Programmazione Distribuita II

A.A. 2010/11

Assignment n. 1

Un *workflow* (vedi http://en.wikipedia.org/wiki/Workflow) è la rappresentazione del flusso di azioni che un insieme di persone deve svolgere in modo coordinato per raggiungere un determinato obiettivo.

Un sistema di gestione dei workflow (workflow management system) è un software che controlla e dà supporto all'esecuzione dei workflow in vari modi. Per esempio, tracciando l'esecuzione del workflow e facendo in modo che quando un'azione deve essere eseguita le persone che devono eseguirla vengano notificate, possano prendere in carico l'azione e, dopo averla eseguita, segnalare al sistema il suo completamento. Chiameremo *processo* una particolare esecuzione di un workflow.

Le interfacce Java (vedi [1]) definite nel package `it.polito.pd2.WF`, danno accesso in sola lettura a informazioni relative a un insieme di workflow e relativi processi. Il javadoc delle interfacce documenta il tipo di informazioni che possono essere ottenute. L'interfaccia principale, dalla quale è possibile accedere a tutte le altre informazioni, è l'interfaccia `WorkflowMonitor`. I metodi in questa interfaccia possono essere usati per ottenere informazioni circa i workflow (`getWorkflows`), e i processi (`getProcesses`). Tali metodi forniscono liste di oggetti che implementano le interfacce `WorkflowReader` e `ProcessReader` rispettivamente. Tali interfacce servono per accedere a tutti i dettagli relativi a workflow e processi.

In [2] è possibile vedere due esempi di workflow e relativi processi, descritti nel modo qui previsto.

Tutto il materiale necessario per svolgere questo esercizio è contenuto nell'archivio *.zip* nel quale avete trovato questo file.

Descrizione dell'esercizio

1. Progettare un formato *XML* che possa essere usato per memorizzare tutte le informazioni che possono essere estratte tramite l'interfaccia *Java* `WorkflowMonitor` definita nel package `it.polito.p2.WF`. Il formato deve essere tale che tutte le informazioni ottenibili tramite l'interfaccia siano anche memorizzabili in un documento *XML* conforme al formato, senza ridondanze. Il formato *XML* deve essere specificato tramite una *DTD*, che deve essere salvata nel file `[root]/dtd/wfInfo.dtd`, dove `[root]` è la radice dell'area di lavoro dove sono stati estratti i file dall'archivio.
2. Scrivere una breve documentazione sulle scelte compiute nel progettare la DTD (massimo una pagina) e salvarla nel file ASCII `[root]/dtd/doc.txt`.
3. Scrivere un documento *XML* valido che faccia riferimento alla *DTD* di cui sopra. Il documento deve essere salvato nel file `[root]/dtd/wfInfo.xml`.

Verifica della correttezza

Prima di sottomettere i file, verificarne la correttezza. La soluzione consegnata deve almeno soddisfare i seguenti requisiti per essere considerata accettabile:

- il file `wfInfo.dtd` deve essere corretto sintatticamente;
- il file `wfInfo.xml` deve essere valido e deve fare riferimento alla DTD memorizzata nel

file `wfInfo.dtd` (nella stessa directory);

- il file `doc.txt` deve esistere.

La validità del file XML può essere verificata tramite un validatore XML. Per esempio, se viene usato il programma `DomParseV` (distribuito insieme gli esempi *XML* sul sito del corso), il file può essere validato lanciando

```
java DomParseV wfInfo.xml.
```

Formato per la sottomissione della slouzione

Deve essere consegnato **un solo file** *.zip*, contenente tutti i file prodotti. Il file *.zip* può essere prodotto lanciando il seguente comando (da prompt, dal folder radice):

```
$ jar cf lab1.zip dtd/wfInfo.dtd dtd/wfInfo.xml dtd/doc.txt
```

Riferimenti

[1] folder doc (online: http://pad.polito.it:8080/enginframe/pd2/assignments/lab1/doc/)

[2] file intro.pdf (online: http://pad.polito.it:8080/enginframe/pd2/assignments/lab1/intro.pdf)