Exercise 7: Enabling Entity Auditing

Understanding the Requirements

We'll implement auditing to track changes made to `Employee` and `Department` entities by adding creation and modification timestamps, as well as creator and modifier information.

Configuring Auditing

To enable auditing, we need to configure Spring Data JPA's auditing features.

Properties

```
spring.jpa.auditing.enabled=true
spring.jpa.auditing.is-always-default-auditor-only=true
```

The first property enables auditing, and the second ensures that the current user is always used as the auditor.

Using Auditing Annotations

We'll use the following annotations to track changes:

- `@CreatedBy`: Indicates the user who created the entity.
- `@LastModifiedBy`: Indicates the user who last modified the entity.
- `@CreatedDate`: Indicates the creation timestamp.
- `@LastModifiedDate`: Indicates the last modification timestamp.

Java

```java
import javax.persistence.*;
import java.time.Instant;

@Entity
@EntityListeners(AuditingEntityListener.class)
public class Employee {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private   Long id;

    @CreatedBy
    private String createdBy;

    @LastModifiedBy
    private String lastModifiedBy;

    @CreatedDate
    private Instant createdDate;

    @LastModifiedDate
    private Instant lastModifiedDate;

    // ... other fields
}
```

You'll need to add the `@EntityListeners` annotation to your entities to enable auditing.

Additional Considerations

- **Auditor Provider:** To populate `createdBy` and `lastModifiedBy` fields, you'll need to implement a `AuditorAware` interface to provide the current user information.
- **Custom Auditing Fields:** You can add custom auditing fields by creating a custom auditing entity listener.
- **Performance Impact:** Be aware of the potential performance impact of auditing, especially in high-traffic systems.