Analyser API

**Analyser**:

When a text value is indexed a so called analyser is used to process the text, it contains 3 blocks character filters, tokenizer, token filters.

Character filters:

Adds, removes, or changes characters. May have zero or more character filters.

E.g. html\_strip filter --- removes html elements and gives plain text.

Tokenizer:

Must contain one tokenizer, tokenizer splits strings into tokens and characters may be stripped as a part of tokenization.

Token filters:

Receives output from the tokenizer and can add, remove or update tokens, can have zero ore more token filters.

E.g. lowercase filter

ANALYSER API

Values for a text field are analysed and the results are stored an inverted index

Each field has dedicated inverted index

Inverted index is mapping between terms and documents contain them

terms are stored alphabetically for performance reasons

creation and maintained by Apache Lucence, not by elastic search.

inverted indices enable fast searches

elasticsearch uses other data structures as well. example BKD trees for numeric values dates and geospecial data.

MAPPING:

mapping two types

dynamic mapping -- elastic generates field mapping for us

explicit mapping-- we defined fields mapping ourselves

DATA TYPES:

object, text, float, date, integer, long, Boolean, double, short, ip

OBJECT data type:

* Used for any JSON object
* Objects may be nested.
* Mapped using the properties
* Objects are not stored as objects in apache lucence

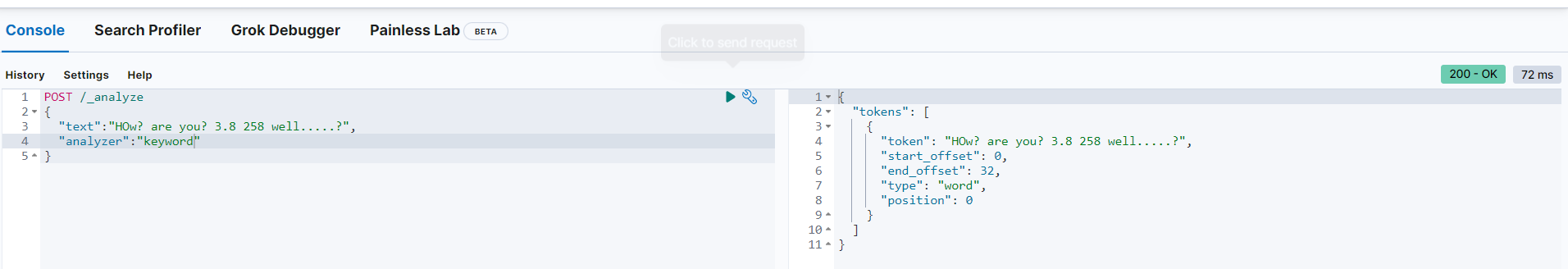
Each level hierarchy is denoted with a dot, such that there are no longer any objects

NESTED data type:

* Similar to object data type, but maintains object relationships.
* Useful when indexing array of objects.
* Enables us to query objects independently must use the nested query
* Nested objects are stored as hidden documents

KEYWORD data type:

* Used for exact matching of values.
* Typically used for filtering, aggregations and sorting.
* For full text searches, use the text data type instead.
* The keyword fields are analysed with keyword analyser is a no-op analyser.



**Type coercion:**

Similar to type casting in programming languages.

When we specify the first index that type will be fixed as data type.

String will be converted into numeric value if string contains a numeric value only and the required data type matches.

Disabling coercion can also be done by users choise.

ARRAYS:

There are no such as arrays.

Any field can contain zero or more values.

Array values should be of same data type or they can be type coercion possible.

NESTED ARRAYS:

Array may contain nested arrays they are flattened during indexing.

**Adding explicit mapping:**

PUT /reviews

{

"mappings" :{

"properties": {

"rating":{"type":"float"},

"content":{"type":"text"},

"product\_id":{"type":"integer"},

"author":{

"properties":{

"First\_name":{"type":"text"},

"Last\_name":{"type":"text"},

"email":{"type":"keyword"}

}

}

}

}

}

**Retrieve mappings**:

GET /reviews/\_mapping

GET /reviews/\_mapping/field/content

GET /reviews/\_mapping/field/author.email

**Append mappings:**

PUT /reviews/\_mapping

{

"properties":{

"created\_at":{"type":"date"}

}

}

DATE data type:

* Specified in one of three ways:
  + A date without time
  + A date with time
  + Milliseconds since the epoch(long)
* Will be default UTC time zone (if not specified)
* Dates must be formatted according to ISO 8061 specification
  + YYYY-MM-DD
  + 2022-03-27
  + 2022-04-18T13:13:13Z
  + 2022-12-13T13:15:00+01:00
* Epoch(1970) counted on seconds since epoch.