Distributed Programming II

A.Y. 2019/20

Lab2

All the material needed for this lab is included in the *.zip* archive where you have found this file. Please extract the archive to the same `[root]` working directory where you have extracted the archive for Lab1, so that the files developed in Lab1 remain under `[root]`. In particular, make sure that the XML and XML Schema files you developed remain under `[root]/xsd`.

Purpose

The aim of this Lab is to experiment with XML processing in Java, using the JAXB framework, and with the abstract factory pattern. This lab focuses specifically on the binding and marshalling operations. The next lab will focus on the unmarshalling operations.

Exercise 1

1. Using the *JAXB* framework, write a ***Java* application** for serialization (marshalling) called `BibInfoSerializer` (in package `it.polito.dp2.BIB.sol1`) that serializes the data about a bibliography into an *XML* file **valid** against the *schema* designed in Lab1 - Exercise 2 (file `[root]/xsd/biblio_e.xsd`). The output *XML* file must include all the information that can be retrieved by accessing the interfaces defined in package `it.polito.dp2.BIB`. The implementation of the interfaces to be used as data source must be created according to the "abstract factory" pattern, so that the actual data source to be used can be selected at runtime by setting a java system property. More specifically, the application must create the data source by means of the factory class `it.polito.dp2.BIB.BibReaderFactory`, which has to be instantiated by calling its static method `newInstance()`.

   The library `[root]/lib/BIBRandom.jar` includes a built-in data source which gets the data from an XML file valid against the `[root]/xsd/biblio.xsd` schema. By default, the XML file this data source uses is `[root]/xsd/biblio.xml`. However, the name of the actual file to be used can be customized by setting the *java system property* `it.polito.dp2.BIB.Random.sourceFileName`.

   The application must receive the name of the output *XML* file on the command line as its first and only argument.

   In practice, the application can be developed by modifying the sample *Java* application `it.polito.dp2.BIB.ass1.BibInfo` (provided in source form in the *.zip* archive), which instantiates the interfaces as specified above, reads information from these interfaces and outputs it to the standard output. Your application can read the information in the same way, but it must output it to a valid *XML* file (rather than to the standard output). Note that the order for data serialization may change in the two applications. `BibInfo` can be run from the `[root]` directory by running the provided ant script (which also compiles, adjusts classpaths, includes libraries as necessary, and sets the system property for specifying and customizing the data source that implements the interfaces). This can be done from the command line, by issuing the command

   ```
   $ ant BibInfo
   ```

if we want to use the default XML file as source of data or

```
$ ant -DsourceFileName=fileName BibInfo
```

if we want to use file *fileName* as source of data.

The application must be portable, even when executed in a distributed environment (there must be no dependency on the local machine, location, and settings).

All the sources of the solution must be stored under the directory `[root]/src/it/polito/dp2/BIB/sol1/`. The ant script provided with the assignment material can also be used to generate the bindings, compile and run the developed application.

The command for generation of the bindings from your schema is

```
$ ant generate-bindings
```

The files will be generated into the `gen` directory.

The command for compilation is

```
$ ant build
```

whereas the command for execution is

```
$ ant -Doutput=fileName BibInfoSerializer
```

This command also calls the targets for binding generation and compilation. Of course, *fileName* is the selected output file name. When running the developed application in this way, the default data source is used, and the `sourceFileName` property can be set, as already shown for the `BibInfo` program. If the ant commands for compilation and execution fail, probably you did not follow the specifications given in the assignment strictly.

2. Test your application by using different source files. In each test, check that the output XML file is valid against your schema (file `[root]/xsd/biblio_e.xsd`) and that the data in the output file matches the data in the source file.

Submission instructions

The solutions of the exercise in this Lab will be submitted along with the solutions of Labs 1 and 3.

A submission will be accepted as valid only if:

- the `BibInfoSerializer` application generates well-formed and valid *XML* files (with respect to your `[root]/xsd/biblio_e.xsd` schema).
- the data stored by the `BibInfoSerializer` application in the output *XML* file are the same that were loaded from the data source.

These requirements will be checked with some sample XML documents. Other checks and evaluations on the code (e.g. programming style, adherence to guidelines) will be done according to the list of criteria for evaluation published in the course web site.