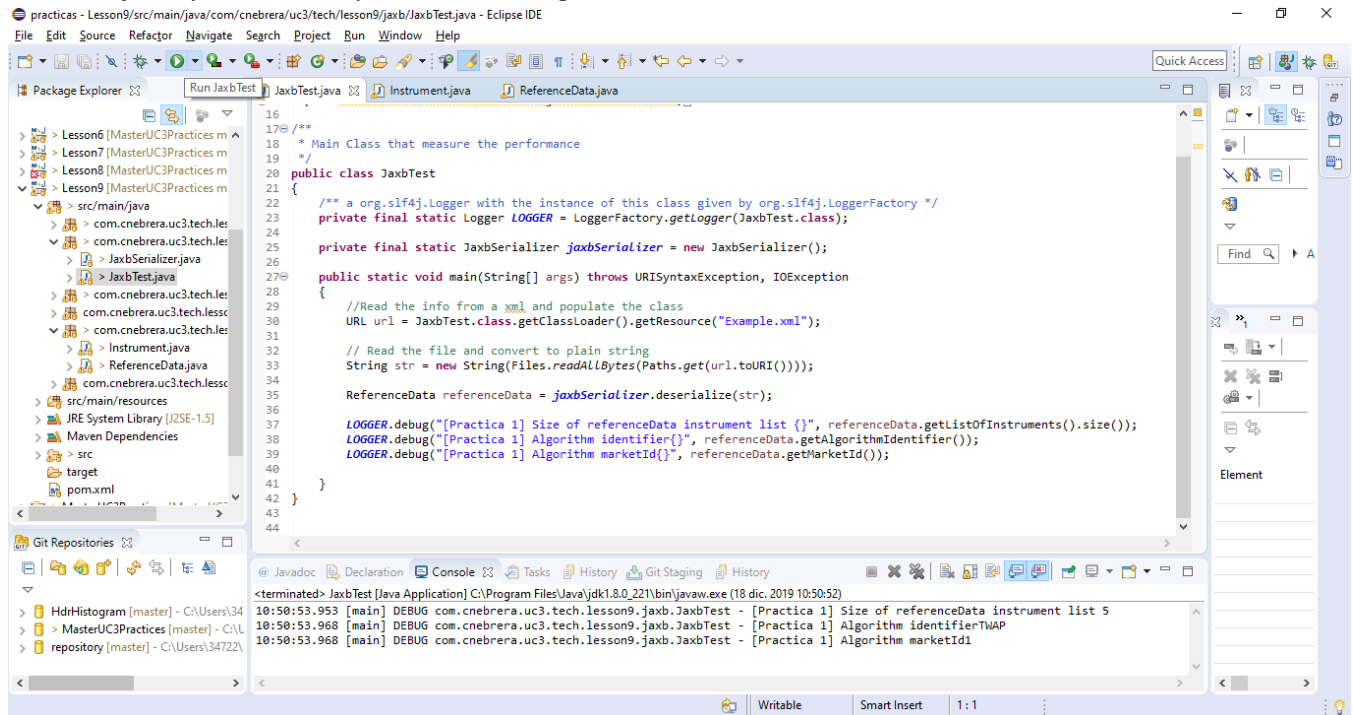


Práctica Lección 9. Serialización de mensajes. Comparación entre herramientas

Practica 1: Jaxb

El objetivo de esta primera parte de la práctica es anotar los modelos con JAXB para que sea capaz de leer el xml de ejemplo y cumpla con el esquema XSD llamado Reference.xsd.

Luego de hacer las anotaciones pertinentes en Instrument.java y ReferenceData.java, se lanza la clase JaxbTest.java, y se visualiza por consola lo siguiente:

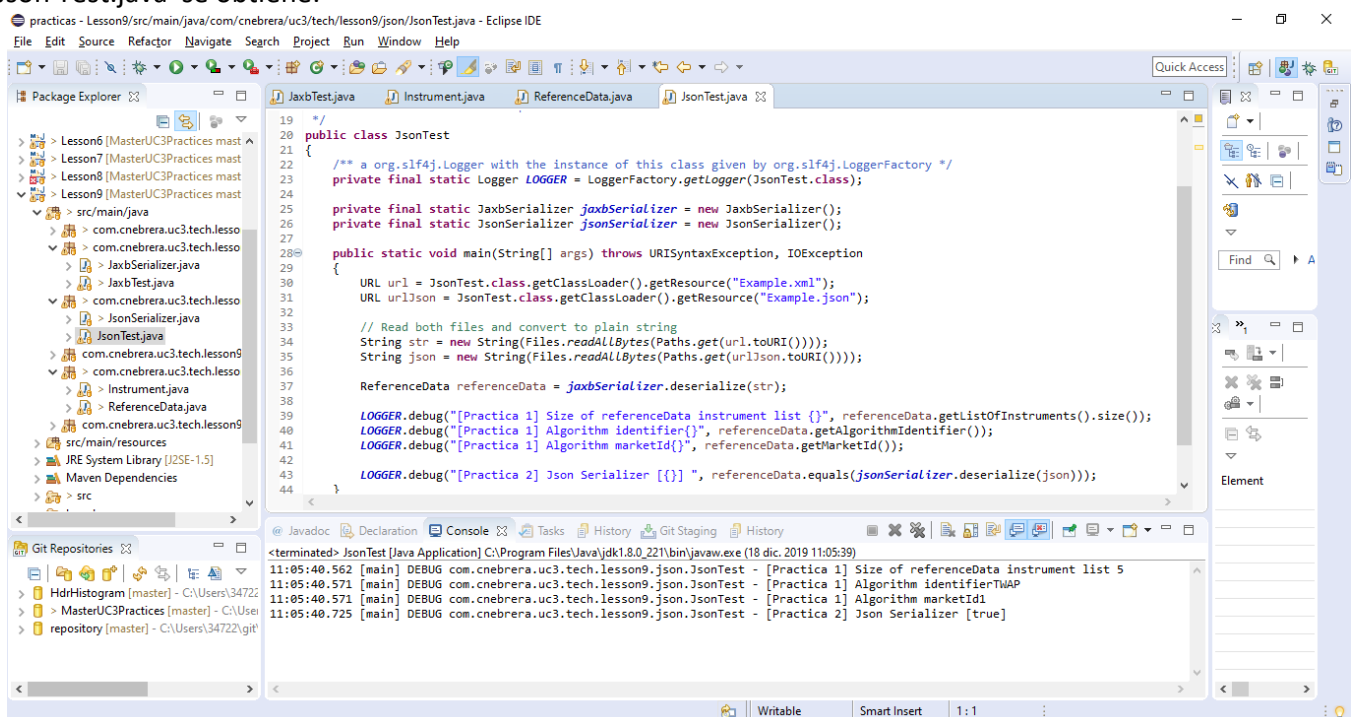


```
16
17 /**
18  * Main Class that measure the performance
19  */
20 public class JaxbTest
21 {
22     /** a org.slf4j.Logger with the instance of this class given by org.slf4j.LoggerFactory */
23     private final static Logger LOGGER = LoggerFactory.getLogger(JaxbTest.class);
24
25     private final static JaxbSerializer jaxbSerializer = new JaxbSerializer();
26
27     public static void main(String[] args) throws URISyntaxException, IOException
28     {
29         //Read the info from a xml and populate the class
30         URL url = JaxbTest.class.getClassLoader().getResource("Example.xml");
31
32         // Read the file and convert to plain string
33         String str = new String(Files.readAllBytes(Paths.get(url.toURI())));
34
35         ReferenceData referenceData = jaxbSerializer.deserialize(str);
36
37         LOGGER.debug("[Practica 1] Size of referenceData instrument list {}", referenceData.getListOfInstruments().size());
38         LOGGER.debug("[Practica 1] Algorithm identifier()", referenceData.getAlgorithmIdentifier());
39         LOGGER.debug("[Practica 1] Algorithm marketId()", referenceData.getMarketId());
40     }
41 }
42
43
44
```

```
<terminated> JaxbTest [Java Application] C:\Program Files\Java\jdk1.8.0_221\bin\javaw.exe (18 dic. 2019 10:50:52)
10:50:53.953 [main] DEBUG com.cnebrera.uc3.tech.lesson9.jaxb.JaxbTest - [Practica 1] Size of referenceData instrument list 5
10:50:53.968 [main] DEBUG com.cnebrera.uc3.tech.lesson9.jaxb.JaxbTest - [Practica 1] Algorithm identifierTWAP
10:50:53.968 [main] DEBUG com.cnebrera.uc3.tech.lesson9.jaxb.JaxbTest - [Practica 1] Algorithm marketId1
```

Práctica 2: Json

En esta parte se añadiran a las clases ReferenceData e Instrument las anotaciones Json pertinentes para que serialice a Json y deserialize de JSON permitiendo leer el fichero "Example.json" de la carpeta resources. Este fichero contiene la misma información que "Example.xml" pero en formato JSON. Luego de lanzar Json Test.java se obtiene:

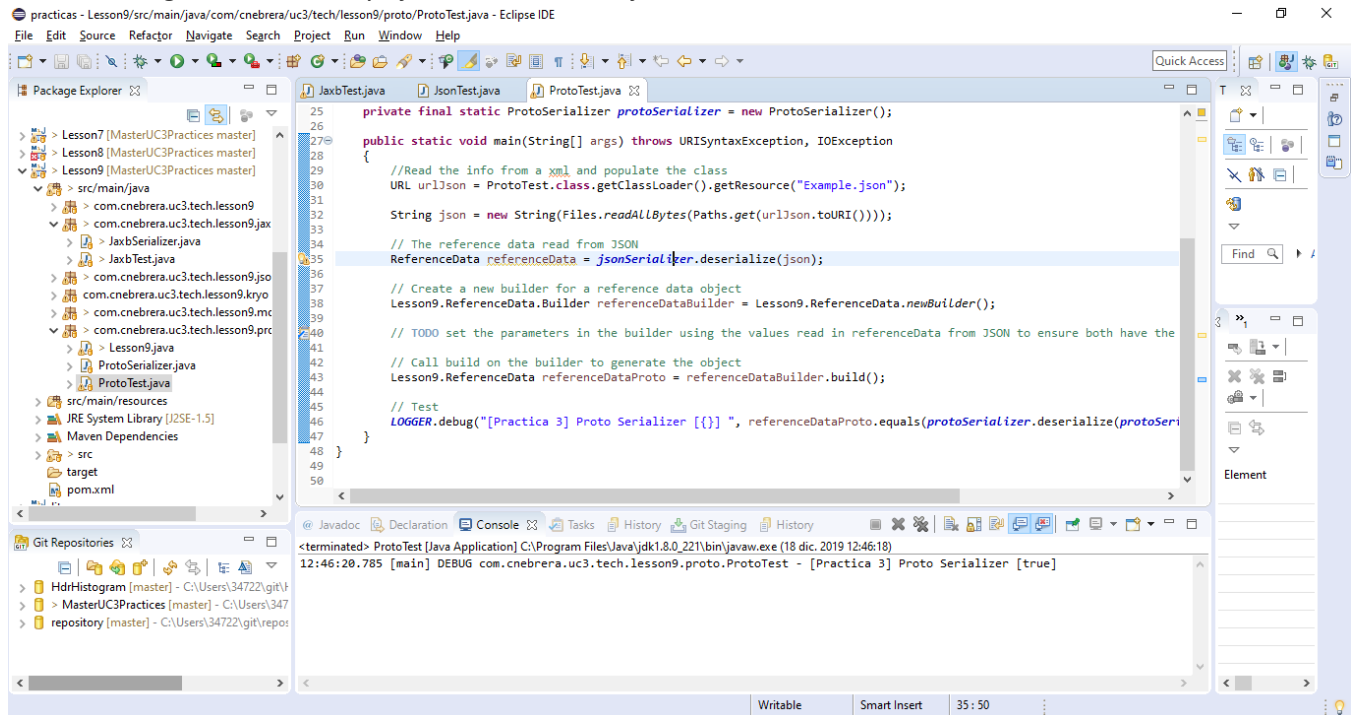


```
19
20 public class JsonTest
21 {
22     /** a org.slf4j.Logger with the instance of this class given by org.slf4j.LoggerFactory */
23     private final static Logger LOGGER = LoggerFactory.getLogger(JsonTest.class);
24
25     private final static JaxbSerializer jaxbSerializer = new JaxbSerializer();
26     private final static JsonSerializer jsonSerializer = new JsonSerializer();
27
28     public static void main(String[] args) throws URISyntaxException, IOException
29     {
30         URL url = JsonTest.class.getClassLoader().getResource("Example.xml");
31         URL urlJson = JsonTest.class.getClassLoader().getResource("Example.json");
32
33         // Read both files and convert to plain string
34         String str = new String(Files.readAllBytes(Paths.get(url.toURI())));
35         String json = new String(Files.readAllBytes(Paths.get(urlJson.toURI())));
36
37         ReferenceData referenceData = jaxbSerializer.deserialize(str);
38
39         LOGGER.debug("[Practica 1] Size of referenceData instrument list {}", referenceData.getListOfInstruments().size());
40         LOGGER.debug("[Practica 1] Algorithm identifier()", referenceData.getAlgorithmIdentifier());
41         LOGGER.debug("[Practica 1] Algorithm marketId()", referenceData.getMarketId());
42
43         LOGGER.debug("[Practica 2] Json Serializer [{}]", referenceData.equals(jsonSerializer.deserialize(json)));
44     }
45 }
```

```
<terminated> JsonTest [Java Application] C:\Program Files\Java\jdk1.8.0_221\bin\javaw.exe (18 dic. 2019 11:05:39)
11:05:40.562 [main] DEBUG com.cnebrera.uc3.tech.lesson9.json.JsonTest - [Practica 1] Size of referenceData instrument list 5
11:05:40.571 [main] DEBUG com.cnebrera.uc3.tech.lesson9.json.JsonTest - [Practica 1] Algorithm identifierTWAP
11:05:40.571 [main] DEBUG com.cnebrera.uc3.tech.lesson9.json.JsonTest - [Practica 1] Algorithm marketId1
11:05:40.725 [main] DEBUG com.cnebrera.uc3.tech.lesson9.json.JsonTest - [Practica 2] Json Serializer [true]
```

Práctica 3: Protocol Buffers

Tras generar la clase y ejecutar ProtoTest.java se obtiene:

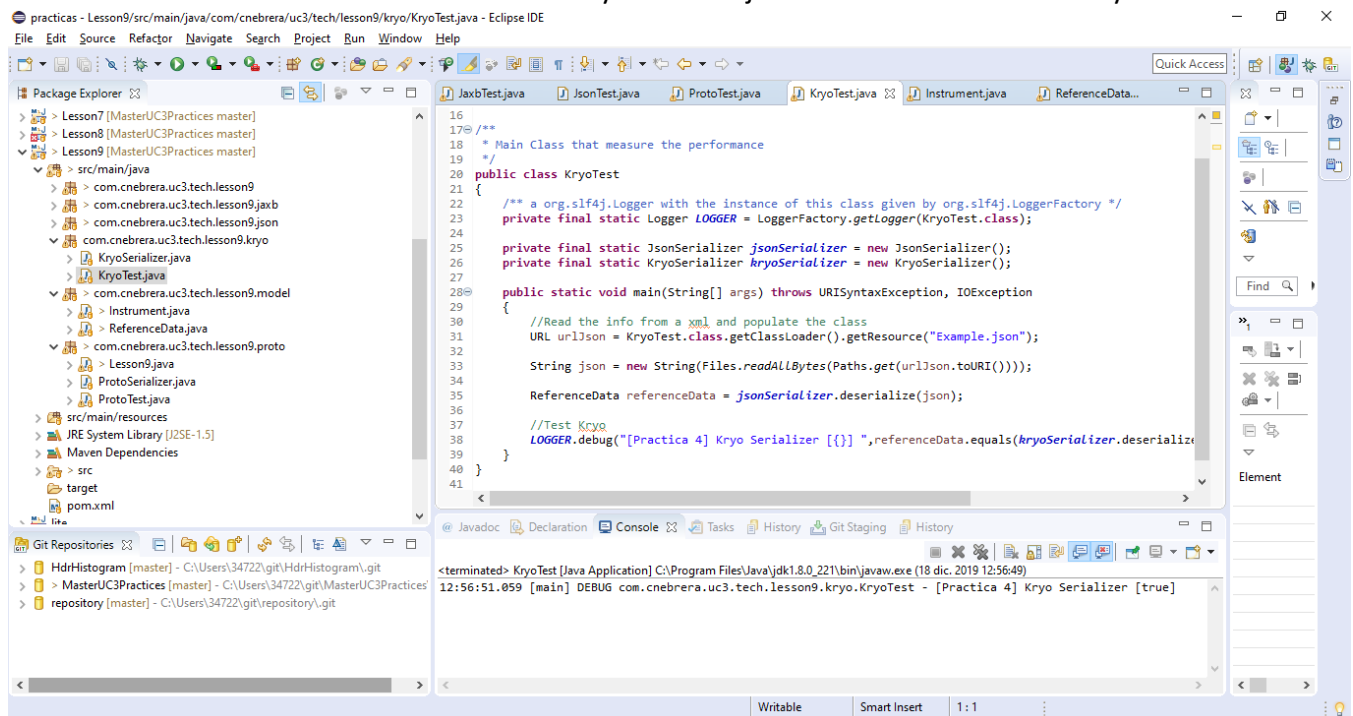


```
25 private final static ProtoSerializer protoSerializer = new ProtoSerializer();
26
27 public static void main(String[] args) throws URISyntaxException, IOException
28 {
29     //Read the info from a xml and populate the class
30     URL urlJson = ProtoTest.class.getClassLoader().getResource("Example.json");
31
32     String json = new String(Files.readAllBytes(Paths.get(urlJson.toURI())));
33
34     // The reference data read from JSON
35     ReferenceData referenceData = jsonSerializer.deserialize(json);
36
37     // Create a new builder for a reference data object
38     Lesson9.ReferenceData.Builder referenceDataBuilder = Lesson9.ReferenceData.newBuilder();
39
40     // TODO set the parameters in the builder using the values read in referenceData from JSON to ensure both have the
41
42     // Call build on the builder to generate the object
43     Lesson9.ReferenceData referenceDataProto = referenceDataBuilder.build();
44
45     // Test
46     LOGGER.debug("[Practica 3] Proto Serializer {}", referenceDataProto.equals(protoSerializer.serialize(referenceDataProto)));
47 }
48
49
50
```

<terminated> ProtoTest [Java Application] C:\Program Files\Java\jdk1.8.0_221\bin\javaw.exe (18 dic. 2019 12:46:18)
12:46:20.785 [main] DEBUG com.cnebrera.uc3.tech.lesson9.proto.ProtoTest - [Practica 3] Proto Serializer [true]

Práctica 4: Kryo.

Realizar una serialización sencilla con Kryo de un objeto ReferenceData a binario y viceversa.



```
16
17 /**
18  * Main Class that measure the performance
19  */
20 public class KryoTest
21 {
22     /** a org.slf4j.Logger with the instance of this class given by org.slf4j.LoggerFactory */
23     private final static Logger LOGGER = LoggerFactory.getLogger(KryoTest.class);
24
25     private final static JsonSerializer jsonSerializer = new JsonSerializer();
26     private final static KryoSerializer kryoSerializer = new KryoSerializer();
27
28     public static void main(String[] args) throws URISyntaxException, IOException
29     {
30         //Read the info from a xml and populate the class
31         URL urlJson = KryoTest.class.getClassLoader().getResource("Example.json");
32
33         String json = new String(Files.readAllBytes(Paths.get(urlJson.toURI())));
34
35         ReferenceData referenceData = jsonSerializer.deserialize(json);
36
37         //Test Kryo
38         LOGGER.debug("[Practica 4] Kryo Serializer {}", referenceData.equals(kryoSerializer.serialize(referenceData)));
39     }
40 }
41
```

<terminated> KryoTest [Java Application] C:\Program Files\Java\jdk1.8.0_221\bin\javaw.exe (18 dic. 2019 12:56:49)
12:56:51.059 [main] DEBUG com.cnebrera.uc3.tech.lesson9.kryo.KryoTest - [Practica 4] Kryo Serializer [true]

Práctica 5: Comparación de rendimiento.

Se lanzaron 100 millones de veces las ejecuciones de serialización y deserialización por cada librería (jaxb, json, proto y kryo). Como resultado se puede observar que proto, es el más eficiente en cuanto a rendimiento;

practicas - Lesson9/src/main/java/com/cnebrera/uc3/tech/lesson9/Measurement.java - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help

Package Explorer

- > Lesson2 [MasterUC3Practices]
- > Lesson3 [MasterUC3Practices]
- > Lesson4 [MasterUC3Practices]
- > Lesson5 [MasterUC3Practices]
- > Lesson7 [MasterUC3Practices]
- > Lesson8 [MasterUC3Practices]
- > Lesson9 [MasterUC3Practices]
 - > src/main/java
 - > com.cnebrera.uc3.tech.lesson9
 - > Measurement.java
 - > Serializer.java
 - > com.cnebrera.uc3.tech.lesson8
 - > KryoSerializer.java
 - > KryoTest.java
 - > com.cnebrera.uc3.tech.lesson7
 - > Instrument.java
 - > ReferenceData.java
 - > com.cnebrera.uc3.tech.lesson6
 - > Instrument.java
 - > src/main/resources
 - > JRE System Library [J2SE-1.5]
 - > Maven Dependencies
 - > src
 - > target
 - > pom.xml
 - > lite
 - > MasterUC3Practices [MasterUC3Practices]
 - > p1

Measurement.java

```

204 byte [] protoSerialized = protoSerializer.serialize(referenceDataProto);
205 protoSerializer.deserialize(protoSerialized);

```

ReferenceData.java

Serializer.java

KryoTest.java

Console

```

<terminated> Measurement [Java Application] C:\Program Files\Java\jdk1.8.0_221\bin\javaw.exe (18 dic 2019 15:17:12)
15:17:14.519 [main] DEBUG com.cnebrera.uc3.tech.lesson9.Measurement - [Practica 1] Size of referenceData instrument list 5
15:17:14.527 [main] DEBUG com.cnebrera.uc3.tech.lesson9.Measurement - [Practica 1] Algorithm identifierTwAP
15:17:14.528 [main] DEBUG com.cnebrera.uc3.tech.lesson9.Measurement - [Practica 1] Algorithm marketId1
15:17:14.656 [main] DEBUG com.cnebrera.uc3.tech.lesson9.Measurement - [Practica 2] Json Serializer [true]
15:17:14.742 [main] DEBUG com.cnebrera.uc3.tech.lesson9.Measurement - [Practica 3] Proto Serializer [true]
15:17:14.778 [main] DEBUG com.cnebrera.uc3.tech.lesson9.Measurement - [Practica 4] Kryo Serializer [true]

Serialización.
media Serialización Jaxb = 9736
media Serialización Json= 2041
media Serialización Proto= 168
media Serialización Kryo= 1463
Deserealización.
media Deserialización Jaxb = 14947
media Deserialización Json= 2683
media Deserialización Proto= 985
media Deserialización Kryo= 1399
Serealización y deserealización.
media Serealización y deserealización Jaxb= 26005
media Serealización y deserealización Json= 4459
media Serealización y deserealización Proto= 1021
media Serealización y deserealización Kryo= 2556

```