

Architetture software

Homework 3

Premessa

Bisogna realizzare un'applicazione di messaging basata sulle API JMS (Java Message Service). L'obiettivo di apprendimento è triplice: (1) comprendere e utilizzare il modello di programmazione di JMS; (2) utilizzare un provider JMS; (3) applicare alcuni pattern per l'integrazione di applicazioni. Come provider JMS, si suggerisce, in particolare, l'uso dell'application server Java EE GlassFish – <http://www.oracle.com/technetwork/java/javaee/overview/index.html>. In alternativa, è possibile utilizzare anche altri provider JMS o application server Java EE open source, tra cui: (1) OpenJMS, <http://openjms.sourceforge.net/>, (2) Apache Active MQ, <http://activemq.apache.org/>, (3) HornetQ, <http://jboss.org/hornetq/>.

L'applicazione potrà essere proficuamente realizzata in piccoli gruppi di lavoro, ciascun gruppo composto da un minimo di due a un massimo di quattro persone.

Riferimenti

- [1] The Java EE 6 Tutorial, Java Message Service Concepts, <http://docs.oracle.com/javaee/6/tutorial/doc/bncdq.html>
- [2] The Java EE 6 Tutorial, Java Message Service Example, <http://docs.oracle.com/javaee/6/tutorial/doc/bncgv.html>
- [3] Eventuale ulteriore documentazione del provider JMS scelto.

Lavoro da svolgere

Bisogna realizzare un'applicazione di messaging basato sulle API JMS (Java Message Service) per la gestione di ordini.

Ciascun ordine è caratterizzato dalle seguenti informazioni:

- un identificatore dell'ordine (un numero intero positivo); il valore speciale 0 indica che l'identificatore non è stato ancora assegnato;
- il nome di un cliente (una stringa, senza spazi al suo interno);
- un elenco di nomi di prodotti (ciascuno è una stringa, senza spazi al suo interno).

L'applicazione sarà composta dalle seguenti tipologie di componenti applicativi (application client), che comunicano mediante alcune destinazioni intermedie, come segue:

- Il primo tipo di componente è un "generatore di ordini casuali"; questo componente genera un flusso di *ordini (senza identificatore)*, e li invia a una coda `jms/asw/CodaOrdiniSenzaId`; in pratica, ciascun messaggio inviato su questa coda è un ordine rappresentato come una stringa della forma "`<cliente> <numero prodotti ordinati (n)> <prodotto 1> ... <prodotto n>`". Nell'applicazione di messaging dovranno essere istanziati due componenti di questo tipo, ciascuno dei quali genera un diverso flusso casuale di ordini e li invia alla medesima coda; questo componente potrebbe generare e inviare, ad esempio, 200 ordini.
- Il secondo componente è un Data Enricher, che ha lo scopo di assegnare a ciascun ordine un identificatore univoco; questo componente riceve un flusso di *ordini (senza identificatore)* dalla coda `jms/asw/CodaOrdiniSenzaId`, associa un identificatore univoco a ciascun ordine, e invia un flusso di *ordini (con identificatore)* a una coda `jms/asw/CodaOrdiniConId`; in pratica, ciascun messaggio inviato su questa coda è un

ordine con identificatore rappresentato come una stringa della forma “<id ordine> <cliente> <numero prodotti ordinati (n)> <prodotto 1> ... <prodotto n>”.

- Il terzo componente è uno Splitter, che ha lo scopo di decomporre ciascun *ordine con identificatore* in una *intestazione d'ordine* (che comprende l'identificatore dell'ordine, il nome del cliente e il numero delle righe dell'ordine) e un insieme di *righe d'ordine* (ciascuna delle quali comprende l'identificatore dell'ordine, il numero della riga, e il prodotto associato alla riga d'ordine); questo componente riceve dunque un flusso di *ordini (con identificatore)* dalla coda `jms/asw/CodaOrdiniConId`, li decompone, e invia un flusso di *intestazioni d'ordine* a una coda `jms/asw/CodaIntestazioniOrdini` (in pratica, ciascuna intestazione d'ordine è rappresentata come una stringa della forma “<id ordine> <cliente> <numero prodotti ordinati (n)>”) e un flusso di *righe d'ordine* a una coda `jms/asw/CodaRigheOrdini` (in pratica, ciascuna riga d'ordine è rappresentata come una stringa della forma “<id ordine> <numero riga d'ordine> <prodotto>”).
- Il quarto componente è un Aggregator, che ha lo scopo di ricomporre *ordini* a partire da un flusso di *intestazioni d'ordine* e da un flusso di *righe d'ordine*; questo componente riceve un flusso di intestazioni di ordini da una coda `jms/asw/CodaIntestazioniOrdini` e un flusso di righe d'ordine da una coda `jms/asw/CodaRigheOrdini`, ricompone ordini (sulla base delle informazioni contenute nelle intestazioni e nelle righe) e invia un flusso di *ordini* (ricomposti) a una coda `jms/asw/CodaOrdini`; in pratica, ciascun messaggio inviato su questa coda è un ordine con identificatore rappresentato come una stringa della forma “<id ordine> <cliente> <numero prodotti ordinati (n)> <prodotto 1> ... <prodotto n>”.
- Il quinto tipo di componente è un “consumatore di ordini”; questo componente riceve un flusso di *ordini* dalla coda `jms/asw/CodaOrdini`, li consuma, e semplicemente li visualizza sullo schermo.

Nella realizzazione di questa applicazione di messaging si può far riferimento al workspace `asw-hw3-messaging`, sul sito del corso:

- Progetto “Preparazione”: Il file `leggimi-jms.txt` descrive le attività da svolgere per definire client JMS (come application client di Eclipse) con Glassfish. Il file `leggimi-hw3.txt` contiene ulteriori informazioni di interesse per questo homework. Inoltre, questo progetto contiene alcuni script per la creazione di risorse JMS (per Glassfish).
- Progetto “Common”: contiene **classi che implementano tutta la logica di dominio dell'applicazione**, per rappresentare ordini, intestazioni d'ordine e righe d'ordine, con diversi metodi per trasformare ordini in messaggi (di testo) e viceversa. Inoltre, questo progetto contiene **classi di utilità per la produzione e il consumo di messaggi JMS** (simili a quelle mostrate nella dispensa `asw840-jms`; può essere utile consultare anche il corrispondente workspace).
- Progetto “ProduttoreOrdini”: realizza il primo tipo di componente (un application client per Glassfish).
- Progetto “ConsumatoreOrdini”: realizza il quinto tipo di componente (un application client per Glassfish). Attenzione, attualmente questo componente legge ordini dalla coda `jms/asw/CodaOrdiniSenzaId` e NON dalla coda `jms/asw/CodaOrdini`, come alla fine dovrebbe fare.

In pratica, lo svolgimento di questo homework richiede:

- di definire, sul provider JMS, come oggetti amministrati, tutte le destinazioni intermedie descritte in precedenza, nonché un'opportuna `ConnectionFactory`;

- se si usa un application server diverso da Glassfish oppure un IDE diverso da Eclipse, di adattare i due tipi di client JMS già definiti (il produttore e il consumatore di ordini) alla piattaforma di sviluppo ed esecuzione utilizzata;
- di modificare il consumatore d'ordini, affinché legga ordini dalla coda jms/asw/CodaOrdini;
- infine, di realizzare i tre tipi di client JMS mancanti (nella forma di application client Java EE, se si usa Eclipse come IDE), ovvero il data enricher, lo splitter e l'aggregator.

Si consulti la documentazione del provider JMS scelto per quanto riguarda la modalità di esecuzione delle applicazioni che devono accedere al provider stesso, nonché per quanto riguarda l'eventuale iniezione delle risorse JMS – o, in alternativa, per ciò che riguarda l'utilizzo di JNDI per accedere agli oggetti amministrati JMS.

Il lavoro si potrà considerare concluso quando, in particolare, sarà possibile mandare in esecuzione l'intera applicazione di messaging. Per eseguire l'intera applicazione, bisogna avviare separatamente (ad esempio in finestre dei comandi separati) ciascun client JMS, a partire da quelli più vicini alla destinazione dei dati.

Che cosa bisogna consegnare

Come documentazione del lavoro svolto, e nell'ipotesi che il lavoro sia stato svolto nell'ambito di un piccolo gruppo, sarà necessario realizzare quanto segue:

- Un singolo file compresso (unico per tutto il gruppo) che contiene tutti i file sorgente e tutti i file di configurazione dei diversi componenti applicativi, da inviare per posta elettronica a cabibbo@dia.uniroma3.it.
- Un foglio (unico per tutto il gruppo) in cui sono elencati i membri del gruppo di lavoro.
- Una descrizione sintetica (unica per tutto il gruppo, scritta a mano oppure, indifferentemente, stampata) delle attività svolte, che affronta separatamente almeno i seguenti punti: (1) quale provider JMS è stato usato, (2) quale IDE è stato usato, (3) quale strumento di amministrazione per JMS è stato usato (ad esempio, file di configurazione oppure script), (4) quali oggetti amministrati JMS sono stati definiti, (5) come fare a mandare in esecuzione l'applicazione di messaging realizzata (ad esempio, con quale linea di comando).
- Un breve tema (obbligatorio, realizzato individualmente da ciascun membro del gruppo di lavoro, scritto a mano, in modo leggibile, su un singolo foglio protocollo) in cui si discutono brevemente il funzionamento e le scelte di progetto fatte nella realizzazione del data enricher, dello splitter e dell'aggregator. Il tema deve essere organizzato in sezioni distinte, una per ciascuno di questi tre componenti.