**What is Criteria API in Hibernate?**

In **Hibernate**, the **Criteria API** is a **programmatic, object-oriented way** to construct queries without writing raw HQL (Hibernate Query Language) or SQL. Instead of writing queries as strings, you **build them using Java objects**, which improves **type safety**, **readability**, and **dynamic query generation**.

**✅ Why Use Criteria in Hibernate?**

| **Reason** | **Explanation** |
| --- | --- |
| 🔁 **Dynamic Query Generation** | Criteria is useful when the query conditions are not fixed and need to be built dynamically at runtime. |
| ✅ **Type Safety** | Queries are constructed with Java code, reducing the chances of syntax errors at runtime. |
| ⚙️ **No String-Based HQL** | Eliminates the need to write HQL/SQL strings manually. |
| 👓 **Readable & Maintainable** | Criteria-based code is often easier to read and refactor compared to large HQL strings. |
| 📦 **Supports Joins, Projections, Sorting, Pagination** | You can easily perform joins, select specific columns, sort, and paginate. |

**✅ Types of Criteria in Hibernate**

| **Type** | **API** | **Status** |
| --- | --- | --- |
| org.hibernate.Criteria | Classic Hibernate Criteria (Hibernate 3-5) | ❌ Deprecated and removed in Hibernate 6 |
| javax.persistence.criteria.\* or jakarta.persistence.criteria.\* | JPA Criteria API | ✅ Supported and recommended in Hibernate 6+ |

**🔍 Line-by-Line Explanation**

**CriteriaBuilder criteriaBuilder = session.getCriteriaBuilder();**

**CriteriaQuery<User> cq = criteriaBuilder.createQuery(User.class);**

**Root<User> root = cq.from(User.class);**

**cq.select(root); // select \* from user**

Here :

CriteriaBuilder criteriaBuilder = session.getCriteriaBuilder();

* **Purpose**: Gets a CriteriaBuilder instance from the Hibernate Session.
* **Why**: CriteriaBuilder is the starting point for building queries. It provides methods to construct query objects, predicates, expressions, etc.

CriteriaQuery<User> cq = criteriaBuilder.createQuery(User.class);

* **Purpose**: Creates a CriteriaQuery object for the User entity.
* **Why**: This defines the type of result the query will return. In this case, you're querying for User objects.

Root<User> root = cq.from(User.class);

* **Purpose**: Defines the root of the query, which is the User entity.
* **Why**: Think of this as the FROM clause in SQL. It tells Hibernate which table (entity) you're querying.

cq.select(root); // select \* from user

* **Purpose**: Specifies that the query should select all fields from the User entity.
* **Why**: This is equivalent to SELECT \* FROM User in SQL. You're telling Hibernate to fetch all columns for each User.