

How to use Java 8 Optional with Hibernate

Optional attributes

If you use *Optional<T>* as a data type, Hibernate cannot determine the type of the attribute and throws a *MappingException*.

To avoid this Exception, you have to use field-type access and keep the original data type of the attribute. The field-type access allows you to implement the getter and setter methods in your own way.

You can, for example, implement a *getPublishingDate()* method which wraps the *publishingDate* attribute in an *Optional<LocalDate>*.

```
@Entity
public class Book {

    @Column
    private LocalDate publishingDate;

    ...

    public Optional getPublishingDate() {
        return Optional.ofNullable(publishingDate);
    }

    public void setPublishingDate(LocalDate publishingDate) {
        this.publishingDate = publishingDate;
    }
}
```

How to use Java 8 Optional with Hibernate

Load optional entities

Hibernate 5.2 also introduced the *loadOptional(Serializable id)* method to the *IdentifierLoadAccess* interface which returns an *Optional<T>*. You should use this method to indicate that the result might be empty when you can't be sure that the database contains a record with the provided id.

The *loadOptional(Serializable id)* method is similar to the *load(Serializable id)* method which you already know from older Hibernate versions. It returns the loaded entity or a *null* value if no entity with the given id was found. The new *loadOptional(Serializable id)* method wraps the entity in an *Optional<T>* and therefore indicates the possibility of a *null* value.

```
Session session = em.unwrap(Session.class);
Optional<Book> book = session.byId(Book.class).loadOptional(1L);

if (book.isPresent()) {
    log.info("Found book with id [" + book.get().getId()
        + "] and title [" + book.get().getTitle() + "].");
} else {
    log.info("Book doesn't exist.");
}
```