Introduction to Hibernate

Overview

Hibernate is a powerful, open-source Object-Relational Mapping (ORM) framework for Java.

It simplifies database interactions by allowing developers to work with Java objects instead of writing complex SQL queries.

History of Hibernate

- Created by Gavin King in 2001 while working on the Cereus project.
- Later maintained and developed by JBoss (a division of Red Hat).
- Continuously improved with contributions from the open-source community.

What is Hibernate?

- Hibernate provides a framework for mapping an object-oriented domain model to a relational database.
- This allows Java developers to work with objects instead of database tables and SQL.

Key Features of Hibernate

- ORM Tool – Maps Java objects to database tables and vice versa.
- Configuration-Based Approach – Uses XML or annotations to define mappings.
- Transaction Management – Abstracts and simplifies database transactions.
- Powerful Querying – Supports HQL (Hibernate Query Language), Criteria API, and native SQL.
- Integration with Java EE and Spring – Easily integrates with enterprise applications.
- Large Community & Support – Active community, extensive documentation, and tutorials

Understanding ORM (Object-Relational Mapping)

ORM is a programming technique for mapping database records to Java objects. Hibernate enables developers to work with objects instead of SQL queries.

Workflow of ORM:

1. Java Application → 2. Objects → 3. ORM Framework (Hibernate) → 4. Database

What is JPA

Jakarta Persistence API (JPA) is a specification that provides certain functionality and standard to ORM tools. The javax.persistence (jakarta.persistence) package contains the JPA classes and interfaces.

Hibernate implements the JPA specification, meaning developers can use Hibernate while adhering to JPA standards.

Relationship between Hibernate and JPA

Feature	Hibernate	JPA
Type	Framework	Specification
Purpose	Implements ORM	Defines ORM standard
Usage	Standalone or with JPA	Implemented by ORM frameworks like Hibernate
Flexibility	Rich set of additional features	More abstract and portable

Hibernate Architecture

Hibernate consists of multiple layers that work together to manage database interactions:

1.Application Layer:

Contains domain classes (entity classes/POJOs) representing business logic.

2. Persistence Layer:

Hibernate sits here, acting as an ORM framework to abstract database operations.

3. Hibernate Configuration:

Uses XML configuration files or annotations to define mappings and connection settings.

Hibernate Architecture

4. Core Components:

- SessionFactory
  - A thread-safe factory for creating Session instances.
  - Typically created once at application startup.
- Session
  - Represents a single unit of work with the database.
  - Manages persistent object lifecycle, executes queries, and performs operations.

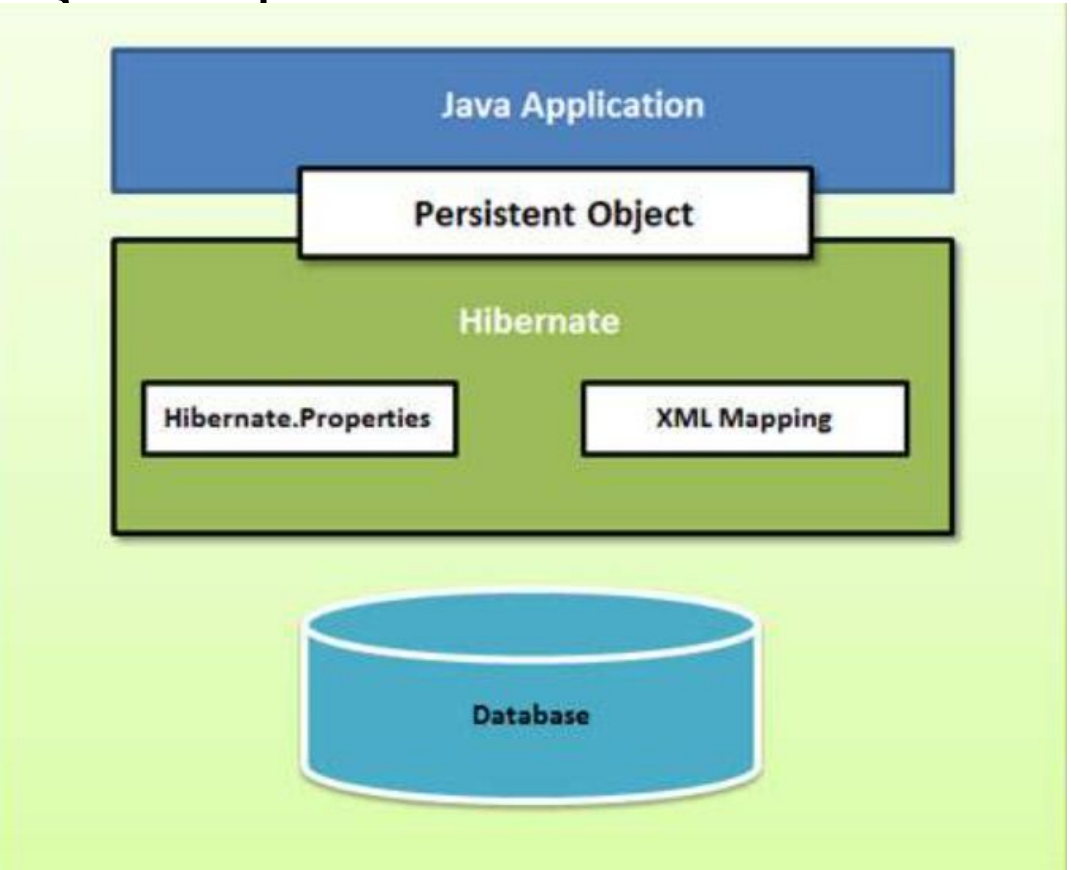
Hibernate Architecture_Core Components

- Transaction Management
  - Ensures data consistency and integrity.
  - Supports programmatic transactions (via Session API) and declarative transactions (via annotations/XML).
- Mapping Layer
  - Defines how Java objects map to database tables.
  - Supports various strategies: Table-per-Class Hierarchy, Table-per-Subclass, Table-per-Concrete-Class.

Hibernate Architecture_Core Components

- Query Language
  - Hibernate provides HQL (Hibernate Query Language), which operates on domain objects instead of tables.
  - Supports native SQL and Criteria API for complex queries.

Diagram of hibernate architecture



Implementing Hibernate

In the next chapter, we will create a simple Hibernate project connected to a database, covering:

- Setting up Hibernate configuration.
- Creating entity classes.
- Performing CRUD operations.