Daro Generic Persistence v1.0 - Software Architecture

Dario A. Palminio

June 7, 2015

Documentation control

Date	Author	Version	Detail
05/05/2015	Dario Palminio	1.0	Creation
04/06/2015	Dario Palminio	1.0	Add info
06/06/2015	Dario Palminio	1.0	Add integration section

Contents

Chapter 1

Introduction

Generic Persistence is un component (package) that implements CRUD (Create, read, update and delete) into DAO (Data Access Object) Generic Persistence in Java. This component uses Hibernate as an Object-Relational Mapping (ORM) framework, Hibernate is concerned with data persistence as it applies to relational databases (via JDBC). This component was designed to deliver a highly scalable architecture and reuse capability. This is configurable and extensible. This integrate hibernate with Spring application. With Hibernate framework, we provide all the database information using anotations and files. We can provide all the information (data source, JDBC connector and mapping) in an Application Context file (xml) from application that uses this library component.

The advantage of using this library component is the relatively simple and rigorous separation between two important parts of an application that can but should not know anything of each other, and which can be expected to evolve frequently and independently. This library component act as an intermediary between the application and the database persistence (CRUD generic on DAO).

1.1 Purpose of the document

The purpose of this document is to show the general architecture of a Java component called "Daro Generic Persistence".

1.2 Audience

Because the document is public license (Creative Commons by-sa 4.0) it was written for anyone interested audience.

Chapter 2

Architecture

2.1 Scope and Context

The Generic Persistence component consists of a JAR package ("com.daro.persistence.generic") named "DaroGenericPersistence-<VERSION>.jar" that can be used from other applications ("Concrete Persistence Service" or application) by extending the core classes.

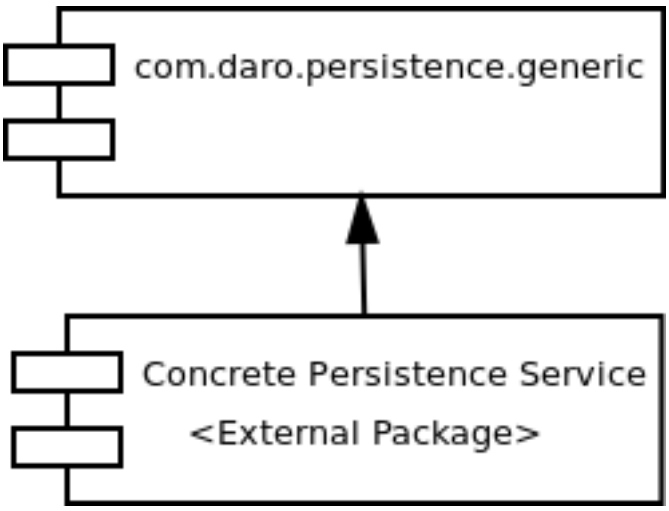

Figure 2.1: Context Diagram - Package Diagram

2.2 High-level Architecture

Class Diagram

The classes described below belong to the package named "com.daro.persistence.generic".

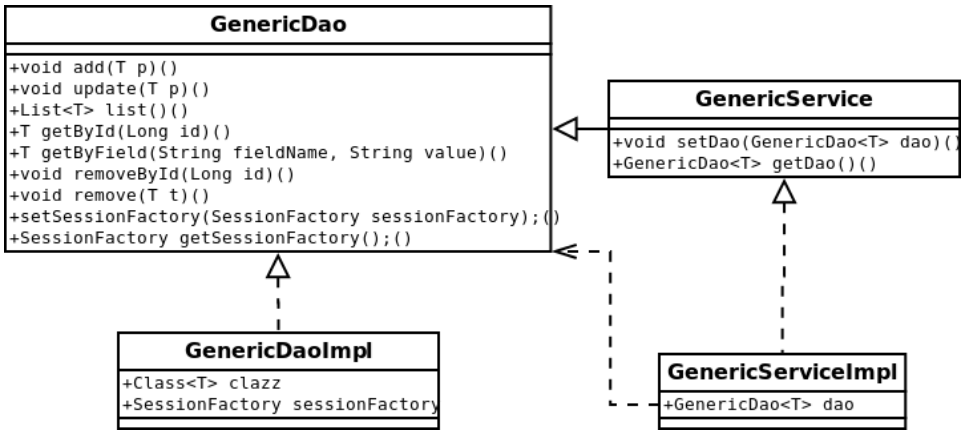

Figure 2.2: General class diagram - high level

Example of a Simple Client

Here's a way to use the component. This example corresponds to the component testing. To use of simple manner the library from code is necessary to extend the GenericDaoImpl class with an entity class (POJO). For example, to persist an entity named PersonEntity can use a class named PersonPersistence that extends GenericDaoImpl. The example shown in the following diagram:

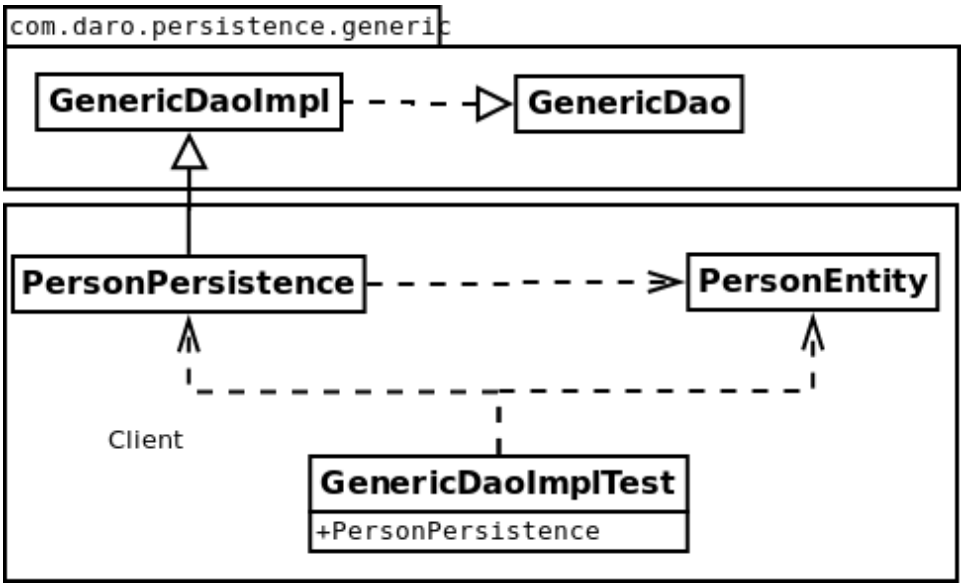

Figure 2.3: Example of how to use on simple manner - diagram

Example of a Complex Client

Here's a way to use the component. This example corresponds to the component testing. The component can also be used in a more complex manner which allows greater scalability. This way is using the services layer. This requires extending the DAO class and Service class. The example is shown in the following diagram:

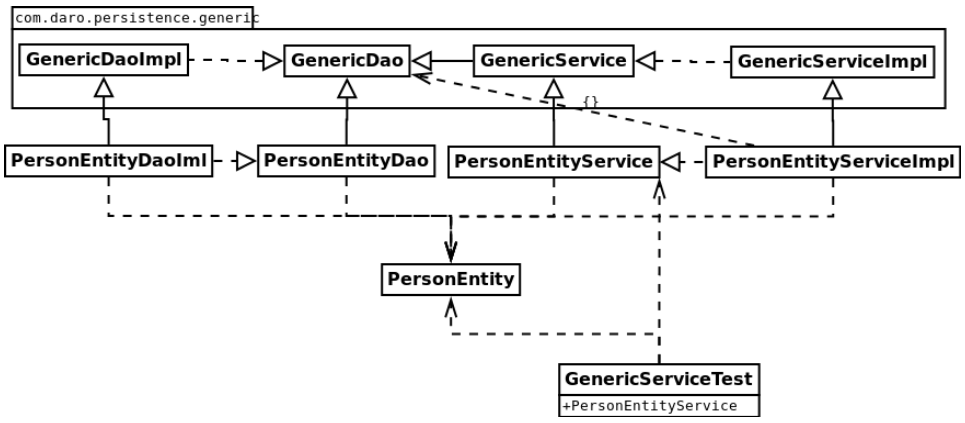

Figure 2.4: Example of how to use - diagram

2.2.1 Integration

The integration is importing this library and configure it via spring application context file (For example you can see test-spring-context.xml file).

The steps required to integrate the library into a java project are:

- Import library into java project.
- Add database connector driver (for example mysql-connector-java).
- To inject DataSourse (via spring application context file).
- To inject SessionFactory and set each entity to be recognized by the SessionFactory (via spring application context file).
- To inject Transaction Manager (via spring application context file).
- Inject SessionFactory each Class Dao we use in the project.

The classes are integrated with Hibernate-Spring as shown in the diagram:

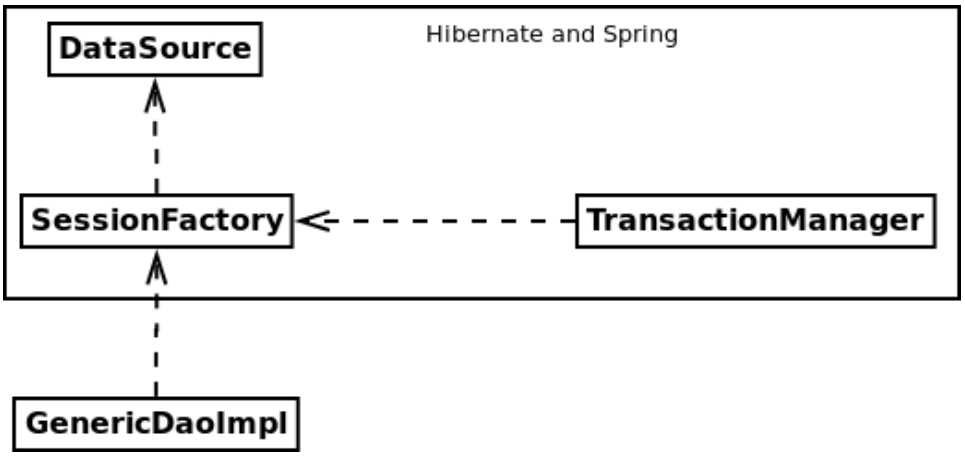

Figure 2.5: Hibernate-Spring Integration diagram