

CONECTAR JAVA CON PROLOG

Nils Murrugarra Llerena nineil.cs@gmail.com
<http://inf.unitru.edu.pe/~nineil/>

Resumen: En este documento veremos la manera de conectar Java con swi-Prolog de una manera simple y sencilla mediante JPL, el ejemplo mostrado ha sido obtenido de los ejemplos que vienen al instalar el swi-prolog, veamos a continuación los pasos que debemos realizar:

1. Crear un proyecto en NetBeans: jpl
2. Crear una clase Family que contenga:

```
package jpl;

import java.util.Hashtable;
import jpl.*;
import jpl.Query;

public class Family
{
    public static void main( String argv[] )
    {
        String t1 = "consult('family.pl')";
        Query q1 = new Query(t1);

        System.out.println( t1 + " " + (q1.hasSolution() ? "succeeded" : "failed") );

        //-----

        String t2 = "child_of(joe, ralf)";
        Query q2 = new Query(t2);

        System.out.println( t2 + " is " + (q2.hasSolution() ? "provable" : "not
provable") );

        //-----

        String t3 = "descendent_of(steve, ralf)";
        Query q3 = new Query(t3);

        System.out.println( t3 + " is " + (q3.hasSolution() ? "provable" : "not
provable") );

        //-----

        String t4 = "descendent_of(X, ralf)";
        Query q4 = new Query(t4);

        System.out.println( "first solution of " + t4 + ": X = " +
q4.oneSolution().get("X"));

        //-----

        java.util.Hashtable[] ss4 = q4.allSolutions();

        System.out.println( "all solutions of " + t4);
```

```

        for ( int i=0 ; i<ss4.length ; i++ ) {
            System.out.println( "X = " + ss4[i].get("X"));
        }

        //-----

        System.out.println( "each solution of " + t4);
        while ( q4.hasMoreSolutions() ){
            java.util.Hashtable s4 = q4.nextSolution();
            System.out.println( "X = " + s4.get("X"));
        }

        //-----

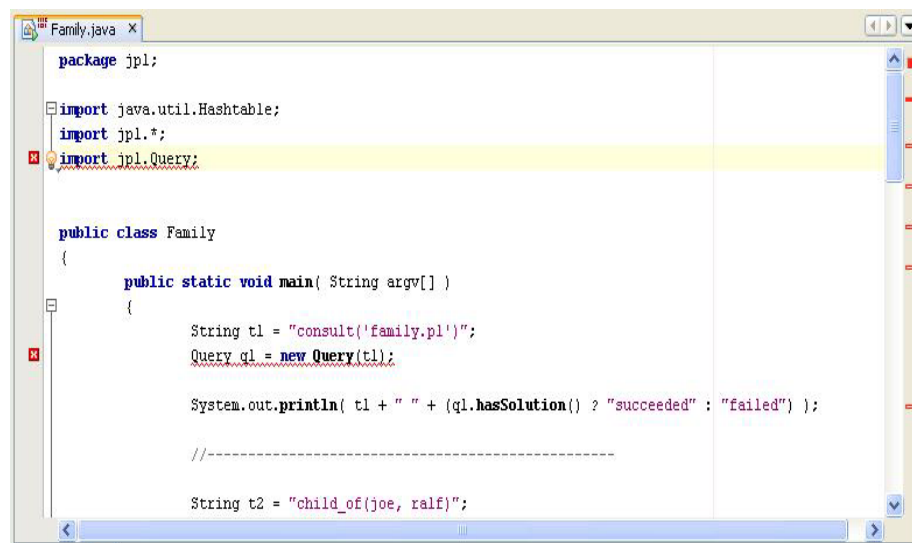
        String t5 = "descendent_of(X,Y)";
        Query q5 = new Query(t5);

        System.out.println( "each solution of " + t5 );
        while ( q5.hasMoreSolutions() ){
            java.util.Hashtable s5 = q5.nextSolution();
            System.out.println( "X = " + s5.get("X") + ", Y = " + s5.get("Y"));
        }
    }
}

```

3. Compilemos

Como podemos ver nos salen 2 errores, veremos como solucionarlos.

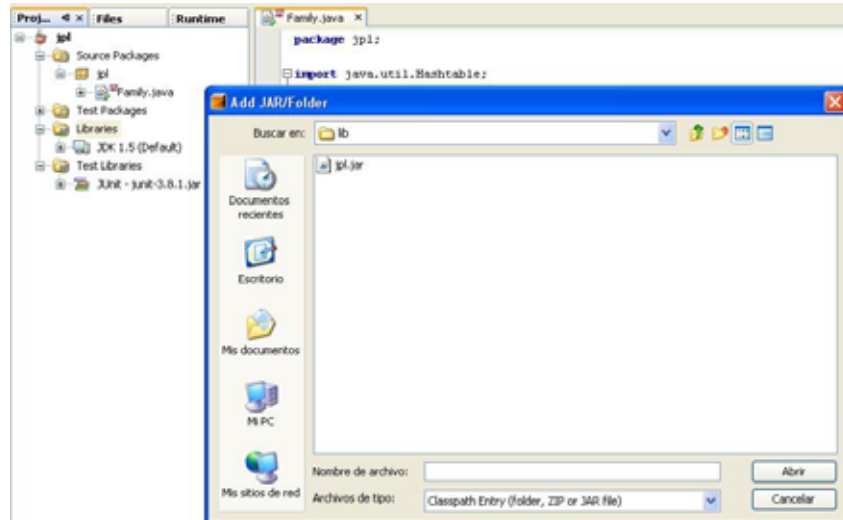


4. Añadir las siguientes rutas al path del sistema:

- C:\Archivos de programa\Java\jdk1.5.0_09\bin;
- C:\Archivos de programa\Java\jdk1.5.0_09\lib\tools.jar;
- C:\Archivos de programa\Java\jdk1.5.0_09\jre\lib\rt.jar;
- C:\Archivos de programa\pl\bin;
- C:\Archivos de programa\pl\lib\jpl.jar;

5. Añadiendo Librería

Añadamos la librería jpl.jar al proyecto, la librería debe encontrarse en: C:\Archivos de programa\pl\lib . Vamos al panel de la izquierda en el ítem libraries, add Jar Fólder y añadimos el jpl.jar.



6. Ahora al realizar la compilación todo sale ok.

7. Crear un archivo en prolog: family.pl con el siguiente contenido

```
child_of(joe, ralf).  
child_of(mary, joe).  
child_of(steve, joe).
```

```
descendent_of(X, Y) :-  
    child_of(X, Y).  
descendent_of(X, Y) :-  
    child_of(Z, Y),  
    descendent_of(X, Z).
```

8. Copiar Family.pl en la carpeta del proyecto actual

9. Analicemos el código

a. *Query q1 = new Query("consult('family.pl')");*

Accedemos al archive de prolog de donde obtendremos nuestras consultas.

b. *Query q2 = new Query("child_of(joe, ralf)");*
q2.hasSolution();

Determinamos si el hecho **child_of(joe, ralf)** nos brinda como respuesta verdadero o falso.

c. *Query q4 = new Query("descendent_of(X, ralf)");*
q4.oneSolution().get("X")

Calcula la primera solución al predicado dado.

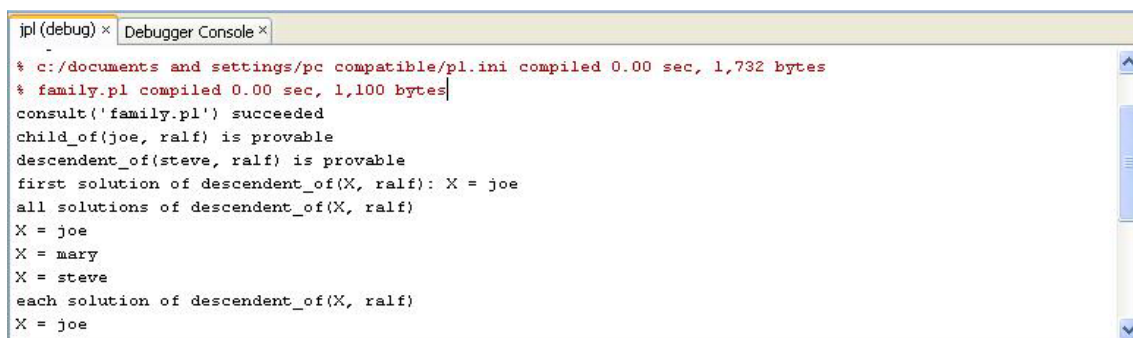
d. `java.util.Hashtable[] ss4 = q4.allSolutions();`

Calcula todas las soluciones posibles para el predicado de q4

10. Compilemos y Ejecutemos

La salida que debemos obtener es la siguiente:

```
% c:/documents and settings/pc compatible/pl.ini compiled 0.00 sec,
1,732 bytes
% family.pl compiled 0.00 sec, 1,100 bytes
consult('family.pl') succeeded
child_of(joe, ralf) is provable
descendent_of(steve, ralf) is provable
first solution of descendent_of(X, ralf): X = joe
all solutions of descendent_of(X, ralf)
X = joe
X = mary
X = steve
each solution of descendent_of(X, ralf)
X = joe
X = mary
X = steve
each solution of descendent_of(X,Y)
X = joe, Y = ralf
X = mary, Y = joe
X = steve, Y = joe
X = mary, Y = ralf
X = steve, Y = ralf
```



```
jpl (debug) x Debugger Console x
% c:/documents and settings/pc compatible/pl.ini compiled 0.00 sec, 1,732 bytes
% family.pl compiled 0.00 sec, 1,100 bytes
consult('family.pl') succeeded
child_of(joe, ralf) is provable
descendent_of(steve, ralf) is provable
first solution of descendent_of(X, ralf): X = joe
all solutions of descendent_of(X, ralf)
X = joe
X = mary
X = steve
each solution of descendent_of(X, ralf)
X = joe
```

11. Los pasos anteriores son suficientes para compilar y ejecutar los programas que usen prolog y Java, pero cuando deseemos usar el .jar generado de nuestro programa debemos tener algunas consideraciones:

a. Crear un archivo .bat que contenga

```
call "C:\Archivos de programa\pl\doc\packages\examples\jpl\java\env.bat"
jpl.jar
```

- b. En nuestro caso nuestro no se vera nada ya que el NetBeans no muestra datos escritos en texto, pero cuando se levante una interfaz esto les será útil.