Programmazione Distribuita II

A.A. 2011/12

Assignment n. 4

Tutto il materiale necessario per questa esercitazione si trova nel file .zip dal quale questo documento è stato estratto. Per lo svolgimento dell'esercitazione si consiglia di estrarre tutto l'archivio .zip in una cartella vuota, all'interno della quale si lavorerà. Il materiale è del tutto analogo a quello usato per l'Assignment 2.

Una volta estratto l'archivio nella cartella che chiameremo `[root]`, copiare il file `wfInfo.xsd` (quello sottomesso come soluzione dell'assignment 3) nella cartella `[root]/xsd` già predisposta.

L'assignment si compone di due parti:

1. Usando il framework *JAXB*, scrivere un'applicazione *Java* per serializzare (cioè scrivere) i dati relativi ad un insieme di workflow in un file *XML* **valido** rispetto allo *schema* progettato nell'assignment 3. L'applicazione deve essere sviluppata interamente nel package `it.polito.pd2.WF.sol4` e la classe principale deve chiamarsi `WFInfoSerializer`. Analogamente a quanto fatto nell'Assignment 2, il file *XML* generato deve contenere tutte le informazioni che è possibile estrarre tramite le interfacce definite nel package `it.polito.pd2.WF` e l'implementazione delle interfacce da usare come sorgente dei dati deve essere selezionata tramite il pattern "abstract factory": l'applicazione `WfInfoSerializer` deve creare la sorgente dei dati istanziando una `WorkflowMonitorFactory` tramite il metodo statico `newInstance()`.

   L'applicazione deve ricevere il nome del file *XML* di output come parametro sulla linea di comando.
   I sorgenti del programma `WFInfoSerializer` devono essere memorizzati in `[root]/src/it/polito/pd2/sol4/`, dove `[root]` come al solito è la cartella dove è stato estratto il file `.zip`. Lo script di ant fornito nell'archivio `.zip` può essere usato per compilare e lanciare l'applicazione sviluppata, come nell'assignment 2.

   Il file *ant* build.xml incluso nell'archivio *.zip* contiene anche un target, chiamato `generate-bindings`, che deve essere usato per generare i *Java* bindings per lo schema sviluppato. Usando questo target, i file saranno generati nella cartella `generated`. Il target viene chiamato automaticamente quando vengono chiamati i target `build` e `WFInfoSerializer`.

2. Scrivere una libreria *Java* che, usando *JAXB*, legga un file *XML*

   Scrivere, usando il framework *JAXB*, una libreria *Java* che possa essere usata per caricare e validare un file *XML* come quello generato dall'applicazione sviluppata nel primo punto di questo assignment. La libreria deve essere sufficientemente robusta per poter essere usata all'interno di un server di rete: deve considerare "non affidabile" il documento di input, e non deve mai lanciare eccezioni di runtime (come per esempio NullPointerException). La libreria deve implementare tutte le interfacce e le classi astratte definite nel package `it.polito.pd2.WF`, restituendo i dati caricati dal file. La libreria deve essere scritta interamente nel package `it.polito.pd2.WF.sol4` e tutti i suoi sorgenti devono essere memorizzati nella cartella `[root]/src/it/polito/pd2/WF/sol4/`. Come per l'assignment 2, la libreria deve includere una classe factory con nome `WorkflowMonitorFactoryImpl`, che estenda la factory astratta `it.polito.pd2.WF.WorkflowMonitorFactory` e, tramite il metodo `newWorkflowMonitor()`, crei un'istanza della vostra classe che implementa l'interfaccia `it.polito.pd2.WF.WorkflowMonitor`. Il nome del file *XML* di input deve essere ottenuto leggendo la proprietà di sistema `it.polito.pd2.WF.sol4.WFInfo.file`.
   Per compilare la libreria si usi:

   ```
   $ ant build
   ```

che automaticamente chiama anche il target `generate-bindings`.
Se questo fallisce significa che non sono state seguite correttamente le istruzioni.

Il serializzatore e la libreria devono essere portabili e interoperabili anche quando eseguiti in ambiente distribuito (per esempio, devono evitare dipendenze dalla "time zone" locale e dalle altre impostazioni locali della macchina).

Verifica finale della correttezza

Prima di sottomettere la soluzione dell'esercizio siete tenuti a verificarne il corretto funzionamento e l'aderenza alle specifiche date. Perché sia accettabile ai fini dell'esame, la soluzione deve includere entrambe le parti e deve superare almeno i test automatici obbligatori. Si noti che i test automatici controllano solo alcuni dei requisiti funzionali! In particolare essi controllano che:

- l'applicazione `WFInfoSerializer` generi file XML well-formed e validi (secondo lo schema `wfInfo.xsd`);
- i dati memorizzati dall'applicazione `WFInfoSerializer` nel file *XML* di uscita vengano caricati dalle classi della libreria sviluppata al punto 2 senza errori.
- la catena serializzazione + rilettura dei dati da parte della libreria non alteri i dati stessi (i dati estratti dalla libreria alla quale è stato passato un file generato dal serializzatore devono essere gli stessi che erano stati forniti al serializzatore per la generazione del file; per date e tempo è richiesta la precisione del minuto).

Altri controlli e valutazioni sul codice verranno fatti in sede d'esame (quindi superare tutti i test non garantisce necessariamente che la valutazione sarà ottima).
Le verifiche di cui sopra vengono fatte usando dati pseudocasuali generati dal generatore incluso nei file jar..
Il file *.zip* fornito per questo assignment include un insieme di test come quelli che verranno fati girare sul sistema di sottomissione. I test sono gli stessi usati per l'assignment 2 e prevedono gli stessi test case **.**

Per lanciarli si usino le stesse istruzioni date nell'assignment 2.

Formato del file da sottomettere

Occorre sottomettere un unico file *.zip*, contenente tutti i file prodotti. Il file *.zip* può essere prodotto con il seguente comando (dalla cartella `[root]`):

```
$ jar -cf lab4.zip src/it/polito/pd2/WF/sol4/*.java xsd/wfInfo.xsd
xsd/doc.txt
```

Si noti che il file .zip non deve includere i file generati automaticamente.