado internamente por el método anterior.

Acciones según el tipo de validación:

DTD: activa la validación básica con DTD.

XSD: activa la validación con esquema XML, habilitando las características necesarias (validation y schema).

NO_VALIDAR: desactiva la validación

Parámetros:

validacion: tipo de validación deseada (NO_VALIDAR, DTD, XSD).

Devuelve:

Una instancia de SAXParserFactory correctamente configurada para crear el parser SAX.

- **Clase para el handler o manejador SAX** que permita procesar el fichero corredores.xml y **construir una lista completa de objetos Corredor**, que pueden ser instancias de Velocista o Fondista. (Por ejemplo CorredoresSaxHandler)

Esta clase se ubicará en el paquete PersistenciaSAX y actuará como manejador de eventos SAX, interpretando las etiquetas del XML y construyendo los objetos correspondientes.

El resultado será una lista de corredores que podrá ser utilizada por otras clases de persistencia.

Hay que implementar los métodos (startElement, characters, endElement, Etc..) para:

- Detectar el tipo de corredor (velocista, fondista).
- Capturar atributos (codigo, dorsal, equipo) y elementos (nombre, fecha_nacimiento, etc.).
- Construir el historial de puntuaciones si existe.
- Añadir cada corredor completo a una lista interna.

Implementar el método getCorredores() que devuelva la lista final de corredores.

- Clase que **permite y encapsula la logica de acceso XML con SAX** (Por ejemplo CorredoresSAX)

El objetivo de esta clase es encapsular la lógica común para **procesar documentos XML mediante SAX**, delegando el análisis en distintos manejadores (DefaultHandler) según las diferentes operaciones que se quieran sacar.

Ejemplo:

```
/**
 * Carga todos los corredores del XML.
 *
 * @param rutaXML
 * @param validacion
 * @return Lista de todos los corredores
 */
public List<Corredor> cargarTodosCorredores(String rutaXML, TipoValidacion validacion) throws ExcepcionXML {
    CorredorSAXHandler handler = new CorredorSAXHandler();
    XMLSAXUtils.cargarDocumentoSAX(rutaXML, handler, validacion);
    return handler.getCorredores();
}
```

- Implementar un método en la clase GestorCorredores (en el paquete de la logica de negocio) que: visualice por pantalla la información de todos los corredores obtenidos desde el XML y procesandolo con SAX, incluyendo:

- ✓ Tipo de corredor (Fondista o Velocista).
- ✓ Datos personales (código, nombre, fecha de nacimiento, dorsal, equipo).
- ✓ Historial de puntuaciones (si existe).
- ✓ Mensaje claro si no tiene puntuaciones registradas.
- ✓ Mensajes de error si ocurre alguna excepción.

Pon ejemplo de llamada a este metodo en el main

2.- Visualizar corredores de un equipo dado

Parte A: Procesamiento con SAX

1. **Crear un manejador SAX personalizado** que procese el fichero corredores.xml y filtre los corredores cuyo equipo coincida con el nombre de equipo pasado como parámetro.
2. **Implementar un método en la capa de persistencia** que invoque el manejador SAX y devuelva la lista de corredores filtrados.
3. **Desde la lógica de negocio**, mostrar por pantalla la información de los corredores obtenidos.
4. Si no se encuentra ningún corredor del equipo indicado, se debe mostrar un mensaje informativo indicando que el equipo no existe en el fichero.

Parte B: Procesamiento con DOM

1. **Crear un método que utilice el parser DOM** para obtener los corredores cuyo equipo coincida con el parámetro recibido.
2. **Implementar este método en la capa de persistencia**, devolviendo la lista de corredores filtrados.
3. **Desde la lógica de negocio**, mostrar por pantalla la información de los corredores obtenidos.
4. Al igual que en la parte A, si no se encuentra ningún corredor del equipo indicado, se debe mostrar un mensaje informativo indicando que el equipo no existe en el fichero.

3.- Actualización de equipos y patrocinadores mediante SAX → 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 SAX**, aplicando los siguientes criterios:

Lectura SAX:

- Procesa el documento actualizaciones.xml usando un manejador SAX.
- 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 equiposUpdate.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>
```