11-06-2020

Jshell help commands :— complete information can be get by using :—

① To know list of options available with jshell
D:\Java> jshell --help ↵

② To know version of jshell
D:\Java> Jshell --version

③ To know the introduction of jshell:— jshell> /help intro ↵

④ For list of commands:— jshell> /help ↵

⑤ To get information about a particular command.
jshell> /help exit ↵
/help imports ↵ etc.

⑥ To know complete information (only names) of all commands :— jshell> /tab_key ↵

⑦ Again tab_key press → Then you get- Synopsys of all commands- in short.
Again tab key press → Then Full description- List of options available with a particular command
jshell> /list -tab key

Press tab key again to see full documentation.

Understanding Jshell snippets :—

jshell> /list ↵ List of all snippets given by us in current session.

V.I. Note:- package snippet is not allowed in jshell
- we cannot use package snippet in jshell.

Declaration of classes, interfaces and enums in Jshell.

```
D:\Java> jshell -V ↵
|  Welcome to JShell -- Version 11.0.2
|  For an introduction type: /help intro

jshell> class Student { } ↵
|  created class Student

jshell> interface Interf { } ↵
|  created interface Interf

jshell> enum color { }
|  created enum Color

jshell> /vars  ⟶ list of variables
        /methods ⟶ list of methods
        /types ⟶ list of all classes, interfaces and enums.

jshell> /edit ↵
```

```
public class Student
{
    String name;
    int rollno;
  public Student(String name, int rollno)
  {
    this.name=name;
    this.rollno=rollno;
  }
public String getName(){
                return name;
                }
public int getRollno(){
                return rollno;
                }
}
```

Save this code : class is created in Jshell

jshell> Student s = new Student("Ram", 105); ↵

jshell> s.getRollno() ↵

jshell> /edit

```
interface Interf {
    public static void m1()
    {
    System.out.println("Interface Static method");
    }
}
```

Save this & return to jShell prompt:-

jshell> /types ↵

```
Class Student
interface Interf
```

jshell> /list ↵

jshell> Interf.m1() ↵

Interface static method.

jshell> /edit ↵

```
enum Fruits{
Mango("Sweet and Sour"), Banana("Sweet"), Apple("Tasty"),
Pineapple("Sour"), Orange("lemon sweet");
String taste;
Fruits(String taste)
{
this.taste = taste;
}
public String getTaste(){ return taste;
}
```

Save and close the editor.

jshell> /edit
Created enum Fruits

jshell> /list Fruits ↵

jshell> Fruits.Orange.taste ↵

$9 ⇒ Lemon Sweet
| created scratch variable $9 : String

jshell> Fruits.Banana.taste ↵

$10 ⇒ Sweet
| Created scratch variable $10 : String

How to reload previous session of Jshell

```
jshell> /reload -restore
```

As soon as we /exit from Jshell then all variables created in previous session will be lost. Now if we require it then open jshell again and type the command jshell> /reload -restore ↵
All previous session scratch variables are restored with the current session.

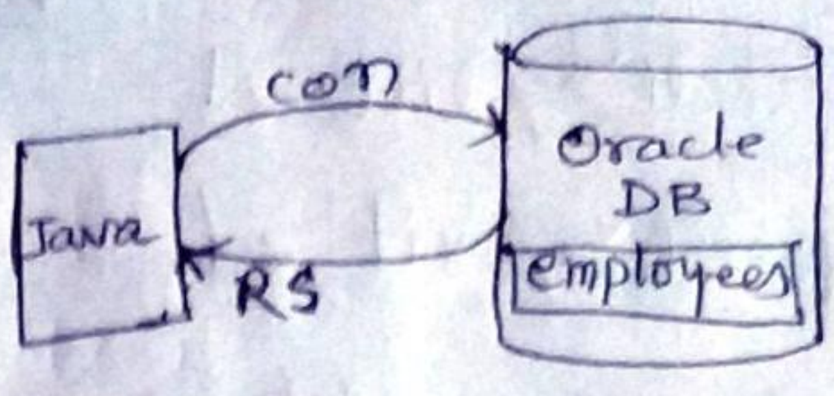
How to reset the Jshell to its default original position →

```
jshell> /reset ↵
```

Now our jshell command prompt settings are restored to default position including settings for editors etc.
After performing /reset command, we are not able to reload previous session variables.

Using Jar files in the Jshell

jshell> /edit ↵



Jar files are collection of classes and methods used to connect with a database. These are provided by third party vendor for working with database generally. JDBC is one of them which allows our program to connect with Oracle database
Name of jar file for JDBC — ojdbc6.jar, ojdbc8.jar, ojdbc11.jar, ojdbc14.jar etc.

for Oracle 11G

```
import java.sql.*;
public void getEmpdata() throws Exception
 {
Connection con = DriverManager.getConnection("jdbc:oracle:
  thin:@localhost:1521:XE", "system", "rranjan");
Statement stmt = con.createStatement();
ResultSet rs = stmt.executeQuery("select * from emp");
System.out.println("Eno\t Name \t Basic \t Address");
System.out.println("_____");
while(rs.next())
 {
System.out.println(rs.getInt(1)+"\t"+rs.getString(2)+"\t"+
                   rs.getDouble(3)+"\t"+rs.getString(4));
 }
con.close();
}
```

Save this file with → db.jsh in current working directory only.

jshell> /open db.jsh ↵

jshell> /list ↵

jshell> /exit ↵

① First way :- Making available jar file while opening jshell :-

D:\Java> Jshell -v --class-path C:\oracleexe\app\

~~Open~~ Oracle database

check there

Select * from emp; ↵

ENO	Name	Basic	Address

Data should be there!
Be sure first

② Second way :- While remaining in the jshell console we can also set path of Jar file ojdbc6.jar

jshell> /env --class-path C:\oraclexe\app\oracle\product\11.2.0\server\jdbc\lib\ojdbc6.jar;

jshell> /open db.jsh ↵

jshell> /list ↵

jshell> getEmpdata() ↵

③ Third way - set classpath explicitly

Open System Properties; Add the classpath value by Specifying ; (semicolon) symbol. and Specify the ojdbc6.jar file's location.

Jshell startup snippets —— Customization of these snippets.

jshell> /list -start ↵

Here 10 packages are by default imported while jshell starting. Now, we want to specify our own startup snippets, which will have to specify within any file as :—

If we want our own snippets will be the default startup snippets for jshell then we have command like —

```
int x=50;
String name="Shashank";
public void m1()
{
System.out.println("Welcome");
}
```

file1.jsh

D:\Java> jshell -v --startup file1.jsh ↵

To check these snippets are default snippets now we can give command —

jshell> /list -start ↵

You can see your three snippets mentioned in file1.jsh

If we want to add our own snippets in addition to 10 default snippets with jshell then we have to specify command —

D:\Java> jshell -v --startup DEFAULT file1.jsh ↵

To check whether the startup snippets now becomes: —

jshell> /list -start ↵

jshell> /list ↵

Now, we want to make available all packages to jshell — total 173 packages.

The command will have to specify — It will took sometime

jshell> /exit ↵

D:\Java> jshell -v --startup JAVASE ↵

D:\Java> jshell -v --startup JAVASE file1.jsh ↵

All packages (173) of Java Runtime environment are imported plus file1.jsh file's three snippets also as startup snippets.

PRINTING option used with jshell

startup — By using this option, we can print any output using print() and println() instead of System.out.print() or System.out.println() functions. There are 21 startup snippets executed with Printing option.

```
jshell> /exit ↵
D:\Java> jshell -v --startup PRINTING ↵
jshell> /list -start ↵
```

shows all 21 startup snippets loaded internally with jshell.

Now, we can use all shortcuts to print any data type specified in the snippets.

```
jshell> print("Hello") ↵
```

Internally calls System.out.print("Hello")

If other options also we need then we can put them all in one statement while startup jshell as:—

```
D:\Java> jshell -v --startup PRINTING DEFAULT ↵
```

Now, with 21 printing snippets, 10 default imports are also available with our jshell command prompt.

We can also load our custom snippets stored into our file name — **file1.jsh** while starting jshell as:—

```
D:\Java> jshell -v --startup PRINTING DEFAULT JAVASE
                                              file1.jsh ↵
```

Now, with jshell prompt we have - printing options (21), Default 10 snippets of import package, Entire java packages (173) with our snippets defined in file1.jsh all gets loaded by jshell. We can use all of them.