UD 1. Manejo de ficheros XML. Introducción a JAXB

Java Architecture for XML Binding (**JAXB**) permite a los desarrolladores Java asignar clases de Java a representaciones XML. JAXB proporciona dos características principales: la capacidad de serializar las referencias de objetos Java a XML y la inversa, es decir, deserializar XML en objetos Java. JAXB provee dos funciones fundamentales:

- La capacidad de presentar un objeto Java en XML (serializar), al proceso lo llamaremos *marshall* o *marshalling*. Java Object a XML.
- Lo contrario, es decir, presenta un XML en un objeto Java (deserializar), al proceso lo llamaremos *unmarshall* o *unmarshalling*. XML a Java Object.

También el compilador que proporciona JAXB nos va a permitir generar clases Java a partir de esquemas XML, que podrán ser llamadas desde las aplicaciones a través de métodos sets o gets para obtener o establecer los datos de un documento XML.

Para crear objetos Java en XML, vamos a utilizar JavaBeans, que serán las clases que se van a mapear. Son las clases primitivas Java con las propiedades getter y setter, el constructor sin parámetros y el constructor con las propiedades. En estas clases que se van a mapear se añadirán las Anotaciones, que son las indicaciones que ayudan a convertir el JavaBean en XML.

Las principales anotaciones son:

- **@XmlRootElement(namespace=”namespace”).** Define la raíz del XML. Namespace es opcional.
- **@XmlType(propOrder={"field2”, “field1”, …})**. Permite definir en qué orden se van a escribir los elementos (o las etiquetas) dentro del XML. Si es una clase que no va a ser raíz añadiremos @XmlType.
- **@XmlElement(name=”nombre”)**. Define el elemento de XML que se va a usar. A cualquiera de ellos podemos ponerle entre paréntesis el nombre de etiqueta que queramos que salga en el documento XML para la clase, añadiendo el atributo name.
- **@XmlAttribute(name =”nombre”).** Define un atributo XML podemos ponerle entre paréntesis el nombre de etiqueta que queramos que salga en el documento XML para el atributo, añadiendo el atributo name.

Para cada atributo de la clase que queramos que salga en el XML, el método get correspondiente a ese atributo debe llevar una anotación @XmlElement, a la que a su vez podemos ponerle un nombre (estas anotaciones no son obligatorias, solo si se desean nombres diferentes del atributo).

Si el atributo es una colección (array, list, etc…) debe llevar dos anotaciones @XmlElementWrapper y @XmlElement, esta última con un nombre si se desea.

Ejemplo lectura XML con JAXB

En primer lugar tenemos que definir las estructuras de datos que nos permitan leer el siguiente fichero xml.

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<libreria>
        <nombre>Libreria técnica</nombre>
        <libros>
                <libro isbn="9788441536258">
                        <titulo>Java 8</titulo>
                        <autor>Herbert Schildt</autor>
                </libro>
                <libro isbn="9788415033370">
                        <titulo>Curso de programación web</titulo>
                        <autor>Scott Mccracken</autor>
                </libro>
                <libro isbn="9788426722126">
                        <titulo>Python fácil</titulo>
                        <autor>Arnaldo Pérez Castaño</autor>
                </libro>
        </libros>
</libreria>
```

Para ello, creamos la clase Librería, con las anotaciones correspondientes que nos permitan luego interpretar correctamente la estructura del documento XML:

```java
package lecturajaxb;

import java.util.ArrayList;
import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlElementWrapper;
import javax.xml.bind.annotation.XmlRootElement;
import javax.xml.bind.annotation.XmlType;

@XmlRootElement(name="libreria") // Le indicamos que es el  elemento raiz
@XmlType(propOrder={"nombre","libros"})
public class Libreria {
    private String nombre;// Atributo que representa el nombre
    private ArrayList<Libro> libros = new ArrayList(); // coleccion de datos que guarde todos los libros

    public Libreria() { // Constructor
    }

    @XmlElement(name="nombre") // Le indicamos
    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    @XmlElementWrapper(name="libros")
    @XmlElement(name="libro")
    public ArrayList<Libro> getLibros() {
        return libros;
    }

    public void setLibros(ArrayList<Libro> libros) {
        this.libros = libros;
    }
```

Y creamos la clase Libro, con las anotaciones correspondientes que nos permitan luego interpretar correctamente la estructura del documento XML:

```java
package lecturajaxb;

import javax.xml.bind.annotation.XmlAttribute;
import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlRootElement;
import javax.xml.bind.annotation.XmlType;

@XmlRootElement(name="libro")
@XmlType(propOrder={"isbn","titulo","autor"})
public class Libro { // Esta clase define cada uno de los libros del documento XML
    private String isbn;
    private String titulo;
    private String autor;

    public Libro() { // Constructor vacío
    }
    @XmlAttribute(name="isbn") // Le indicamos que es atributo
    public String getIsbn() {
        return isbn;
    }
    public void setIsbn(String isbn) {
        this.isbn = isbn;
    }
    @XmlElement(name="titulo") // Le indicamos que es elemento
    public String getTitulo() {
        return titulo;
    }
    public void setTitulo(String titulo) {
        this.titulo = titulo;
    }
    @XmlElement(name="autor") // Le indicamos que es elemento
    public String getAutor() {
        return autor;
    }
    public void setAutor(String autor) {
        this.autor = autor;
    }
}
```

Por último definimos en el MAIN el Unmarshaller que nos va a permitir leer el XML.

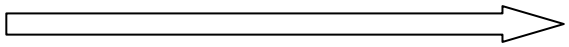
```java
package lecturajaxb;

import java.io.File;
import java.util.ArrayList;
import javax.xml.bind.JAXBContext;
import javax.xml.bind.JAXBException;
import javax.xml.bind.Unmarshaller;


public class LecturaJAXB {


    public static void main(String[] args) throws JAXBException {
        JAXBContext context = JAXBContext.newInstance(Libreria.class); // Necesito instancia de JAXBContext
        Unmarshaller unmarshaller = context.createUnmarshaller(); // Objeto que me permite leer XML
        Libreria libreria = (Libreria) unmarshaller.unmarshal(new File("libreria.xml"));

        System.out.println("Nombre: "+libreria.getNombre());

        ArrayList<Libro> libros = libreria.getLibros();

        for(Libro l: libros){
            System.out.println(l.getIsbn()+" "+l.getTitulo()+" "+l.getAutor());
        }
    }

}
```

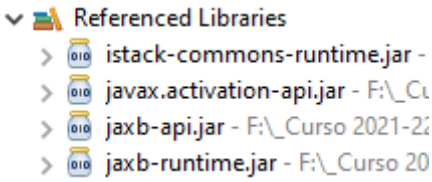
- Para leer los datos de un documento XML y convertirlos a objetos Java.
- Instanciamos el contexto indicando la clase que será el RootElement, en este caso Libreria.class.

- Se crea un **Unmarshaller** en el contexto de la clase Librería.
- Utilizamos el método **unmarshal** para obtener datos de un Reader (file).
- Recuperamos un atributo del objeto: el nombre de la librería
- Recuperamos el arraylist de objetos si lo tiene con .getLibros() y lo visualizamos

Nota: Dependiendo de la versión del JRE de Java puede ser necesario añadir archivos jar de la API JAXB => https://javaee.github.io/jaxb-v2/

Por ejemplo para el JavaSE-1. 8 necesitamos añadir los archivos de la imagen adjunta





Ejemplo escritura XML con JAXB

```
package lecturajaxb;

import java.io.FileWriter;
import java.io.IOException;
import java.util.ArrayList;
import javax.xml.bind.JAXBContext;
import javax.xml.bind.JAXBException;
import javax.xml.bind.Marshaller;

public class EscrituraJAXB {
    public static void main(String[] args) throws JAXBException, IOException{
        JAXBContext context = JAXBContext.newInstance(Libreria.class); // Necesito instancia de JAXBContext y
                                                                       // le indico la clase que define el documento XML
        Marshaller marshaller = context.createMarshaller();// Objeto que me permite escribir un XML
        // Ahora vamos a definir la informacion que queremos meter en el fichero
        Libreria libreria = new Libreria();//Creamos objeto Libreria
        libreria.setNombre("Libreria_1");
        // Añado a la Libreria objetos libro
        ArrayList<Libro> libros = new ArrayList();
        Libro libro = new Libro();
        libro.setIsbn("12345678");
        libro.setTitulo("Titulo1");
        libro.setAutor("Autor1");
        libros.add(libro);
        Libro libro2 = new Libro();
        libro2.setIsbn("87654321");
        libro2.setTitulo("Titulo2");
        libro2.setAutor("Autor2");
        libros.add(libro2);

        libreria.setLibros(libros); // Relleno la libreria con el array de libros
        //Volcamos esta información en memoria sobre un fichero XML
        marshaller.setProperty(Marshaller.JAXB_FORMATTED_OUTPUT, Boolean.TRUE);
        //marshaller.marshal(libreria, System.out);
        marshaller.marshal(libreria, new FileWriter("LibreriaEscritura.xml"));
    }
}
```

- Instanciamos el contexto, indicando la clase que será el **RootElement**, en nuestro ejemplo es la clase Librería (Librería.class)
- Creamos un **Marshaller** , que es la clase capaz de convertir nuestro JavaBean en una cadena XML.
- Definimos la información que queremos meter en el fichero.
- Indicamos que vamos a querer el XML con un formato amigable (saltos de línea, sangrías, etc) (JAXB_FORMATTED_OUTPUT TRUE)
- Hacemos la conversión llamando al método Marshal, pasando una instancia del JavaBean que queremos convertir a XML y un OutputStream donde queremos que salga el XML, por ejemplo la salida estándar, o en este caso un fichero.