```
<?xml encoding="UTF-8"?>

<!ELEMENT nota (remitent, destinatari, titol, missatge?)>
<!ATTLIST nota importancia (alta|normal) normal)>
<!ELEMENT remitent (#PCDATA)>
<!ELEMENT destinatari (#PCDATA)>
<!ELEMENT titol (#PCDATA)>
<!ELEMENT missatge (#PCDATA)>
```

Accés a dades.

Java Api for XML Binding.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
      <xsd:element name="Libro">
          <xsd:complexType>
              <xsd:sequence>
                  <xsd:element name="Titulo" type="xsd:string"/>
                  <xsd:element name="Autores" type="xsd:string" maxOccurs="10"/>
                  <xsd:element name="Editorial" type="xsd:string"/>
              </xsd:sequence>
              <xsd:attribute name="precio" type="xsd:double"/>
          </xsd:complexType>
      </xsd:element>
</xsd:schema>
```

Binding

El **Binding** és una tècnica que consisteix a vincular classes Java amb formats específics d'emmagatzematge de manera automatitzada. Sovint un conjunt idèntic de classes pot donar lloc a múltiples esquemes XML. Això significa que normalment l'automatització de l'escriptura de dades en format XML, que en termes tècnics s'anomena *marshalling*, requereix de certa ajuda per tal de decidir quin format s'automatitzarà d'entre els molts possibles. D'això se'n diu *mapar*, perquè ve a ser com si configuréssim un mapa on s'indiquen els vincles entre les classes i els seus atributs, i els elements i atributs XML.

En Java existeixen diverses biblioteques per gestionar el binding, com per exemple JAXB, JiBX, XMLBinding, etc. Des de la versió 6.0 s'ha incorporat en el JDK estàndard JAXB i a partir de la versió 9 s'ha eliminat..

Incloure JAXB al projecte

S'han d'incloure els següents jar al classpath del projecte:

- FastInfoset-1.2.16.jar
- istack-commons-runtime-3.0.8.jar
- jakarta.activation-api-1.2.1.jar
- jakarta.xml.-bind-api-2.3.2.jar
- jaxb-runtime-2.3.2.jar
- stax-ex-1.8.1.jar
- txw2-2.3.2.jar

Una altra manera és utilitzant el gestor de dependències *maven* i incorporar l'artefacte:

- jaxb-runtime de *org.glassfish.jaxb*

Anotacions

JAXB utilitza Anotacions per aconseguir la informació extra necessària per mapar el binding. Serveixen per associar informació i funcionalitat als objectes sense interferir en l'estructura del model de dades.

@XmlRootElement

Decora l'element arrel del document. En la majoria dels casos es sol posar a sobre de la declaració de la classe, de manera que l'element arrel prendrà el nom de la classe.

@XmlAccessorType(tipus)

Opcional. Determina com es decidiran els elements que formaran part de l'xml.

- XmlAccessType.PUBLIC_MEMBER és el tipus per defecte. Agafarà com a elements els atributs públics, els atributs anotats expressament i els getters
- XmlAccessType.FIELD Agafarà com a elements els atributs i els getters anotats expressament
- XmlAccessType.PROPERTY Agafarà com a elements els getters i els atributs anotats expressament
- XmlAccessType.None Només inclourà a l'XML els atributs i els getters anotats expressament.

@XmlType(propOrder={"prop1", ... , "propN"})

Opcional. Permet definir l'ordre en el que apareixeran els camps a l'xml generat.

@XmlElement

Indica que l’atribut o getter que decora és un element xml. No fa falta a no ser que s’hagi d’especificar algun atribut de l'anotació:

- *name* Opcional. Permet canviar el nom de l’element XML. Per defecte és el mateix que el de l'atribut java.
- *nillable* Opcional. Si és true, els atributs java amb valor null apareixen com un element xml buid. Si és fals o no ho posam, no apareixeran com a elements xml.

@XmlAttribute

L'atribut java es mapeja a un atribut xml en lloc de a un element.

- *name* Opcional. permet canviar el nom de l'atribut xml, per defecte és el mateix que java.

@XmlTransient

Indica que l'element que decora no s'ha d'incloure al document xml

@XmlValue

Quan volem generar un element xml sense elements anidats, només amb un valor i atributs. Restriccions: No hi pot haver res anotat amb @XmlElement i l'atribut o getter anotat amb @XmlValue ha de tornar un valor simple.

<valueExample name="name">A very large value</valueExample>

@XmlElementWrapper

Quan un atribut java és una llista (ArrayList, HashSet, ...) es genera un element xml per a cada membre de la llista, directament a l'XML. Si volem que aquests elements estiguin tancats dins d'un element superior posam aquesta anotació. Necessita l'atribut *name* per definir el nom de l'element xml que crearà.

Context JAXB

La classe JAXBContext és el punt d'entrada a l'entorn JAXB. S'ha de configurar donant-li el nom de la classe arrel de l'xml que volguem tractar. A partir d'aquesta classe obtindrem els objectes necessaris per poder fer la conversió de java a xml i de xml a java.

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

O bé

```
JAXBContext context = JAXBContext.newInstance(objecte.getClass());
```

Unmarshalling

Obtenir objectes java a partir de la informació del document XML

Aquest procés es coneix com a *unmarshalling*. Hem d'obtenir una instància de la classe *Unmarshaller* a partir del context.

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

objectRoot = unmarshaller.unmarshal(fitxer);
```

El mètode *unmarshall* s'encarregarà de fer la transformació de l'xml a objectes java i ens tornarà l'objecte arrel que contendrà tots els altres objectes creats.

El mètode està sobrecarregat i li podem passar com a paràmetre, entre d'altres opcions, un objecte File, URL, InputStream, Reader, ...

Marshalling

Obtenir el document XML a partir d'objectes Java

Aquest procés es coneix com a *marshalling*. Hem d'obtenir una instància de la classe *Marshaller* a partir del context.

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

marshaller.marshal(objecteRoot, fitxer);
```

Per realitzar transvasament d'informació del model al document XML serà necessari l'objecte del model corresponent a l'objecte contenidor principal de tota la informació (és a dir, l'equivalent al qual serà l'arrel del document XML) i un objecte que representi l'emmagatzematge on volem l'xml, per exemple un OutputStream, Writer, File, ...

Per generar xml ben formatat amb canvis de línia i sagnat:

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