Elasticsearch Wiki

1. Konfigurace (Docker, konfigurační třídy)
2. Implementace
3. Provoz
4. Přechod na nového klienta

Důležité zdroje:

- Oficiální dokumentace (https://www.elastic.co/guide/en/elasticsearch/reference/current/rest-apis.html)
    - Pozor, na projektech používáme verzi 7.17., API tedy může být lehce odlišné od nejnovější verze.
- Diskusní fórum (https://discuss.elastic.co/)
- OpsGPT – AI zaměřená na Elasticsearch a OpenSearch (https://opster.com/opsgpt/)

Konfigurace

---

1. Zprovoznění Elasticsearch (Docker, application.yml)
2. Konfigurační třídy

Zprovoznění Elasticsearch

Elasticsearch běží v samostatném kontejneru v Dockeru. Je třeba jej nakonfigurovat v rámci `docker-compose.yml` takto.

```
  index:
    build: ./index
    image: docker.elastic.co/elasticsearch/elasticsearch:7.17.9
    networks:
      - backend
    expose:
      - 9200
    deploy:
      resources:
        limits:
          memory: 1000m
    environment:
      - discovery.type=single-node
      - "ES_JAVA_OPTS=-Xms1024m -Xmx1024m"
      - TZ="Europe/Prague"
      - ELASTIC_PASSWORD=${INDEX_PASS}
      - xpack.security.enabled=true
    ulimits:
      memlock:
        soft: -1
        hard: -1
    volumes:
      - index_volume:/usr/share/elasticsearch/data
```

Dále je třeba nastavit `application.yml`.

```
spring:
  elasticsearch:
    uris: index:9200
    username: elastic
    password: password
```

Doprovodným nástrojem pro práci s Elasticsearch pomocí grafického rozhraní je Kibana. Kibana běží také v samostatném kontejneru a je ji možné definovat takto:

```
indexfe:
  image: docker.elastic.co/kibana/kibana:7.17.9
  depends_on:
    - index
  expose:
    - 80
  networks:
    - backend
  deploy:
    resources:
      limits:
        memory: 350m
  environment:
    SERVER_PORT: "80"
    SERVER_BASEPATH: "/index"
    SERVER_REWRITEBASEPATH: "true"
    ELASTICSEARCH_HOSTS: "http://index:9200"
    ELASTICSEARCH_USERNAME: ${INDEX_USER}
    ELASTICSEARCH_PASSWORD: ${INDEX_PASS}
  ports:
    - "8083:80"
```

Konfigurační třídy

IndexConfiguration (povinné)

Tato konfigurace slouží ke specifikaci:

- Elasticsearch Java API Client, který slouží pro komunikaci s Elasticsearch.
- Nastavení indexu přes `ElasticSearchIndexSettingsConfigurer`.
- Povolených filtrů a způsobů řazení.

ElasticSearchIndexSettingsConfigurer

V rámci instance `ElasticSearchIndexSettingsConfigurer` specifikujeme nastavení, které se použije pro indexy při jejich vytváření. Mimo jiné zde můžeme nastavit limity pro mapování (definice atributů indexovaných objektů) a analyzátory pro textovou analýzu. Analýza textu spočívá v transformaci textových atributů indexovaných objektů do struktury umožňující optimální vyhledávání. Můžeme použít buď již existující analyzéry nebo definovat vlastní.

Analyzér se skládá z následujících částí:

- *character filters*
  - Předzpracovávají znaky textu, např. odstraní HTML značky nebo nahradí speciální znaky. Vybírají se z již existujících a může jich být specifikováno nula až více.
- *tokenizer*
  - Rozděluje text na jednotlivé výrazy, tzv. tokeny, které jsou samostatně vyhledatelné. Analyzér jej musí mít právě jeden z dostupných, např. `whitespace` tokenizer text rozdělí na základě mezer.
- *token filters*
  - Upravují (např. na malá písmena), odstraňují (např. výplňková slova) a přidávají (např. synonyma) tokeny. Opět se vybírají z již existujících a může jich být nula a více.

Definice vlastních analyzérů může být vhodná pro podporu jazyka, zohlednění specifik domény nebo zpracování speciálních znaků. Pro atributy typu `keyword` (text je vždy právě jeden token) se používá alternativa analyzérů, normalizéry, které pochopitelně neobsahují tokenizer.

IndexMappingConfiguration (nepovinné)

S touto konfigurací máme k dispozici API pro získání mapování (definice atributů indexovaných objektů) pro danou entitní třídu.

ReindexConfiguration (nepovinné)

Pokud byl Elasticsearch delší dobu mimo provoz, mohou v něm být nekonzistentní hodnoty vůči DB. Pro tyto případy tato konfigurace přidává API pro manuální reindex, který vytvoří nové indexy na základě dat v DB.

Poznámka: Dříve bylo nutné reindexovat i při změně mapování nebo jiném nastavení indexu. Reindex se nyní v těchto případech provádí automaticky při startu aplikace.

IndexReferenceConfiguration (nepovinné)

Některé indexované objekty mohou mít jako indexovaný atribut atribut jiné třídy (např. student má zaindexován název školy). Pokud se hodnota tohoto atributu v DB změní (škola změní svůj oficiální název), ve výchozím nastavení zůstane u objektu s referencí stará hodnota. Konfigurace `IndexReferenceConfiguration` přidává `IndexReferenceScanner`, který monitoruje změny v referencích a automaticky aplikuje změny.

ElasticsearchMonitoringConfiguration (nepovinné)

Tato konfigurace přidává podporu pro monitorování a alerty stavu Elasticsearch. Více informací v sekci Provoz.

Poznámka: `IndexMappingConfiguration`, `ReindexConfiguration`, `IndexReferenceConfiguration` je aktuálně možné využít jen pro indexované objekty navázané na DB entitu.

Viz konfigurační třídy v kódu.

Implementace

---

1. Definice indexů a indexovaných objektů
2. Anotace (datové typy atributů indexovaného objektu)
3. Modifikace requestů pro vyhledávání
4. PIT (Point in time)
5. Optimalizace vyhledávání bez nutnosti řazení

Definice indexů a indexovaných objektů

Když v EASu pracujeme s Elasticsearch, využíváme pro to dva typy komponent: indexy a indexované objekty.

- Indexy
  - Tyto třídy definují a zajišťují dostupnou funkcionalitu nad Elasticsearch. Existují předdefinované třídy `DomainIndex`, `DatedIndex`, `AuthoredIndex` a `DictionaryIndex`, které se mohou přímo využít pro konkrétní typy objektů nebo mohou jejich funkcionalitu modifikovat či rozšířit.
- Indexované objekty
  - Tyto třídy definují, jak bude daný objekt indexován (s jakými atributy). Indexované objekty mohou být navázané na DB entitu (pak implementují `DomainIndexed` a rozšiřují `DomainIndexedObject`) nebo nikoliv (pak implementují `DomainCoreIndexed` a rozšiřují `DomainCoreIndexedObject`). Opět existují specifické třídy pro objekty na úrovni `Dated`, `Authored` a `Dictionary`.

Většinou používáme indexované objekty v návaznosti na DB tabulky. Toto nám umožňuje mít v DB uložena data, která nejsou potřeba pro vyhledávání (např. přihlašovací údaje). Naopak v Elasticsearch můžeme mít data uložena způsobem, který vyhledávání optimalizuje. V jistých případech může ale dávat smysl mít objekt uložen jen v indexu bez návaznosti na DB. Takovým příkladem mohou být malé třídy, jejichž všechny atributy slouží k vyhledávání, nebo třídy, u kterých se atributy často nebo vůbec nemění, jako jsou logy.

V demo projektu jsou k dispozici ukázky využití indexů pro zážitkové akce (entita navázaná na DB) a dopravní nehody (bez návaznosti na DB). Pro oba typy objektů jsou připravena testovací data, která můžete použít pro testování funkcionality Elasticsearch.

Anotace (datové typy atributů indexovaného objektu)

Aktuálně EAS využívá pro definici indexu a jeho atributů anotace z knihovny Spring Data Elasticsearch (více info zde). Klíčové jsou pro nás tyto tři typy anotací:

- `@Document`
  - Slouží pro specifikaci názvu indexu pro danou třídu a nastavení, zda se má index vytvořit.
- `@Field`
  - Slouží pro specifikaci názvu atributu, datový typ a další nastavení jako jsou analyzéry.
- `@MultiField`
  - Slouží pro indexování atributu ve více formách (např. jinak pro vyhledávání, řazení a agregace).

Každý atribut indexovaného objektu má své specifické požadavky na svoji definici, které mají vliv na podporu operací a jejich optimalizaci. Dostupné typy jsou k nalezení zde. Při výběru anotací můžete vycházet ze třídy `MultipleFieldsIndexedObject`, která obsahuje anotace pro všechny typy atributů. Nicméně, ne pro všechny atributy jsou `MultipleFieldsIndexedObject` vhodné!

Modifikace requestů pro vyhledávání

V rámci některých metod pro vyhledávání (metody `list*` v `DomainIndex`), je možné pomocí parametru `Consumer<SearchRequest.Builder> queryModifier` upravovat request pro vyhledávání. Tohoto se využívá např. na úrovni `DatedIndex`, kde se rozšiřuje dotaz pro vyhledávání o podmínku, že nesmí mít atribut smazání.

PIT (Point in time)

Metody pro stránkování a infinite scroll se ve své základní formě v Elasticsearch aplikují vždy na aktuálním stavu indexů. Pokud v průběhu do indexů přibydou nové záznamy, mohou tyto metody vrátit nekonzistentní výsledky. Např. některý záznam se vrátí podruhé, protože přidání nového záznamu jej posunulo na další stránku. Pro tyto účely tu existuje parametr PIT (point in time), který vytvoří bod v čase a vrátí záznamy, které v daném bodě existovaly. PIT můžeme nastavit v rámci `Params`, kde specifikujeme dobu, po kterou má PIT zůstat aktivní. Při prvním volání se PIT vytvoří a ve výsledku se vrátí jeho ID, které se následně použije v dalších voláních. Ukázku užití naleznete v `PitFilterTest`.

Optimalizace vyhledávání bez nutnosti řazení

Pokud vyhledáváme bez specifikace řazení, použije se pro řazení relevance score. Pokud potřebujete vyhledat všechny záznamy bez nutnosti řazení rychleji, můžete použít pro řazení `_doc` (alternativně `_shard_doc` při PIT a více shards). Více info zde.

Provoz

Aktuálně jsou v EASu k dispozici dva nástroje pro provozní podporu, `ElasticsearchMonitoring` (monitoring a alerty) a Kibana (průzkumník, debugování a mnoho dalšího).

---

1. ElasticsearchMonitoring
2. Kibana

ElasticsearchMonitoring

`ElasticsearchMonitoring` je třída reprezentující kontrolu zdravotního stavu Elasticsearch. V rámci její metody `check()` se zavolá Elasticsearch API pro zdravotní statistiky a provedou se nad nimi kontroly (pokud se překročí omezení, vyhodí se `ElasticsearchMonitoringEvent` pro další zpracování).
`ElasticsearchMonitoring` obsahuje metodu `schedule()` určenou pro pravidelné volání kontroly. Ukázku užití naleznete např. ve třídě `ClusterMonitoring`.
Podporu pro tyto kontroly do své aplikace přidáte pomocí `ElasticsearchMonitoringConfiguration`.

Je možné vytvořit vlastní kontroly rozšířením třídy `ElasticsearchMonitoring`. K dispozici jsou také již předdefinované kontroly `ClusterMonitoring` (pro kontrolu zdravotního stavu clusteru) a `NodeStatsMonitoring` (pro kontrolu statistik Elasticsearch uzlů), které můžete rovnou použít, rozšířit, nebo z nich vycházet při vytváření vlastních kontrol.

V demo projektu naleznete:

- konfiguraci pro podporu `ElasticsearchMonitoring`
- konfiguraci spouštění kontrol v `application.yml`
- ukázku listeneru na `ElasticsearchMonitoringEvent` události (lze např. upravit na zasílání mailu)

Kibana

Kibana je grafické rozhraní pro práci s Elasticsearch a doprovodnými nástroji.

Dá se použít pro:

- průzkum indexovaných dat (alternativa k pgAdmin) a jejich vizualizaci
- debugování a testování funkcionality
- správu a monitoring Elasticsearch a dalších nástrojů od Elasticu

Následující video má za cíl představit Kibanu a několik základních funkcionalit, pro které můžete Kibanu začít používat.

Odkaz na video

Poznámky k videu:

- Zapomněla jsem zmínit, že pro vizualizaci textových fieldů je potřeba, aby byly typu `keyword`. Jinak se text nebude správně zobrazovat.
- V části s konzolí ve skutečnosti neukazuji analyzéry, ale tokenizéry.
- Více informací o Kibaně naleznete zde.

Přechod na nového klienta

V rámci změn spojených s využitím Elasticsearch, došlo ke změně klienta z High Level Rest Client na novější Elasticsearch Java API Client. Pro přechod je třeba na projektech upravit následující.

- Je třeba přidat dependency `implementation "org.glassfish:jakarta.json:2.0.1"` do modulu aplikace, pro podporu object mapperu.
- Pokud používáte vlastní definici `es_settings.json`, lehce se změnil formát JSONu. Viz zde.
- Je třeba mít alespoň následující verzi Spring Boot. Starší verze obsahují starší verzi klienta, která se v některých bodech liší od té, kterou používáme v `DomainIndex`.

```
id "org.springframework.boot" version "2.7.14"
id "io.spring.dependency-management" version "1.1.3"
```

- Nový klient používá jiné třídy. Proto, pokud s nimi v kódu pracujete, je třeba tyto části upravit. Vycházet můžete z tříd indexů a `DictionaryRepository`.
- Nový klient využívá bulder pattern, přes který lze parametry requestů pouze nastavovat a nelze napřímo číst aktuální hodnoty. Výsledný request objekt po vytvoření již nelze modifikovat. Aby bylo možné zachovat funkcionalitu vyhledávání s queryModifier parametrem, využívá se nyní vlastní definice `SearchRequest`. Ukázka použití je v `DatedIndex.list()`.
- Nelze využít `org.elasticsearch.common.geo.GeoPint` pro field indexovaného obejtku (obsahuje nepodporované fieldy). Jako náhradu může použít `EasGeoPoint`. Jinde v kódu se vlastnosti `GeoPint` mohou dále používat, problém je jen při indexování objektu.