Hibernate Search 6.2.4.Final

Migration Guide from 6.1

2024-04-10

Table of Contents

Introduction

The aim of this guide is to assist you migrating an existing application using any version `6.1.x` of Hibernate Search to the latest of the `6.2.x` series.

If you think something is missing or something does not work, please contact us.

If you're looking to migrate from an earlier version, you should migrate step-by-step, from one minor version to the next, following the migration guide of each version.

To Hibernate Search 5 users

Be aware that a lot of APIs have changed since Hibernate Search 5, some only because of a package change, others because of more fundamental changes (like moving away from using Lucene types in Hibernate Search APIs).

When migrating from Hibernate Search 5, you are encouraged to migrate first to Hibernate Search 6.0 using the 6.0 migration guide, and only then to later versions (which will be significantly easier).

Requirements

Hibernate Search's requirements did not change in version 6.2.4.Final.

Data format and schema changes

Indexes

Indexes created with Hibernate Search 6.1 can be read from and written to with Hibernate Search 6.2.4.Final.

If your Hibernate Search mapping includes `GeoPoint` fields that are using the default value for the `projectable` option, and are using either the default value or `Sortable.NO` for the `sortable` option, Elasticsearch schema validation will fail on startup because of missing docvalues on those fields. To address that, either:

- Revert to the previous defaults by adding `projectable = Projectable.NO` to the mapping annotation of relevant `GeoPoint` fields.
- Or recreate your Elasticsearch indexes and reindex your database. The easiest way to do so is to use the `MassIndexer` with `dropAndCreateSchemaOnStart(true)`.

Outbox polling system tables

If you use the incubating `outbox-polling` coordination strategy, you will be impacted by changes to the entities that represents the outbox event and agent, requiring database schema changes. For the outbox event table the column `processAfter` is now non-nullable, the `id` column changes its type from integer to varchar, except MSSQL where the `id` type becomes binary. For the agent table `id` column changes its type from integer to varchar, except MSSQL where the `id` type becomes binary. You can find suggested migration scripts for the tested databases below:

Postgresql:

```
-- adjust `processAfter` values for existing events so that older events (i.e. ones with
smaller ID) will have older timestamp:
update hsearch_outbox_event
set processAfter = now() - make_interval(secs => (select max(id) from hsearch_outbox_event) -
id)
where processAfter is null;
alter table hsearch_outbox_event
    alter column processAfter set not null;

-- change outbox event `id` column type to varchar and generate uuids to replace previous int
values:
alter table hsearch_outbox_event
    alter column id TYPE varchar(36) USING gen_random_uuid();

-- change agent `id` column type to varchar and generate uuids to replace previous int values:
alter table hsearch_agent
    alter column id TYPE varchar(36) USING gen_random_uuid();
```

When using Hibernate ORM 6, i.e. when using `-orm6` artifacts the new `id` column type is `char(36)` instead of `varchar(36)`.

CockroachDB:

```
-- adjust `processAfter` values for existing events so that older events (i.e. ones with
smaller ID) will have older timestamp:
```

```
update hsearch_outbox_event
set processAfter = now() - cast(((select max(id) from hsearch_outbox_event) - id) || ' SECOND'
as interval)
where processAfter is null;
alter table hsearch_outbox_event
    alter column processAfter set not null;

-- change outbox event `id` column type to varchar and generate uuids to replace previous int
values:
-- altering type directly is not supported: https://go.crdb.dev/issue-v/47636/v22.1
alter table hsearch_outbox_event
    add tmp varchar(36) default cast(gen_random_uuid() as varchar) not null;
alter table hsearch_outbox_event
    alter primary key using columns (tmp);
alter table hsearch_outbox_event
    drop column id;
alter table hsearch_outbox_event
    rename column tmp to id;

-- change agent `id` column type to varchar and generate uuids to replace previous int values:
alter table hsearch_agent
    add tmp varchar(36) default cast(gen_random_uuid() as varchar) not null;
alter table hsearch_agent
    alter primary key using columns (tmp);
alter table hsearch_agent
    drop column id;
alter table hsearch_agent
    rename column tmp to id;
```

When using Hibernate ORM 6, i.e. when using `-orm6` artifacts the new `id` column type is `char(36)` instead of `varchar(36)`.

MySQL:

```
-- adjust `processAfter` values for existing events so that older events (i.e. ones with
smaller ID) will have older timestamp:
with max_id as (
    select max(id) as id from hsearch_outbox_event
)
update hsearch_outbox_event
set created = subtime(now(), sec_to_time((select id from max_id) - id))
where processAfter is null;
alter table hsearch_outbox_event
    modify column processAfter datetime not null;

-- change outbox event `id` column type to varchar and generate uuids to replace previous int
values:
alter table hsearch_outbox_event
    modify column id varchar(36);
update hsearch_outbox_event
set id = uuid()
where 1 = 1;

-- change agent `id` column type to varchar and generate uuids to replace previous int values:
alter table hsearch_agent
    modify column id varchar(36);
update hsearch_agent
set id = uuid()
where 1 = 1;
```

When using Hibernate ORM 6, i.e. when using `-orm6` artifacts the new `id` column type is `char(36)` instead of `varchar(36)`.

MariaDB:

```
-- adjust `processAfter` values for existing events so that older events (i.e. ones with
smaller ID) will have older timestamp:
update hsearch_outbox_event
set processAfter = subtime(now(), sec_to_time((select max(id) as id from hsearch_outbox_event)
- id))
where processAfter is null;
alter table hsearch_outbox_event
    modify column processAfter datetime not null;

-- change outbox event `id` column type to varchar and generate uuids to replace previous int
values:
alter table hsearch_outbox_event
    modify column id varchar(36);
update hsearch_outbox_event
set id = uuid()
where 1 = 1;

-- change agent `id` column type to varchar and generate uuids to replace previous int values:
alter table hsearch_agent
    modify column id varchar(36);
update hsearch_agent
set id = uuid()
where 1 = 1;
```

When using Hibernate ORM 6, i.e. when using `-orm6` artifacts the new `id` column type is `char(36)` instead of `varchar(36)`.

DB2:

```
-- adjust `processAfter` values for existing events so that older events (i.e. ones with
smaller ID) will have older timestamp:
update hsearch_outbox_event
set processAfter = current_timestamp - ((select max(id) from hsearch_outbox_event) - id)
seconds
where processAfter is null;
alter table hsearch_outbox_event
    alter column processAfter set not null;

-- change outbox event `id` column type to varchar and generate uuids to replace previous int
values:
alter table hsearch_outbox_event
    drop primary key;
alter table hsearch_outbox_event
    alter column id set data type varchar(36);
-- make this call if the adding constraint fails:
call sysproc.admin_cmd('reorg table hsearch_outbox_event');
alter table hsearch_outbox_event
    add constraint hsearch_outbox_event_pkey primary key (id);
update hsearch_outbox_event
set id = regexp_replace(concat(rawtohex(generate_unique()), 'AAAAAA'), '([A-F0-9]{8})([A-F0-
9]{4})([A-F0-9]{4})([A-F0-9]{4})([A-F0-9]{12})', '\1-\2-\3-\4-\5')
where 1 = 1;

-- change agent `id` column type to varchar and generate uuids to replace previous int values:
alter table hsearch_agent
    drop primary key;
alter table hsearch_agent
    alter column id set data type varchar(36);
-- make this call if the adding constraint fails:
call sysproc.admin_cmd('reorg table hsearch_agent');
alter table hsearch_agent
    add constraint hsearch_agent_pkey primary key (id);
update hsearch_agent
```

```
set id = regexp_replace(concat(rawtohex(generate_unique()), 'AAAAAA'), '([A-F0-9]{8})([A-F0-
9]{4})([A-F0-9]{4})([A-F0-9]{4})([A-F0-9]{12})', '\1-\2-\3-\4-\5')
where 1 = 1;
```

When using Hibernate ORM 6, i.e. when using `-orm6` artifacts the new `id` column type is `character(36)` instead of `varchar(36)`.

Oracle:

```
-- adjust `processAfter` values for existing events so that older events (i.e. ones with
smaller ID) will have older timestamp:
update hsearch_outbox_event
set processAfter = current_timestamp - numToDSInterval( (select max(id) from
hsearch_outbox_event) - id, 'second' )
where processAfter is null;
alter table hsearch_outbox_event
    modify (processAfter not null);

-- change outbox event `id` column type to varchar and generate uuids to replace previous int
values:
alter table hsearch_outbox_event
    add tmp varchar(36) default REGEXP_REPLACE(RAWTOHEX(SYS_GUID()), '([A-F0-9]{8})([A-F0-
9]{4})([A-F0-9]{4})([A-F0-9]{4})([A-F0-9]{12})', '\1-\2-\3-\4-\5') not null;
alter table hsearch_outbox_event
    drop column id;
alter table hsearch_outbox_event
    rename column tmp to id;
alter table hsearch_outbox_event
    add constraint hsearch_outbox_event_pkey primary key (id);

-- change agent `id` column type to varchar and generate uuids to replace previous int values:
alter table hsearch_agent
    add tmp varchar(36) default REGEXP_REPLACE(RAWTOHEX(SYS_GUID()), '([A-F0-9]{8})([A-F0-
9]{4})([A-F0-9]{4})([A-F0-9]{4})([A-F0-9]{12})', '\1-\2-\3-\4-\5') not null;
alter table hsearch_agent
    drop column id;
alter table hsearch_agent
    rename column tmp to id;
alter table hsearch_agent
    add constraint hsearch_agent_pkey primary key (id);
```

When using Hibernate ORM 6, i.e. when using `-orm6` artifacts the new `id` column type is `char(36)` instead of `varchar(36)`.

MSSQL:

```
-- adjust `processAfter` values for existing events so that older events (i.e. ones with
smaller ID) will have older timestamp:
update hsearch_outbox_event
set processAfter = dateadd(ss, -(select max(id) from hsearch_outbox_event) + id,
current_timestamp)
where processAfter is null;
alter table hsearch_outbox_event
    alter column processAfter datetime not null;

-- change publox event `id` column type to varchar and generate uuids to replace previous int
values:
alter table hsearch_outbox_event
    drop constraint if exists hsearch_outbox_event_pkey;
alter table hsearch_outbox_event
    alter column id binary(255) not null;
alter table hsearch_outbox_event
    add constraint hsearch_outbox_event_pkey primary key (id);
```

```
update hsearch_outbox_event
set id = convert(binary, newid())
where 1 = 1;

-- change agent `id` column type to varchar and generate uuids to replace previous int values:
alter table hsearch_agent
    drop constraint if exists hsearch_agent_pkey;
alter table hsearch_agent
    alter column id binary(255) not null;
alter table hsearch_agent
    add constraint hsearch_agent_pkey primary key (id);
update hsearch_agent
set id = convert(binary, newid())
where 1 = 1;
```

When using Hibernate ORM 6, i.e. when using `-orm6` artifacts the new `id` column type is `binary(16)` instead of `binary(255)`.

H2:

```
-- adjust `processAfter` values for existing events so that older events (i.e. ones with
smaller ID) will have older timestamp:
update hsearch_outbox_event
set processAfter = dateadd(ss, -(select max(id) from hsearch_outbox_event) + id,
current_timestamp)
where processAfter is null;
alter table hsearch_outbox_event
    alter column processAfter set not null;

-- change outbox event `id` column type to varchar and generate uuids to replace previous int
values:
alter table hsearch_outbox_event
    alter column id varchar(36) not null;
update hsearch_outbox_event
set id = random_uuid()
where 1 = 1;

-- change agent `id` column type to varchar and generate uuids to replace previous int values:
alter table hsearch_agent
    alter column id varchar(36) not null;
update hsearch_agent
set id = random_uuid()
where 1 = 1;
```

When using Hibernate ORM 6, i.e. when using `-orm6` artifacts the new `id` column type is `char(36)` instead of `varchar(36)`.

Configuration changes

The configuration properties are for the most part backward-compatible with Hibernate Search 6.1.

However, some changes may have an impact on exotic configuration:

- Configuration properties expecting references to "configurer" beans now accept multiple references, separated by commas. If your bean reference contains a comma, it may no longer be interpreted correctly.

  The suggested workaround is to avoid using commas in bean names.

  This affects the following configuration properties:

  - `hibernate.search.backend.analysis.configurer`
  - `hibernate.search.backend.query.caching.configurer`
  - `hibernate.search.mapping.configurer`

Additionally, some configuration properties have been deprecated:

- `hibernate.search.automatic_indexing.synchronization.strategy` is now deprecated in favor of `hibernate.search.indexing.plan.synchronization.strategy`.
- `hibernate.search.automatic_indexing.enabled` is now deprecated in favor of `hibernate.search.indexing.listeners.enabled`.
- `hibernate.search.automatic_indexing.enable_dirty_check` is now deprecated with no alternative to replace it. After its removal in a future version, a dirty check will always be performed when considering whether to trigger reindexing.

API changes

The API is backward-compatible with Hibernate Search 6.1.

Some incubating API changed:

- org.hibernate.search.engine.search.predicate.factories.NamedPredicateProvider` is now `org.hibernate.search.engine.search.predicate.definition.PredicateDefinition`.
- org.hibernate.search.engine.search.predicate.factories.NamedPredicateProviderContext` is now `org.hibernate.search.engine.search.predicate.definition.PredicateDefinitionContext`.

Parts of the API have been deprecated, and may be removed in the next major version:

- `org.hibernate.search.mapper.orm.common.EntityReference`: use `org.hibernate.search.engine.common.EntityReference` instead.
- `SearchPredicateFactory#bool(Consumer)`, which enables the syntax `f.bool(b → { b.must(…); b.must(…); })`: use the syntax `f.bool().with(b → { b.must(…); b.must(…); })` instead, or (if possible) take advantage of the new `.where(BiConsumer)` method in the Search Query DSL: `.where((f, b) → { b.must(…); b.must(…); })`.
- `SearchPredicateFactory#nested()`, which enables the syntax `f.nested().objectFieldPath("someField").nest(f.bool().must(…).must(…))`: use the syntax `f.nested("someField").must(…).must(…)` instead.
- `SearchProjectionFactory#composite((Function, SearchProjection …)` /`SearchProjectionFactory#composite((Function, ProjectionFinalStep …)` which enable the syntax `f.composite(list → …, <some projection>, <some projection>, …)`: use the (more flexible) syntax `f.composite().from(<some projection>, <some projection>, …).asList(list → …)` instead.
- `SearchProjectionFactory#composite((Function, SearchProjection)` /`SearchProjectionFactory#composite((Function, ProjectionFinalStep)` which enable the syntax `f.composite(p1 → …, <some projection>)`: use the (more flexible) syntax `f.composite().from(<some projection>).as(p1 → …)` instead.
- `SearchProjectionFactory#composite((BiFunction, SearchProjection, SearchProjection)`/`SearchProjectionFactory#composite((BiFunction, ProjectionFinalStep, ProjectionFinalStep)` which enable the syntax `f.composite((p1, p2) → …, <some projection>, <some projection>)`: use the (more flexible) syntax `f.composite().from(<some projection>, <some projection>).as((p1, p2) → …)` instead.
- `SearchProjectionFactory#composite((TriFunction, SearchProjection, SearchProjection, SearchProjection)` /`SearchProjectionFactory#composite((TriFunction, ProjectionFinalStep, ProjectionFinalStep, ProjectionFinalStep)` which enable the syntax `f.composite((p1, p2, p3) → …, <some projection>, <some projection>, <some projection>)`: use the (more flexible) syntax `f.composite().from(<some projection>, <some projection>, <some projection>).as((p1, p2, p3) → …)` instead.

- `SearchSession#automaticIndexingSynchronizationStrategy(..)` and related `AutomaticIndexingSynchronizationStrategy`/`AutomaticIndexingSynchronizationConfigurationContext`/`AutomaticIndexingSynchronizationStrategyNames`: use `SearchSession#indexingPlanSynchronizationStrategy(..)` and `IndexingPlanSynchronizationStrategy`/`IndexingPlanSynchronizationStrategyConfigurationContext`/`IndexingPlanSynchronizationStrategyNames` instead. Note the new API is still incubating and might change.
- The complement operator (`~`) used for matching regular expression patterns with flags is now deprecated for removal with no alternative to replace it.

SPI changes

Below are the most notable SPI changes compared to 6.1:

- `PojoGenericTypeModel` no longer exists; its methods moved to `PojoTypeModel`.
- `org.hibernate.search.mapper.pojo.mapping.spi.AbstractPojoMappingInitiator#annotatedTypeDiscoveryEnabled` is deprecated. Use `.annotationMapping().discoverAnnotationsFromReferencedTypes(…)` instead.
- `org.hibernate.search.util.common.reflect.spi.ValueReadHandleFactory` is deprecated. Use/implement `org.hibernate.search.util.common.reflect.spi.ValueHandleFactory` instead.
- `PojoAdditionalMetadataCollectorTypeNode#markAsEntity(String, org.hibernate.search.mapper.pojo.model.path.spi.PojoPathsDefinition)` is deprecated. Use `PojoAdditionalMetadataCollectorTypeNode#markAsEntity(String, org.hibernate.search.mapper.pojo.model.path.spi.PojoPathDefinitionProvider)` instead.
- `AutomaticIndexingStrategyStartContext` is deprecated. It was introduced by mistake and does not have any use.
- Mappers are no longer expected to provide a custom class to represent entity references, e.g. in search projections or in indexing failure reports. They should use `org.hibernate.search.engine.common.EntityReference` instead, which is the type that will be instantiated by default. Mappers that for some reason still need to rely on custom entity references classes will need to have their custom entity references class implement `org.hibernate.search.engine.common.EntityReference`, and will need to call `org.hibernate.search.mapper.pojo.mapping.spi.AbstractPojoMappingImplementor.AbstractPojoMappingImplementor(org.hibernate.search.mapper.pojo.mapping.spi.PojoMappingDelegate, org.hibernate.search.mapper.pojo.mapping.spi.PojoEntityReferenceFactory)` when their mapping gets instantiated.
- Many `execute*(…)`/`send*(…)` methods related to indexing plans now take an `OperationSubmitter` as an argument (see the javadoc of `OperationSubmitter`) and no longer take an `EntityReferenceFactory` as an argument (which is provided through `AbstractPojoMappingImplementor#entityReferenceFactory` instead).

Behavior changes

Due to bugfixes, parts of Hibernate Search now behave differently:

- The boolean predicate, `SearchPredicateFactory#bool()`, when used without any clause, used to match no documents with the Lucene backend, but all documents with the Elasticsearch backend. A boolean predicate with no clause will now consistently match no documents regardless of the backend.
- API methods matching `*Async(..)` pattern (e.g. `SearchWorkspace#purgeAsync()`) will no longer block if internal queues of operations are full, but will throw `RejectedOperationException` instead.

Due to switching from `new URL(..)` to `new URI(..)` in the Hibernate Search internals indexing behaviour of `URL` properties might change. In particular malformed URLs won't be accepted anymore and would result in a runtime exception.

Due to some optimizations applied to bool queries, the resulting query might get replaced with a more straightforward query that returns the same results. Possible changes can include: some clauses can be rearranged, nested bool queries might be flattened, a bool query might be replaced with its clause.

A bool query with a single `mustNot` clause and applied boost would implicitly add a `must` with `match_all` clause.