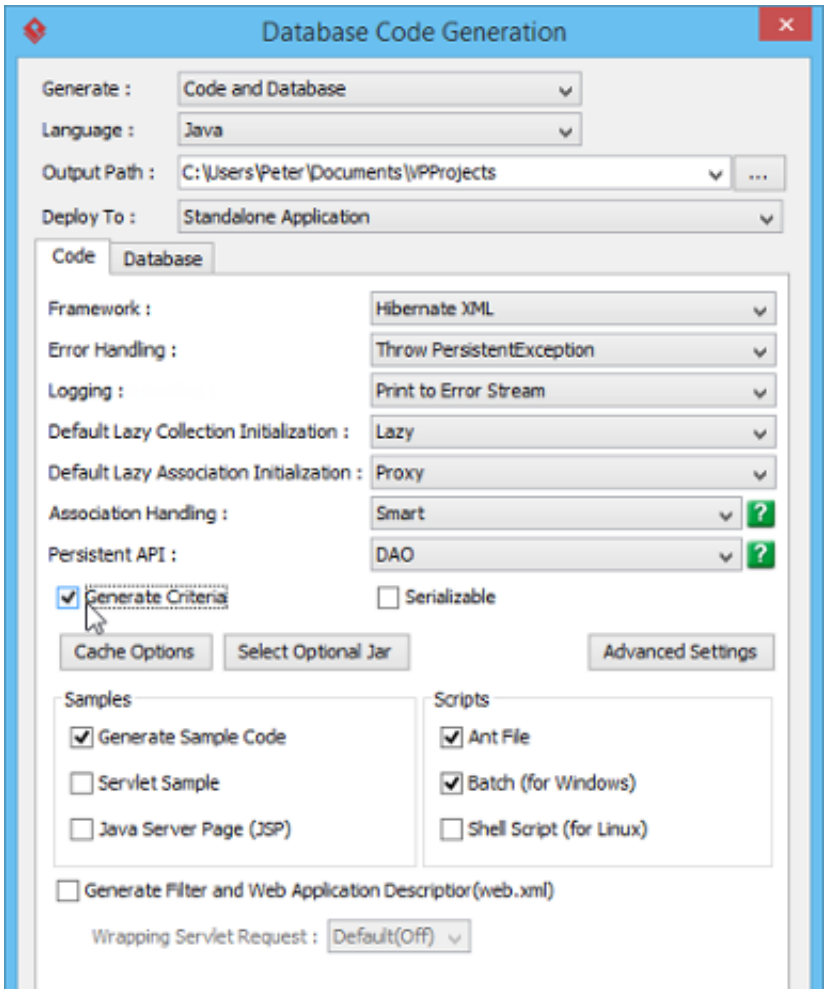
Using ORM Criteria

ORM Criteria is one of the handy approaches that aids data retrieval in writing a program. Instead of writing a lot of if-then-else in checking and filtering the records you need, ORM Criteria groups all the required conditions as a single criteria object, and you can retrieve data by loading the records from that criteria object.

In order to use ORM Criteria, make sure you have checked **Generate Criteria** in the **Database Code Generation** window.



Understanding the Criteria class

The following is an example of ORM Criteria class. Its name and attributes respect the corresponding ORM Persistable class in your object model.

```
public class StaffCriteria extends AbstractORMCriteria {       public
final StringExpression name;     public final IntegerExpression age;
   public final CharacterExpression gender;     public final DateExpre
ssion dob;     public final IntegerExpression ID;       public StaffCr
iteria(PersistentSession session) {        super(session.createCriteri
a(Staff.class));        name = new StringExpression("name", this);
    age = new IntegerExpression("age", this);        gender = new Char
acterExpression("gender", this);        dob = new DateExpression("dob"
```

```
, this);        ID = new IntegerExpression("ID", this);     }       pu
blic StaffCriteria() throws PersistentException {        this(com.Unti
tledPersistentManager.instance().getSession());     }       public Sta
ff uniqueStaff() {        return (Staff) super.uniqueResult();     }
     public Staff[] listStaff() {        return (Staff[]) super.list()
.toArray(new Staff[super.list().size()]);     }  }
```

Using ORM Criteria in programming

To use an ORM Criteria, you need to first specify the conditions and then retrieve data from the Criteria class. To specify conditions, you need to write the following in source code:

```
criteria.property.expression(parameter);
```

where criteria is the instance of the criteria class; property is the property of the criteria; expression is the expression to be applied on the property; parameter is the parameter(s) of the expression. The table below shows the expression that can be used for specifying the condition for query.

Expression	Description
eq(value)	The value of the property is equal to the specified value.
ne(value)	The value of the property is not equal to the specified value.
gt(value)	The value of the property is greater than to the specified value.
ge(value)	The value of the property is greater than or equal to the specified value.
lt(value)	The value of the property is less than the specified value.
le(value)	The value of the property is less than or equal to the specified value.
isEmpty()	The value of the property is empty.
isNotEmpty()	The value of the property is not empty.
isNull()	The value of the property is NULL.
isNotNull()	The value of the property is not NULL.
in(value)	The value of the property contains the specified values in the array.
between(value1, value2)	The value of the property is between the two specified values, value1 and value2.
like(value)	The value of the property matches the string pattern of value; use % in value for wildcard.
ilike(value)	The value of the property matches the string pattern of value, ignoring case differences.

Here is an example of specifying conditions with ORM criteria:

```
staffCriteria.age.ge(13);
```

There are two types of ordering to sort the retrieved records, that is, ascending and descending order. To sort the retrieved records with respect to the property, use the code template:

```
criteria.property.order(ascending_order);
```

where the value of ascending_order is either true or false. True refers to sort the property in ascending order while false refers to sort the property in descending order. For example:

```
staffCriteria.age.order(true);
```

To set the range of the number of records to be retrieved by using one of the two methods:

- setFirstResult(int i) - Retrieve the i-th record from the results as the first result.
- setMaxResult(int i) - Set the maximum number of retrieved records by specified value, i.

For example:

```
staffCriteria.setMaxResults(100);
```

The StaffCriteria class contains two methods to load the retrieved record(s) to an object or array.

- uniqueClass() - Retrieve a single record matching the specified condition(s) for the criteria; Exception will be thrown if the number of retrieved record is not equal to 1.
- listClass() - Retrieve the records matched with the specified condition(s) for the criteria.

For example:

```
com.Staff[] lcomStaffs = staffCriteria.listStaff();
```

Comparison between Criteria class and SQL query

Both SQL Query and Criteria Class can help you find records from database. However, SQL Query is usually long and complex. It is easy to make syntax mistake when writing SQL Query and when a mistake happens, it is hard to debug, too. On the contrary, Criteria class is easy to use. It also supports getting persistent objects directly from database while SQL Query can only retrieve individual data from database.