Programmazione Distribuita II

A.A. 2011/12

Assignment n. 5

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.

L'assignment consiste nello sviluppare un client per il web service WorkflowInfoService usando la programmazione "dynamic proxy" del framework JAX-WS basata sul WSDL.

Il servizio WorkflowInfoService fornisce accesso in lettura ai dati relativi ad un insieme di workflow (senza le informazioni relative ai processi). Per lanciarlo ed installarlo sul proprio PC è sufficiente dare il seguente comando:

```
$ ant run-server
```

Una volta lanciato il servizio, il WSDL può essere ottenuto come di consueto all'URL dove il servizio è in attesa, che in questo caso è `http://localhost:8181/WorkflowInfoService?wsdl`

La semantica delle operazioni fornite dal servizio è intuitiva: un'operazione (getWorkflowNames) permette di ottenere la lista dei nomi dei workflow disponibili, mentre l'altra operazione (getWorkflows) permette di ottenere tutte le informazioni di un insieme di workflow dati i loro nomi.

Il client da sviluppare deve assumere la forma di una libreria, analoga a quelle sviluppate nella seconda parte degli Assognment 2 e 4. La libreria deve implementare tutte le interfacce e le classi astratte definite nel package `it.polito.pd2.WF`, restituendo i dati ottenuti dal servizio (non essendoci dati sui processi, la libreria deve sempre restituire liste vuote di processi). La libreria deve essere scritta interamente nel package `it.polito.pd2.WF.sol5` e tutti i suoi sorgenti devono essere memorizzati nella cartella `[root]/src/it/polito/pd2/WF/sol5/`. Come per l'assignment 2 e 4, 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 classe della soluzione che implementa l'interfaccia `it.polito.pd2.WF.WorkflowMonitor`. Più precisamente, la libreria deve funzionare nel seguente modo: ogni volta che viene invocato il metodo factory `newWorkflowMonitor()` viene contattato il servizio e viene fatta una fotografia dei dati ottenuti in quel momento. Il `WorkflowMonitor` restituito fornirà sempre questa fotografia dei dati, senza più contattare il servizio. Un eventuale aggiornamento dei dati potrà quindi essere ottenuto solo creando un nuovo `WorkflowMonitor` tramite la factory. Se per qualche motivo il servizio non può essere contattato al momento della creazione del `WorkflowMonitor` o si verifica qualche errore nell'interazione con il servizio che impedisce il recupero dei dati, deve essere sollevata l'eccezione WorkflowMonitorError, prevista per il metodo `newWorkflowMonitor()`.

Il client deve poter funzionare anche nel caso in cui il servizio venga fornito su un diverso URL. L'URL effettivo da usare sarà memorizzato nella proprietà di sistema `it.polito.pd2.WF.sol5.URL`.

Per generare gli artifacts dal WSDL è possibile utilizzare lo script ant nel seguente modo:

```
$ ant compile-wsdl
```

Questo target, come si può vedere, provvede a recuperare il WSDL in base alla proprietà di sistema di cui sopra.

La libreria deve essere robusta e portabile.

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 superare almeno i test automatici obbligatori. Si noti che i test automatici controllano solo alcuni dei requisiti funzionali! In

particolare essi controllano che i dati estratti dalla libreria siano gli stessi che erano stati generati.

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 nel file jar del server.
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 analoghi a quelli usati per gli assignment 2 e 4 e prevedono gli stessi test case.

Per lanciarli si può usare il file `build.xml` fornito, che provvede automaticamente anche a compilare la soluzione. Prima di lanciare i test occorre ovviamente lanciare il server. Successivamente si possono lanciare i test sul client usando 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 lab5.zip src/it/polito/pd2/WF/sol5/*.java
```

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