

# Lección 8 - Protocolos de mensajería

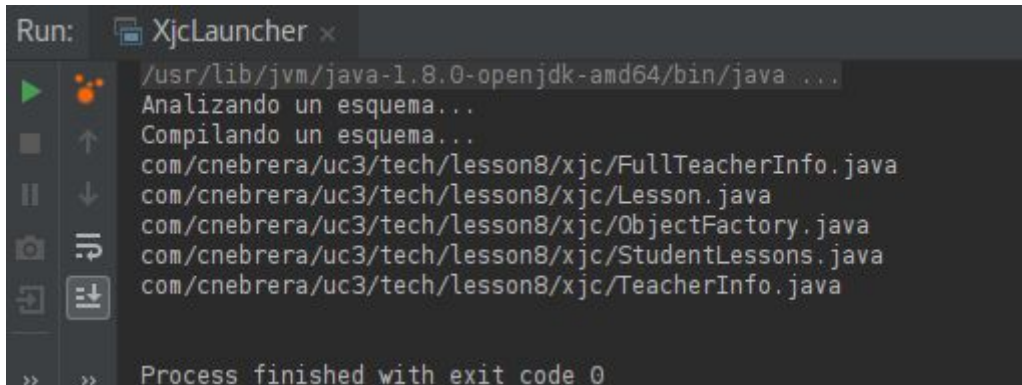
Tecnologías del sector financiero

Asier Sampietro Alberdi

# Práctica 1: Generación de código Java mediante XJC a partir de un XSD.

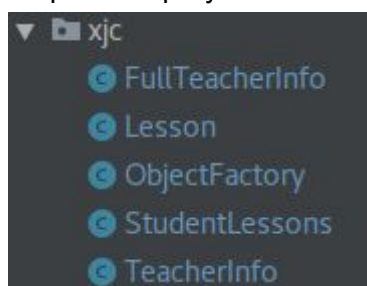
Se ha modificado la clase *XjcLauncher* para que ejecute la orden en línea de comandos del runtime. Tras esto, se han generado las clases a partir de los XSD que se disponían. El resultado de la ejecución ha sido el siguiente mensaje en consola, y el paquete de java “xjc” con su contenido que se mostrará a continuación.

Mensaje en consola:



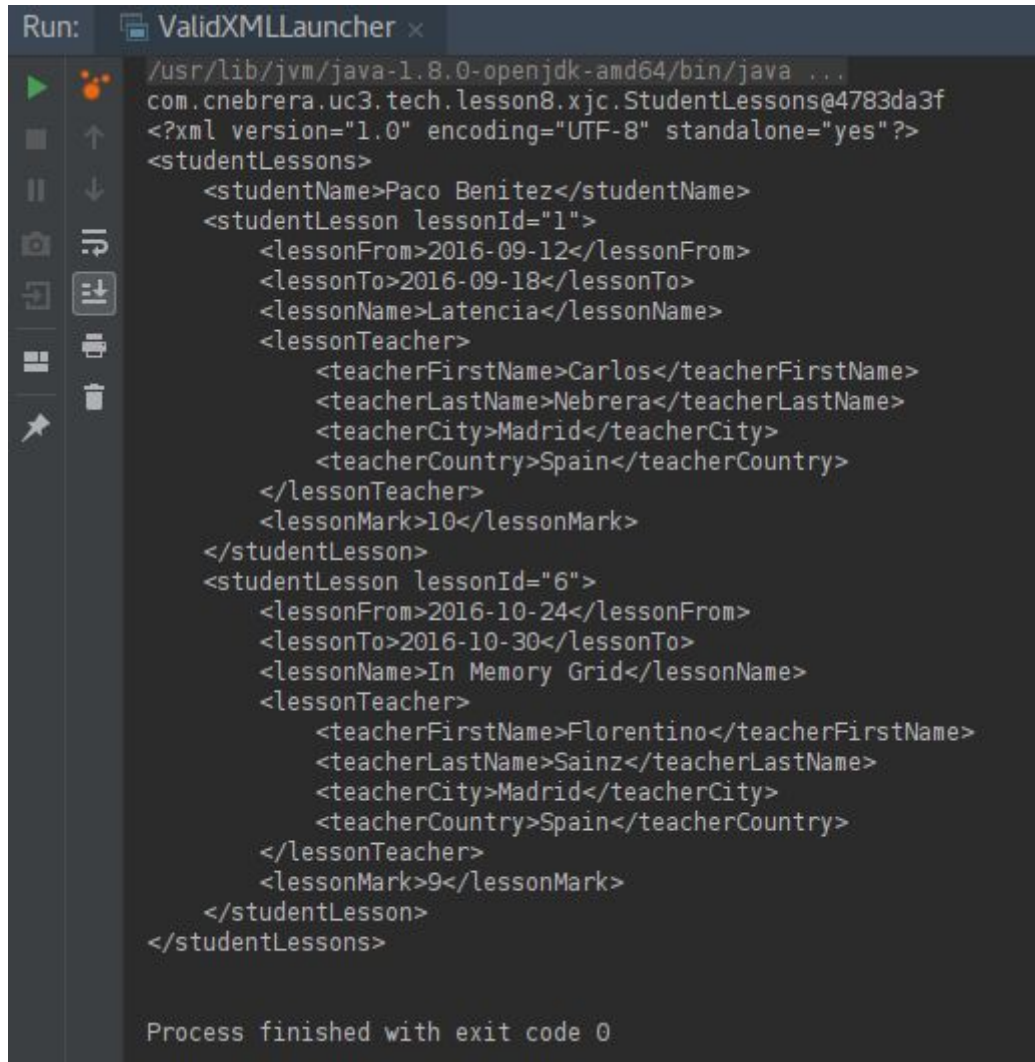
```
Run: XjcLauncher x
/usr/lib/jvm/java-1.8.0-openjdk-amd64/bin/java ...
Analizando un esquema...
Compilando un esquema...
com/cnebrera/uc3/tech/lesson8/xjc/FullTeacherInfo.java
com/cnebrera/uc3/tech/lesson8/xjc/Lesson.java
com/cnebrera/uc3/tech/lesson8/xjc/ObjectFactory.java
com/cnebrera/uc3/tech/lesson8/xjc/StudentLessons.java
com/cnebrera/uc3/tech/lesson8/xjc/TeacherInfo.java
Process finished with exit code 0
```

Paquete en proyecto:



## Práctica 2: Generación de instancias de clases a partir de XMLs y viceversa.

Añadiendo la clase al contexto de JAXB, y generando un *marshaller* y un *unmarshaller*. Usando estos, se ha conseguido generar un objeto java del XML válido, y luego pasar este objeto a un XML legible en memoria de nuevo. Las trazas de la ejecución son los siguientes:



```
Run: ValidXMLLauncher x
/usr/lib/jvm/java-1.8.0-openjdk-amd64/bin/java ...
com.cnebrera.uc3.tech.lesson8.xjc.StudentLessons@4783da3f
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<studentLessons>
  <studentName>Paco Benitez</studentName>
  <studentLesson lessonId="1">
    <lessonFrom>2016-09-12</lessonFrom>
    <lessonTo>2016-09-18</lessonTo>
    <lessonName>Latencia</lessonName>
    <lessonTeacher>
      <teacherFirstName>Carlos</teacherFirstName>
      <teacherLastName>Nebrera</teacherLastName>
      <teacherCity>Madrid</teacherCity>
      <teacherCountry>Spain</teacherCountry>
    </lessonTeacher>
    <lessonMark>10</lessonMark>
  </studentLesson>
  <studentLesson lessonId="6">
    <lessonFrom>2016-10-24</lessonFrom>
    <lessonTo>2016-10-30</lessonTo>
    <lessonName>In Memory Grid</lessonName>
    <lessonTeacher>
      <teacherFirstName>Florentino</teacherFirstName>
      <teacherLastName>Sainz</teacherLastName>
      <teacherCity>Madrid</teacherCity>
      <teacherCountry>Spain</teacherCountry>
    </lessonTeacher>
    <lessonMark>9</lessonMark>
  </studentLesson>
</studentLessons>

Process finished with exit code 0
```

Con el XML no válido no ha podido ser posible ejecutarlo. Como *unmarshaller* comprueba el documento que se está intentando convertir el objeto, comprueba que este esté bien formado. Por eso muestra la siguiente excepción al intentar ejecutarlo:

```
Run: InvalidXMLLauncher x
/usr/lib/jvm/java-1.8.0-openjdk-amd64/bin/java ...
Exception in thread "main" javax.xml.bind.UnmarshalException
- with linked exception:
[org.xml.sax.SAXParseException; systemId: file:/media/sampru/Datuak/eskola/tec-sec-fin/MasterUC3Practices/Lesson8/src/main/resources/invalid.xml;
at javax.xml.bind.helpers.AbstractUnmarshallerImpl.createUnmarshalException(AbstractUnmarshallerImpl.java:335)
at com.sun.xml.internal.bind.v2.runtime.unmarshaller.UnmarshallerImpl.createUnmarshalException(UnmarshallerImpl.java:563)
at com.sun.xml.internal.bind.v2.runtime.unmarshaller.UnmarshallerImpl.unmarshal0(UnmarshallerImpl.java:249)
at com.sun.xml.internal.bind.v2.runtime.unmarshaller.UnmarshallerImpl.unmarshal(UnmarshallerImpl.java:214)
at javax.xml.bind.helpers.AbstractUnmarshallerImpl.unmarshal(AbstractUnmarshallerImpl.java:157)
at javax.xml.bind.helpers.AbstractUnmarshallerImpl.unmarshal(AbstractUnmarshallerImpl.java:162)
at javax.xml.bind.helpers.AbstractUnmarshallerImpl.unmarshal(AbstractUnmarshallerImpl.java:171)
at javax.xml.bind.helpers.AbstractUnmarshallerImpl.unmarshal(AbstractUnmarshallerImpl.java:189)
at com.cnebrera.uc3.tech.lesson8.jaxb.JAXBHandler.convertToObject(JAXBHandler.java:61)
at com.cnebrera.uc3.tech.lesson8.AbstractXMLLauncher.handleXML(AbstractXMLLauncher.java:32)
at com.cnebrera.uc3.tech.lesson8.InvalidXMLLauncher.main(InvalidXMLLauncher.java:25)
Caused by: org.xml.sax.SAXParseException; systemId: file:/media/sampru/Datuak/eskola/tec-sec-fin/MasterUC3Practices/Lesson8/src/main/resources/inv
at com.sun.org.apache.xerces.internal.util.ErrorHandlerWrapper.createSAXParseException(ErrorHandlerWrapper.java:203)
at com.sun.org.apache.xerces.internal.util.ErrorHandlerWrapper.error(ErrorHandlerWrapper.java:134)
at com.sun.org.apache.xerces.internal.impl.XMLErrorReporter.reportError(XMLErrorReporter.java:396)
at com.sun.org.apache.xerces.internal.impl.XMLErrorReporter.reportError(XMLErrorReporter.java:327)
at com.sun.org.apache.xerces.internal.impl.XMLErrorReporter.reportError(XMLErrorReporter.java:284)
at com.sun.org.apache.xerces.internal.impl.xs.XMLSchemaValidator$XSIErrorReporter.reportError(XMLSchemaValidator.java:453)
at com.sun.org.apache.xerces.internal.impl.xs.XMLSchemaValidator.reportSchemaError(XMLSchemaValidator.java:3231)
at com.sun.org.apache.xerces.internal.impl.xs.XMLSchemaValidator.handleStartElement(XMLSchemaValidator.java:1791)
at com.sun.org.apache.xerces.internal.impl.xs.XMLSchemaValidator.startElement(XMLSchemaValidator.java:741)
at com.sun.org.apache.xerces.internal.jaxp.validation.ValidatorHandlerImpl.startElement(ValidatorHandlerImpl.java:568)
at com.sun.xml.internal.bind.v2.runtime.unmarshaller.ValidatingUnmarshaller.startElement(ValidatingUnmarshaller.java:86)
at com.sun.xml.internal.bind.v2.runtime.unmarshaller.SAXConnector.startElement(SAXConnector.java:153)
at com.sun.org.apache.xerces.internal.parsers.AbstractSAXParser.startElement(AbstractSAXParser.java:509)
at com.sun.org.apache.xerces.internal.impl.XMLNSDocumentScannerImpl.scanStartElement(XMLNSDocumentScannerImpl.java:374)
at com.sun.org.apache.xerces.internal.impl.XMLDocumentFragmentScannerImpl$FragmentContentDriver.next(XMLDocumentFragmentScannerImpl.java:2784)
```

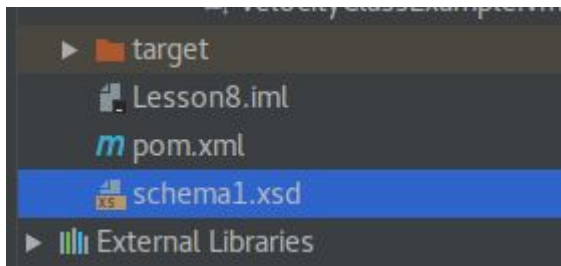
Como el error no entra en pantalla, cabe destacar que el *unmarshaller* indica cuál ha sido el causante de la excepción:

```
Se ha encontrado contenido no válido a partir del elemento
'lessonTeacher'. Se esperaba uno de '{lessonName}'.
```

## Práctica 3: Generación de XSD a partir de clases anotadas con JAXB.

De la misma manera que JAXB comprueba que el XML sea adaptable al objeto Java, también dispone de herramientas para generar un XSD que valide el XML partiendo del objeto Java. Cabe destacar que la utilidad no piensa como lo haría un humano, y se limita a replicar lo que necesita para Java, así que (en mi opinión) no se puede fiar al 100% de un XSD generado automáticamente, pudiendo haber indeterminaciones.

La ejecución no deja trazas, pero en la carpeta raíz del proyecto aparece un documento nuevo con el nombre definido en las constantes:



Como se ha mencionado antes, y al compararlo con el XSD de base, se puede observar una pequeña diferencia, que pese a no comprometer la integridad del programa, permite diferentes tipos de XML. La diferencia en cuestión es esta línea de aquí, que restringe a un mínimo de 1 el siguiente elemento:

Creado:

```
<xs:element name="studentLesson" type="lesson" maxOccurs="unbounded" />
```

Original:

```
<xs:element name="studentLesson" type="lesson" minOccurs="1" maxOccurs="unbounded" />
```

## Práctica 4: Generación de XSD a partir de clases anotadas con JAXB.

Tras escribir la plantilla que se desea generar, y con ayuda de java, se crea la clase VelocityClassExample que una vez ejecutado printa el siguiente output:

```
/usr/lib/jvm/java-1.8.0-openjdk-amd64
Velocity injected values!
[Hello, World!]

Process finished with exit code 0
```

Como la práctica constaba de dos partes, se ha almacenado el contenido de la primera parte en el fichero "VelocityClassExample.java.old". En ella se puede observar que sin especificar nada, la librería coge el formato por defecto y se lo aplica. Cuando se especifica la conducta que ha de tener a la hora de generar el código, genera ya una clase ejecutable con todo lo que necesita.