Hibernate Envers

Docs

https://docs.jboss.org/hibernate/orm/4.2/devguide/en-US/html/ch15.html

Goal

- Historical versioning of your applications
- Similar to version control management tools (SVN, Git)
- It works with the revision of the data.
- It uses revision numbers.
- Each transaction relates to a revision.
- Revision number has a date.

Basics

Setting up

1. hibernate-envers.jar on the classpath.
2. @Audited annotation on the entity.
3. Audit DB tables
    a. DDL: hibernate.hbm2ddl.auto = create|create-drop|update
    b. Programmatic: org.hibernate.tool.EnversSchemaGenerator
    c. Buid tool: ANT task

Making an entity auditable (@Audited)

- https://docs.jboss.org/hibernate/orm/4.2/javadocs/org/hibernate/envers/Audited.html
- Retention: type, method, field
- When applied to a class -> all of its properties should be audited.
- When applied to a field -> this field should be audited.
- Elements:
    - auditParents: Class[]
        - Deprecated. Use @AuditOverride(forClass=YourClass.class)
    - targetAuditMode: RelationTargetAuditMode
        - Specifies if the entity that is the target of the relation should be audited or not.
        - If not, then when reading a historic version an audited entity, the relation will always point to the "current" entity.
        - Useful for dictionary like entities.
        - https://docs.jboss.org/hibernate/orm/4.2/javadocs/org/hibernate/envers/RelationTargetAuditMode.html
    - withModifiedFlag: boolean
        - Should a modification flag be stored for each property in the annotated class or for the annotated property.
        - The flag stores information if a property has been changed at a given revision. This can be used for example in queries.

Refining audit mappings

Audit table names

- By default the audit table names generated from the audited tables.
    - Use the org.hibernate.envers.audit_table_prefix system property (Empty by default)
    - Use the org.hibernate.envers.audit_table_suffix system property (Defaulted to _AUD)
- Set audit table name with the @AuditTable
    - https://docs.jboss.org/hibernate/orm/4.2/javadocs/org/hibernate/envers/AuditTable.html

    - You can define the schema/catalog/table-name
- It is easyer to use the audited table name and some prefix/postfix

Audit table names for secondary tables

- The default name generation still applies.
- Override it with the `@SecondaryAuditTable and with the @SecondaryAuditTables`
    - https://docs.jboss.org/hibernate/orm/4.2/javadocs/org/hibernate/envers/SecondaryAuditTable.html
    - `https://docs.jboss.org/hibernate/orm/4.2/javadocs/org/hibernate/envers/SecondaryAuditTables.html`

Inherited attributes / Empedded components

- Overriding auditing behaviour fields/properties inherited from @MappedSuperclass / in an Embedded component
- Apply @AuditOverride on the subclass.
- https://docs.jboss.org/hibernate/orm/4.2/javadocs/org/hibernate/envers/AuditOverride.html
- Properties
    - forClass: Class which audit mappins is being overridden.
    - name: name of the field to audit override. Empty means all fields.
    - isAudited: true -> audit, false -> do not audit
    - auditJoinTable: New AuditJoinTable used for this field (or property).
- Example:

```
@Entity
@Audited
@AuditOverrides({
  @AuditOverride(forClass = BaseEntity.class, name = "prop1", isAudited = false),
  @AuditOverride(forClass = BaseEntity.class, name = "prop2", isAudited = true)
})
public class DescendantEntity extends BaseEntity
{ ... }
```

Handling @OneToMany+@JoinColumn

- Unidirectional one-to-many association using a foreign key mapping

```
@Enity
public class Customer implements Serializable
{
 ...
 @OneToMany
 @JoinColumn(name="CUST_ID") // join column is in table for Order
 public Set<Order> getOrders() {return orders;}
 ...
}
```

- Hibernate will not generate a join table in that case.
- Envers require a join table.
- Use the @AuditJoinTable annotation to resolve that contradiction.

```
@Enity
@Audited
public class Customer implements Serializable
{
 ...
 @OneToMany
 @JoinColumn(name="CUST_ID") // join column is in table for Order
 @AuditJoinTable(name="CustomerOrders_AUD")
 public Set<Order> getOrders() {return orders;}
 ...
}
```

- https://docs.jboss.org/hibernate/orm/4.2/javadocs/org/hibernate/envers/AuditJoinTable.html

@OneToMany+@JoinColumn and ManyToOne+@JoinColumn(insertable=false,updatable=false)

- @OneToMany + @JoinColumn on the one side
- @ManyToOne+@JoinColumn(insertable=false, updatable=false) on the many side.
- This relationship is
  - bidirectional
  - the owning side is the collection
- Use the @AuditMappedBy to specify a reverse property using the

Audit Strategy

- It defines the process of the audit system.
- Set it via the org.hibernate.envers.audit_strategy property.
- Possible values are the fully qualified class names of classes that implement the org.hibernate.envers.strategy.AuditStrategy
- https://docs.jboss.org/hibernate/orm/4.2/javadocs/org/hibernate/envers/strategy/AuditStrategy.html
- Implementations:
  - DefaultAuditStrategy
  - ValidityAuditStrategy
  - ValidTimeAuditStrategy
- The chosen implementation effects the performance of the application.

DefaultAuditStrategy

- https://docs.jboss.org/hibernate/orm/4.2/javadocs/org/hibernate/envers/strategy/DefaultAuditStrategy.html
- It is the default strategy.
- Quick write, slow read.
- It persists the audit data with the revision start.
- For each row inserted, updated or deleted in an audited table, one or more rows are inserted in the audit tables, together with the start revision of its validity.
- Queries of audit information use subqueries to select the applicable rows in the audit tables.
- The audit data is hard to index. -> Slow read.

ValidityAuditStrategy

- https://docs.jboss.org/hibernate/orm/4.2/javadocs/org/hibernate/envers/strategy/ValidityAuditStrategy.html
- Slower write, quicker read than the DefaultAuditStrategy.
- Persists the audit data with the start revision.
- For each row inserted, updated or deleted in an audited table, one or more rows are inserted in the audit tables, together with the start revision of its validity.
- At the same time the end-revision field of the previous audit rows (if available) are set to this revision.
- Audit information can be queried between start and end date instead of subqueries. -> Easier indexing, no subselects

ValidTimeAuditStrategy

- https://docs.jboss.org/hibernate/orm/4.2/javadocs/org/hibernate/envers/strategy/ValidTimeAuditStrategy.html
- Deprecated. Do not use it.

Revision Log

Default revision entity:

- Every revision is represented by a org.hibernate.envers.DefaultRevisionEntity object.
- https://docs.jboss.org/hibernate/orm/4.2/javadocs/org/hibernate/envers/DefaultRevisionEntity.html
- Attributes:
  - id
    - The revision number
  - revisionDate
    - the instant when the revision is made.
    - It can be long/Long or java.util.Date.
- It is mapped to the `REVINFO` table.

Custom revision entity:

- Annotate your revision entity with the `@org.hibernate.envers.RevisionEntity` annotation.
- Annotate the revision number attribute with the `@org.hibernate.envers.RevisionNumber`
- Annotate the revision timestamp attribute with the `@org.hibernate.envers.RevisionTimestamp`
- `Tip: extemd tje org.hibernate.envers.DefaultRevisionEntity`
- Create your own RevisionListener implementation,
- Define you custom RevisionListener on the @RevisionEntity as it's value.
- Alternatively set the org.hibernate.envers.revision_listener property to it's fully qualified name.
- Example:

```
@Entity
@RevisionEntity( MyCustomRevisionListener.class )
public class MyCustomRevisionEntity
{
   ...
}

public class MyCustomRevisionListener implements RevisionListener
{
   public void newRevision(Object revisionEntity)
 {
      ...( (MyCustomRevisionEntity) revisionEntity )...;
   }
}
```

Tracking entity names modified during revisions:

- It is not enabled by default. -> Need to query all tables.
- Envers enables us to track entity names modified during revisions in the REVCHANGES table.
- The REVCHANGES contains a name column and a FK to the REVINFO table.
- Enabling modified entity name tracking:
  - Set org.hibernate.envers.track_entities_changed_in_revision parameter to true. In this case org.hibernate.envers.DefaultTrackingModifiedEntitiesRevisionEntity will be implicitly used as the revision log entity.
  - Create a custom revision entity that extends org.hibernate.envers.DefaultTrackingModifiedEntitiesRevisionEntity class.

```
@Entity
@RevisionEntity
public class ExtendedRevisionEntity
        extends DefaultTrackingModifiedEntitiesRevisionEntity
{
   ...
}
```

- Mark an appropriate field of a custom revision entity with @org.hibernate.envers.ModifiedEntityNames annotation. The property is required to be of Set<String> type.

```
@Entity
@RevisionEntity
public class AnnotatedTrackingRevisionEntity
{
   ...

   @ElementCollection
   @JoinTable(name = "REVCHANGES", joinColumns = @JoinColumn(name = "REV"))
   @Column(name = "ENTITYNAME")
   @ModifiedEntityNames
   private Set<String> modifiedEntityNames;

   ...
}
```

Custom revision mechanism

- Implement a custom org.hibernate.envers.EntityTrackingRevisionListener and use it as a value of the @org.hibernate.envers.RevisionEntity
- Override the entityChanged() method

Tracking entity changes at property level (Modification steps)

- By default Envers only stores the revisions of the modified entities.
- Modification flags enable Envers to track which properties of the audited entities changed.
- This can be enabled
  - Globally by setting the org.hibernate.envers.global_with_modified_flag to true.
  - Using the @Audited(withModifiedFlag=true) on a property or on an entity.
- Consequences:
  - Bigger tables.
  - Negligible write time.
- Use it sparingly.

Reading audit data

- The audit (history) of an entity can be accessed using the `AuditReader` interface, which can be obtained having an open `EntityManager` or `Session` via the `AuditReaderFactory`. See the javadocs for these classes for details on the functionality offered.
- Two dimensions:
  - State of the DB at a given revision (horizontal)
  - Changes in a given revision (vertical)

Envers queries

- The speed depends on the audit strategy
- The API resembles to the Hibernate Criteria API

- Entry point:
  - AuditReaderFactory#get(Session): AuditReader
  - https://docs.jboss.org/hibernate/orm/4.2/javadocs/org/hibernate/envers/AuditReaderFactory.html#get(org.hibernate.Session)
- Use the createQuery() method of the AuditReader to create and AuditQueryCreator.

Querying for entities of a class at a given revision

- AuditQueryCreator#forEntitiesAtRevision()
- Example:

```
AuditReaderFactory.get( em ).createQuery().forEntitiesAtRevision( Email.class, 1);
```

Querying for revisions, at which entities of a given class changed

- AuditQueryCreator#forRevisionsOfEntity
- Example:

```
AuditReaderFactory.get( em ).createQuery()
      .forRevisionsOfEntity( Email.class, false, true);
```

Querying for revisions of entity that modified given property

- AuditQueryCreator#forRevisionsOfEntity
- Use the AuditQuery#Add(AuditEntity.property("your prop").hasChanged())
- Example

```
AuditReaderFactory.get( em ).createQuery()
      .forRevisionsOfEntity( Email.class, false, true)
      .add(AuditEntity.property("email").hasChanged());
```

Querying for entities modified in a given revision

- AuditReader.getCrossTypeRevisionChangesReader().findEntityTypes(revisionNumber);
- CrossTypeRevisionChangesReader
  - https://docs.jboss.org/hibernate/orm/4.2/javadocs/org/hibernate/envers/CrossTypeRevisionChangesReader.html
  - Queries that allow retrieving snapshots of all entities (regardless of their particular type) changed in the given revision.
  - Note that this API can be legally used only when default mechanism of tracking modified entity names is enabled.

Configuration

- https://docs.jboss.org/hibernate/orm/4.2/devguide/en-US/html/ch15.html#d5e4079

Source

- https://newcircle.com/s/post/115/easy_auditing_versioning_for_your_hibernate_entities_with_envers
- https://github.com/hibernate/hibernate-orm/blob/master/hibernate-envers/src/test/java/org/hibernate/envers/test/integration/superclass/auditoverride/PropertyOverrideEntity.java
- http://stackoverflow.com/questions/20269600/jpa-hibernate-and-auditjointable