Below is a detailed document on implementing auditing with Spring Data Envers, using an Employee table as an example. This document includes design considerations, code snippets, and a step-by-step guide for developers. Since you’ve requested not to use the auto DDL feature, I’ve also included a section for the database administrator to create the necessary tables using DDL scripts for SQL Server.

**Document Title:** Implementing Auditing with Spring Data Envers for Employee Table

**Table of Contents:**

1. **Introduction**
2. **Database Setup (For DB Administrator)**
3. **Project Setup and Dependencies**
4. **Entity and Auditor Configuration**
5. **Envers Configuration**
6. **Usage and Testing**
7. **Generic Steps for Any Table**

### 1. Introduction

**Purpose:** This document outlines the process of implementing auditing for an Employee table using Spring Data Envers. Auditing allows for the tracking of changes to the Employee entity, enabling the retrieval of historical data.

**Technology Stack:**

* **Backend:** SQL Server
* **Framework:** Spring Framework
* **Auditing Tool:** Spring Data Envers

### 2. Database Setup (For DB Administrator)

**Task:** Create the necessary tables for the Employee entity and its audit trail using the provided DDL scripts for SQL Server.

-- Table for storing Employee data  
CREATE TABLE Employees (  
 Id INT PRIMARY KEY IDENTITY(1,1),  
 FirstName VARCHAR(255) NOT NULL,  
 LastName VARCHAR(255) NOT NULL,  
 Email VARCHAR(255) UNIQUE NOT NULL,  
 CreatedBy VARCHAR(255),  
 CreatedDate DATETIME DEFAULT GETDATE(),  
 LastModifiedBy VARCHAR(255),  
 LastModifiedDate DATETIME  
);  
  
-- Tables required by Envers for auditing (prefix with your schema if needed)  
CREATE TABLE revinfo (  
 rev INTEGER PRIMARY KEY,  
 revtstmp BIGINT,  
 username VARCHAR(255)  
);  
  
CREATE TABLE Employees\_Audit (  
 Id INT,  
 rev INTEGER NOT NULL,  
 revtype tinyint,  
 FirstName VARCHAR(255),  
 LastName VARCHAR(255),  
 Email VARCHAR(255),  
 PRIMARY KEY (Id, rev),  
 CONSTRAINT FK\_Employees\_Audit\_revinfo FOREIGN KEY (rev) REFERENCES revinfo (rev)  
);

### 3. Project Setup and Dependencies

**Step 1:** Include the necessary dependencies in your pom.xml (if using Maven) or your build.gradle file (if using Gradle).

**Maven:**

<dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-data-jpa</artifactId>  
</dependency>  
<dependency>  
 <groupId>org.hibernate</groupId>  
 <artifactId>hibernate-envers</artifactId>  
</dependency>  
<dependency>  
 <groupId>com.microsoft.sqlserver</groupId>  
 <artifactId>mssql-jdbc</artifactId>  
</dependency>

**Gradle:**

dependencies {  
 implementation 'org.springframework.boot:spring-boot-starter-data-jpa'  
 implementation 'org.hibernate:hibernate-envers'  
 implementation 'com.microsoft.sqlserver:mssql-jdbc'  
}

### 4. Entity and Auditor Configuration

**Employee Entity:**

@Entity  
@EntityListeners(AuditingEntityListener.class)  
@Table(name = "Employees")  
public class Employee {  
  
 @Id  
 @GeneratedValue(strategy = GenerationType.IDENTITY)  
 private Long id;  
  
 private String firstName;  
 private String lastName;  
 private String email;  
  
 @CreatedBy  
 private String createdBy;  
  
 @CreatedDate  
 private Instant createdDate;  
  
 @LastModifiedBy  
 private String lastModifiedBy;  
  
 @LastModifiedDate  
 private Instant lastModifiedDate;  
  
 // Getters and Setters  
}

**Auditor Configuration (in a @Configuration class or application.properties/yml):**

@Configuration  
@EnableJpaAuditing auditor = "springSecurityAuditor")  
public class AuditingConfig {  
   
 @Bean  
 public AuditorAware<String> springSecurityAuditor() {  
 return () -> Optional.ofNullable(SecurityContextHolder.getContext().getAuthentication().  
 getPrincipal().toString());  
 }  
}

**OR (in application.properties)**:

spring.jpa.properties.hibernate.envers.audit\_strategy = org.springframework.boot.autoconfigure.orm.jpa.hibernate.SpringJUnitAuditingStrategy  
spring.jpa.properties.hibernate.envers.auditor\_provider = org.springframework.security.core.userdetails.UserDetailsAuditorAware

### 5. Envers Configuration

**Step 1:** Activate Envers in your application’s main configuration class or a separate configuration file.

@SpringBootApplication  
@EnableJpaRepositories(basePackages = "com.example.repo") // Ensure your repository package is scanned  
@EnableJpaAuditing  
public class DemoApplication {  
 public static void main(String[] args) {  
 SpringApplication.run(DemoApplication.class, args);  
 }  
}

**Step 2 (Optional but Recommended for Global Configuration):** Define global Envers settings if needed (e.g., revision listeners, custom revision entities). For the Employee example, we’ll stick with the defaults.

### 6. Usage and Testing

* **Insert/Update Operations:** Perform CRUD operations on the Employee entity using your Spring Data JPA repositories. Envers automatically tracks changes.
* **Retrieve Audit History:**
* @Autowired  
  private EmployeeRepository employeeRepository;  
    
  @Autowired  
  private RevisionRepository revisionRepository, Employee.class, Integer>.class>;  
    
  public void testAudit() {  
   // Retrieve all revisions for an employee  
   List<Revision<Integer, Employee>> revisions = revisionRepository.findRevisions(Employee.class, employeeId);  
    
   // Iterate and access revisions  
   for (Revision<Integer, Employee> revision : revisions) {  
   Employee empAtRevision = revision.getEntity();  
   int revisionNumber = revision.getMetadata().getRevisionNumber();  
   AuditRevisionListener.RevisionType revisionType = revision.getMetadata().getRevisionType();  
   // Process or log as needed  
   }  
  }

### 7. Generic Steps for Any Table

1. **Database Setup:**
   * Create the main table for your entity.
   * Create an audit table with a structure mirroring the main table, plus rev (foreign key to revinfo) and revtype columns.
2. **Project Setup:**
   * Include Spring Data JPA and Hibernate Envers dependencies.
3. **Entity Configuration:**
   * Annotate your entity with @Entity and add auditing fields (@CreatedBy, @CreatedDate, etc.) as needed.
4. **Envers Activation:**
   * Enable JPA Auditing and JPA Repositories in your application configuration.
5. **Usage:**
   * Perform CRUD operations; Envers tracks changes.
   * Use RevisionRepository to retrieve audit history.

**DDL Script Template for Any Table (SQL Server):**

-- Main Table  
CREATE TABLE [TableName] (  
 Id INT PRIMARY KEY IDENTITY(1,1),  
 -- Your Columns Here  
 CreatedBy VARCHAR(255),  
 CreatedDate DATETIME DEFAULT GETDATE(),  
 LastModifiedBy VARCHAR(255),  
 LastModifiedDate DATETIME  
);  
  
-- Audit Table  
CREATE TABLE [TableName]\_Audit (  
 Id INT,  
 rev INTEGER NOT NULL,  
 revtype tinyint,  
 -- Mirror columns from the main table  
 PRIMARY KEY (Id, rev),  
 CONSTRAINT FK\_[TableName]\_Audit\_revinfo FOREIGN KEY (rev) REFERENCES revinfo (rev)  
);