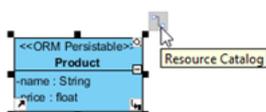# Define custom implementations for ORM Persistable class
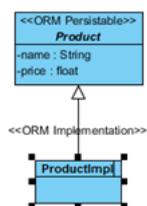
Written Date : March 01, 2016

Hibernate is one of the most popular Object Relational Mapping (ORM) framework on the market which helps in map Java objects with relational database in object-oriented programming. With Visual Paradigm you can generate Hibernate source code from UML class models and Entity Relationship model for building database applications. For most of the case the generated Hibernate source code are just used out-of-the box as the persistence layer and developers can building business logic for their applications on top of it. But in some case developers may want to add simple business logic to the generated persistence layer to ease their job. To do this we can extend the functionality of the persistence layer by defining custom implementations. In this tutorial, we will show you step-by-step procedures to define custom implementation for ORM classes. Visual Paradigm, Eclipse Mars.1 and Microsoft SQL Server will be used in this tutorial.

We assume you already have Visual Paradigm Standard installed and integrated with Eclipse. SQL Server is also being setup and ready to use. Suppose we have a simple ORM class, **Product**, and we want to define extra features for calculate the product price after volume discount:
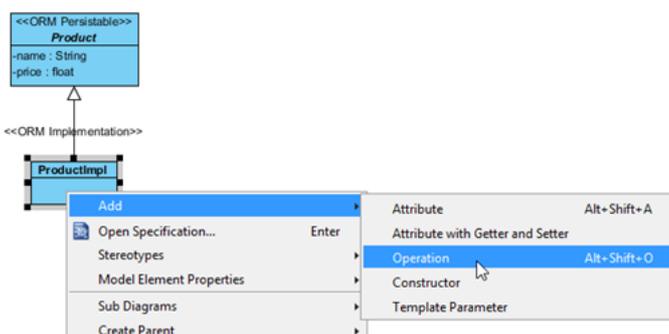
1. Click on the *Product* class and drag out the resource icon.
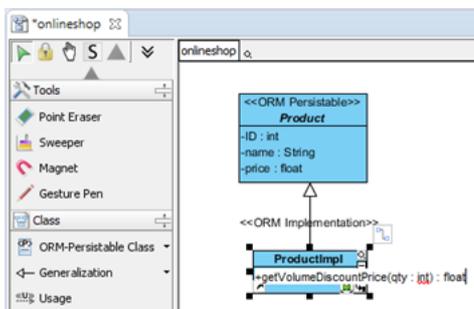

2. Release the mouse button on the desire location in the diagram.

3. Select **Create ORM Implementation Class** in **Resource Catalog**..

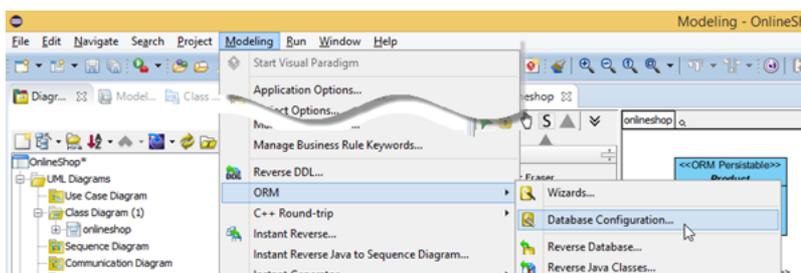4. Name the created class as *ProductImpl*.

5. Right-click on *ProductImpl* and select **Add > Operation**.


6. Name the operation as *getVolumeDiscountPrice(qty : int) : float*

7. Now the model is ready and we can proceed to generate ER model and hibernate code. First we define the default database for our project. Select **Modeling > ORM > Database Configuration...**

8. Select **SQL Server** as our default database. Specify the connection details to the SQL Server. Use the **Test Connection** button to make sure your configuration detail works. Then press OK to close the **Database Configuration** dialog.
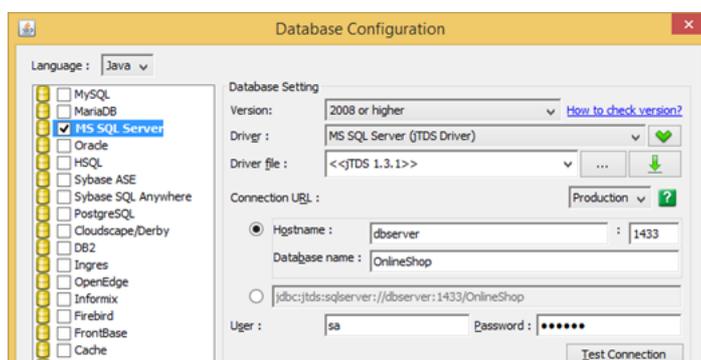
9. Right-click on the blank area of the class diagram and select **Synchronize to Entity Relationship Diagram**, follow the wizard to perform the synchronization with default settings.
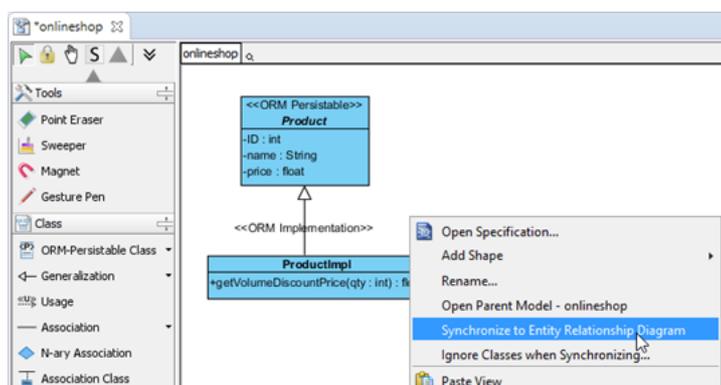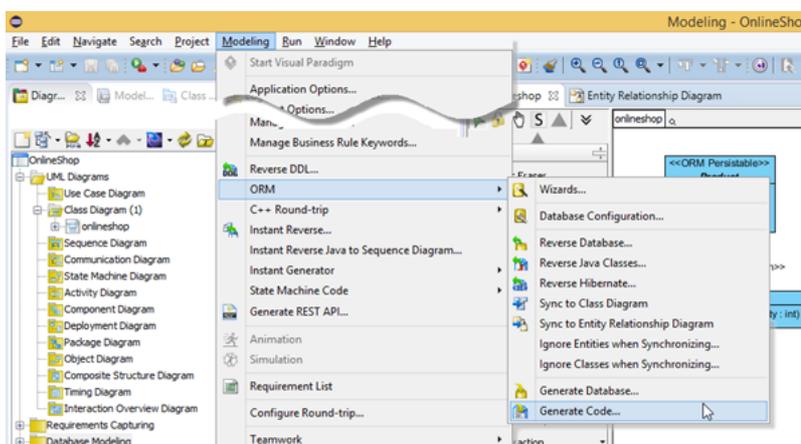
10. The ERD for our model is generated. To simplify the testing we can predefine some sample data for our database. Right-click on the blank area of the ERD and select **Show Table Record Editor or View Editor**.
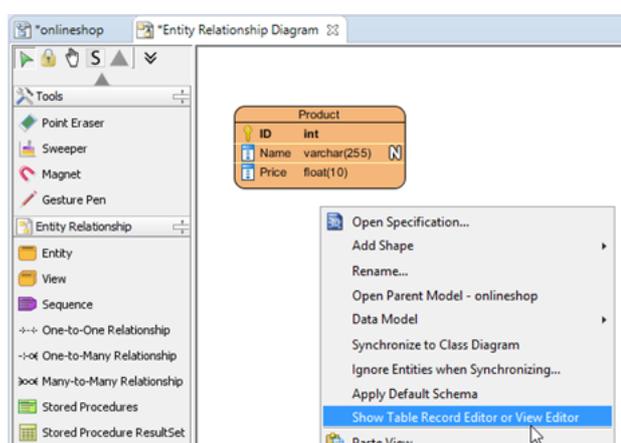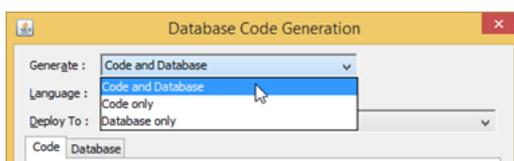
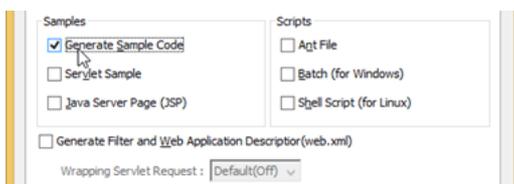11.   Enter the sample record below to the **Table Record Editor**.
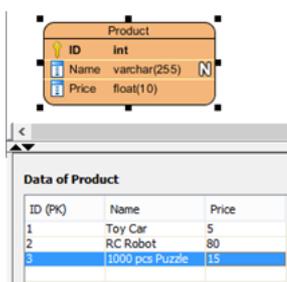
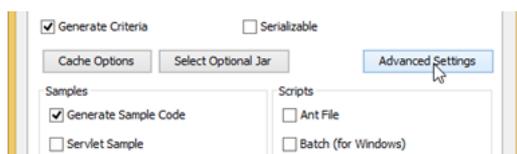12.   We are almost there. Select **Modeling > ORM > Generate Code...**

13.   In **Database Code Generation** dialog, select generate **Code and Database**.
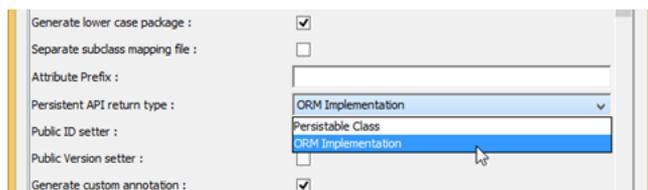

14.   Select **Generate Sample Code** so that we can test the implementation with the generated testing program.
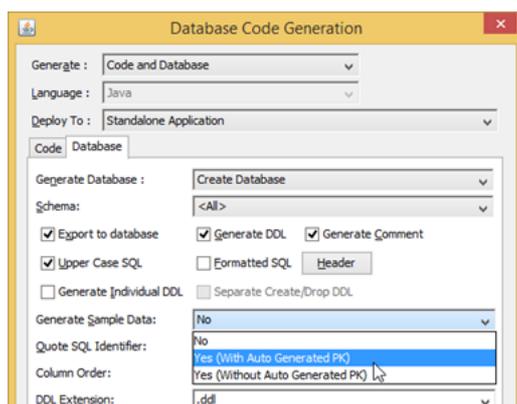
15. Press the **Advanced Settings** button.

16. In the **Advanced Settings** dialog, select **ORM Implementation** as **Persistent API return type**.

17. Switch to **Database** tab and select **Export to database**. Select **Yes (With Auto Generated PK)** in **Generate Sample Data**. Press **OK** to proceed code generation.

18. Now the Hibernate code are being generated, and we can start define the custom implementation in the getVolumeDiscountPrice method. i.e. we offer 10% off for purchase 5+ copies, and 20% off for purchase 10+ copies.

19. Done. Let's modify printout section in the list data sample to test our custom implementation.

```
onlineshop    ProductImpl.java    ListOnlineShopData.java ⊠
/**
 * Licensee: Developer
 * License Type: Purchased
 */
package ormsamples;

import org.orm.*;

import onlineshop.ProductImpl;
public class ListOnlineShopData {
    private static final int ROW_COUNT = 100;

    public void listTestData() throws PersistentException {
        System.out.println("Listing Product...");
        onlineshop.ProductImpl[] onlineshopProducts = onlineshop.ProductDAO.listProductByQuery(null, null);
        int length = Math.min(onlineshopProducts.length, ROW_COUNT);
        for (int i = 0; i < length; i++) {
            ProductImpl product = onlineshopProducts[i];
            System.out.println(product.getName() + ", Original Price: " + product.getPrice());
            System.out.println("Discounted price for 5+  qty: " + product.getVolumeDiscountPrice(5));
            System.out.println("Discounted price for 10+ qty: " + product.getVolumeDiscountPrice(10));

        }
        System.out.println(length + " record(s) retrieved.");

    }
}
```

20. Run the modified list data sample and the custom implementation method was being called.

```
Message  Console ⊠
<terminated> ListOnlineShopData [Java Application] C:\Program Files\Java\jre1.8.0_40\b
Listing Product...
Toy Car, Original Price: 5.0
Discounted price for 5+  qty: 4.5
Discounted price for 10+ qty: 4.0
RC Robot, Original Price: 80.0
Discounted price for 5+  qty: 72.0
Discounted price for 10+ qty: 64.0
1000 pcs Puzzle, Original Price: 15.0
Discounted price for 5+  qty: 13.5
Discounted price for 10+ qty: 12.0
3 record(s) retrieved.
```

Related Links

- Tutorial - Generate Hibernate Mapping for Oracle database

- Tutorial - Begin UML Modeling in Eclipse

- Tutorial - Working with Hibernate in Eclipse

- User's Guide - Eclipse Integration

- What is Entity Relationship Diagram (ERD)?

Visual Paradigm

Visual Paradigm home page
(https://www.visual-paradigm.com/)

Visual Paradigm tutorials
(https://www.visual-paradigm.com/tutorials/)