**What is hibernate**

Hibernate is an open source [object relational mapping](https://en.wikipedia.org/wiki/Object-relational_mapping) tool for Java. It provides a framework for mapping an object-oriented domain model to a traditional [relational database](https://en.wikipedia.org/wiki/Relational_database). Hibernate not only takes care of the mapping from Java classes to database tables (and from Java data types to SQL data types), but also provides data query and retrieval facilities and can significantly reduce development time otherwise spent with manual data handling in SQL and JDBC.

Mapping Java classes to database tables is accomplished through the configuration of an XML file or by using Java [annotations](https://howtodoinjava.com/hibernate/hibernate-jpa-2-persistence-annotations-tutorial/). There are facilities to arrange [one-to-many](https://howtodoinjava.com/hibernate/hibernate-one-to-many-mapping-using-annotations/) and [many-to-many](https://howtodoinjava.com/hibernate/hibernate-many-to-many-mapping-using-annotations/)relationships between classes are provided.

In addition to managing associations between objects, Hibernate can also manage **reflexive associations** where an object has a one-to-many relationship with other instances of its own type.

**How hibernate works**

Hibernate doesn’t get in our way; nor does it force us to change the way our objects behave. The objects don’t need to implement any magical interfaces in order to be blessed with the ability to persist. All we have to do to put some meta data in form of **annotations** telling hibernate that how to use them when mapping them with database. At runtime, hibernate reads these annotations and use this information to build queries to send to some relational database.

There is a simple, intuitive API in Hibernate to perform queries against the objects represented by the database, to change those objects we just interact with them normally in the program, and then tell Hibernate to save the changes. Creating new objects is similarly simple; we just create them in the normal way and tell Hibernate about them using annotations so they can get stored to the database.

**Relation of hibernate with JPA**

[JPA (Java Persistence API)](https://en.wikipedia.org/wiki/Java_Persistence_API) is an specification for persistence providers to implement. Hibernate is one such implementation of JPA specification. We can annotate our classes as much as we would like with JPA annotations, however without an implementation nothing will happen.

Think of JPA as the guidelines/specification that must be followed or an interface, while Hibernates JPA implementation is code that meets the API as defined by JPA and provides the under the hood functionality.

*When we use hibernate with JPA we are actually using the Hibernate JPA implementation.* The benefit of this is that we can swap out hibernates implementation of JPA for another implementation of the JPA specification.

When we use straight hibernate your locking into the implementation because other ORMs may use different methods/configurations and annotations, therefore we cannot just switch over to another ORM.

**Java Hibernate hello world example**

Lets create our step by step hibernate hello world example. In this example, I have created an Employeeclass and declared four attributes id, email, firstname and lastname.

I want the id attribute should be generated automatically so that application code does not store a local cache of employee ids.

So far we targeted what we want to make in our first application. Lets identify the files need to be created.

1. **hibernate.cfg.xml** -This configuration file will be used to store database connection information and schema level settings.
2. **EmployeeEntity.java** – This class will refer Java POJOs having hibernate annotations.
3. **HibernateUtil.java**– This class will have utility methods which will be used for creating session factory and session objects.
4. **TestHibernate.java** – This class will be used to test our configuration settings and Emplyee entity annotations.

Before moving into code, lets see the project setup and adding [**maven**](https://maven.apache.org/) dependencies which need to added to pom.xml to include all compile time and runtime dependencies.