User Activity Tracking - Quick Start

🎯What You Get

Track **who** did **what** and **when** in your entire app with just 2 simple steps!

Approach 1: Automatic Auditing ⚡

```
createdBy: "employee1"
createdAt: "2026-02-15 10:30:00"
updatedBy: "admin1"
updatedAt: "2026-02-15 14:45:00"
```

Automatically added to every record - NO code changes needed!

Approach 2: Activity Logging 📝

```
Activity Log:
- 2026-02-15 10:30:00 | employee1 | CREATE | News #123: Breaking News
- 2026-02-15 14:45:00 | admin1    | UPDATE | News #123: Breaking News
- 2026-02-16 09:00:00 | admin1    | DELETE | News #123: Breaking News
```

Complete audit trail of all actions

📦Files to Copy (5 files)

1. **JpaAuditingConfig.java** → `config/`
2. **AuditableEntity.java** → `model/`
3. **ActivityLog.java** → `model/activity/`
4. **ActivityLogRepository.java** → `repository/activity/`
5. **ActivityLogService.java** → `service/activity/`

🚀Setup (3 Steps)

Step 1: Run Database Migration

```sql
-- Add audit columns to all tables
ALTER TABLE news
ADD COLUMN created_by VARCHAR(100),
ADD COLUMN created_at TIMESTAMP,
ADD COLUMN updated_by VARCHAR(100),
ADD COLUMN updated_at TIMESTAMP;

-- Create activity_logs table
CREATE TABLE activity_logs (
    id BIGSERIAL PRIMARY KEY,
    username VARCHAR(100) NOT NULL,
    action VARCHAR(50) NOT NULL,
    entity_type VARCHAR(100) NOT NULL,
    entity_id BIGINT,
    entity_name VARCHAR(500),
    details TEXT,
    performed_at TIMESTAMP NOT NULL
);

-- Run: database_migration_activity_tracking.sql
```

Step 2: Update Your Entities

Before:

```java
@Entity
public class News {
    @Id
    private Long id;
    private String title;
}
```

After:

```java
@Entity
public class News extends AuditableEntity {  // ✅Just add this!
    @Id
    private Long id;
    private String title;

    // createdBy, createdAt, updatedBy, updatedAt
    // are now automatic!
}
```

Apply to all 6 entities:

- ✅News
- ✅Project
- ✅Album
- ✅Film
- ✅ImageCollection
- ✅Soundtrack

Step 3: Add Activity Logging to Controllers

Before:

```java
@RestController
@RequiredArgsConstructor
public class NewsController {
    private final NewsService newsService;

    @PostMapping
    public ResponseEntity<?> create(@RequestBody NewsDto dto) {
        News created = newsService.create(dto);
        return ResponseEntity.ok(created);
    }
}
```

After:

```java
@RestController
@RequiredArgsConstructor
public class NewsController {
    private final NewsService newsService;
    private final ActivityLogService activityLogService; // ✅ Add this

    @PostMapping
    public ResponseEntity<?> create(@RequestBody NewsDto dto) {
        News created = newsService.create(dto);

        // ✅ Add one line to log activity
        activityLogService.logCreate("News", created.getId(), created.getTitle());

        return ResponseEntity.ok(created);
    }
}
```

Apply to all CRUD operations:

```java
// CREATE
activityLogService.logCreate("News", id, title);

// UPDATE
activityLogService.logUpdate("News", id, title);

// DELETE
activityLogService.logDelete("News", id, title);
```

📊 What Gets Tracked

Automatic Auditing (in database columns)

Column	Value	When
created_by	"employee1"	On INSERT
created_at	2026-02-15 10:30:00	On INSERT
updated_by	"admin1"	On UPDATE
updated_at	2026-02-15 14:45:00	On UPDATE

Activity Logs (in activity_logs table)

Field	Example
username	"employee1"
action	"CREATE"
entity_type	"News"
entity_id	123
entity_name	"Breaking News"
details	"Created news article: Breaking News"
ip_address	"192.168.1.100"
http_method	"POST"
endpoint	"/api/v1/news"
performed_at	2026-02-15 10:30:00

🔍Query Examples

Get all activities by user

```java
List<ActivityLog> activities =
    activityLogService.getUserActivities("employee1");
```

Get entity history

```java
List<ActivityLog> history =
    activityLogService.getEntityHistory("News", newsId);

// Result:
// 10:30:00 | employee1 | CREATE | Created news article
// 14:45:00 | admin1    | UPDATE | Updated news article
// 09:00:00 | admin1    | DELETE | Deleted news article
```

Get recent activities

```java
List<ActivityLog> recent =
    activityLogService.getRecentActivities();
```

SQL Query

```sql
-- Get all activities
SELECT
    username,
    action,
    entity_type,
    entity_name,
    performed_at
FROM activity_logs
ORDER BY performed_at DESC
LIMIT 100;

-- Get user activities
SELECT * FROM activity_logs
WHERE username = 'employee1'
ORDER BY performed_at DESC;

-- Get entity history
SELECT * FROM activity_logs
WHERE entity_type = 'News' AND entity_id = 123
ORDER BY performed_at DESC;
```

✅Benefits

Automatic Auditing

✅ **Zero effort** - No code changes in controllers
✅ **Always accurate** - Can't forget to update
✅ **Built into entity** - `news.getCreatedBy()`

Activity Logging

✅ **Complete history** - Every action tracked
✅ **Searchable** - Query by user, date, entity, action
✅ **IP tracking** - Know where actions came from
✅ **Async** - Doesn't slow down requests

📋 Checklist

Database

- [ ] Run migration script
- [ ] Add audit columns to all tables
- [ ] Create activity_logs table
- [ ] Create indexes

Code

- [ ] Copy 5 Java files
- [ ] Make all 6 entities extend AuditableEntity
- [ ] Add ActivityLogService to all 6 controllers
- [ ] Add logCreate/Update/Delete calls in all CRUD methods

Testing

- [ ] Create a news article (check created_by, created_at)
- [ ] Update a project (check updated_by, updated_at)
- [ ] Delete an album (check activity_logs table)
- [ ] Query activities: `SELECT * FROM activity_logs ORDER BY performed_at DESC`

🎯 Example Result

After setup, when `employee1` creates a news article:

Database (news table):

```
id: 123
title: "Breaking News"
created_by: "employee1"          ← Automatic!
created_at: 2026-02-15 10:30:00  ← Automatic!
updated_by: null
updated_at: null
```

Activity Log (activity_logs table):

```
id: 456
username: "employee1"
action: "CREATE"
entity_type: "News"
entity_id: 123
entity_name: "Breaking News"
details: "Created News: Breaking News"
ip_address: "192.168.1.100"
http_method: "POST"
endpoint: "/api/v1/news"
performed_at: 2026-02-15 10:30:00
```

📚Files Reference

- **ACTIVITY_TRACKING_GUIDE.md** - Complete guide with all details
- **database_migration_activity_tracking.sql** - Database migration script
- **JpaAuditingConfig.java** - Automatic auditing configuration
- **AuditableEntity.java** - Base class for entities
- **ActivityLog.java** - Activity log entity
- **ActivityLogRepository.java** - Repository for queries
- **ActivityLogService.java** - Service with simple methods

That's it! Your app now tracks everything automatically. 📊🔍