

Hibernate

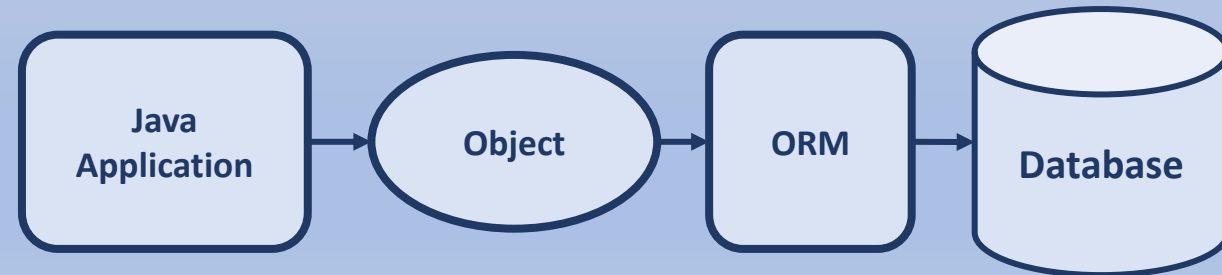
Unit - 2

Topics to be Covered...

- ORM
- JDBC V/s Hibernate
- Introduction to Hibernate
- Hibernate Architecture
- Setting Up Hibernate
- Exploring HQL
- Understanding O/R Mapping
- Working with O/R Mapping
 - Developing Hibernate Configuration File
 - Hibernate Mapping File
 - Java Beans

ORM: Object – Relational Model

- ORM is a programming technique for converting data between incompatible systems, specifically **between object-oriented programming languages and relational databases**.
- Simplifies data manipulation** by abstracting database interactions into higher-level object-oriented constructs.
- Its benefits are:
 - Productivity:** Reduces boilerplate code and speeds up development.
 - Maintainability:** Promotes cleaner, more maintainable code by separating business logic from data access logic.
 - Portability:** Makes it easier to switch between different database systems.
 - Type Safety:** Ensures compile-time checking of data access code.
- Popular ORM Frameworks:
 - Hibernate:** For Java applications.
 - Entity Framework:** For .NET applications.
 - SQLAlchemy:** For Python applications.
 - ActiveRecord:** For Ruby on Rails applications.
 - Django ORM:** For Django applications in Python



JDBC v/s Hibernate

Feature	JDBC	Hibernate
Mapping Java Classes to Database Tables	Manual mapping required	Automatic mapping by Hibernate
Query Generation	Developers write SQL queries	Hibernate generates queries
Portability	Requires specific JDBC driver for databases	More portable, works with various databases
Complexity	Complex, developers manage mapping and SQL	Simplified, Hibernate handles mapping
SQL Support	Supports native SQL queries	Provides HQL (Hibernate Query Language)
Lazy Coding	No Supported by default	Allows loading of Objects at run-time and on-demand.
Complexity	Requires writing and managing complex SQL Queries	Reduces boilerplate coding and simplifies DB Operations

Hibernate Introduction

- ❏ Hibernate is the most used Object/Relational persistence and query service and It is licensed under the open-source GNU Lesser General Public License (LGPL).
- ❏ It is Open-sourced.
- ❏ It simplifies database interactions by mapping Java classes to database tables.
- ❏ Also provides data query and recovery facilities.
- ❏ Commonly used in enterprise applications for managing complex database interactions.
- ❏ Suitable for applications requiring robust data persistence, transaction management, and efficient data retrieval.

Hibernate Introduction

It has features like,

- **ORM:** Automatically maps Java objects to database tables.
- **HQL:** Hibernate Query Language, similar to SQL but operates on Hibernate objects.
- **Fast performance:** The performance of hibernate framework is fast because cache is internally used in hibernate framework.
- **Caching:** Supports both first-level (session) and second-level (session factory) caching.
- **Lazy Loading:** Fetches data only when it's actually needed, improving performance.
- **Automatic Table Generation:** Can automatically generate database tables based on Java class definitions.
- **Simplifies complex join:** To fetch data from multiple tables is easy in hibernate framework.

Hibernate Introduction

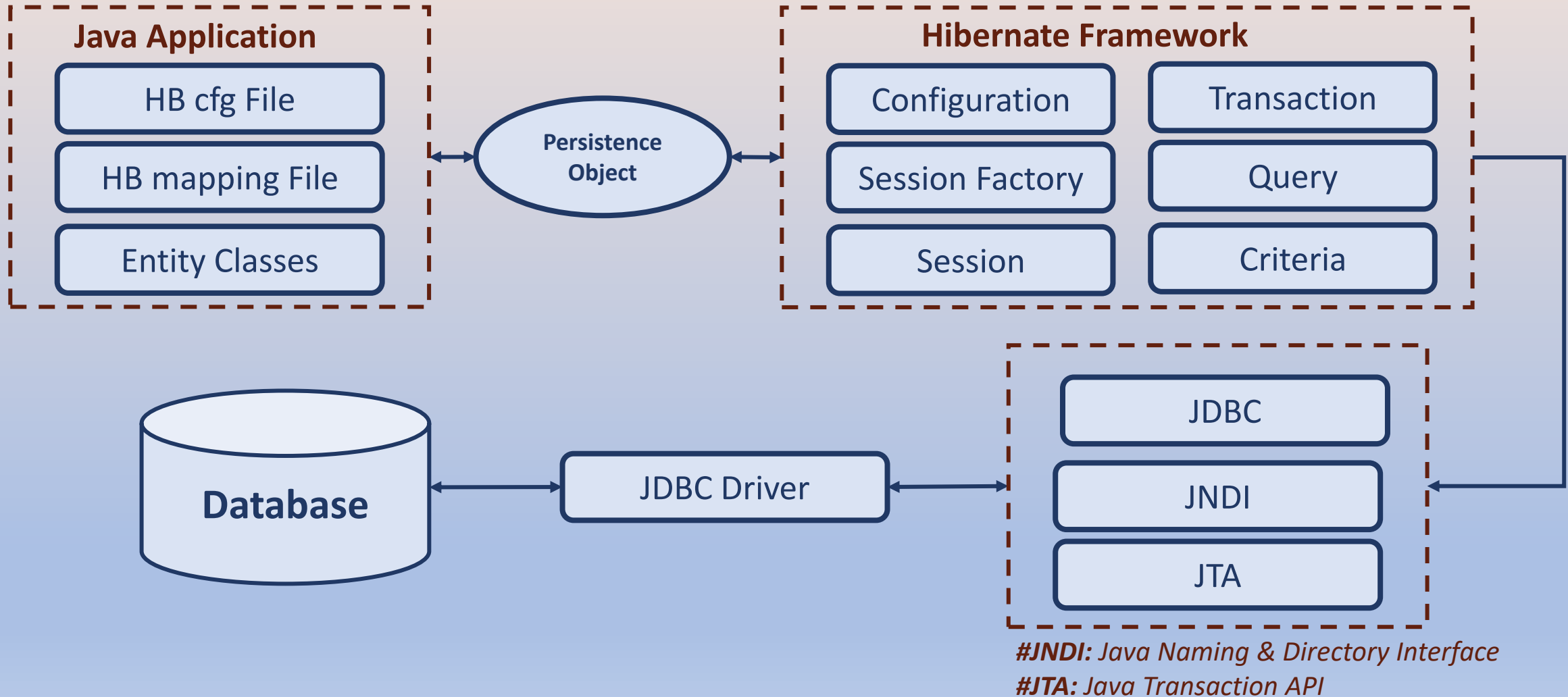
💻 Its benefits include,

- 💻 **Simplified Data Access:** Reduces boilerplate code for database operations.
- 💻 **Portability:** Abstracts away database-specific details, making it easier to switch databases.
- 💻 **Transaction Management:** Provides integrated transaction management, making it easier to manage database transactions.
- 💻 **Performance Optimization:** Supports batch processing, caching, and lazy loading to enhance performance

Hibernate Introduction

- ❏ Typically configured using XML files (**hibernate.cfg.xml**) or annotations within Java classes.
- ❏ Supports **various configurations** for **database connections**, **caching**, and **query optimization**.
- ❏ Uses **Java annotations** (e.g., **@Entity**, **@Table**, **@Id** etc.) to define mappings between Java classes and database tables
- ❏ Manages database operations through **Session** objects, which are lightweight and designed for short-lived use.
- ❏ Provides methods for CRUD operations (**save**, **update**, **delete**, **get**, **load**)
- ❏ **Java Persistence API (JPA)** is a Java specification that provides specific functionality and is a standard for ORM tools.
 - ❏ **javax.persistence** package contains the JPA classes and interfaces.

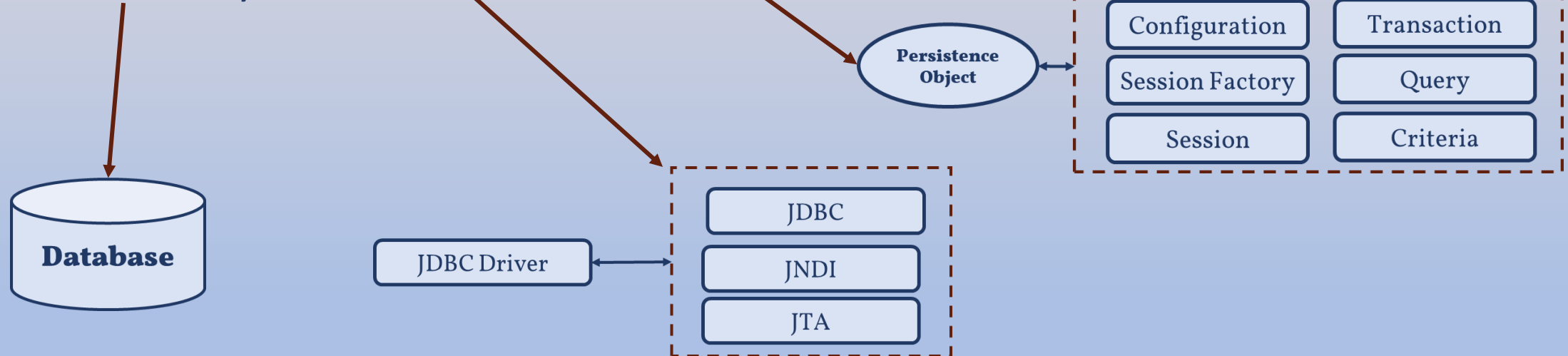
Hibernate Architecture



Hibernate Architecture

It includes mainly 4 layers:

- Java Application Layer
- Hibernate Framework Layer
- Backend API Layer
- Database Layer



Hibernate Architecture

- 💻 Hibernate uses persistence, what is that?
- 💻 Persistence simply means that we would like **our application's data to outlive the applications process**. In Java terms, we would like the state of (some of) our objects to live beyond the scope of the JVM so that the same state is available later.

Hibernate Architecture

Internal API used by Hibernate:

JDBC (Java Database Connectivity):

- A standard Java API for connecting and executing queries against databases. (Already discussed in Unit-I)
- Provides methods for querying and updating data in a relational database.
- Supports executing SQL statements, managing connections, and processing results.

JTA (Java Transaction API):

- A Java API for managing and coordinating transactions across multiple resources.
- Allows applications to perform distributed transactions, ensuring all involved resources commit or rollback together.
- Provides interfaces for demarcating transactions and managing transaction lifecycle.

JNDI (Java Naming Directory Interface):

- A Java API for directory service and naming operations.
- Allows Java applications to access various naming and directory services (e.g., LDAP, DNS).
- Facilitates resource lookup (e.g., database connections, EJBs) in a standardized way across different environments.

Hibernate Architecture

☞ There are various key elements in Hibernate Architecture.

☞ Configuration:

☞ It is the first required object for any Hibernate application.

☞ Usually created during application initialization.


☞ This object provides 2 key components:

☞ **Database Connection:** Handled through one or more hibernate supported config files (`hibernate.properties`, `hibernate.cfg.xml` etc.)



☞ **Class Mapping Setup:** Component responsible for creating connection between Java Classes and Database Tables.

Hibernate Architecture

Configuration:






-  Configuration is a class which is present in `org.hibernate.cfg` package. It **activates Hibernate framework**. It **reads both configuration file and mapping files**.

```
Configuration cfg = new Configuration(); //Activates Hibernate  
cfg.configure(); //reads both cfg file and mapping file.
```

-  It checks whether the config file is syntactically correct or not.
-  If the config file is not valid then it will throw an exception. If it is valid then it creates a meta-data in memory and returns the meta-data to object to represent the config file.







Hibernate Architecture

SessionFactory:

-  **SessionFactory** is an **Interface** which is present in `org.hibernate` package and it is used to create **Session Object**.
-  It is **immutable and thread-safe** in nature.
-  You would need **one SessionFactory object per database** using a **separate configuration file**.
-  **`buildSessionFactory()`**: method gathers the meta-data which is in the `cfg` Object.
-  From `cfg` object it takes the JDBC information and create a JDBC Connection.

Hibernate Architecture





Session:

-  `Session` is an **interface** which is present in `org.hibernate` package. Session object is created **based upon SessionFactory object** i.e. factory.
-  It **opens the Connection/Session with Database** software through Hibernate Framework.
-  It is a **light-weight object** and it is not thread-safe.
-  Session object is used **to perform CRUD** operations.
-  Instantiated each time an interaction is needed with the database.
-  The session objects **should not be kept open for a long time** because they are **not usually thread safe** and they should be created and destroyed as needed.

```
Session session = factory.buildSession();
```


Hibernate Architecture








Transaction:

-  Transaction object is used whenever we perform any operation and based upon that operation there is some change in database.
-  It represents a unit of work with the database.
-  Transactions in Hibernate are handled by an underlying transaction manager and transaction (From JDBC or JTA).
-  Transaction object is used to give the instruction to the database to make the changes that happen because of operation as a permanent by using `commit()` method.

```
Transaction tx = session.beginTransaction();  
tx.commit();
```




Hibernate Architecture

Query:

-  `Query` is an interface that present inside `org.hibernate` package.
-  Query objects use `SQL` or `Hibernate Query Language (HQL)` string to store/retrieve data to/from the database and create objects.
-  A Query instance is **obtained by** calling `Session.createQuery()`.
-  A Query instance is used to **bind query parameters**, limit the number of results returned **by the query**, and finally to **execute the query**.
-  This interface exposes some extra functionality beyond that provided by `Session.iterate()` and `Session.find()`:
 -  A particular page of the result set may be selected by calling `setMaxResults()`, `setFirstResult()`.
 -  Named query parameters may be used.

Hibernate Architecture

Criteria:

-  Criteria is a **simplified API** for retrieving entities by composing **Criterion** objects.
-  The **Session** is a **factory** for **Criteria**. Criterion instances are usually obtained via the factory methods on Restrictions.
-  Criteria objects are used to **create and execute object oriented criteria queries** to retrieve objects.

Setting-Up Hibernate Environment

Download and Install Eclipse:

 Download Eclipse IDE: <https://www.eclipse.org/>

 Install Eclipse IDE using “Eclipse IDE for Java Developers” option.

Download Hibernate Repository (.zip):

<https://sourceforge.net/projects/hibernate/files/>

 Extract the .zip into a “hibernate” directory.

Create Hibernate Application:

 Create a new Project: File -> New -> Project -> Java Project, give name and Finish.

 Add .jar files: Right click on Project -> Build Path -> Add External Archives

 In the new dialog, Select required .jar files

HQL: Hibernate Query Language

O/R Mapping in Hibernate

Three most important mapping are as follows:

- Collections Mappings
- Association Mappings
- Component Mappings