🔍 What Is JPA Auditing in Spring Boot?

JPA Auditing is a feature in **Spring Data JPA** that automatically tracks and populates **audit-related metadata** such as:

- When an entity was created (`@CreatedDate`)
- Who created it (`@CreatedBy`)
- When it was last modified (`@LastModifiedDate`)
- Who last modified it (`@LastModifiedBy`)

✅ Benefits of JPA Auditing

- **No need to manually set timestamps or user info**
- Clean and consistent audit logic across all entities

By Eslam Khder


```java
@CreatedDate
@Column(updatable = false)
private LocalDateTime createdAt;

@CreatedBy
@Column(updatable = false)
private String createdBy;

@LastModifiedDate
@Column(insertable = false)
private LocalDateTime updatedAt;

@LastModifiedBy
@Column(insertable = false)
private String updatedBy;
```

By Eslam Khder

When Are These Fields Populated?

Annotation	Triggered On	Example Value
`@CreatedDate`	First insert	`2025-06-21T16:00:00`
`@CreatedBy`	First insert	`"admin"`
`@LastModifiedDate`	Every update	`2025-06-21T16:30:00`
`@LastModifiedBy`	Every update	`"eslam.khder"`

By Eslam Khder

spring

EraaSoft

✅ How to Enable JPA Auditing

1. Enable it via Annotation

In a configuration class (e.g., `@SpringBootApplication` or a dedicated `@Configuration`):

```
@EnableJpaAuditing
public class ResturantBackendApplication {
```

By Eslam Khder

spring can create bean with this way

EraaSoft

If you're using `@CreatedBy` or `@LastModifiedBy`, you must also specify:

```java
@Component("auditAwareImpl")
public class AuditAwareImpl implements AuditorAware<String> {

    /**
     * Returns the current auditor of the application.
     *
     * @return the current auditor.
     */
    1 usage
    @Override
    public Optional<String> getCurrentAuditor() {
        return Optional.of( value: "ACCOUNTS_MS");
    }

}
```

By Eslam Khder


```java
@EnableJpaAuditing(auditorAwareRef = "auditAwareImpl")
```

```java
@EntityListeners(AuditingEntityListener.class)
public class Category {
```

By Eslam Khder

all entities has 4 column you must apply inheritance

🔍 What Is `@MappedSuperclass` in JPA?

In **JPA**, `@MappedSuperclass` is used to create a **base class** that defines **common fields** (like `createdAt`, `updatedBy`, `id`, etc.) that can be **inherited** by multiple entities — **but it's not an entity itself**.

✅ Key Points

- It's **not mapped to a table** on its own.
- Fields in the superclass are mapped to **the table of the subclass**.
- It's a way to reuse common mappings (like auditing fields, primary keys, etc.).

By Eslam Khder

By Eslam Khder

By Eslam Khder

By Eslam Khder

By Eslam Khder

By Eslam Khder