Ebean in Java explained

give me the deep and detailed explaination of Ebean in java so that i can able to understand it well and do not have any type of doubt in this.

so give me the deep and detailed explaination

Alright Piyush 👍
Since you prefer **deep, structured, and doubt-free explanations**, I'll explain **Ebean in Java** from **absolute basics → internals → real usage → comparison with Hibernate/JPA**, in **simple language**, step by step.

🌱 What is Ebean in Java?

Ebean is a **Java ORM (Object Relational Mapping) framework**.

📌 In simple words:

> **Ebean helps you work with a database using Java objects instead of writing SQL manually.**

It maps:

- Java **classes** → **database tables**
- Java **objects** → **table rows**
- Java **fields** → **table columns**

Ebean is officially known as **Ebean ORM**.

🧠 Why Was Ebean Created?

Most Java developers use:

- **Hibernate**
- **JPA (Java Persistence API)**

But these have problems:

- Too many annotations
- Complex configurations
- Lazy loading confusion
- Heavy boilerplate code

👉 **Ebean was created to be:**

- Simple
- Clean
- Less verbose
- More SQL-friendly
- Easier to learn

🆚 Ebean vs Traditional JDBC

❌ JDBC Way

```
java

Connection con = DriverManager.getConnection(...);
PreparedStatement ps = con.prepareStatement("SELECT * FROM user");
ResultSet rs = ps.executeQuery();
```

Problems:

- Too much boilerplate
- Manual mapping
- Error-prone
- Hard to maintain

✅ Ebean Way

```
java

List<User> users = User.find.all();
```

✔️ Clean
✔️ Readable
✔️ Less code
✔️ Object-oriented

🧱 Core Idea of Ebean (Very Important)

Ebean follows **Active Record Pattern**

📌 This is DIFFERENT from JPA.

Active Record Pattern

> The **entity class itself** contains database operations.

Example:

```
java

User.find.byId(1);
User.find.all();
```

👉 No Repository / DAO layer required.

🏗️ Architecture of Ebean

```
pgsql

Java Class
   ↓
Ebean ORM
   ↓
SQL Generation
   ↓
Database
```

Ebean:

- Generates SQL automatically
- Executes SQL
- Maps result back to Java objects

🧩 Key Components of Ebean

1️⃣ Entity Class

A normal Java class mapped to a DB table.

```
java

@Entity
public class User {
    @Id
    Long id;

    String name;
}
```

2️⃣ Finder (Most Important Concept)

Ebean provides a **Finder** object for querying.

```
java

public static final Finder<Long, User> find =
        new Finder<>(User.class);
```

Now you can write:

```java
User user = User.find.byId(1);
```

📌 This is why Ebean is simple.

3️⃣ Database Object

Ebean uses a **`Database`** object internally.

```java
Database db = DB.getDefault();
```

This object:

- Executes queries
- Manages transactions
- Handles caching

🔄 How Ebean Works Internally (Step-by-Step)

Let's say you write:

```java
User user = User.find.byId(1);
```

Internally:

1. Ebean reads metadata from **`@Entity`**
2. Generates SQL:

   ```sql
   SELECT * FROM user WHERE id = 1
   ```

3. Executes SQL
4. Converts row → **`User`** object
5. Returns Java object

👉 **You never touch SQL unless you want to**

🧠 Annotations in Ebean

Ebean uses **JPA-style annotations**, but fewer.

Common Annotations

Annotation	Purpose
`@Entity`	Marks class as DB table
`@Id`	Primary key
`@Column`	Column mapping
`@ManyToOne`	Relationship
`@OneToMany`	Relationship
`@Version`	Optimistic locking

Example:

```java
@Entity
public class Order {
    @Id
    Long id;

    @ManyToOne
    Customer customer;
}
```

🔗 Relationships in Ebean

One-to-Many

```java
@OneToMany(mappedBy = "user")
List<Order> orders;
```

Many-to-One

```java
@ManyToOne
User user;
```

✔️ Same as JPA
✔️ But easier query handling

🔍 Querying in Ebean (Very Powerful)

Find All

```java
List<User> users = User.find.all();
```

Find by ID

```java
User user = User.find.byId(1);
```

Conditional Query

```java
List<User> users = User.find
        .query()
        .where()
        .eq("name", "Piyush")
        .findList();
```

✔️ Type-safe
✔️ SQL-like
✔️ Clean syntax

🔪 Raw SQL Support (Important Feature)

Ebean does NOT hide SQL from you.

```java
SqlQuery query = DB.sqlQuery(
    "select * from user where name = :name"
);
query.setParameter("name", "Piyush");
```

👉 Best of **ORM + SQL**

🔁 Transactions in Ebean

Automatic Transaction

```java
User user = new User();
user.save();
```

Ebean:

- Starts transaction
- Commits

- Closes automatically

Manual Transaction

```
java

try (Transaction txn = DB.beginTransaction()) {
    user.save();
    txn.commit();
}
```

✔️ Clean
✔️ Safe
✔️ Simple

🚀 Ebean with Spring Boot

Ebean integrates with **Spring Boot**.

Benefits:

- Auto configuration
- No repository layer
- Faster development

📌 Many developers prefer Ebean over Spring Data JPA for simplicity.

🆚 Ebean vs Hibernate / JPA

Feature	Ebean	Hibernate
Learning Curve	Easy	Steep
Boilerplate	Very Low	High
Active Record	✅ Yes	❌ No
Repository Required	❌ No	✅ Yes
SQL Control	Excellent	Limited
Performance	Fast	Good but heavy

👉 **Hibernate = Powerful but complex**
👉 **Ebean = Simple and practical**

🧠 When Should You Use Ebean?

Use Ebean if:

- You want **simple ORM**
- You hate boilerplate
- You want SQL control
- You want faster development
- You prefer Active Record style

Avoid Ebean if:

- Project strictly follows JPA standard
- Enterprise legacy systems
- Need advanced JPA features

📌 Summary (Very Important)

✔️ Ebean is a **lightweight Java ORM**
✔️ Uses **Active Record Pattern**
✔️ Minimal configuration
✔️ Clean queries
✔️ SQL-friendly
✔️ Faster learning than Hibernate

If you want, next I can explain:
1️⃣ **Ebean vs Spring Data JPA (deep comparison)**
2️⃣ **Ebean full project setup step-by-step**
3️⃣ **How Ebean handles lazy loading internally**
4️⃣ **Ebean caching & performance tuning**

Just tell me 👍