ElasticSearch 分词器，了解一下

武培轩
广联达科技股份有限公司 Java工程师

这篇文章主要来介绍下什么是 Analysis，什么是分词器，以及 ElasticSearch 自带的分词器是怎么工作的，最后会介绍下中文分词是怎么做的。

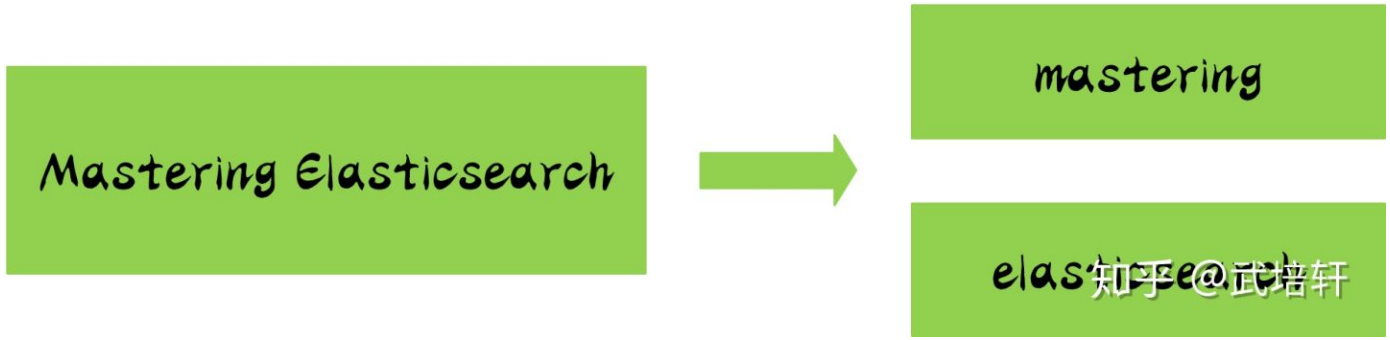
首先来说下什么是 Analysis：

什么是 Analysis?

顾名思义，文本分析就是**把全文本转换成一系列单词（term/token）的过程**，也叫**分词**。在 ES 中，Analysis 是通过**分词器（Analyzer）** 来实现的，可使用 ES 内置的分析器或者按需定制化分

析器。

举一个分词简单的例子：比如你输入 `Mastering Elasticsearch`，会自动帮你分成两个单词，一个是 `mastering`，另一个是 `elasticsearch`，可以看出单词也被转化成了小写的。

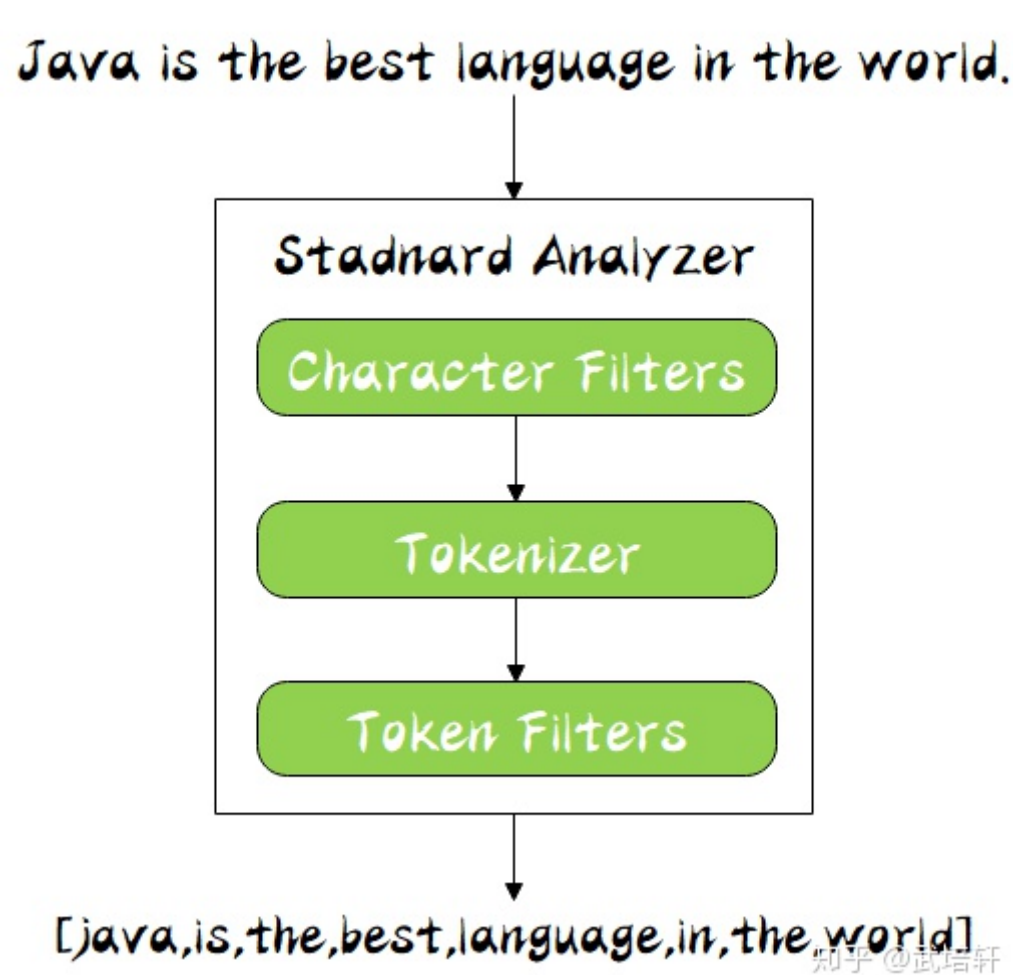


再简单了解了 Analysis 与 Analyzer 之后，让我们来看下分词器的组成：

分词器的组成

分词器是专门处理分词的组件，分词器由以下三部分组成：

- **Character Filters**：针对原始文本处理，比如去除 html 标签
- **Tokenizer**：按照规则切分为单词，比如按照空格切分
- **Token Filters**：将切分的单词进行加工，比如大写转小写，删除 stopwords，增加同义语



同时 Analyzer 三个部分也是有顺序的，从图中可以看出，从上到下依次经过 `Character Filters`，`Tokenizer` 以及 `Token Filters`，这个顺序比较好理解，一个文本进来肯定要先对文本数据进行处理，再去分词，最后对分词的结果进行过滤。

其中，ES 内置了许多分词器：

- **Standard Analyzer** - 默认分词器，按词切分，小写处理
- **Simple Analyzer** - 按照非字母切分（符号被过滤），小写处理
- **Stop Analyzer** - 小写处理，停用词过滤（the ，a，is）
- **Whitespace Analyzer** - 按照空格切分，不转小写

- **Keyword Analyzer** - 不分词，直接将输入当做输出
- **Pattern Analyzer** - 正则表达式，默认 \W+
- **Language** - 提供了 30 多种常见语言的分词器
- **Customer Analyzer** - 自定义分词器

接下来会对以上分词器进行讲解，在讲解之前先来看下很有用的 API：`_analyzer API`：

Analyzer API

它可以通过以下三种方式来查看分词器是怎么样工作的：

- **直接指定 Analyzer 进行测试**

```
GET _analyze
{
    "analyzer": "standard",
    "text" : "Mastering Elasticsearch , elasticsearch in Action"
}
```

- **指定索引的字段进行测试**

```
POST books/_analyze
{
    "field": "title",
    "text": "Mastering Elasticesearch"
}
```

- **自定义分词进行测试**

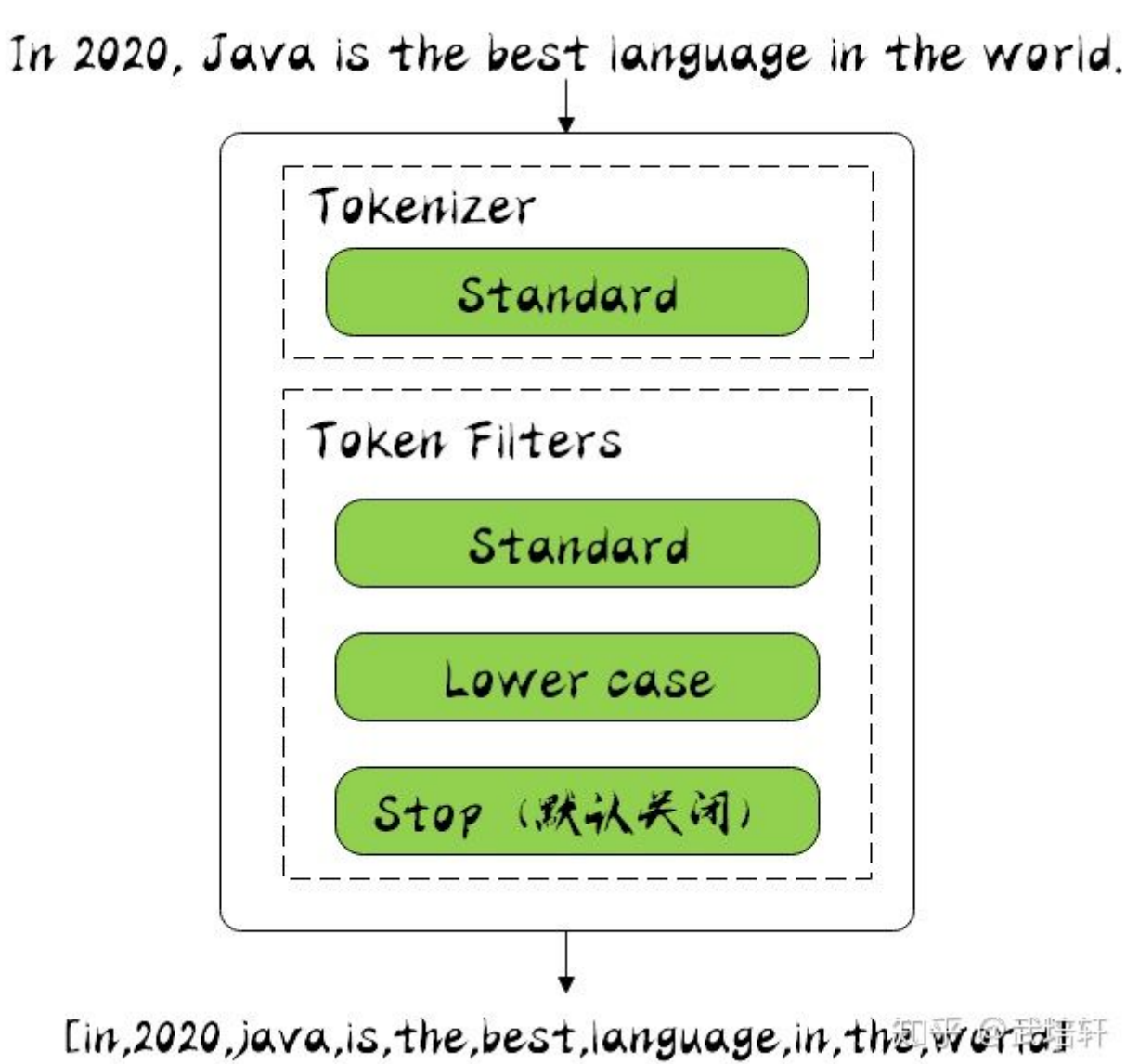
```
POST /_analyze
{
    "tokenizer": "standard",
    "filter": ["lowercase"],
    "text": "Mastering Elasticesearch"
}
```

再了解了 Analyzer API 后，让我们一起看下 ES 内置的分词器：

ES 分词器

首先来介绍下 `Stamdard Analyzer` 分词器：

Stamdard Analyzer



它是 ES **默认的分词器**，它会对输入的文本**按词的方式进行切分**，切分好以后会进行**转小写**处理，**默认的 stopwords 是关闭的。**

下面使用 Kibana 看一下它是怎么样进行工作的，在 Kibana 的开发工具（Dev Tools）中指定 Analyzer 为 `standard`，并输入文本 `In 2020, Java is the best language in the world.`，然后我们运行一下：

```
GET _analyze
{
  "analyzer": "standard",
  "text": "In 2020, Java is the best language in the world."
}
```

运行结果如下：

```
{
  "tokens" : [
    {
      "token" : "in",
      "start_offset" : 0,
      "end_offset" : 2,
      "type" : "<ALPHANUM>",
      "position" : 0
    },
    {
      "token" : "2020",
      "start_offset" : 3,
      "end_offset" : 7,
      "type" : "<NUM>",
      "position" : 1
    },
    {
      "token" : "java",
      "start_offset" : 9,
      "end_offset" : 13,
      "type" : "<ALPHANUM>",
      "position" : 2
    },
    {
      "token" : "is",
```

```
      "start_offset" : 14,
      "end_offset" : 16,
      "type" : "<ALPHANUM>",
      "position" : 3
    },
    {
      "token" : "the",
      "start_offset" : 17,
      "end_offset" : 20,
      "type" : "<ALPHANUM>",
      "position" : 4
    },
    {
      "token" : "best",
      "start_offset" : 21,
      "end_offset" : 25,
      "type" : "<ALPHANUM>",
      "position" : 5
    },
    {
      "token" : "language",
      "start_offset" : 26,
      "end_offset" : 34,
      "type" : "<ALPHANUM>",
      "position" : 6
    },
    {
      "token" : "in",
      "start_offset" : 35,
      "end_offset" : 37,
      "type" : "<ALPHANUM>",
      "position" : 7
    },
    {
```

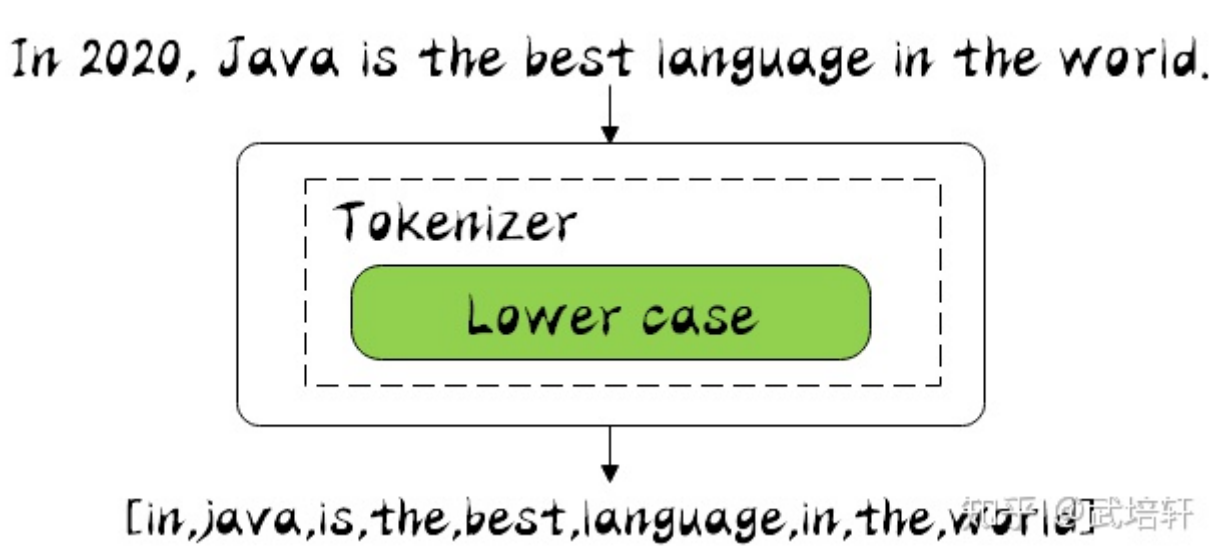
```
      "token" : "the",
      "start_offset" : 38,
      "end_offset" : 41,
      "type" : "<ALPHANUM>",
      "position" : 8
    },
    {
      "token" : "world",
      "start_offset" : 42,
      "end_offset" : 47,
      "type" : "<ALPHANUM>",
      "position" : 9
    }
  ]
}
```

可以看出是按照空格、非字母的方式对输入的文本进行了转换，比如对 `Java` 做了转小写，对一些停用词也没有去掉，比如 `in`。

其中 `token` 为分词结果； `start_offset` 为起始偏移； `end_offset` 为结束偏移； `position` 为分词位置。

下面来看下 `Simple Analyzer` 分词器：

Simple Analyzer



它只包括了 `Lower Case` 的 `Tokenizer`，它会按照**非字母切分**，**非字母的会被去除**，最后对切分好的做**转小写**处理，然后接着用刚才的输入文本，分词器换成 `simple` 来进行分词，运行结果如下：

```
{
  "tokens" : [
    {
      "token" : "in",
      "start_offset" : 0,
      "end_offset" : 2,
      "type" : "word",
      "position" : 0
    },
    {
      "token" : "java",
      "start_offset" : 9,
      "end_offset" : 13,
      "type" : "word",
      "position" : 1
    },
```

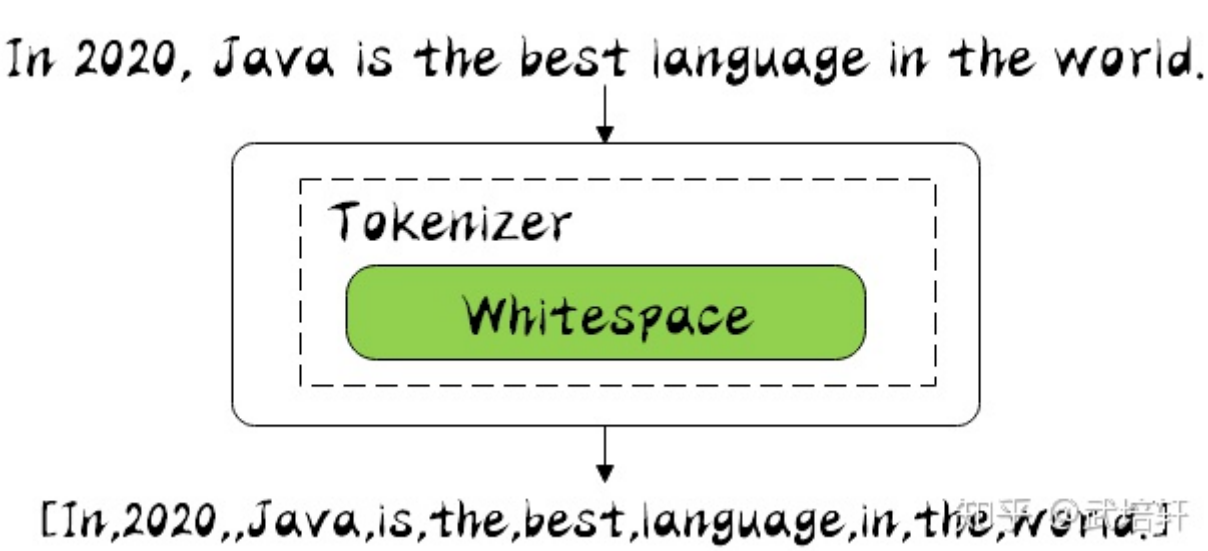
```
    {
      "token" : "is",
      "start_offset" : 14,
      "end_offset" : 16,
      "type" : "word",
      "position" : 2
    },
    {
      "token" : "the",
      "start_offset" : 17,
      "end_offset" : 20,
      "type" : "word",
      "position" : 3
    },
    {
      "token" : "best",
      "start_offset" : 21,
      "end_offset" : 25,
      "type" : "word",
      "position" : 4
    },
    {
      "token" : "language",
      "start_offset" : 26,
      "end_offset" : 34,
      "type" : "word",
      "position" : 5
    },
    {
      "token" : "in",
      "start_offset" : 35,
      "end_offset" : 37,
      "type" : "word",
      "position" : 6
```

```
    },
    {
      "token" : "the",
      "start_offset" : 38,
      "end_offset" : 41,
      "type" : "word",
      "position" : 7
    },
    {
      "token" : "world",
      "start_offset" : 42,
      "end_offset" : 47,
      "type" : "word",
      "position" : 8
    }
  ]
}
```

从结果中可以看出，数字 `2020` 被去除掉了，说明非字母的的确会被去除，所有的词也都做了小写转换。

现在，我们来看下 `Whitespace Analyzer` 分词器：

Whitespace Analyzer



它非常简单，根据名称也可以看出是**按照空格进行切分**的，下面我们来看下它是怎么样工作的：

```
{
  "tokens" : [
    {
      "token" : "In",
      "start_offset" : 0,
      "end_offset" : 2,
      "type" : "word",
      "position" : 0
    },
    {
      "token" : "2020,",
      "start_offset" : 3,
      "end_offset" : 8,
      "type" : "word",
      "position" : 1
    },
    {
      "token" : "Java",
```

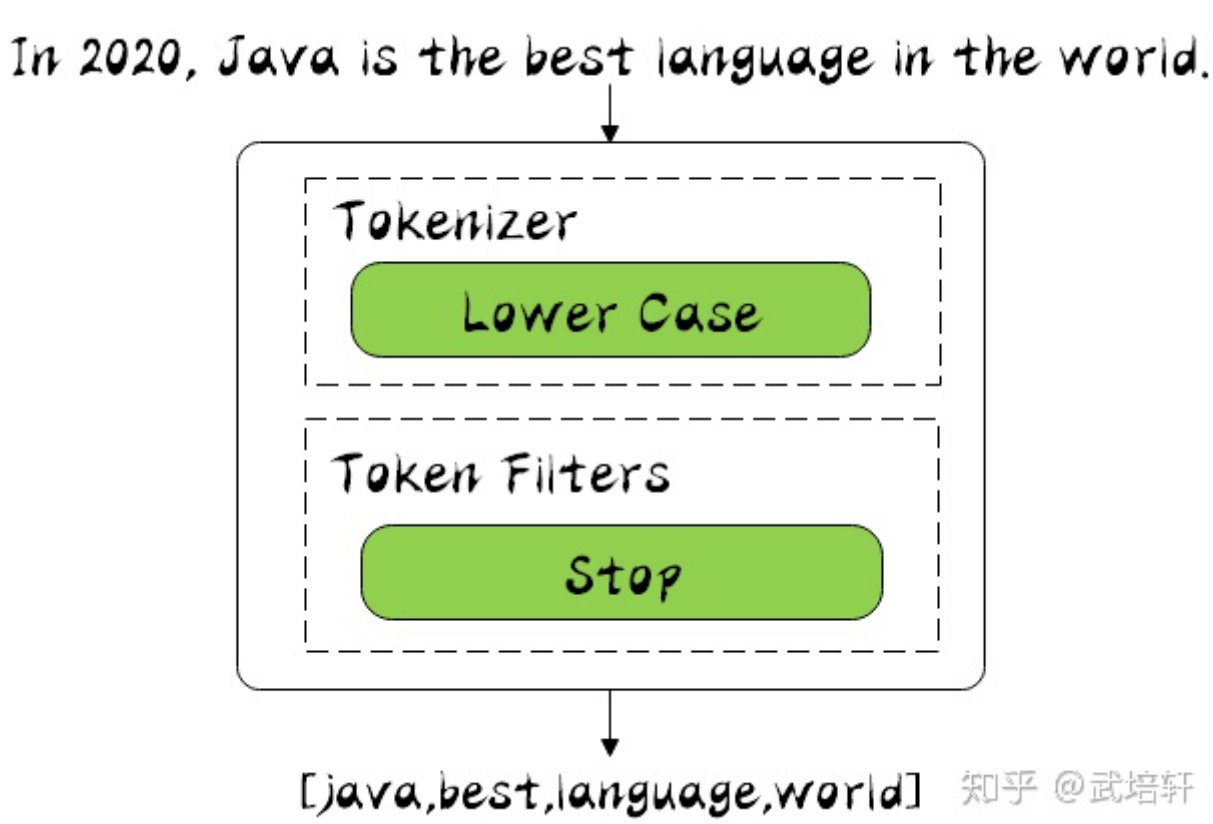
```
    "start_offset" : 9,
    "end_offset" : 13,
    "type" : "word",
    "position" : 2
  },
  {
    "token" : "is",
    "start_offset" : 14,
    "end_offset" : 16,
    "type" : "word",
    "position" : 3
  },
  {
    "token" : "the",
    "start_offset" : 17,
    "end_offset" : 20,
    "type" : "word",
    "position" : 4
  },
  {
    "token" : "best",
    "start_offset" : 21,
    "end_offset" : 25,
    "type" : "word",
    "position" : 5
  },
  {
    "token" : "language",
    "start_offset" : 26,
    "end_offset" : 34,
    "type" : "word",
    "position" : 6
  },
  {
```

```
      "token" : "in",
      "start_offset" : 35,
      "end_offset" : 37,
      "type" : "word",
      "position" : 7
    },
    {
      "token" : "the",
      "start_offset" : 38,
      "end_offset" : 41,
      "type" : "word",
      "position" : 8
    },
    {
      "token" : "world.",
      "start_offset" : 42,
      "end_offset" : 48,
      "type" : "word",
      "position" : 9
    }
  ]
}
```

可以看出，只是按照空格进行切分，`2020` 数字还是在的，`Java` 的首字母还是大写的，`,` 还是保留的。

接下来看 `Stop Analyzer` 分词器：

Stop Analyzer



它由 `Lowe Case` 的 `Tokenizer` 和 `Stop` 的 `Token Filters` 组成的，相较于刚才提到的 `Simple Analyzer`，多了 stop 过滤，stop 就是会把 `the`，`a`，`is` 等修饰词去除，同样让我们看下运行结果：

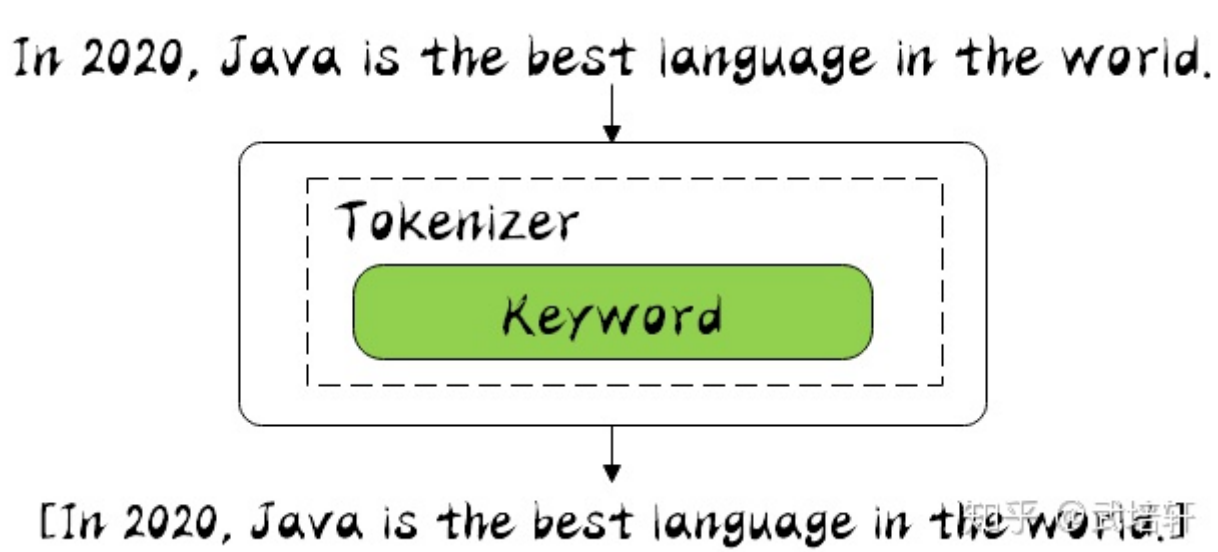
```
{
  "tokens" : [
    {
      "token" : "java",
      "start_offset" : 9,
      "end_offset" : 13,
      "type" : "word",
      "position" : 1
    },
    {
```

```
      "token" : "best",
      "start_offset" : 21,
      "end_offset" : 25,
      "type" : "word",
      "position" : 4
    },
    {
      "token" : "language",
      "start_offset" : 26,
      "end_offset" : 34,
      "type" : "word",
      "position" : 5
    },
    {
      "token" : "world",
      "start_offset" : 42,
      "end_offset" : 47,
      "type" : "word",
      "position" : 8
    }
  ]
}
```

可以看到 `in` `is` `the` 等词都被 `stop filter` 过滤掉了。

接下来看下 `Keyword Analyzer`：

Keyword Analyzer



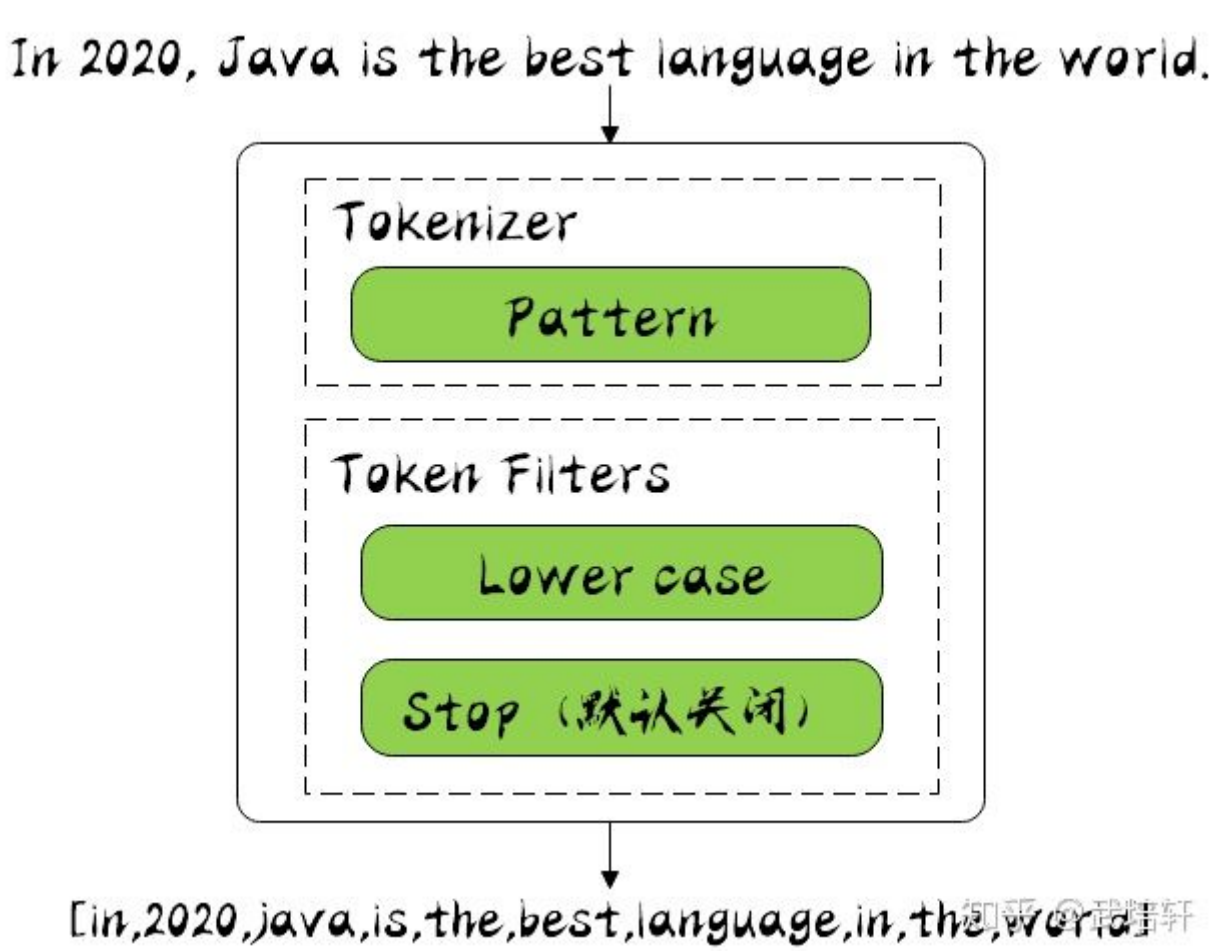
它其实不做分词处理，只是将输入作为 Term 输出，我们来看下运行结果：

```
{
  "tokens" : [
    {
      "token" : "In 2020, Java is the best language in the world.",
      "start_offset" : 0,
      "end_offset" : 48,
      "type" : "word",
      "position" : 0
    }
  ]
}
```

我们可以看到，没有对输入文本进行分词，而是直接作为 Term 输出了。

接下来看下 `Pattern Analyzer`：

Pattern Analyzer



它可以通过**正则表达式的方式进行分词**，默认是用 `\W+` 进行分割的，也就是非字母的符合进行切分的，由于运行结果和 `Stamdard Analyzer` 一样，就不展示了。

Language Analyzer

ES 为不同国家语言的输入提供了 `Language Analyzer` 分词器，在里面可以指定不同的语言，我们用 `english` 进行分词看下：

```
{
  "tokens" : [
```

```
  {
    "token" : "2020",
    "start_offset" : 3,
    "end_offset" : 7,
    "type" : "<NUM>",
    "position" : 1
  },
  {
    "token" : "java",
    "start_offset" : 9,
    "end_offset" : 13,
    "type" : "<ALPHANUM>",
    "position" : 2
  },
  {
    "token" : "best",
    "start_offset" : 21,
    "end_offset" : 25,
    "type" : "<ALPHANUM>",
    "position" : 5
  },
  {
    "token" : "languag",
    "start_offset" : 26,
    "end_offset" : 34,
    "type" : "<ALPHANUM>",
    "position" : 6
  },
  {
    "token" : "world",
    "start_offset" : 42,
    "end_offset" : 47,
    "type" : "<ALPHANUM>",
    "position" : 9
```

```
    }
  ]
}
```

可以看出 `language` 被改成了 `languag`，同时它也是有 stop 过滤器的，比如 `in`，`is` 等词也被去除了。

最后，让我们看下中文分词：

中文分词

中文分词有特定的难点，不像英文，单词有自然的空格作为分隔，在中文句子中，不能简单地切分成一个个的字，而是需要分成有含义的词，但是在不同的上下文，是有不同的理解的。

比如以下例子：

```
在这些，企业中，国有，企业，有十个/在这些，企业，中国，有企业，有十个
各国，有，企业，相继，倒闭/各，国有，企业，相继，倒闭
羽毛球，拍卖，完了/羽毛球拍，卖，完了
```

那么，让我们来看下 `ICU Analyzer` 分词器，它提供了 Unicode 的支持，更好的支持亚洲语言！

我们先用 `standard` 来分词，以便于和 ICU 进行对比。

```
GET _analyze
{
  "analyzer": "standard",
  "text": "各国有企业相继倒闭"
}
```

运行结果就不展示了，分词是一个字一个字切分的，明显效果不是很好，接下来用 ICU 进行分词，分词结果如下：


```
{
  "tokens" : [
    {
      "token" : "各国",
      "start_offset" : 0,
      "end_offset" : 2,
      "type" : "<IDEOGRAPHIC>",
      "position" : 0
    },
    {
      "token" : "有",
      "start_offset" : 2,
      "end_offset" : 3,
      "type" : "<IDEOGRAPHIC>",
      "position" : 1
    },
    {
      "token" : "企业",
      "start_offset" : 3,
      "end_offset" : 5,
      "type" : "<IDEOGRAPHIC>",
      "position" : 2
    },
    {
      "token" : "相继",
      "start_offset" : 5,
      "end_offset" : 7,
      "type" : "<IDEOGRAPHIC>",
      "position" : 3
    },
```

```
    {
      "token" : "倒闭",
      "start_offset" : 7,
      "end_offset" : 9,
      "type" : "<IDEOGRAPHIC>",
      "position" : 4
    }
  ]
}
```

可以看到分成了 `各国`， `有`， `企业`， `相继`， `倒闭`，显然比刚才的效果好了很多。

还有许多中文分词器，在这里列举几个：

IK：

- 支持自定义词库，支持热更新分词字典
- github.com/medcl/elasti...

jieba：

- Python 中最流行的分词系统，支持分词和词性标注
- 支持繁体分词、自定义词典、并行分词等
- github.com/sing1ee/elas...

THULAC：

- THU Lexucal Analyzer for Chinese, 清华大学自然语言处理和社会人文计算实验室的一套中文分词器
- github.com/thunlp/THULA...

大家可以自己安装下，看下它中文分词效果。

总结

本文主要介绍了 ElasticSearch 自带的分词器，学习了使用 `_analyzer API` 去查看它的分词情况，最后还介绍下中文分词是怎么做的。

> 参考文献
> Elasticsearch顶尖高手系列
> Elasticsearch核心技术与实战
> elastic.co/guide/en/ela...
> elastic.co/guide/en/ela...

发布于 03-08

分词 Elasticsearch 中文分词