Oct 27th, 3:15 PM - 4:45 PM

# Performance Analysis of Java Persistence API Providers

Besart Pllana

*University for Business and Technology*, besart.pllana@ubt-uni.net

## Recommended Citation

Pllana, Besart, "Performance Analysis of Java Persistence API Providers" (2018). *UBT International Conference*. 101.
https://knowledgecenter.ubt-uni.net/conference/2018/all-events/101

# Performance Analysis of Java Persistence API Providers

Besart Xh. Pllana

UBT – Higher Education Institution, Faculty of Computer Science and Engineering,

Kalabria n.n, 10000 Prishtina, Kosovo.

(e-mail: besart.pllana@ubt-uni.net, besartpllana@gmail.com)

**Abstract.** Nowadays, fast and accurate access to data is very important. Usually data is managed and processed through software applications. In recent years, the most preferred programming model by most application developers is Object Oriented Programming (OOP) where data is represented through objects. These data must be persistent and therefore needs to be stored, and storage can be done on a variety of databases. The most common databases are Relational Database Management Systems (RDBMS). While persistence of objects in RDBMS is limited by object-relational mismatch which is the inconsistency of the direct interaction between two components based on different approaches, OOP object on one side and RDBMS table data on the other, Object-relational mapping (ORM) can be used as a solution. ORM maps the data stored in database tables into the application objects. In other words, ORM persists data from application environment to that of the database. In this paper, we use the Java Persistence API (JPA) specification which provides the characteristics of the ORM technique for developing Java applications. A comparison of three JPA providers was performed by implementing three JPA applications in order to conclude which JPA provider has a better performance.

**Keywords**: Java, Persistence, JPA, EclipseLink, Hibernate, OpenJPA, Object, Relational, Database, ORM, OOP

## 1 Introduction

In general, the existence of a software application is directly related to data. In recent years, the most preferred programming model by most application developers is Object Oriented Programming (OOP). In OOP, data is represented by objects [1]. When an object is created during the execution of an application, its scope is limited by the application lifecycle. So the scope of the object ends with the completion of the application that created the object. It is therefore necessary for the objects be accessible beyond the process of their creation and to achieve such an objective the objects need to be persistent [2].

Persistence of an object means preserving the state of the object so that we can later access it for reading, processing and representation. Objects information can be stored in any database and, in terms of storage, this can be accomplished by simply storing this information in an ordinary text file. However as per the part of reading and processing this information from the file, the process is rather difficult, based on the fact that there isn't any rule according to which the structure of this information will be adapted from ordinary text files back to objects [3].

There exist different data models used for persistence purposes such as the less used data models like hierarchical model (Hierarchical Data Model) and network model (Network Data Model) and also the more popular ones where the most popular is the relational model (Relational Model)[4][5]. The most widely used databases are Database Management Systems (DBMS) which offer reliability, higher performance, faster access, and concurrent transactions, while the most common DBMS are the Relational Database Management Systems (RDBMS) [3].

Although RDBMS are very popular, the persistence of objects in RDBMS is limited by something known as object-relational impedance mismatch. "Object-relational impedance mismatch" is caused by the inconsistency of the direct interaction of two components based on different approaches, object-oriented approach (object in OOP) on one side and relational approach (table data in RDBMS) on the other side [6]. The most common method for solving this mismatch is the use of Object-Relational Mapping (ORM) technique. The most common task of an ORM is to map (transform) the data stored in the relational database tables into the corresponding objects depending on the type of relation, and vice versa [7].

In this paper we will focus on the Java Persistence API (JPA) specification / standard that enables software application developers using Java programming language to benefit from the properties of the ORM technique while developing software applications using Java. More specifically, we will focus on analyzing and comparing the performance of JPA standard providers/ implementations.

## 2 Persistence in Java

### 2.1. Java Database Connectivity (JDBC)

One of the ways to achieve data persistence in Java is to use Java Database Connectivity (JDBC). JDBC is a Java Application Programming Interface (API) that defines how an application interacts with databases through which SQL statements can be executed. JDBC also provides database transaction management[8].

Since different databases use different SQL dials, though JDBC is independent of the database, its use should be made through the specific "driver" of the database. Figure 1 shows the communication scheme of a Java application with a database via JDBC. Thus, using only JDBC, most database actions such as returning data to objects and vice versa should be handled manually by application developers. These include actions such as creating a database connection, creating and retrieving data from the database etc.[8].