# Classpath e Esercizi su RMI

## Esercitazione di Laboratorio di Programmazione di Rete A

Daniele Sgandurra

Università di Pisa

26/11/2008

# Classpath

- Quando si avvia l'interprete, esso deve localizzare le classi necessarie.
- Una classe viene cercata nelle sottodirectory delle directory elencate dal classpath.
- Le classi hanno un nome gerarchico che comprende il loro package: il package serve a indicare in quale sottodirectory bisogna cercare la classe.
- Le directory da cui parte la ricerca sono elencate nel classpath.
- Il classpath è un parametro fornito all'interprete Java o da riga di comando o tramite la variabile di ambiente `CLASSPATH`.

# Percorso delle Classi

Supponiamo che i file `.class` stiano nella directory `~/javaclasses/`. Inoltre supponiamo il file Foo.java stia nel pacchetto `lpr.prova`.

```
package lpr.prova;

public class Foo
{
   public Foo()
   {
      System.out.println("Hello World!");
   }
}
```

Si deve procedere così

- Si compila il file `Foo.java`, che produce `Foo.class`.
- Il file `Foo.class` deve trovarsi in `~/javaclasses/lpr/prova/`
- Si definisce il classpath: il percorso delle classi.

Per definire il percorso delle classi ci sono due alternative: si può utilizzare l'opzione `-classpath` (`-cp`) di `javac` e `java`, o impostare la variabile d'ambiente `CLASSPATH` a seconda del S.O. e dalla shell.

# Classpath e rmiregistry

- Attenzione: quando avviate l'`rmiregistry`, deve essere avviato nella directory base del classpath:
  - se non usate package, avviare l'`rmiregistry` dalla directory dove sono situati i file `.class` necessari per l'rmi;
  - se usate package, dalla directory radice del package;
  - in caso di callback, può essere necessario settare la proprietà `java.rmi.server.codebase` (per il momento no).

# Classpath su Linux

In Linux, gli elementi del classpath sono separati dal simbolo due punti (`:`). Ad es.:

```
.:~/javaclasses/:~/archives/archive.jar
```

è un classpath che identifica:

- la directory corrente (`.`), quella visualizzata tramite `pwd`,
- la directory base `~/javaclasses/`,
- l'archivio `~/archives/archive.jar`,

Il file della libreria di runtime di java `rt.jar` e i file jar nelle directory di java `jre/lib` e `jre/lib/ext` sono inclusi di default nel classpath.

```
[~/java] -> ls /usr/local/java/jdk1.6.0_02/jre/lib/*.jar
 6556 /usr/local/java/jdk1.6.0_02/jre/lib/charsets.jar
 2624 /usr/local/java/jdk1.6.0_02/jre/lib/deploy.jar
  704 /usr/local/java/jdk1.6.0_02/jre/lib/javaws.jar
   92 /usr/local/java/jdk1.6.0_02/jre/lib/jce.jar
  608 /usr/local/java/jdk1.6.0_02/jre/lib/jsse.jar
    4 /usr/local/java/jdk1.6.0_02/jre/lib/management-agent.jar
  940 /usr/local/java/jdk1.6.0_02/jre/lib/plugin.jar
 1080 /usr/local/java/jdk1.6.0_02/jre/lib/resources.jar
47188 /usr/local/java/jdk1.6.0_02/jre/lib/rt.jar
```

# Classpath su Linux: Impostazione Temporanea

A questo punto, supponiamo che `Foo.class` stia in `~/javaclasses/` e un file di test `TestFoo.java` sia in `~/java`:

```
import lpr.prova.*;

public class TestFoo
{
   public static void main(String[] args)
   {
      Foo foo = new Foo();
   }
}
```

Per compilare TestFoo.java o si utilizza l'opzione `-classpath`:

```
[~/java] -> javac -classpath ~/javaclasses/ TestFoo.java
[~/java] -> java -cp ~/javaclasses/:./ TestFoo
Hello World!
```

oppure, si setta la variabile d'ambiente `CLASSPATH` (shell `tcsh`):

```
[~/java] -> setenv CLASSPATH .:$HOME/javaclasses
[~/java] -> javac TestFoo.java
[~/java] -> java TestFoo
Hello World!
```

# Classpath su Linux: Impostazione Permanente

Per rendere la modifica al CLASSHPATH permanente, aggiungete al file `.cshrc` nella home directory la riga:

```
setenv CLASSPATH .:$HOME/javaclasses
```

ad es., nella sezione del file dove sono settate tutte le variabili:

```
#Settaggio variabili ambiente globali
setenv CLASSPATH .:$HOME/javaclasses
setenv EDITOR emacs
setenv CSHEDIT emacs
setenv SHELL tcsh
setenv PAGER less
```

Nel caso in cui la vostra shell sia `bash` e non `(t)csh` (come nei casi precedenti), il file da modificare è `~/.bashrc`, e la riga da aggiungere è:

```
export CLASSPATH=.:$HOME/javaclasses
```

Per conoscere che tipo di shell state usando, digitate:

```
-> echo $SHELL
tcsh
```

# I File JAR

Per creare un archivio, usate il comando jar (Java ARchive).

```
jar cvf nomearchivio.jar file1 file2 ...
```

Il classpath può contenere anche una lista di file jar. Con l'esempio precedente:

```
[~/javaclasses] -> jar cvf lprclasses.jar lpr/prova/*.class
added manifest
adding: lpr/prova/Foo.class(in = 355) (out= 268)(deflated 24%)
[~/javaclasses] -> mv lprclasses.jar ~/archives/
[~/javaclasses] -> cd ~/java
[~/java] -> java -cp ~/archives/lprclasses.jar:. TestFoo
Hello World!
```

Per vedere il contenuto di un file jar:

```
-> jar tvf ~/archives/lprclasses.jar
     0 Wed Nov 28 14:42:52 CET 2007 META-INF/
    71 Wed Nov 28 14:42:52 CET 2007 META-INF/MANIFEST.MF
   355 Wed Nov 28 08:14:44 CET 2007 lpr/prova/Foo.class
```

# Esercizio 1

Sviluppare un'applicazione RMI per la gestione di un'elezione. Il server esporta un insieme di metodi:

- `public void vota(String nome)`: accetta come parametro il nome del candidato. Non restituisce alcun valore. Registra il voto di un candidato in una struttura dati opportunamente scelta.
- `public int risultato(String nome)`: accetta come parametro il nome di un candidato e restituisce i voti accumulati da tale candidato fino a quel momento.
- un metodo che consenta di ottenere i nomi di tutti i candidati, con i rispettivi voti, ordinati rispetto ai voti ottenuti.

# 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.