## 1.9 INTRODUCCIÓN A JAXB.

JAXB es una tecnología java que permite mapear clases Java a representaciones XML, y viceversa, es decir serializar objetos java a representaciones XML. 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.

```
public class Libro {private String nombre;
private String autor;
private String editorial;
private String isbn;
public Libro(String nombre, String autor, String
editorial,
         String isbn) {
        super();
        this.nombre = nombre;
        this.autor = autor;
        this.editorial = editorial;
        this.isbn = isbn;
}
public Libro() {}
public String getNombre() { return nombre; }
public String getAutor() { return autor;  }
public String getEditorial() {return editorial; }
public String getIsbn() { return isbn;}
public void setNombre(String nombre)
    { this.nombre = nombre; }
public void setAutor(String autor)
    { this.autor = autor; }
public void setEditorial(String editorial)
    { this.editorial = editorial;  }
public void setIsbn(String isbn)
    {  this.isbn = isbn;  }
}
```

Creo un libro

```
Libro milibro = new Libro("Entornos de
Desarrollo","Alicia Ramos", "Garceta",
"978-84-1545-297-3" );
```

Y generaré este documento.

```
<?xml version="1.0" encoding="UTF-8"
standalone="yes"?>
<libro>
    <autor>Alicia Ramos</autor>
    <editorial>Garceta</editorial>
    <isbn>978-84-1545-297-3</isbn>
    <nombre>Entornos de Desarrollo</nombre>
</libro>
```

- Lo contrario, es decir presentar **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 y gets para obtener o establecer los datos de un documento XML.

### 1.9.1 Mapear clases java a representaciones XML.

Para crear objetos java en XML, vamos a utilizar ***JavaBeans***, que serán las clases que se van a mapear. Son clases primitivas java (POJOS) 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. Si una clase va a ser la raíz del documento se añadirá esta anotación, el *namespace* es opcional.

```
@XmlRootElement
   public class ClaseRaiz {
...
}
```

- **@XmlType**(propOrder = { "field2", "field1",.. }): Permite definir en qué orden se van escribir los elementos (o las etiquetas) dentro del XML.

  Si es una clase **que no va a ser raíz** añadiremos @**XmlType.**

  Por ejemplo, esta es una clase Producto, en la que indico que es un tipo, no es clase raíz, e indico el orden de las etiquetas con propOrder, los nombres tienen que coincidir con los nombres de los atributos:

```
. . . . . .
import javax.xml.bind.annotation.XmlType;

@XmlType(propOrder = {"codigo", "nombre", "existencias", "precio",
"unidadesvendidas","importe", "estado"})
public class Producto {
     private int codigo;
     private String nombre;
     private int existencias;
     private int unidadesvendidas;
     private double precio;
     private double importe;
     private String estado;


     public Productoxml() {
     }
. . . . . .
```

- @**XmlElement**(name = "nombre"): Define el elemento de XML que se va 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***. Sería algo como esto

```
@XmlRootElement(name="Un_Nombre_para_la_raiz")

@XmlType(name="Otro_Nombre")
```

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):

```
@XmlRootElement(name="La_ClaseRaiz")
public class UnaClase {
   private String unAtributo;
```

```
    @XmlElement(name="El_Atributo")
    String getUnAtributo() {
        return this.unAtributo;
    }
}
```

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

```
@XmlRootElement(name="La_ClaseRaiz")
public class UnaClase {
    private String [] unArray;

    @XmlElementWrapper
    @XmlElement(name="Elemento_Array")
    String [] getUnArray() {
        return this.unArray;
    }
}
```

Si el atributo es otra clase (otro java bean), le ponemos igualmente @**XmlElement** al método get, pero la clase que hace de atributo debería llevar a la vez sus anotaciones correspondientes.

**Ejemplo1:** se desea generar el siguiente documento XML:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<libro>
    <autor>Alicia Ramos</autor>
    <editorial>Garceta</editorial>
    <isbn>978-84-1545-297-3</isbn>
    <nombre>Entornos de Desarrollo</nombre>
</libro>
```

Se trata de presentar un documento XML con la información de un único libro

Necesitamos crear la clase libro con la anotación de elemento raíz **@XmlRootElement()**, pues sólo presentamos un libro:

```
import javax.xml.bind.annotation.XmlRootElement;

@XmlRootElement()
public class Libro {private String nombre;
private String autor;
private String editorial;
private String isbn;
public Libro(String nombre, String autor, String editorial,
        String isbn) {
     super();
     this.nombre = nombre;
     this.autor = autor;
     this.editorial = editorial;
     this.isbn = isbn;
```

```
}
public Libro() {}
public String getNombre() { return nombre; }
public String getAutor() { return autor;  }
public String getEditorial() {return editorial; }
public String getIsbn() { return isbn;}
public void setNombre(String nombre) { this.nombre = nombre; }
public void setAutor(String autor) { this.autor = autor; }
public void setEditorial(String editorial)
    { this.editorial = editorial;  }
public void setIsbn(String isbn) {  this.isbn = isbn;  }
  }
```

Una vez que tenemos la clase definida, lo siguiente es ver el **código java para mapear los objetos** que definamos de esta clase.

Utilizando la anotación ***@XmlRootElement***. El código java para conseguir el fichero XML es el siguiente:

- Instanciamos el contexto, indicando la clase que será el ***RootElement***, en nuestro ejemplo es la clase **Libro**:

```
JAXBContext context = JAXBContext.newInstance(Libro.class);
```

- Creamos un ***Marshaller***, que es la clase capaz de convertir nuestro *Java Bean*, en una cadena XML:

```
Marshaller m = context.createMarshaller();
```

- Indicamos que vamos a querer el XML con un formato amigable (saltos de línea, sangrado, etc)

```
m.setProperty(Marshaller.JAXB_FORMATTED_OUTPUT, Boolean.TRUE);
```

- Hacemos la conversión llamando al método ***marshal***, pasando una instancia del *Java Bean* que queramos convertir a XML y un OutpuStream donde queramos que salga el XML, por ejemplo la salida estándar, o también podría ser un fichero o cualquier otro Stream:

```
m.marshal(milibro, System.out); //por pantalla
m.marshal(milibro, new File(MIARCHIVO_XML)); //salida en archivo
```

Esta sería nuestra clase principal para crear el xml:

```
import java.io.File;
import javax.xml.bind.JAXBContext;
import javax.xml.bind.JAXBException;
import javax.xml.bind.Marshaller;

public class Principal {

   private static final String MIARCHIVO_XML = "./unlibro.xml";

   public static void main(String[] args) {
         Libro milibro = new Libro("Entornos de Desarrollo",
                "Alicia Ramos","Garceta","978-84-1545-297-3" );
         try {
               JAXBContext context = JAXBContext.newInstance(Libro.class);
               Marshaller m = context.createMarshaller();
               m.setProperty(Marshaller.JAXB_FORMATTED_OUTPUT,
Boolean.TRUE);
               m.marshal(milibro, System.out);
               m.marshal(milibro, new File(MIARCHIVO_XML));
         } catch (JAXBException e) {
               // TODO Auto-generated catch block
               e.printStackTrace();
         }
   }

  }
```

**Prueba a añadir las anotaciones en la clase Libro para obtener la siguiente salida:**

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<milibrito>
    <autorcito>Alicia Ramos</autorcito>
    <mieditorial>Garceta</mieditorial>
    <miisbn>978-84-1545-297-3</miisbn>
    <nombrecito>Entornos de Desarrollo</nombrecito>
</milibrito>
```

Si no se pone orden en las etiquetas, estas se visualizan por orden alfabético.

**Prueba a añadir las anotaciones en la clase Libro para que el orden de las etiquetas sea este:**

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<milibrito>
    <nombre>Entornos de Desarrollo</nombre>
    <editorial>Garceta</editorial>
    <isbn>978-84-1545-297-3</isbn>
    <autor>Alicia Ramos</autor>
</milibrito>
```

**Ejemplo2:** se desea generar el siguiente documento XML (este tiene dos libros, es decir varias etiquetas Libro, que se van a incluir dentro de una etiqueta ListaLibro):

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<libreria>
    <ListaLibro>
        <Libro>
            <autor>Alicia Ramos</autor>
            <nombre>Entornos de Desarrollo</nombre>
            <editorial>Garceta</editorial>
            <isbn>978-84-1545-297-3</isbn>
        </Libro>
        <Libro>
            <autor>María Jesús Ramos</autor>
            <nombre>Acceso a Datos</nombre>
            <editorial>Garceta</editorial>
            <isbn>978-84-1545-228-7</isbn>
        </Libro>
    </ListaLibro>
    <lugar>Talavera, como no</lugar>
    <nombre>Prueba de libreria JAXB</nombre>
</libreria>
```

Se trata de representar los libros de una librería. Crearemos las siguientes clases:

- La clase *Libreria*, con la lista de libros, el lugar y el nombre de la librería.
- La clase *Libro*, con los datos del autor, el nombre, la editorial y el ISBN.

En la clase *Libro*, vamos a indicar la anotación ***@XmlType*** pues es una clase que no es raíz, y además indicamos el orden de las etiquetas con ***propOrder***, es decir cómo se desea que salgan en el documento XML. La clase tendrá la siguiente descripción:

```
package clasesjaxb;

import javax.xml.bind.annotation.XmlType;

@XmlType(propOrder = {"autor", "nombre", "editorial", "isbn"})
public class Libro {
    private String nombre;
    private String autor;
    private String editorial;
    private String isbn;
    public Libro(String nombre, String autor, String editorial,
             String isbn) {
          super();
          this.nombre = nombre;
          this.autor = autor;
          this.editorial = editorial;
          this.isbn = isbn;
     }
    public Libro() {}
    public String getNombre() { return nombre; }
```

```
    public String getAutor() { return autor;  }
    public String getEditorial() {return editorial; }
    public String getIsbn() { return isbn;}
    public void setNombre(String nombre) { this.nombre = nombre; }
    public void setAutor(String autor) { this.autor = autor; }
    public void setEditorial(String editorial)
        { this.editorial = editorial;  }
    public void setIsbn(String isbn) {  this.isbn = isbn;  }
}
```

En la clase *Libreria*, vamos a indicar la anotación ***@XmlRootElement*** pues es una clase raíz. También tenemos que indicar que hay un atributo qué es una colección, con lo que hay que añadir con las anotaciones ***@XmlElementWrapper*** y ***@XmlElement*** , en el método get. En estas anotaciones indicamos como se van a llamar las etiquetas dentro del documento generado. La clase tendrá la siguiente descripción:

```
package clasesjaxb;
import java.util.ArrayList;
import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlElementWrapper;
import javax.xml.bind.annotation.XmlRootElement;

@XmlRootElement()
public class Libreria {
    private ArrayList<Libro> listaLibro;
    private String nombre;
    private String lugar;

    public Libreria(ArrayList<Libro> listaLibro, String nombre,
        String lugar) {
          super();
          this.listaLibro = listaLibro;
          this.nombre = nombre;
          this.lugar = lugar; }
    public Libreria(){}
    public void setNombre(String nombre) { this.nombre = nombre; }
    public void setLugar(String lugar) { this.lugar = lugar;    }
    public String getNombre() {return nombre;    }
    public String getLugar() { return lugar;    }

    //Wrapper, envoltura alrededor la representación XML
    @XmlElementWrapper(name = "ListaLibro")
    @XmlElement(name = "Libro")
    public ArrayList<Libro> getListaLibro() {
        return listaLibro;    }

    public void setListaLibro(ArrayList<Libro> listaLibro) {
        this.listaLibro = listaLibro;    }
 }
```

Una vez que tenemos las clases ya definidas, lo siguiente es ver el **código java para mapear los objetos** que definamos de esas clases.

Utilizando la anotación ***@XmlRootElement***. El código java para conseguir el fichero XML es el siguiente:

- Instanciamos el contexto, indicando la clase que será el ***RootElement***, en nuestro ejemplo es la clase Libreria:

```
JAXBContext jaxbContext = JAXBContext.newInstance(Libreria.class);
```

- Creamos un ***Marshaller***, que es la clase capaz de convertir nuestro *Java Bean*, en una cadena XML:

```
Marshaller jaxbMarshaller = jaxbContext.createMarshaller();
```

- Indicamos que vamos a querer el XML con un formato amigable (saltos de línea, sangrado, etc)

```
jaxbMarshaller.setProperty(Marshaller.JAXB_FORMATTED_OUTPUT, true);
```

- Hacemos la conversión llamando al método ***marshal***, pasando una instancia del *Java Bean* que queramos convertir a XML y un OutpuStream donde queramos que salga el XML, por ejemplo la salida estándar, o también podría ser un fichero o cualquier otro Stream:

```
jaxbMarshaller.marshal(unaInstanciaDeUnaClase, System.out);
//Si ponemos un fichero
jaxbMarshaller.marshal(unaInstanciaDeUnaClase, new
                    File("./mifichero.xml"));
```

**Ahora en el ejemplo, vamos a crear objetos de las clases y vamos a ver como generar el XML:**

```
package clasesjaxb;
import java.io.File;
import java.io.FileReader;
import java.io.IOException;
import java.util.ArrayList;
import javax.xml.bind.JAXBContext;
import javax.xml.bind.JAXBException;
import javax.xml.bind.Marshaller;
import javax.xml.bind.Unmarshaller;

public class Ejemplo1_JAXB {
    private static final String MIARCHIVO_XML = "./libreria.xml";
    public static void main(String[] args)
       throws JAXBException, IOException {
        //Se crea la lista de libros
        ArrayList<Libro> libroLista = new ArrayList<Libro>();
        // Creamos dos libros y los añadimos
        Libro libro1 = new Libro("Entornos de Desarrollo",
                 "Alicia Ramos","Garceta","978-84-1545-297-3" );
        libroLista.add(libro1);
        Libro libro2 = new Libro("Acceso a Datos","Maria Jesús
Ramos",
                 "Garceta","978-84-1545-228-7" );
        libroLista.add(libro2);
```

```
        // Se crea La libreria y se le asigna la lista de libros
        Libreria milibreria = new Libreria();
        milibreria.setNombre("Prueba de libreria JAXB");
        milibreria.setLugar("Talavera, como no");
        milibreria.setListaLibro(libroLista);

        // Creamos el contexto indicando la clase raíz
        JAXBContext context =
JAXBContext.newInstance(Libreria.class);
        //Creamos el Marshaller, convierte el java bean en una cadena
XML
        Marshaller m = context.createMarshaller();
       //Formateamos el xml para que quede bien
        m.setProperty(Marshaller.JAXB_FORMATTED_OUTPUT,
Boolean.TRUE);
        // Lo visualizamos con system out
        m.marshal(milibreria, System.out);
         // Escribimos en el archivo
        m.marshal(milibreria, new File(MIARCHIVO_XML));
  }
}
```

```
//Salida con el envolvente
@XmlElementWrapper(name = "ListaLibro")

<?xml version="1.0" encoding="UTF-8"
standalone="yes"?>
<libreria>
    <ListaLibro>
        <Libro>
            <autor>Alicia Ramos</autor>
            <nombre>Entornos de Desarrollo</nombre>
            <editorial>Garceta</editorial>
            <isbn>978-84-1545-297-3</isbn>
        </Libro>
        <Libro>
            <autor>Maria JesÃºs Ramos</autor>
            <nombre>Acceso a Datos</nombre>
            <editorial>Garceta</editorial>
            <isbn>978-84-1545-228-7</isbn>
        </Libro>
    </ListaLibro>
    <lugar>Talavera, como no</lugar>
    <nombre>Prueba de libreria JAXB</nombre>
</libreria>
```

```
//Salida si no ponemos el envolvente
@XmlElementWrapper(name = "ListaLibro")

<?xml version="1.0" encoding="UTF-8"
standalone="yes"?>
<libreria>
    <Libro>
        <autor>Alicia Ramos</autor>
        <nombre>Entornos de Desarrollo</nombre>
        <editorial>Garceta</editorial>
        <isbn>978-84-1545-297-3</isbn>
    </Libro>
    <Libro>
        <autor>Maria JesÃºs Ramos</autor>
        <nombre>Acceso a Datos</nombre>
        <editorial>Garceta</editorial>
        <isbn>978-84-1545-228-7</isbn>
    </Libro>
    <lugar>Talavera, como no</lugar>
    <nombre>Prueba de libreria JAXB</nombre>
</libreria>
```

# LEER DATOS DE UN DOCUMENTO XML

## Si ahora deseamos hacer lo contrario, es decir, leer los datos del documento XML y convertirlos a objetos java, utilizaremos las siguientes órdenes:

- Instanciamos el contexto, indicando la clase que será el RootElement, en nuestro ejemplo *Libreria*:

```
JAXBContext context = JAXBContext.newInstance(Libreria.class);
```

- Se crea ***Unmarshaller*** en el cotexto de la clase Libreria:

```
Unmarshaller unmars = context.createUnmarshaller();
```

- Utilizamos el método ***unmarshal***, para obtener datos de un ***Reader*** (un file):

```
UnaClase objeto = (UnaClase) unmars.unmarshal(new FileReader("mifichero.xml"));
```

- Recuperamos un atributo del objeto:

```
System.out.println(objeto.getUnAtributo());
```

- Recuperamos el array list, si lo tiene y visualizamos:

```
ArrayList<ClaseDelArray> lista = objeto.getListadeobjetos();
for (ClaseDelArray obarray : lista) {
    System.out.println("Atributo array: " + obarray.getAtributo());
```

En nuestro ejercicio el código para visualizar el contenido del fichero XML es el siguiente:

```
    // Visualizamos ahora los datos del documento XML creado
     System.out.println("------------ Leo el XML ---------");
     //Se crea Unmarshaller en el cotexto de la clase Libreria
     Unmarshaller unmars = context.createUnmarshaller();

     //Utilizamos el método unmarshal, para obtener datos de un Reader
     Libreria libreria2 =(Libreria)
          unmars.unmarshal(new FileReader(MIARCHIVO_XML));

     //Recuperamos los datos y visualizamos
     System.out.println("Nombre de libreria: "+ libreria2.getNombre());
     System.out.println("Lugar de la libreria: "+
        libreria2.getLugar());
     System.out.println("Libros de la librería: ");

     ArrayList<Libro> lista = libreria2.getListaLibro();
     for (Libro libro : lista) {
            System.out.println("\tTítulo del libro: "
              + libro.getNombre()
              + " , autora: " + libro.getAutor());
      }
```

**ACTIVIDAD 1.7.**

Realiza cambios en las clases anteriores y añade las clases que se necesitan, para generar un documento XML que agrupe a varias librerías con varios libros. Haz el programa java que utilice esas clases, cree dos objetos librerías, una con dos libros, y otra con tres libros y genere un documento con nombre ***Librerias.xml*** con esta estructura:

```
<MISLIBRERIAS>
 <Libreria>
  <nombre>xxxxx</nombre>
  <lugar>xxxxx</lugar>
  <MiListaLibros>
    <Libro>
      <nombre>xxxxx</nombre>
      <autor>xxxxx</autor>
      <editorial>xxx</editorial>
      <isbn>xxxx</isbn>
    </Libro>
    <Libro>
            .........
            .........
    </Libro>
  </MiListaLibros>
 </Libreria>
 <Libreria>
   <nombre>xxxxx</nombre>
   <lugar>xxxxx</lugar>
   <MiListaLibros>
     <Libro>
       <nombre>xxxxx</nombre>
       <autor>xxxxx</autor>
       <editorial>xxxx</editorial>
       <isbn>xxxx</isbn>
      </Libro>
      <Libro>
         .........
         .........
       </Libro>
   </MiListaLibros>
 </Libreria>
</MISLIBRERIAS>
```

## LEER XML

CREAR UN PROGRAMA JAVA PARA LEER EL DOCUMENTO NuevosDep.xml**,** que está en la carpeta de recursos, y visualizar todos sus datos.

Ídem con **ventasarticulos.xml**