Hibernate - @Embeddable and @Embedded Annotation

Last Updated : 28 Apr, 2025

The @Embeddable and @Embedded annotations in Hibernate are used to map an object’s properties to columns in a database table. These annotations are used in combination to allow the properties of one class to be included as a value type in another class and then be persisted in the database as part of the containing class.

Overview

- The @Embeddable annotation is used to mark a class as being embeddable, meaning its properties can be included in another class as a value type. The class marked with @Embeddable is called the embeddable class.
- The @Embedded annotation is used to mark a field in a class as being an embeddable object, and it is used in the class that contains the embeddable object.
- By using these annotations, Hibernate can automatically persist the properties of the embeddable class within the containing class to the database table, without the need to create a separate table for the embeddable class.
- Using @Embeddable and @Embedded annotations in Hibernate allows for better data modeling, code reusability, normalization, and better performance. The annotations also allow to encapsulation of the business logic within the embeddable class.

Benefits

The main benefits of using the @Embeddable and @Embedded annotations in Hibernate are:

1. **Code Reusability:** You can reuse the embeddable class in multiple entities, avoiding duplication of code.
2. **Normalization:** It helps in normalizing the database by reducing the number of tables, which in turn improves the performance.
3. **Data Integrity:** It ensures data integrity by maintaining the relationship between the embeddable and the containing class.
4. **Simplicity:** It simplifies the development process by reducing the number of classes and tables required to map the data.
5. **Better Data Modelling:** It allows for better data modeling by allowing you to encapsulate the properties of an object within another object, making the data structure more intuitive and easy to understand.
6. **Ease of maintenance:** it makes the maintenance of the codebase more manageable and easy.
7. **Flexibility:** The embeddable classes can be used in multiple entities, and it also supports complex data modeling with the use of @Embedded and @AttributeOverrides.
8. **Readability:** The use of @Embeddable and @Embedded annotations makes the code more readable and self-explanatory.
9. **Performance:** it improves the performance of the application by reducing the number of joins required to fetch the data.
10. **Business logic:** It allows to encapsulation of the business logic within the embeddable class, making it more manageable and easy to understand.

Example

Embeddable Class:

```
@Embeddable
public class Address {
    private String street;
    private String city;
    private String state;
    private String zip;
    // getters and setters
}
```

Entity Class:

```
@Entity
public class Employee {
    @Id
    private int id;
    private String name;
    @Embedded
    private Address address;
    // getters and setters
}
```

- **In this example**, the Address class is marked as @Embeddable, which means it can be included as a value type in another class. The Employee class has an Address field which is marked with the @Embedded annotation. This tells Hibernate that the Address object is an embeddable object and its properties should be mapped to columns in the same table as the Employee class.
- When the Employee object is saved to the database, the properties of the embedded Address object will also be saved to the same table, with the column names being prefixed with the name of the field in the Employee class (in this case, "address_").

> ***Note:*** *You can also use the @AttributeOverrides to customize the column name mapping.*

```
@Embedded
@AttributeOverrides({
    @AttributeOverride(name = "street", column = @column(name =
"home_street"))
    })
    private Address address;
```

The above example will map the street property of the Address class to home_street in the database table.

Conclusion

- embeddable and embedded annotations are useful tools in both web development and Hibernate for adding additional information and context to a piece of content or defining the relationship between entities in an ORM configuration. They allow for better organization and reuse of complex value objects, making the code more maintainable and efficient.
- It’s important to understand the difference and use cases of these annotations in order to effectively utilize them in your projects. Whether you are creating an educational website, a research paper, or professional software, embeddable and embedded annotations can help to enhance the user experience and understanding of the content.