Tutorial 4

IS2150
Assoc Prof Kaushik Dutta
Department of Information Systems
School Computing
National University of Singapore

CONTINUATION of MySQL

P.S: Make sure you have the table Persons in your database, if not, please create and insert some values just like in last week's tutorials.

UPDATE STATEMENT:

To view all the rows in the table, we use the statement

```
SELECT * FROM persons;
```

Now suppose we want to change the address of John Lee in the table, we can use an update statement like this

```
UPDATE persons SET address = 'newaddress' WHERE lastname = 'Lee';
```

The general syntax would be

Update TABLE_NAME set COLUMN_NAME1 = new value where COLUMN_NAME2 = value;

The important thing here is the **where clause**. The condition you set in the where clause will determine which rows in the table will be affected.

DELETE STATEMENT:

To delete a particular row from a table, again the where clause is the most important. We can delete rows from the table using the following query

```
DELETE FROM persons WHERE lastname = 'Lee';
```

As in the previous case, the condition specified in the where clause will determine which rows will be deleted.

GROOVY TUTORIAL

1. Netbeans Plugin

1. Download and install **Netbeans** IDE version 6.5 or above.
2. In Netbeans, choose **Tools>Plugins**. In the **Available Plugins** tab, choose Groovy to install the Groovy Grails plugin.

2. Groovy VS Java

Perhaps the best way to capture the elegance and power of the Groovy language is to compare it with Java.

2.1 The Java File

1. Choose **File>New Project** and select **Java Application** from the **Java** category.
2. For the **Project Name**, type "JavaDemo".
3. Unselect the **Create Main Class** checkbox and click **Finish**.
4. Right-click the project and choose **New>Java Class**.
5. For the **Class Name**, type "Todo" and click **Finish**.
6. Replace the code of **Todo.java** with the following:

```
import java.util.*;

public class Todo {
    private String name;
    private String note;

    public Todo(){}

    public Todo(String name, String note){
        this.name = name;
        this.note = note;
    }

    public String getName(){
        return name;
    }

    public void setName(String name){
        this.name = name;
    }

    public String getNote(){
        return note;
    }

    public void setNote(String note){
        this.note = note;
    }

    public static void main(String[] a){
        List todos = new ArrayList();
        todos.add(new Todo("1", "one"));
        todos.add(new Todo("2", "two"));
        todos.add(new Todo("3", "three"));

        for(Iterator iter = todos.iterator();iter.hasNext();){
            Todo todo = (Todo)iter.next();
            System.out.println(todo.getName()+" "+todo.getNote());
        }
    }
}
```

7. Study the code. It is nothing but a POJO (Plain Old Java Object).

8. Right-click the java file and click **Run File** to see the program output.

2.2 The Groovy File

1. Choose **File>New Project** and select **Java Application** from the **Java** category.
2. For the **Project Name**, type "GroovyDemo".
3. Unselect the **Create Main Class** checkbox and click **Finish**.
4. Right-click the project and choose **New>Groovy Class**.
5. For the **Class Name**, type "Todo" and click **Finish**.
6. Replace the code of **Todo.groovy** with the same code as **Todo.java**
7. Right-click the file and click **Run File** and notice that the program works.

Implicit Imports

Remove, from **Todo.groovy**, the import statement:

```
import java.util.*;
```

We do not need this because in Groovy, the package is implicitly imported.
Other implicitly included packages are **java.lang.***, **java.net.***, **java.io.***, **groovy.lang.*** and **groovy.util.***

Constructors

Remove the constructors:

```
public Todo(){}

public Todo(String name, String note){
        this.name = name;
        this.note = note;
}
```

We need not explicitly create a convenience constructor for object initialization. When creating the object, named parameters can be passed to initialize any properties required. To illustrate, change the lines

```
todos.add(new Todo("1", "one"));
todos.add(new Todo("2", "two"));
todos.add(new Todo("3", "three"));
```

to

```
todos.add(new Todo(name:"1", note:"one"));
todos.add(new Todo(name:"2", note:"two"));
todos.add(new Todo(name:"3", note:"three"));
```

Run the file to see that the program still works.

Getters and Setters

In Groovy, the getters and setters of attributes are automatically generated in the byte-code. Therefore we can remove them:

```
public String getName(){
        return name;
    }
```

```
    public void setName(String name){
        this.name = name;
    }

    public String getNote(){
        return note;
    }

    public void setNote(String note){
        this.note = note;
    }
```

Syntactic Sugar

Syntactic sugar refers to syntax within a programming language designed to make things easier to read or express. It makes the language "sweeter" to use.

Change the lines

```
private String name;
private String note;
```

to

```
def name;
def note;
```

The attributes **name** and **note** have now the type of **def**. This is **optional typing** in Groovy. We do not care what type the variable is, as long as it can be handled by the object.

Next, change the lines

```
List todos = new ArrayList();
todos.add(new Todo(name:"1", note:"one"));
todos.add(new Todo(name:"2", note:"two"));
todos.add(new Todo(name:"3", note:"three"));
```

to

```
def todos = [
     new Todo(name:"1", note:"one"),
     new Todo(name:"2", note:"two"),
     new Todo(name:"3", note:"three")
 ]
```

Note that the ArrayList is replaced with **[]**. Groovy instantiates an ArrayList for us.

Next, change the lines

```
for(Iterator iter = todos.iterator();iter.hasNext();){
     Todo todo = (Todo)iter.next();
     System.out.println(todo.getName()+" "+todo.getNote());
}
```

to

```
todos.each{
     println "${it.name} ${it.note}"
}
```

We have replaced the complicated **for** statement with an elegant closure. The iteration is simplifies with the **each()** method, passing a closure that prints out a string. By default, the iteration variable is **it**.

Also, notice the shorter **println()** method. Groovy supports a more advanced string called a **GString**. GStrings are expressions that are embedded within a string in the form **${…}**.

Lastly, syntactic sugar included in the Groovy language is making **semicolons** and **parenthesis** optional.

Result

Your **Todo.groovy** file should look like this:

```
public class Todo {

    def name
    def note

    public static void main(String[] a){

        def todos = [
            new Todo(name:"1", note:"one"),
            new Todo(name:"2", note:"two"),
            new Todo(name:"3", note:"three")
        ]

        todos.each{
            println "${it.name} ${it.note}"
        }
    }
}
```

This is the equivalent of the Java class in Groovy. Notice how dramatically the code has been reduced, while **readability** and **expressiveness** has been increased.