#######LAB 5 ############################Analyzers

```
index :
    analysis :
        analyzer :
            standard :
                type : standard
                stopwords : [stop1, stop2]
            myAnalyzer1 :
                type : standard
                stopwords : [stop1, stop2, stop3]
                max_token_length : 500
            # configure a custom analyzer which is
            # exactly like the default standard analyzer
            myAnalyzer2 :
                tokenizer : standard
                filter : [standard, lowercase, stop]
        tokenizer :
            myTokenizer1 :
                type : standard
                max_token_length : 900
            myTokenizer2 :
                type : keyword
                buffer_size : 512
        filter :
            myTokenFilter1 :
                type : stop
                stopwords : [stop1, stop2, stop3, stop4]
            myTokenFilter2 :
                type : length
                min : 0
                max : 2000

index :
  analysis :
    analyzer :
      default :
        tokenizer : keyword

index :
  analysis :
    analyzer :
      standard :
```

```
        alias: [alias1, alias2]
        type : standard
        stopwords : [test1, test2, test3]

        DELETE test

PUT /test
{
  "settings": {
    "analysis": {
      "analyzer": {
        "whitespace": {
          "type": "pattern",
          "pattern": "\\s+"
        }
      }
    }
  }
}

GET /test/_analyze?analyzer=whitespace&text=foo,bar baz

# "foo,bar", "baz"


DELETE test

PUT /test
{
  "settings": {
    "analysis": {
      "analyzer": {
        "nonword": {
          "type": "pattern",
          "pattern": "[^\\w]+"
        }
      }
    }
  }
}

GET /test/_analyze?analyzer=nonword&text=foo,bar baz
# "foo,bar baz" becomes "foo", "bar", "baz"
```

```
GET /test/_analyze?analyzer=nonword&text=type_1-type_4
# "type_1","type_4"

DELETE test

PUT /test?pretty=1
{
  "settings": {
    "analysis": {
      "analyzer": {
        "camel": {
          "type": "pattern",
          "pattern":
"([^\\p{L}\\d]+)|(?<=\\D)(?=\\d)|(?<=\\d)(?=\\D)|(?<=[\\p{L}&&[^\\p{Lu}]])(?=\\p{Lu})|(?<=\\p{Lu})(?=\
\p{Lu}[\\p{L}&&[^\\p{Lu}]])"
        }
      }
    }
  }
}

GET /test/_analyze?analyzer=camel&text=MooseX::FTPClass2_beta
# "moose","x","ftp","class","2","beta"

The regex above is easier to understand as:

  ([^\p{L}\d]+)                # swallow non letters and numbers,
| (?<=\D)(?=\d)                 # or non-number followed by number,
| (?<=\d)(?=\D)                 # or number followed by non-number,
| (?<=[ \p{L} && [^\p{Lu}]])    # or lower case
  (?=\p{Lu})                   #   followed by upper case,
| (?<=\p{Lu})                  # or upper case
  (?=\p{Lu}                    #   followed by upper case
    [\p{L}&&[^\p{Lu}]]         #   then lower case
  )
###arabic analyser
{
  "settings": {
    "analysis": {
      "filter": {
        "arabic_stop": {
          "type":       "stop",
```

```
        "stopwords":  "_arabic_"
       },
       "arabic_keywords": {
        "type":       "keyword_marker",
        "keywords":   []
       },
       "arabic_stemmer": {
        "type":       "stemmer",
        "language":   "arabic"
       }
      },
      "analyzer": {
       "arabic": {
        "tokenizer":  "standard",
        "filter": [
         "lowercase",
         "arabic_stop",
         "arabic_normalization",
         "arabic_keywords",
         "arabic_stemmer"
        ]
       }
      }
    }
  }
}
###armenian Analyzer
{
  "settings": {
    "analysis": {
      "filter": {
       "armenian_stop": {
        "type":       "stop",
        "stopwords":  "_armenian_"
       },
       "armenian_keywords": {
        "type":       "keyword_marker",
        "keywords":   []
       },
       "armenian_stemmer": {
        "type":       "stemmer",
        "language":   "armenian"
       }
```

```
      },
      "analyzer": {
        "armenian": {
          "tokenizer":  "standard",
          "filter": [
            "lowercase",
            "armenian_stop",
            "armenian_keywords",
            "armenian_stemmer"
          ]
        }
      }
    }
  }
}
####basque analyzer
{
  "settings": {
    "analysis": {
      "filter": {
        "basque_stop": {
          "type":       "stop",
          "stopwords":  "_basque_"
        },
        "basque_keywords": {
          "type":       "keyword_marker",
          "keywords":   []
        },
        "basque_stemmer": {
          "type":       "stemmer",
          "language":   "basque"
        }
      },
      "analyzer": {
        "basque": {
          "tokenizer":  "standard",
          "filter": [
            "lowercase",
            "basque_stop",
            "basque_keywords",
            "basque_stemmer"
          ]
        }
```

```
      }
    }
  }
}
###brazilian analyzer
{
  "settings": {
    "analysis": {
      "filter": {
        "brazilian_stop": {
          "type":       "stop",
          "stopwords":  "_brazilian_"
        },
        "brazilian_keywords": {
          "type":       "keyword_marker",
          "keywords":   []
        },
        "brazilian_stemmer": {
          "type":       "stemmer",
          "language":   "brazilian"
        }
      },
      "analyzer": {
        "brazilian": {
          "tokenizer":  "standard",
          "filter": [
            "lowercase",
            "brazilian_stop",
            "brazilian_keywords",
            "brazilian_stemmer"
          ]
        }
      }
    }
  }
}
###english analyzer
{
  "settings": {
    "analysis": {
      "filter": {
        "english_stop": {
          "type":       "stop",
```

```
        "stopwords":  "_english_"
      },
      "english_keywords": {
        "type":       "keyword_marker",
        "keywords":   []
      },
      "english_stemmer": {
        "type":       "stemmer",
        "language":   "english"
      },
      "english_possessive_stemmer": {
        "type":       "stemmer",
        "language":   "possessive_english"
      }
    },
    "analyzer": {
      "english": {
        "tokenizer":  "standard",
        "filter": [
          "english_possessive_stemmer",
          "lowercase",
          "english_stop",
          "english_keywords",
          "english_stemmer"
        ]
      }
    }
  }
}

index :
    analysis :
        analyzer :
            myAnalyzer2 :
                type : custom
                tokenizer : myTokenizer1
                filter : [myTokenFilter1, myTokenFilter2]
                char_filter : [my_html]
                position_increment_gap: 256
        tokenizer :
            myTokenizer1 :
                type : standard
```

```
            max_token_length : 900
        filter :
           myTokenFilter1 :
              type : stop
              stopwords : [stop1, stop2, stop3, stop4]
           myTokenFilter2 :
              type : length
              min : 0
              max : 2000
        char_filter :
             my_html :
              type : html_strip
              escaped_tags : [xxx, yyy]
              read_ahead : 1024

#### edge ngram tokenizer #######

  curl -XPUT 'localhost:9200/test' -d '
   {
      "settings" : {
         "analysis" : {
            "analyzer" : {
               "my_edge_ngram_analyzer" : {
                  "tokenizer" : "my_edge_ngram_tokenizer"
               }
            },
            "tokenizer" : {
               "my_edge_ngram_tokenizer" : {
                  "type" : "edgeNGram",
                  "min_gram" : "2",
                  "max_gram" : "5",
                  "token_chars": [ "letter", "digit" ]
               }
            }
         }
      }
   }'

    curl 'localhost:9200/test/_analyze?pretty=1&analyzer=my_edge_ngram_analyzer' -d 'FC
Schalke 04'
    # FC, Sc, Sch, Scha, Schal, 04

    #### ngram tokenizer#####
```

```
curl -XPUT 'localhost:9200/test' -d '
{
    "settings" : {
        "analysis" : {
            "analyzer" : {
                "my_ngram_analyzer" : {
                    "tokenizer" : "my_ngram_tokenizer"
                }
            },
            "tokenizer" : {
                "my_ngram_tokenizer" : {
                    "type" : "nGram",
                    "min_gram" : "2",
                    "max_gram" : "3",
                    "token_chars": [ "letter", "digit" ]
                }
            }
        }
    }
}'

curl 'localhost:9200/test/_analyze?pretty=1&analyzer=my_ngram_analyzer' -d 'FC Schalke
04'
# FC, Sc, Sch, ch, cha, ha, hal, al, alk, lk, lke, ke, 04


######Lowercase token filter

index :
    analysis :
        analyzer :
            myAnalyzer2 :
                type : custom
                tokenizer : myTokenizer1
                filter : [myTokenFilter1, myGreekLowerCaseFilter]
                char_filter : [my_html]
        tokenizer :
            myTokenizer1 :
                type : standard
                max_token_length : 900
        filter :
            myTokenFilter1 :
                type : stop
```

```
            stopwords : [stop1, stop2, stop3, stop4]
          myGreekLowerCaseFilter :
            type : lowercase
            language : greek
      char_filter :
           my_html :
            type : html_strip
            escaped_tags : [xxx, yyy]
            read_ahead : 1024

##### STop Token Filter

PUT /my_index
{
   "settings": {
      "analysis": {
         "filter": {
            "my_stop": {
               "type":       "stop",
               "stopwords": ["and", "is", "the"]
            }
         }
      }
   }
}

PUT /my_index2
{
   "settings": {
      "analysis": {
         "filter": {
            "my_stop": {
               "type":       "stop",
               "stopwords":  "_english_"
            }
         }
      }
   }
}
```

Elasticsearch provides the following predefined list of languages:

arabic, _armenian_, _basque_, _brazilian_, _bulgarian_, _catalan_, _czech_, _danish_, _dutch_, _english_, _finnish_, _french_, _galician_, _german_, _greek_, _hindi_, _hungarian_, _indonesian_, _irish_, _italian_, _latvian_, _norwegian_, _persian_, _portuguese_, _romanian_, _russian_, _sorani_, _spanish_, _swedish_, _thai_, _turkish_.

For the empty stopwords list (to disable stopwords) use: _none_.

###character filters

```
{
   "index" : {
      "analysis" : {
         "char_filter" : {
            "my_mapping" : {
               "type" : "mapping",
               "mappings" : [
                 "ph => f",
                 "qu => k"
               ]
            }
         },
         "analyzer" : {
            "custom_with_char_filter" : {
               "tokenizer" : "standard",
               "char_filter" : ["my_mapping"]
            }
         }
      }
   }
}
```

```
{
   "index" : {
      "analysis" : {
         "char_filter" : {
            "my_pattern":{
               "type":"pattern_replace",
               "pattern":"sample(.*)",
               "replacement":"replacedSample $1"
            }
         },
         "analyzer" : {
            "custom_with_char_filter" : {
```

```
                "tokenizer" : "standard",
                "char_filter" : ["my_pattern"]
            }
        }
    }
  }
}
```