Hibernate Search 6.2.0.Alpha1

Migration Guide from 6.1

2022-07-12
# Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction</td>
<td>1</td>
</tr>
<tr>
<td>Requirements</td>
<td>2</td>
</tr>
<tr>
<td>Data format and schema changes</td>
<td>3</td>
</tr>
<tr>
<td>Configuration changes</td>
<td>4</td>
</tr>
<tr>
<td>API changes</td>
<td>5</td>
</tr>
<tr>
<td>SPI changes</td>
<td>6</td>
</tr>
<tr>
<td>Behavior changes</td>
<td>7</td>
</tr>
</tbody>
</table>
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.
Requirements

Hibernate Search’s requirements did not change in version 6.2.0.Alpha1.
Data format and schema changes

Indexes created with Hibernate Search 6.1 can be read from and written to with Hibernate Search 6.2.0.Alpha1.

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)`.
Configuration changes

The configuration properties are backward-compatible with Hibernate Search 6.1.
API changes

The API is backward-compatible with Hibernate Search 6.1.

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

- `SearchPredicateFactory#bool(Consumer)`, which enables the syntax `f.bool(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)` / `SearchProjectionFactory#composite((TriFunction, 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.
SPI changes

The SPI is mostly backward-compatible with Hibernate Search 6.1.

Below are the most notable SPI changes:

- `PojoGenericTypeModel` no longer exists; its methods moved to `PojoTypeModel`. 
Behavior changes

No behavior changes to report.