

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 <u>Visual</u> <u>Paradigm</u> you can generate <u>Hibernate</u> source code from <u>UML</u> class models and <u>Entity Relationship</u> 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. <u>Visual Paradigm</u>, <u>Eclipse Mars.1</u> and <u>Microsoft SQL Server</u> will be used in this tutorial.

We assume you already have <u>Visual Paradigm Standard</u> installed and <u>integrated with Eclipse</u>. 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



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...

0					Modeling - OnlineSt
<u>File Edit Navigate Search Project</u>	Mod	eling <u>R</u> un <u>W</u> indow <u>H</u> e	elp	_	
📑 🕶 📰 🗣 🐘 🐚 💁 🖝	\$	Start Visual Paradigm			◙:::: €, €, €, €, • = +1: +:::
🛅 Diagr 😢 📔 Model 🛅 Class	-	Application Options			reshop 🔀
		Mu	100	1	S ▲ 🛛 S ▲ onlineshop o
📑 🗟 • 👷 🎼 • 🐟 - 🔛 • 🤣 🗃		Manage Business Rule Key	words	ŀ	
OnlineShop*	2	Reverse DDL			Frager Brocket
Use Case Diagram		ORM	,	•	强 Wizards
- Class Diagram (1)		C++ Round-trip		1	Q Database Configuration
Englishing	*	Instant Reverse			Reverse Database
Communication Diagram		Instant Reverse Java to Sequence Instant Generator	uence Diagram		n Reverse Java Classes

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.

<u>s</u>	Datab	ase Configuration	
Language : Java v MySQL MariaD8 V HS SQL Server Crode HSQL Sybase ASE Sybase ASE Sybase SQL Anywhere	Database Setting Version: Driver : Driver file : Connection UBL :	2008 or higher MS SQL Server (JTDS Driver) < <jtds 1.3.1="">></jtds>	v How to check version? v ♥ v … ↓ Production v 2
PostgreSQL Cloudscape,Derby B2 DopenEdge Informix Firebird FrontBase Cache	Hostnam Database jdbc:jtds Uger :	e : dbserver e name : OnlineShop ::sqlserver://dbserver:1433/Or sa :	ilneShop Password : ••••••

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.

😤 *onlineshop 🕴		
🕨 🕤 🕈 🔺 👻	onlineshop o	
Tools	<crm persistable="">> Product -ID : Int -name : String -price : float</crm>	
/ Gesture Pen	<u>А</u>	
Class	< <orm implementation="">> Den Specification</orm>	
ORM-Persistable Class •	Productimpl Add Shape	+
← Generalization •	+getVolumeDiscountPrice(qty:int):fit Rename	
	Open Parent Model - onlineshop	
Association -	Synchronize to Entity Relationship Diag	Iram
N-ary Association	Ignore Classes when Synchronizing	
🛓 Association Class	🛍 Paste View	

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**.

😭 *onlineshop 🛛 🔭 *Entity	Relationship Diagram 🔀	
🕨 🖻 🖏 🖉 🖌 👻		
Tools -		
Point Eraser	Product	
Sweeper	VID int	
💎 Magnet	Price float(10)	
🧪 Gesture Pen		_
🐒 Entity Relationship 🛛 🕂	Open Specification	
Entity	Add Shape	+
View	Rename	
Sequence	Open Parent Model - onlineshop	
+-+ One-to-One Relationship	Data Model	•
-:-⊶ One-to-Many Relationship	Synchronize to Class Diagram	
≫≪ Many-to-Many Relationship	Ignore Entities when Synchronizing	
Stored Procedures	Apply Default Schema	_
Stored Procedure ResultSet	Paste View	

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**.

1	Database Code Generation	×
Gener <u>a</u> te :	Code and Database 🗸 🗸	
Language :	Code and Database	
Deploy To :	Database only	~
Code Data	base	

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.

Generate lower case package :	•	_
Separate subclass mapping file :		
Attribute Prefix :		
Persistent API return type :	ORM Implementation	~
Public ID setter :	Persistable Class	
Public Version setter :		
Generate custom annotation :	•	

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.

🌆 Da	tabase Code Generation	×
Generate : Code and Datab	ase V	
Deploy To : Standalone Appl Code Database	ication	~
Generate Database :	Create Database	~
Schema:	<ai></ai>	~
 Export to database 	Generate DDL Generate Comment	
Upper Case SQL	Eormatted SQL Header	
Generate Individual DDL	Separate Create/Drop DDL	
Generate Sample Data:	No	~
Quote SQL Identifier:	No Kas Alšth Auto Consented BKA	
Column Order:	Yes (Without Auto Generated PK)	
DDL Extension:	.ddl	~

 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.



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

Related Links

- Tutorial Generate Hibernate Mapping for Oracle database
- Tutorial Begin UML Modeling in Eclipse
- <u>Tutorial Working with Hibernate in Eclipse</u>
- User's Guide Eclipse Integration
- What is Entity Relationship Diagram (ERD)?



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

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

https://www.visual-paradigm.com/tutorials/define-custom-implementations-for-orm.jsp