Esercizi sui Callable e InetAddress

Esercitazione di Laboratorio di Programmazione di Rete A

Daniele Sgandurra

Università di Pisa

01/10/2008

L'interfaccia Callable

- Simile a `Runnable`, ma restituisce un valore.
  - Il metodo `run` di un oggetto `Runnable` è dichiarato `void`.
- Interfaccia di tipo parametrizzato.
- Contiene il solo metodo call.
- Es.:

```
import java.util.concurrent.*;
public class MyCallable implements Callable<Integer>
{
    public Integer call()
    {
        return 1;
    }
}
```

L'oggetto Future

- Non è possibile passare un oggetto `Callable` direttamente ad un Thread. Bisogna usare un pool di thread, tramite la classe `ExecutorService`.
- Per passare un task che implementa `Callable` ad un oggetto `ExecutorService`, si invoca il metodo submit:

```
<T> Future<T> submit(Callable<T> task)
```

- Il metodo `submit` ritorna un oggetto Future. Il metodo `get()` di `Future` si blocca fino a che il task descritto dal metodo `call` non è completato.
- Alla fine, viene restituito il risultato del metodo `call`.

Pool di Thread e Callable

1. Usare il metodo statico newCachedThreadPool o newFixedThreadPool della classe `Executors` per creare un pool di thread.
2. Usare il metodo submit per sottomettere un oggetto `Runnable` o `Callable` al pool.
3. Nel caso di `Callable`, il metodo `submit` ritorna un oggetto Future, che conterrà il risultato del Task.
4. Si invoca il metodo `get()` per ottenere il valore di ritorno dall'oggetto `Future`.
5. Invocare il metodo shutdown sul pool se non si vogliono sottomettere più task al pool.

La Classe InetAddress

La classe InetAddress rappresenta un indirizzo IP.

- `String getHostAddress()`: ritorna l'IP in forma testuale.
- `String getHostName()`: ritorna il nome simbolico associato all'IP.
- `static InetAddress[] getAllByName(String host)`: prende il nome di un host, e ritorna un array di indirizzi IP associati all'host.
- `static InetAddress getByAddress(byte[] addr)`: ritorna un oggetto InetAddress a partire da un indirizzo IP rappresentato come array di byte.
- `static InetAddress getByName(String host)`: determina l'IP di un host a partire dal suo nome simbolico.

La Classe NetworkInterface

La classe NetworkInterface rappresenta un'interfaccia di rete.

- `static Enumeration getNetworkInterfaces()`: ritorna tutte le interfacce presenti sulla macchina.
- `static NetworkInterface getByInetAddress(InetAddress addr)`: ritorna l'interfaccia di rete associata all'indirizzo IP specificato.
- `String getName()`: ritorna il nome dell'interfaccia (es., `eth0`).
- `String getDisplayName()`: ritorna informazioni sull'interfaccia.

La Classe NetworkInterface

- Il metodo `getNetworkInterfaces()` ritorna la lista di tutte le interfacce di un host.
- Ritorna anche le interfacce di loopback (o altre interfacce locali).
- L'ordine con cui sono ritornate le interfacce non è specificato.
- È errato prendere la prima interfaccia, ottenere l'`InetAddress`, e pensare che sia una interfaccia raggiungibile da remoto.
- Per questo, bisogna usare alcuni metodi della classe `InetAddress`, ad es.:
  - `isLinkLocalAddress()`.
  - `isLoopbackAddress()`.

Esercizio 1

- Scrivere un programma Java, Resolve che traduca una sequenza di nomi simbolici di host nei corrispondenti indirizzi IP. Resolve legge i nomi simbolici da un file.
- Si deve definire un task che estenda l'interfaccia Callable che, ricevuto come parametro un nome simbolico, provvede a tradurre il nome ritornando un InetAddress.
- Per ottimizzare la ricerca, si deve attivare un pool di thread che esegua i task in modo concorrente.
- Ogni volta che si sottomette al pool di thread un task, si ottiene un oggetto Future<InetAddress>, che deve essere aggiunto ad un ArrayList. Infine, si scorre l'ArrayList, stampando a video gli InetAddress.

Esercizio 2

- Scrivere un programma che enumeri e stampi a video tutte le interfacce di rete del computer, usando i metodi della classe java.net.NetworkInterface.
- Usare il metodo statico getNetworkInterfaces() per ottenere una Enumeration di NetworkInterface.
- Per ogni NetworkInterface, stampare gli indirizzi IP associati ad essa (IPv4 e IPv6) e il nome dell'interfaccia.

Esercizio 3

- Scrivere un programma che ricerca una parola chiave (K) nei file contenuti in una directory e nelle sue sottodirectory,
- Per ogni file che contiene K, si deve visualizzare il nome dei file e il contenuto della prima riga trovata che contiene K.
- Creare una classe FindKeyword che implementa Callable, che ritorna un array di stringhe del formato *nome file:contenuto riga che contiene K*.
- Creare un pool di thread a cui vengono sottomessi FindKeyword per ogni directory/sottodirectory. Il metodo Search della classe FindKeyword implementa la ricerca di K all'interno del file e ritorna la stringa formattata come sopra.
- Alla fine, si otterrà un oggetto Future da cui ricavare i risultati e stamparli a video.

Soluzioni

Inviare la soluzione degli esercizi (solo i file .java) a :

`ricci@di.unipi.it`
`sgandurra@di.unipi.it`

Tra due settimane saranno disponibili le soluzioni.