Hibernate Named Query

The hibernate named query is way to use any query by some meaningful name. It is like using alias names. The Hibernate framework provides the concept of named queries so that application programmer need not to scatter queries to all the java code.

There are two ways to define the named query in hibernate:

- by annotation
- by mapping file.

Hibernate Named Query by annotation

If you want to use named query in hibernate, you need to have knowledge of @NamedQueries and @NamedQuery annotations.

@NameQueries annotation is used to define the multiple named queries.

@NameQuery annotation is used to define the single named query.

Let's see the example of using the named queries:

```
@NamedQueries(
   {
      @NamedQuery(
      name = "findEmployeeByName",
      query = "from Employee e where e.name = :name"
      )
   }
)
```

Example of Hibernate Named Query by annotation

In this example, we are using annotations to defined the named query in the persistent class. There are three files only:

- Employee.java
- hibernate.cfg.xml
- FetchDemo

In this example, we are assuming that there is em table in the database containing 4 columns id, name, job and salary and there are some records in this table.

Employee.java

It is a persistent class that uses annotations to define named query and marks this class as entity.

```
package com.javatpoint;

import javax.persistence.*;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;

@NamedQueries(
   {
      @NamedQuery(
      name = "findEmployeeByName",
      query = "from Employee e where e.name = :name"
      )
   }
)

@Entity
@Table(name="em")
public class Employee {

   public String toString(){return id+" "+name+" "+salary+" "+job;}

   int id;
   String name;
   int salary;
   String job;
   @Id
   @GeneratedValue(strategy=GenerationType.AUTO)
```

```
    //getters and setters
}
```

hibernate.cfg.xml

It is a configuration file that stores the informations about database such as driver class, url, username, password and mapping class etc.

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE hibernate-configuration PUBLIC
        "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
        "http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">

<hibernate-configuration>

    <session-factory>
        <property name="hbm2ddl.auto">update</property>
        <property name="dialect">org.hibernate.dialect.Oracle9Dialect</property>
        <property name="connection.url">jdbc:oracle:thin:@localhost:1521:xe</property>
        <property name="connection.username">system</property>
        <property name="connection.password">oracle</property>
        <property name="connection.driver_class">oracle.jdbc.driver.OracleDriver</property>
    <mapping class="com.javatpoint.Employee"/>
    </session-factory>

</hibernate-configuration>
```

FetchData.java

It is a java class that uses the named query and prints the informations based on the query. The **getNamedQuery** method uses the named query and returns the instance of Query.

```
package com.javatpoint;

import java.util.Iterator;
import java.util.List;
```

```
import org.hibernate.cfg.AnnotationConfiguration;
import org.hibernate.*;

public class FetchData {
 public static void main(String[] args) {

  AnnotationConfiguration configuration=new AnnotationConfiguration();
   configuration.configure("hibernate.cfg.xml");
   SessionFactory sFactory=configuration.buildSessionFactory();
    Session session=sFactory.openSession();

    //Hibernate Named Query
    Query query = session.getNamedQuery("findEmployeeByName");
    query.setString("name", "amit");

    List<Employee> employees=query.list();

    Iterator<Employee> itr=employees.iterator();
     while(itr.hasNext()){
    Employee e=itr.next();
    System.out.println(e);
     }

    session.close();

  }
}
```

download this hibernate example (developed using Myeclipse IDE)

Hibernate Named Query by mapping file

If want to define named query by mapping file, you need to use **query** element of hibernate-mapping to define the named query.

In such case, you need to create hbm file that defines the named query. Other resources are same as given in the above example except Persistent class Employee.java where you don't need to use any annotation and hibernate.cfg.xml file where you need to specify mapping resource of the hbm file.

The hbm file should be like this:

emp.hbm.xml

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE hibernate-mapping PUBLIC
        "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
        "http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">

<hibernate-mapping>
<class name="com.javatpoint.Employee" table="em">
<id name="id">
<generator class="native"></generator>
</id>
<property name="name"></property>
<property name="job"></property>
<property name="salary"></property>
</class>

<query name="findEmployeeByName">
<![CDATA[from Employee e where e.name = :name]]>
</query>

</hibernate-mapping>
```

The persistent class should be like this:

Employee.java

```
package com.javatpoint;
public class Employee {
    int id;
```

```
    String name;
    int salary;
    String job;
    //getters and setters
}
```

Now include the mapping resource in the hbm file as:

hibernate.cfg.xml

```
<mapping resource="emp.hbm.xml"/>
```

download this hibernate example (developed using Myeclipse IDE)

← prev next →