**How To Write an Index Value Maker**

**Last modified: 12/05/2022**

In G-SRS, an *Index Value Maker* is a procedure for taking an Entity and adding searchable and indexable fields, without changing the underlying data model.

***When you might want to use one:***

* You want a calculated value to be searchable, but not stored on the Entity itself
* You want to have facets for data, but do not want a direct hard link to that data in the data model
* You want to add a new custom sorter for an existing Entity
* You want to add a new type-ahead field

***When you might NOT want to use one:***

* You want to add a new top-level Entity (you must change the data model, or implement your own components for this)
* You want to add a new field to the JSON or Entity itself (you must change the data model, or implement your own components for this)
* You want an index which cannot be reduced to simple text or numeric data (e.g. an Index for searching for nearest-neighbor documents -- you must implement your own components for this)

**IndexValueMaker Interface**

The **IndexValueMaker** interface takes a genetic **T**, and **MUST** have a public 0 argument constructor. It defines the following methods, which must be implemented by an implementation:

|  |
| --- |
| Class<T> getIndexedEntityClass ();  public void createIndexableValues(T t, Consumer<IndexableValue> consumer); |

* getIndexedEntityClass returns the class of Type T. For example, if T is Substance, it returns Substance.class. T can be kept fairly generic, and can even be Object.class.
* createIndexableValues creates IndexableValue objects from a supplied Entity object t, and passes them into the supplied Consumer
* IndexableValue is an interface, which supplies a value to be indexed, as well as a field name, and information on how it should be processed (e.g. whether it should be a sortable column, facet, ranged numeric facet, or suggestible field)

**Example Implementation: Add Search Field and Facet for computed value**

For example, we may want to have some String value computable from a Substance be available as a searchable and faceted field. In this example, we’ll use a simple case where we index the first 4 characters of the substance UUIDs. The example could be expanded to do something like fetch information from another database, flat file, etc.

|  |
| --- |
| package ix.ginas.indexers;  import java.util.function.Consumer;  import ix.core.search.text.IndexValueMaker;  import ix.core.search.text.IndexableValue;  import ix.ginas.models.v1.Substance;  public class SubstanceUUIDIndexValueMaker implements IndexValueMaker<Substance>{  @Override  public Class<Substance> getIndexedEntityClass() {  return Substance.class;  }  @Override  public void createIndexableValues(Substance t, Consumer<IndexableValue> consumer) {  String firstFour = t.getUuid().toString().substring(0, 4);  consumer.accept(IndexableValue.simpleFacetStringValue("First Four", firstFour));  }  } |

IndexableValue, as an interface, has a helper method “simpleFacetStringValue” which can construct a basic IndexableValue that will be stored as searchable, and as a facet. After making this basic IndexableValue, we have the consumer consume it, which will trigger the deeper processing and indexing. The IndexableValue interface also includes the ability to flag a field as sortable, suggestable (can be used for type-ahead), and several other specifics about how the field should be named and how it should be parsed.

**Enabling the Index Value Maker**

After writing an IndexValueMaker, it must be enabled in the application. To do this, first make sure that it is visible to the classpath. Next, in the config file for the application (substances-core.conf), it must be explicitly registered for the given Entity type that it’s processing. This is done by adding an entry to the ` gsrs.indexers.list ` list. For the above example, this can be done by adding these lines to the end of the conf file:

|  |
| --- |
| gsrs.indexers.list += {  "indexer":"ix.ginas.indexers.SubstanceUUIDIndexValueMaker"  } |

You can also specify any initialization parameters that the new IndexValueMaker may allow in the JSON entry in the config file. For example, in the above case where the first 4 characters are extracted from the UUID, we could instead write this to extract the first **n** characters and allow **n** to be set in the config file. We can do this by adding the @Data annotation, and adding 2 fields meant to be settable by the config “numberChars” and “fieldName” below, for example:

|  |
| --- |
| package ix.ginas.indexers;  import java.util.function.Consumer;  import ix.core.search.text.IndexValueMaker;  import ix.core.search.text.IndexableValue;  import ix.ginas.models.v1.Substance; import lombok.Data;  @Data  public class SubstanceUUIDIndexValueMaker implements IndexValueMaker<Substance>{  private int numberChars = 4;  Private String fieldName = "First Four";  @Override  public Class<Substance> getIndexedEntityClass() {  return Substance.class;  }  @Override  public void createIndexableValues(Substance t, Consumer<IndexableValue> consumer) {  String firstN = t.getUuid().toString().substring(0, numberChars);  consumer.accept(IndexableValue.simpleFacetStringValue(fieldName, firstN));  }  } |

Then, to specify the parameters at launch time, the configuration file can be changed accordingly:

|  |
| --- |
| gsrs.indexers.list += {  "indexer":"ix.ginas.indexers.SubstanceUUIDIndexValueMaker",  "fieldName": "First Five",  "numberChars": 5  } |

These examples above adds the index value for indexing, and will affect the way the REST API behaves. They do *not*, however, change the UI to make a new facet appear by default. To do this, you can also add any facets you’ve made to the appropriate category of facets in the UI conf file `config.json` by adding the names of the facets to the `facetView` section. If a facet belongs to the `Record Data`, then add it to that section in the “facets” array.

|  |
| --- |
| {  "category": "Record Data",  "facets": [  ...  "First Four",  ...  ]  } |

To add the facet to the default set of facets, it can be added to the “default” category. Facets can be added to more than one category.

**Some Notes:**

* You can register multiple IndexValueMaker for the same class, or for different classes.
* When you register an IndexValueMaker for a class, it will be registered for all subtypes of that class which are entities. Currently most GSRS indexValueMakers are in gsrs-module-subtances-core project.
* There is currently no guarantee an execution order of the value makers, if there is more than 1 registered for an entity.
* There is no need to make only one value, or even one named *field* per IndexValueMaker. Those field names can be dynamic, and you can have multiple values for the same field name, which just makes each instance searchable.