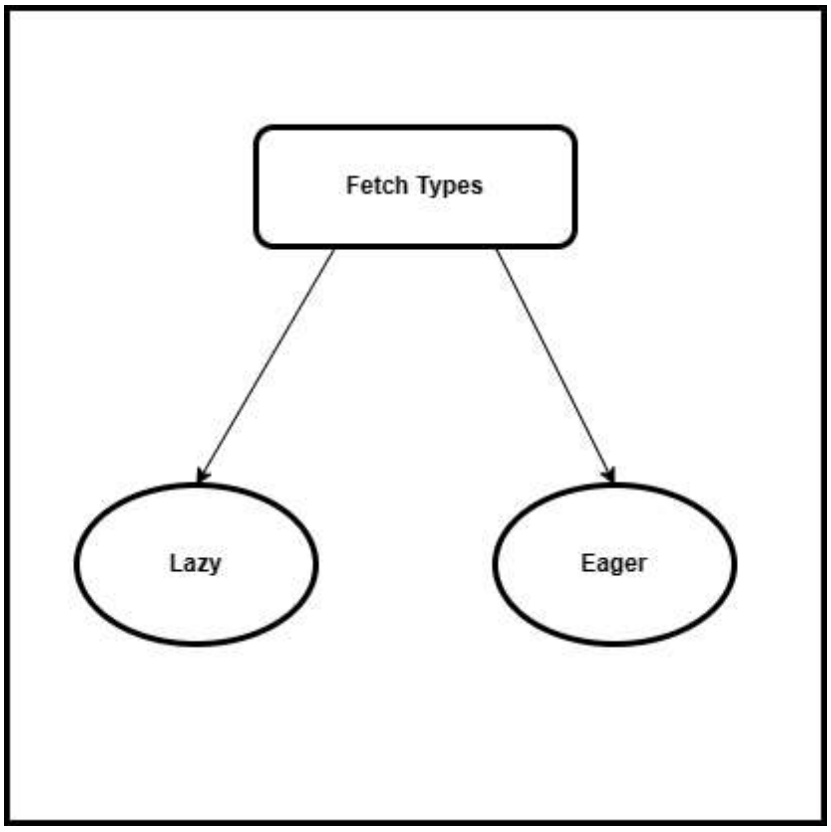
Hibernate - Eager/Lazy Loading

Last Updated : 28 Apr, 2025

FetchType is an enumerated type in the Java Persistence API (JPA) that specifies whether the field or property should be lazily loaded or eagerly loaded. It is used in the ***javax.persistence.FetchType*** enum. In Hibernate, the FetchType is used to specify the fetching strategy to be used for an association. The FetchType can be specified for associations at the time of mapping the association. There are two FetchType options available: **LAZY and EAGER**.



Note: *You can specify the fetch type of an association by using the fetch attribute of the* ***@OneToMany, @ManyToOne, @OneToOne,*** *or* ***@ManyToMany annotations.***

Overview

LAZY:

- This is the default FetchType in Hibernate. It means that the associated entity will be fetched only when it is accessed for the first time. This can improve performance in cases where the associated entity is not required most of the time.
- This can be more efficient than eagerly fetching the entity, especially if the entity has a lot of data and is not needed for every use of the parent entity.
- It's important to note that using **FetchType.LAZY** can result in additional database queries being issued when the associated entity is accessed, so it may not always be the most efficient option. It's a good idea to profile your application to determine the best fetch strategy for your use case.
- **Here is an example of how FetchType.LAZY can be used in a Hibernate @OneToOne mapping:**

```
@Entity
public class Employee {
    @OneToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "address_id")
    private Address address;

    // other fields and methods
}
```

> ***Note:*** *In this example, the Address entity associated with an Employee will be fetched lazily when it is accessed for the first time.*

EAGER:

- This FetchType means that the associated entity will be fetched together with the main entity when the main entity is fetched from the database. This can be useful in cases where the associated entity is always required, but can also result in a performance decrease if the associated entity is large and/or has many associations itself.
- The **FetchType.EAGER** option indicates that the associated entity should be fetched eagerly, which means that it will be fetched at the same time as the parent entity.
- Using **FetchType.EAGER** can be more efficient than using FetchType.LAZY if the associated entity is needed for most uses of the parent entity, as it avoids the need for additional database queries to fetch the associated entity when it is accessed. However, it can also be less efficient if the associated entity has a lot of data and is not needed for every use of the parent entity, as it will always be fetched along with the parent entity. It's a good idea to profile your application to determine the best fetch strategy for your use case.
- **Here is an example of how FetchType.EAGER can be used in a Hibernate @OneToOnemapping:**

```
@Entity
public class Employee {
    @OneToOne(fetch = FetchType.EAGER)
    @JoinColumn(name = "address_id")
    private Address address;

    // other fields and methods
}
```

Note: *In this example, the Address entity associated with an Employee will be fetched eagerly when the Employee is loaded from the database.*

Summary

- As I mentioned before, you must select the appropriate fetch type for your use case in order to prevent frequent Hibernate performance difficulties. The bulk of use cases calls for the fetch type. A good option is lazy.
- Let's quickly review the various fetch types. Hibernate is told to get the entities related to the initial query by EAGER fetching. Because only one query is required to retrieve all entities, this can be incredibly effective. However, most of the time you choose entities that are unnecessary for your use case, which results in significant overhead.
- This is something that **FetchType.LAZY** may stop. This instructs Hibernate to postpone initializing the relationship until your business code has access to it. The negative aspect of this strategy.