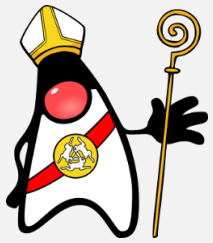
Wer hat was wann geändert? Einfache Auditierung mit Envers

Independent consultant,
trainer and author

Thorben Janssen

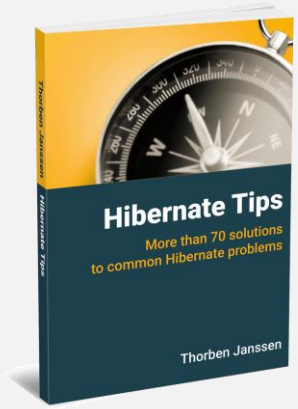


CDI 2.0 Expert
Group Member

Co-Organizer
JUG Paderborn

@thjanssen123
/c/ThoughtsOnJava
/ThoughtsOnJava

www.thoughts-on-java.org



Hibernate Tips
More than 70 solutions to common Hibernate problems
www.hibernate-tips.com

Audit Log?

- Definition Audit Log

> “An audit trail (also called audit log) is a security-relevant chronological record, set of records, and/or destination and source of records that provide documentary evidence of the sequence of activities that have affected at any time a specific operation, procedure, or event.”
>
> Wikipedia (https://en.wikipedia.org/wiki/Audit_trail)

- What it means:
  - Documents all changes in your database

Hibernate Envers

- Integrates with Hibernate ORM
- Automatically writes the audit log
  - Writes an audit record when an entity gets created, updated and deleted
  - Uses lifecycle listener
- Powerful query API

Setup Hibernate Envers

Dependencies

- Maven dependency

```
<dependency>
        <groupId>org.hibernate</groupId>
        <artifactId>hibernate-envers</artifactId>
</dependency>
```

  - Requires Hibernate ORM

Database Tables

- Required tables
  - Revision table
    - Default name: REVINFO
  - An audit table for each audited entity

Revision Table

- Revision Table

```
CREATE TABLE revinfo
(
        rev integer NOT NULL,
        revtstmp bigint,
        CONSTRAINT revinfo_pkey PRIMARY KEY (rev)
)
```

Audit table

- One for each audited entity
  - Default name: <name of audited table>_AUD
    - Configure custom name with *@AuditTable* annotation
  - Columns
    - Primary key
    - Audited fields
    - Revision number and type

Audit table

- Audit table of *Author* entity

```
CREATE TABLE author_aud
(
    id bigint NOT NULL,
    rev integer NOT NULL,
    revtype smallint,
    firstname character varying(255),
    lastname character varying(255),
    CONSTRAINT author_aud_pkey PRIMARY KEY (id, rev),
    CONSTRAINT author_aud_revinfo FOREIGN KEY (rev)
    REFERENCES revinfo (rev) MATCH SIMPLE
    ON UPDATE NO ACTION ON DELETE NO ACTION
)
```

Audit entities

Audit an entity

- Annotate entity or attribute with @Audited

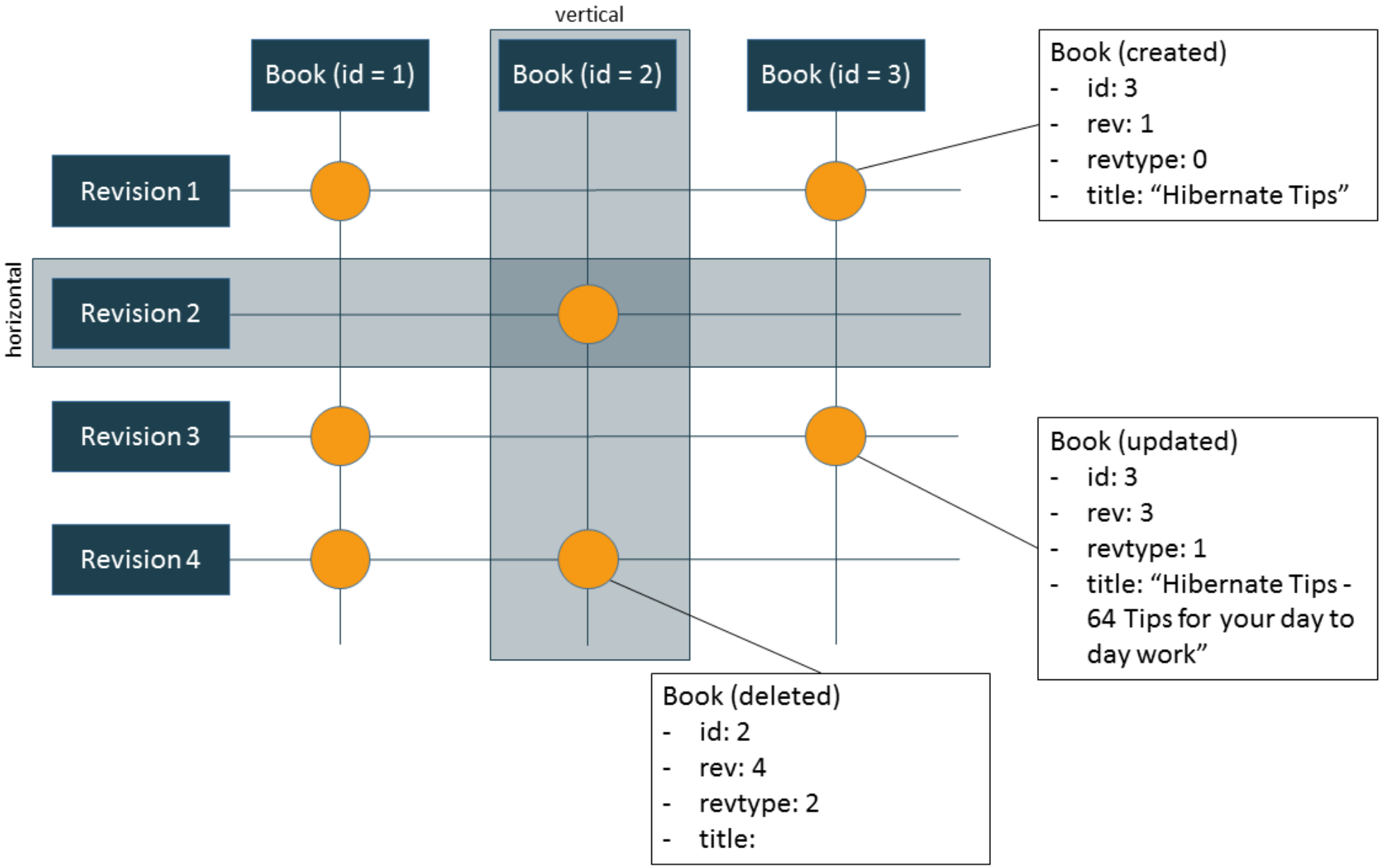
```
@Entity
@Audited
public class Author { … }
```

- No additional configuration required!

Demo

Read audit information

2 Dimensions



Vertical Queries

- Retrieves the revisions of an entity
  - *forRevisionsOfEntity*(*Class* c, *boolean* selectedEntitiesOnly, *boolean* selectDeletedEntities)

```
AuditQuery q = auditReader.createQuery()
                                .forRevisionsOfEntity(Book.class, true, true);
q.add(AuditEntity.id().eq(b.getId()));
List<Book> audit = q.getResultList();
```

Demo

Horizontal Queries

- Retrieves the entities of a revision
  - *forEntitiesAtRevision*(*Class*<?> c, *Number* revision)

```
AuditQuery q = auditReader.createQuery().forEntitiesAtRevision(Book.class, 2);
q.add(AuditEntity.property("title").ilike("Hibernate", MatchMode.ANYWHERE));
q.addOrder(AuditEntity.property("title").asc());
List<Book> audit = q.getResultList();
```

Demo

Audit Strategies

Audit Strategies

- 2 Strategies
  - Default audit strategy
    - Persists audit data with a start revision
  - Validity audit strategy
    - Persist audit data with a start and end revision

Audit Strategy

- Defined by *org.hibernate.envers.audit_strategy*
  - *org.hibernate.envers.strategy.DefaultAuditStrategy*
  - *org.hibernate.envers.strategy.ValidityAuditStrategy*

```
<persistence>
   <persistence-unit name="my-persistence-unit">
      <properties>
         …

         <property name="org.hibernate.envers.audit_strategy"
                   value="org.hibernate.envers.strategy.ValidityAuditStrategy" />
```

Demo

Custom Revisions

Additional Information

- Only minimal set of revision information
- Often required information
  - User name
  - IP address

Custom Revision

- Requires 2 classes
  - A revision entity annotated with *@RevisionEntity*
  - Implementation of *RevisionListener*

Revision Entity

- Required attributes
  - Revision number
    - Of type *int/Integer* or *long/Long*
    - Annotated with *@RevisionNumber*
  - Revision timestamp
    - Of type *long/Long* or *java.util.Date*
    - Annotated with *@RevisionTimestamp*

Revision Entity

- Easiest option: extend *DefaultRevisionEntity*
  - Good approach to add a few attributes

```
@Entity
@RevisionEntity(MyRevisionListener.class)
public class MyRevision extends DefaultRevisionEntity {

        private String userName;

        ...
}
```

RevisionListener

- Sets all additional revision attributes

```
public class MyRevisionListener implements RevisionListener {

        @Override
        public void newRevision(Object revisionEntity) {
                MyRevision rev = (MyRevision) revisionEntity;
                rev.setUserName(getUserName());
        }

        ...
}
```

Revision Queries

- Use it in queries as any other audited attributes

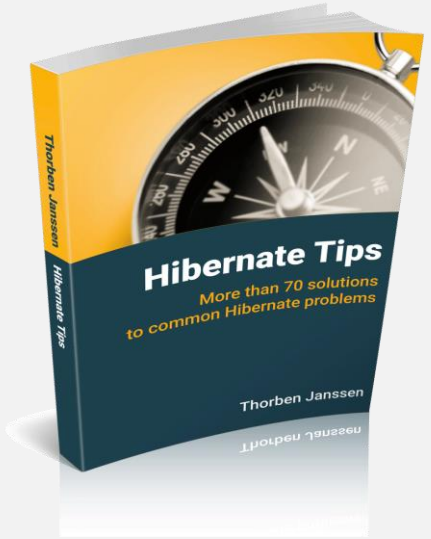
```
AuditQuery q = auditReader.createQuery()
                                .forRevisionsOfEntity(Book.class, false, true);
q.addProjection(AuditEntity.revisionNumber());
q.add(AuditEntity.revisionProperty("userName").eq("User 1"));
List<Number> revisionNumbers = q.getResultList();
```

Demo

Tutorials and Online Courses

www.thoughts-on-java.org

More Hibernate Tips



Hibernate Tips
More than 70 solutions to common Hibernate problems

www.hibernate-tips.com