JPA Tutorials

**JPA Tutorials and Example:**

Here we are providing easy to learn JPA tutorials. All the JPA concepts discussed here are supported by running example code.

**What is JPA?**

The Java Persistence API or JPA for sort is one to the specification of JEE5, which allows the programmer to develop the persistence layer for their desktop and web applications. JPA is developed to ease the development of persistence layer and it has also standardized the Java ORM technologies.

**Let's get started with JPA.**

1. JDBC vs ORM

In this section we will examine the differences and similarities between JDBC and ORM technologies. Both these technologies are used to save the data into persistence storage for future reference.

1. JPA

This section introduces you with the Java Persistence API (JPA). We will learn the benefits of JPA specification. We will also list down the ORM frameworks that can be used with the JPA specification.

1. JPA Architectures

In this section we will discuss the architecture of JPA specification. Java Persistence API or JPA for short is a lightweight, POJO-based Java framework to persist the Java Objects to the relational database.

1. JPA Features   
   In this section we will discuss about the features of JPA (Java Persistence API).

JDBC vs ORM

In this section we will examine the differences and similarities between JDBC and ORM technologies. Both these technologies are used to save the data into persistence storage for future reference.

**What is JDBC?**

JDBC stands for Java Database Connectivity, is a set of Java API for accessing the relational databases from Java program. The Java API enables the programmers to execute the SQL statements against the JDBC complaint database.

JDBC allows the programmers to quickly develop small Java applications that interact with the databases. To develop an application programmer has to develop the code to connect to the database and then write the code to execute the query. Once the query is successfully executed and result is retrieved from the database, programmer can read the data programmatically from the result set object. Here is the simple example of JDBC example program.

|  |
| --- |
| **import** java.sql.\*;   **public class** AllTableName{     **public static void** main(String[] args) {     System.out.println( "Listing all table name in Database!" );     Connection con =  **null** ;     String url =  "jdbc:mysql://localhost:3306/" ;     String db =  "jdbctutorial" ;     String driver =  "com.mysql.jdbc.Driver" ;     String user =  "root" ;     String pass =  "root" ;     **try** {     Class.forName(driver);     con = DriverManager.getConnection(url+db, user, pass);     **try** {     DatabaseMetaData dbm = con.getMetaData();     String[] types = { "TABLE" };     ResultSet rs = dbm.getTables(null,null, "%" ,types);     System.out.println( "Table name:" );     **while** (rs.next()){     String table = rs.getString( "TABLE\_NAME" );     System.out.println(table);     con.close();     }     }     **catch** (SQLException s){     System.out.println( "No any table in the database" );     }     }     **catch** (Exception e){     e.printStackTrace();     }     }  } |

Above program retrieves the names of all tables present in the database and then displays on the console.

**Advantages of JDBC**

* Clean and easily for small programs
* JDBC provides good performance with large amount of data
* Small JDBC programs can be developed very easily
* Very good for small applications

**Disadvantages of JDBC**

* JDBC is not easily if it is used in large projects. There is a big programming overhead.
* Programmer must hardcode the Transactions and concurrency code in the application.
* Handling the JDBC connections and properly closing the connection is also a big issue. Properly closing the connection is must.
* JDBC is not good for big applications

**What is ORM?**

ORM stands for Object Relational Mapping, is another technology to access the data databases. Here business object is directly mapped to the database tables with the help of ORM framework.

**Here are the benefits of ORM technology**

* No need to deal with the SQL Queries to save and retrieve the data
* Simple configuration
* Standardized API to persist the business objects
* Fast development of application
* Concurrency support
* Excellent cashing support for better performance of the application
* Injected transaction management
* Configurable logging
* Easy to learn and use

**Disadvantages of ORM**

* Slow performance in case of large batch updates
* Little slower than JDBC

JPA Introduction

This section introduces you with the Java Persistence API (JPA). We will learn the benefits of JPA specification. We will also list down the ORM frameworks that can be used with the JPA specification.

The Java Persistence API or JPA for short is Java Specification for persisting the Java Objects to relational database using popular ORM technology. JPA API provides enough tools to enable the java developers to create database driven applications quickly. The JPA API can be used to persist the business object (POJO) to the relational database. Retrieving the data from database in the form of business objects (Java Objects) is so simple with the help of JPA API.

**What is JPA?**

JPA is just an specification from Sun, which is released under JEE 5 specification. JPA standardized the ORM persistence technology for Java developers. JPA is not a product and can't be used as it is for persistence. It needs an ORM implementation to work and persist the Java Objects. ORM frameworks that can be used with JPA are Hibernate, Toplink, Open JPA etc.

These days most of the persistence vendors are releasing the JPA implementation of their persistence frameworks. So, developers can choose the best ORM implementation according to the application requirement. For example, production can be started from the free versions of ORM implementation and when the needs arise it can be switched to the commercial version of the ORM framework. You can switch the persistence provides without changing the code. So, ORM framework independence is another another big benefit of JPA.

**Here are the benefits of JPA**

* Simplified Persistence technology
* ORM frameworks independence: Any ORM framework can be used
* Data can be saved in ORM way
* Supported by industry leaders

**ORM frameworks**

Here are the list of ORM frameworks that can be used with JPA specification.

* Hibernate
* Toplink
* iBatis
* Open JPA

**Why JPA?**

* JPA is standardized specification and part of EJB3 specification
* Many free ORM frameworks are available with can be used to develop applications of any size
* Application developed in JPA is portable across many servers and persistence products (ORM frameworks).
* Can be used with both JEE and JSE applications
* JSE 5 features such as annotations can be used
* Both annotations and xml based configuration support

**What is the current version of JPA Specification?**

The current version of JPA specification is 1.0 at the time of writing of the tutorial. You can develop enterprise grade application using JPA 1.0. Future versions of JPA will provide many features.

In the next session we will learn the architecture of JPA specification.

JPA Architecture

**JPA Architecture:** In this section we will discuss the architecture of JPA specification. Java Persistence API or JPA for short is a lightweight, POJO-based Java framework to persist the Java Objects to the relational database. JPA is uses metadata to map the persistence objects with the database table. JPA supports SQL like query language to ease the process of querying the database. JPA Query language can be used to execute both static and dynamic queries.

JPA supports many ORM frameworks available these days. You can use either free or commercial ORM framework in your JPA based applications. It's also very easy to switch to different ORM frameworks.

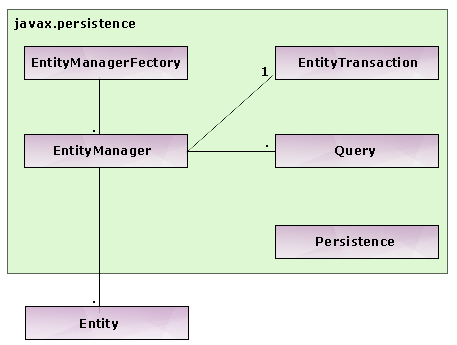
**List of ORM frameworks:**

1. Hibernate
2. Toplink from oracle
3. iBatis
4. Open JPA

You can easily plug any persistence provider into your JPA application.

**JPA Concepts**

JPA concept includes the three components Entity, EntityManager and EntityManagerFactory. Following diagram shows the primary components of JPA architecture.



**Entity**

Entity is the persistence (POJO) objects that represent one record in the table. The Entity is simple annoted POJO class, which is easy to develop. Here are the characteristics of an Entity:

1. Entity can be persisted to the relational database
2. Entity is identified by persistence identity (the primary key of the table)
3. Entity supports transactions
4. Entity supports inheritance

**EntityManager**

The EntityManager interface is providing the API for interacting with the Entity. Some of the functions provided by EntityManager API are:

1. persist ? this method is used to save a new entity
2. merge ? this method is used to update the sate of entity into database
3. remove ? this method is used to remove the entity instance

You will learn about all these functions in next sections. We have developed many JPA examples to help you in learning JPA.

**EntityManagerFactory**

The **EntityManagerFactory** is used to create an instance of EntityManager. In your application when there is no use of **EntityManagerFactory** or application shuts down then it is necessary to close the instance of **EntityManagerFactory** . Once the **EntityManagerFactory** is closed, all its EntityManagers are also closed.

JPA Features

In this section we will discuss about the features of JPA (Java Persistence API). This section will give you an idea about the capabilities of JPA based applications. These days' java developers are using Java Persistence API to develop complex applications.

The introduction of Java Persistence API (JPA) into JEE specification is another major steps towards simplifying the development processes. JPA simplifies the entity persistence model and adds new capabilities. These new capabilities were not present in the earlier versions of EJB (EJB 2.1). Now developers can directly map the persistence object (POJO classes) with the relational database. The Java Persistence API has standardized the object-relational mapping technique.

The JPA can be used outside the container, which was earlier not possible in the EJB 2.1. You can use JPA in your swing applications also.

**Here are the features of JPA:**

The Java Persistence API is now here to simplify the developer's life. Developers can now use the JPA API to develop the applications easily. Here are the features of JPA:

* JPA supports pluggable, third party persistence providers such as Hibernate and Toplink
* JDK 5 annotations are fully supported
* Few java classes are required to develop the persistence applications
* JPA application can run outside the container also. So, developers can use JPA capabilities in desktop applications also
* No need to write deployment descriptors. Annotations based meta-data are supported in JPA applications
* Annotations defaults can be used in model class, which saves a lot of development time
* Provides cleaner, easier, standardized object-relational mapping
* JPA supports inheritance, polymorphism, and polymorphic queries.
* JPA also supports named (static) and dynamic queries.
* JEB QL is very powerfully query language provided by JPA
* JPA helps you build a persistence layer that is vendor neutral and any persistence provider can be used
* Many IDE's are also available to ease the development of JPA applications
* Some IDE's can generate the model and persistence code from database schema
* JPA application can also be configured to generate database schema based on persistence model
* It is also very easy to switch to most performing persistence provider. You can easily move to any commercial persistence providers

In the next sections will discuss the steps to develop application using JPA specifications. We will be using MySQL and hibernate to develop the example applications.