Service Oriented Architecture

Sistemi Operativi e Programmazione Distribuita
Anno Accademico 2010-2011
Specifiche progetto Unix

Giuseppe Lipari
g.lipari@sssup.it

23 maggio 2011

1 Struttura generale

Lo scopo del progetto è di realizzare una infrastruttura che realizzi una architettura "Service Oriented". L'uso di tale architettura deve essere dimostrato implementando una semplice applicazione di processamento di immagini.

Il sistema consiste di un **registro dei servizi**, uno o più **service provider** e uno o più **client**.

Il progetto va realizzato in C++.

1.1 Servizi e Service provider

Un **service-provider** è un processo che mette a disposizione alcuni **servizi** per i client che ne fanno richiesta. Un servizio può essere visto come una funzione che:

- Ha un nome (che identifica univocamente il servizio nel sistema)
- prende dei parametri in ingresso
- fornisce dei parametri in uscita

Un servizio è quindi descritto da una stringa contenente il nome e da una lista ordinata di parametri, ognuno dei quali contiene un *tipo* e una direzione (**IN** o **OUT**). Il tipo del parametro può essere uno dei seguenti:

- numero intero
- numero floating point (double)

- stringa
- buffer di bytes

Il service provider è un processo di tipo server che sta in attesa di richieste di utilizzo di un servizio da parte dei client. Quando una richiesta arriva, il service provider fa il parsing della richiesta e di tutti i parametri, ne verifica la correttezza, e la processa. Al termine manda la risposta al client. La risposta contiene i parametri di uscita (se ci sono).

Il service provider è un programma concorrente; esso serve tutte le richieste in arrivo in parallelo, se necessario utilizzando dei mutex o delle variabili condition per regolare l'accesso alle risorse condivise interne (dipendentemente dal tipo di servizio).

1.2 Registro dei servizi

Il registro dei servizi è un processo che si occupa di registrare la corrispondenza tra servizio e service provider. Infatti, service provider differenti possono supportare servizi diversi oppure lo stesso servizio. Ogni service provider è identificato univocamente dalla coppia (*indirizzo IP, porta*). Il registro dei servizi associa il nome del servizio con il service provider.

Un cliente che vuole usufruire di un servizio per prima cosa chiede al registro dei servizi quali service provider forniscono il determinato servizio; quindi indirizza la propria richiesta direttamente al service provider indicato dal registro.

Poiché più di un service provider può fornire lo stesso servizio, il registro dei servizi usa una qualunque politica (definita da voi) per selezionare uno dei service provider corrispondenti al servizio richiesto (ad esempio round robin).

Il registro è dinamico: all'inizio, nessun servizio è registrato. Un service provider registra un servizio mandando un opportuno messaggio al registro; successivamente, un service provider può volersi cancellare dal registro, oppure de-registrare un suo servizio.

1.3 Client

Il client che ha bisogno di utilizzare un servizio per prima cosa si rivolge al registro dei servizi, chiedendo l'indirizzo e la porta del service provider a cui rivolgersi. Successivamente interagisce direttamente con il service provider.

2 Implementazione

Lo scopo principale del progetto è la realizzazione del registro dei servizi e di una libreria per poter realizzare un service provider e un client in maniera semplice ed efficace.

Lo scopo secondario è la realizzazione di alcuni service provider che utilizzano la libreria, in maniera da dimostrarne il funzionamento.

2.1 Libreria

La libreria deve supportare una classe base chiamata **Service** che permetta di realizzare dei servizi. La classe Service deve supportare le seguenti funzionalità:

- *Interfaccia*: la classe deve fornire un metodo per descrivere il servizio, in termini del nome e del numero e tipo dei parametri. Tale informazione andrà poi registrata nel registro dei servizi. Inoltre, può essere usata per controllare la consistenza delle richieste dei client (ovvero che il client specifichi il numero e il tipo corretto dei parametri.

- *Comunicazione*: la classe deve fornire un metodo semplice per inviare la richiesta di un servizio al service provider su un socket. La classe deve occuparsi (direttamente o indirettamente tramite altre classi di appoggio) di codificare un messaggio o una sequenza di messaggi contenente il nome del servizio e i parametri di ingresso. La classe deve parimenti preoccuparsi di realizzare la decodifica di un messaggio o di una sequenza di messaggi;

- *Realizzazione*: le classi derivate dalla classe base servizio si occuperanno di realizzare il servizio, ad esempio chiamando metodi di altre classi, oppure implementando il codice in proprio. Tali classi vanno scritte dal programmatore di uno specifico service provider e non fanno parte della libreria.

Per ogni classe derivata da Service, ci sarà una classe **Response** che si occupa di codificare la risposta. Essa avrà metodi per codificare e trasmettere i parametri di uscita (da parte del service provider) e per riceverli e decodificarli (da parte del client).

La libreria inoltre contiene funzioni per:

- Settare globalmente l'indirizzo e la porta del registro dei servizi
- Registrare un servizio presso il registro dei servizi
- De-registrare un servizio presso il registro dei servizi
- Cancellare un service provider dal registro dei servizi
- Richiedere l'indirizzo di un service provider al registro dei servizi

2.2 Registro dei servizi

Il registro dei servizi deve essere realizzato in maniera indipendente dalla particolare applicazione e dai particolari service provider, in modo da poter essere riutilizzato in qualunque momento.

2.3 Applicazione

Dopo aver realizzato la libreria, essa va testata realizzando due semplici istanze di service provider. Il primo service provider realizza i seguenti due servizi:

- *Rotate Image(in int direction, in buffer img, out buffer img2)*: prende in ingresso un'immagine JPG e la ruota
- *Horizontal Flip image(in buffer img, out buffer img2)*: prende in ingresso un'immagine JPG e ne produce una speculare, scambiando destra con sinistra.

Il secondo service provider realizza un semplice storage per delle immagini, associando un nome ad un'immagine. I due servizi forniti sono:

- *Store image(in string name, in buffer img)*: conserva in memoria una certa immagine associata ad un nome;
- *Get image(in string name, out buffer img)*: restituisce l'immagine associata al nome specificato nel parametro in ingresso;
- *Get List(out buffer list*: restituisce la lista dei nomi delle immagini memorizzate nel buffer list.

Inoltre, va realizzato un client che ciclicamente:

- legga un file JPG a caso dal disco, oppure lo legga dal secondo provider (ancora una volta a caso);
- ne chieda la rotazione o il flip (a caso) al primo provider,
- e cerchi di memorizzarlo sul secondo provider con nome pari al nome del file.

L'applicazione finale vede in esecuzione il registro dei servizi, i due service providers, e 5 client. Prevedere almeno 5 file JPG diversi (delle foto vanno bene).

Per realizzare il primo service provider, è possibile utilizzare una libreria open source esistente. Ce ne sono molte, eccone alcune di esempio:

- The CImg Library `http://cimg.sourceforge.net/`
- Leptonica `http://code.google.com/p/leptonica/`
- Image Magick `http://www.imagemagick.org/script/index.php`, and Magick C++ API `http://www.imagemagick.org/script/magick++.php?ImageMagick=5bcjsoq1di68polhr6muajas91`

Per realizzare il secondo service provider, utilizzare la politica di locking *lettori/scrittori* per permettere a più clienti contemporaneamente di leggere una immagine, mentre i client che vogliono sovrascrivere un'immagine devono prendere un lock esclusivo.

3 Modalità di consegna

Il progetto va svolto in gruppi di massimo 2 persone. Il progetto va spedito per e-mail almeno 2 giorni prima dell'esame all'indirizzo `g.lipari@sssup.it`, specificando nel subject:

```
[SISOP] Consegna progettino
```

e indicando chiaramente nel testo dell'e-email i nomi degli studenti del gruppo, e l'appello a cui si intende partecipare.