## Project Setup

1. Add Hibernate Envers to the classpath of your application.

```
<dependencies>
    <dependency>
        <groupId>org.hibernate</groupId>
        <artifactId>hibernate-envers</artifactId>
        <version>${hibernate.version}</version>
    </dependency>
</dependencies>
```

2. Setup audit tables

   The *REVINFO* table stores the revision information. By default, Hibernate persists only the revision number as an integer and the creation timestamp as a long.

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

   You need to create an audit table for each entity you want to audit. By default, Hibernate adds the "*_AUD*" suffix to the table name of the audited entity. You can define a different table name with the @AuditTable annotation or by configuring a different prefix or suffix in the configuration.

   Each audit table contains the primary key of the original entity, all audited fields, the revision number and the revision type. The revision number has to match a record in the revision table and is used together with the id column to create a combined

primary key. The revision type persists the type of operation that was performed on the entity in the given revision. Envers uses the integer values 0, 1 and 2 to store that an entity was added, updated or deleted.

```
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 an entity

When you annotate an entity with @Audited, Hibernate Envers creates a new revision for each transaction and documents all changes to the entity in the audit tables.

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

## Retrieve basic audit information

Hibernate Envers provides an extensive Query API which you can use to extract the required information from your audit log.

### Get all revisions of an entity

The first thing you need to do to access your audit information is to create an AuditReader via the AuditReaderFactory. You can see an example of it in the first line of the following code snippet. I call the get method of the AuditReaderFactor with the current instance of the EntityManager.

The getRevisions method of the AuditReader returns all revision numbers of a given entity. You can use these numbers to get an entity with all the attribute values it had at a given revision. I do that in the 5th line of the code snippet. I iterate through the List of revision numbers and call the find method for each of them to get the Book entity that was active at the given revision.

```
AuditReader auditReader = AuditReaderFactory.get(em);

List revisionNumbers = auditReader
        .getRevisions(Book.class, b.getId());

for (Number rev : revisionNumbers) {
        Book auditedBook = auditReader.
                find(Book.class, b.getId(), rev);
        log.info("Book ["+auditedBook+"] at revision ["+rev+"].");
}
```

### Get active revision at a given date

If you just want to get an entity that was active at a given time, you can call the find method of the AuditReader and provide a java.util.Date instead of a revision number. You can see an example of it in the following code snippet.

```
AuditReader auditReader = AuditReaderFactory.get(em);


Book auditedBook = auditReader
        .find(Book.class, b.getId(), created);
log.info("Book ["+auditedBook+"] at ["+created+"].");
```