Programowanie z Solrj

RDF RIOTECH DATA FACTORY

Co w module

- Programowanie w Javie z Solr.
- Jak zrobić bardzo szybko prostą bibliotekę do wyszukiwania?
- Indeksacja danych w Solr

Podstawy Solrj

RDF RIOTECH DATA FACTORY

Solrj

- Biblioteka Javy, która pozwala na komunikowanie z Solr

- Można dzięki niej stworzyć aplikację, która wysyła dane do indeksacji oraz przeszukuje Solr

WAŻNE!

Solrj to jedynie API

Odzwierciedlenie w kodzie tego, co już wiesz!

Podstawy

- Wszystkie requesty wysyłane są przez SolrClient.

- To podstawowy obiekt, który “zarządza” wieloma rzeczami.

- SolrClient posiada różne implementacje

Podstawy

SolrClient

- **HttpSolrClient**
- Http2SolrClient
- LBHttpSolrClient
- LBHttp2SolrClient
- **CloudSolrClient**
- ConcurrentUpdateSolrClient
- ConcurrentUpdateHttp2SolrClient

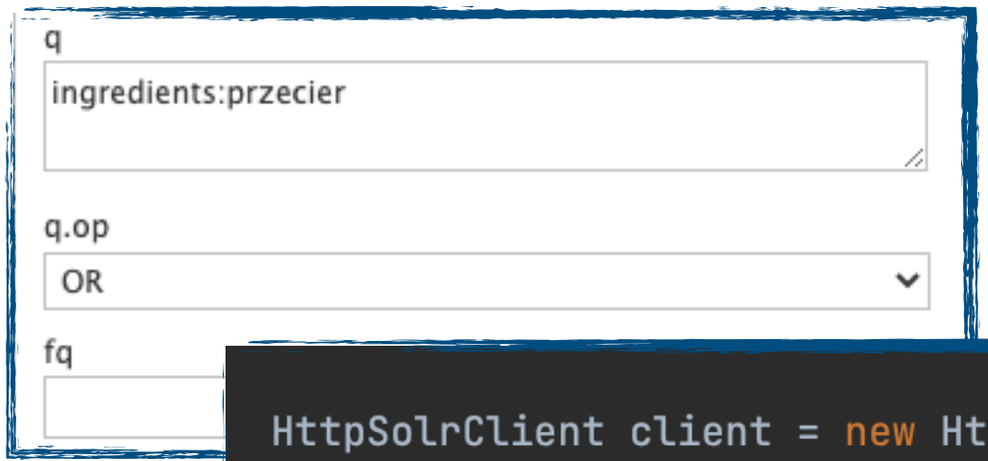
Podstawy

- Requesty reprezentowane są przez SolrRequest
- Odpowiedzi reprezentowane sa przez SolrResponse

Maven

```xml
<dependencies>
    <dependency>
        <groupId>org.apache.solr</groupId>
        <artifactId>solr-solrj</artifactId>
        <version>8.11.0</version>
    </dependency>
</dependencies>
```

Podstawy



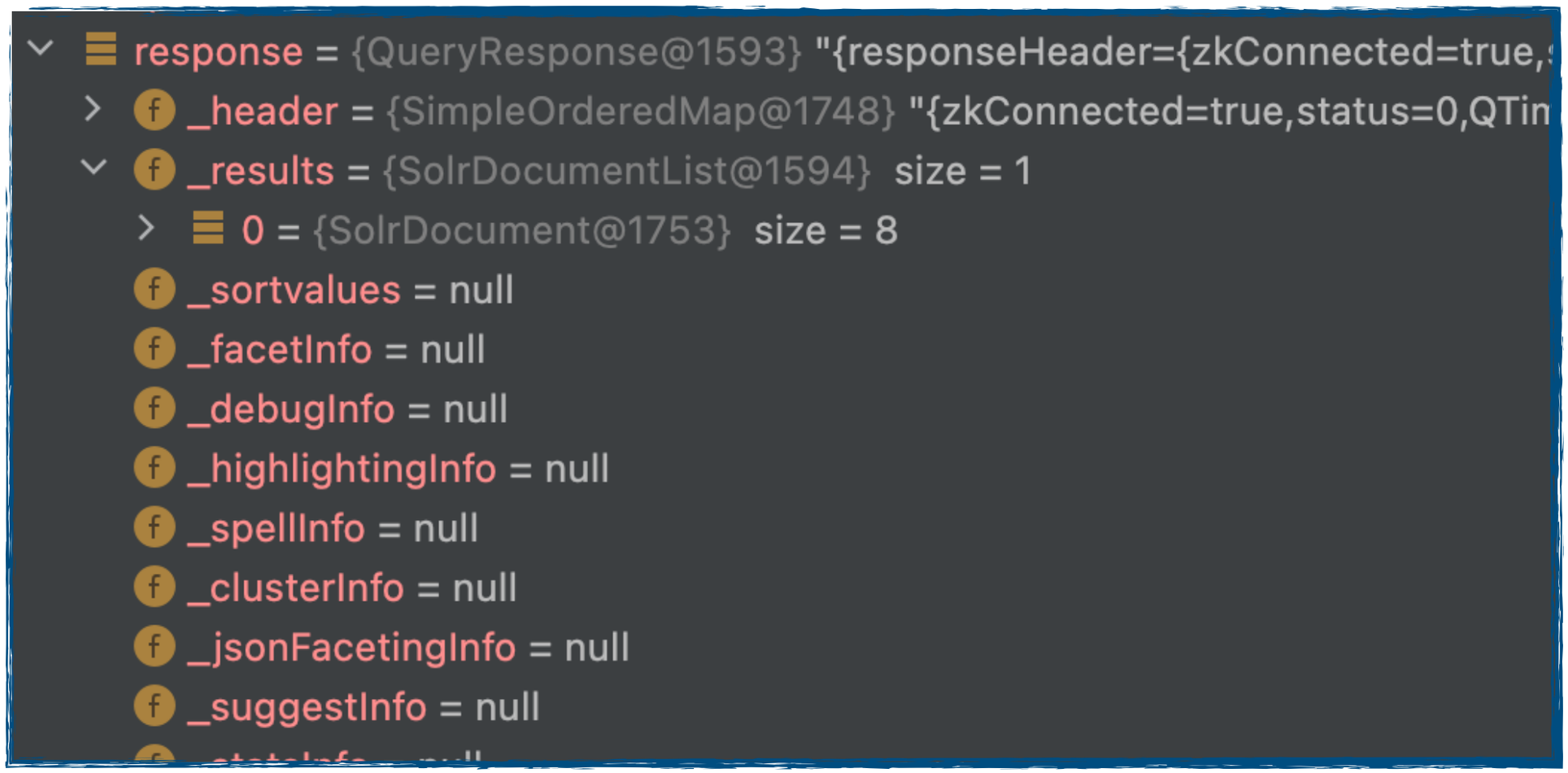
```
HttpSolrClient client = new HttpSolrClient.Builder( baseSolrUrl: "http://localhost:8983/solr/")
        .build();

Map<String, String> queryParamsString = new HashMap<String, String>();
queryParamsString.put("q", "ingredients:przecier");

MapSolrParams queryParams = new MapSolrParams(queryParamsString);

QueryResponse response = client.query( collection: "recipes", queryParams);
SolrDocumentList documents = response.getResults();
```

Response

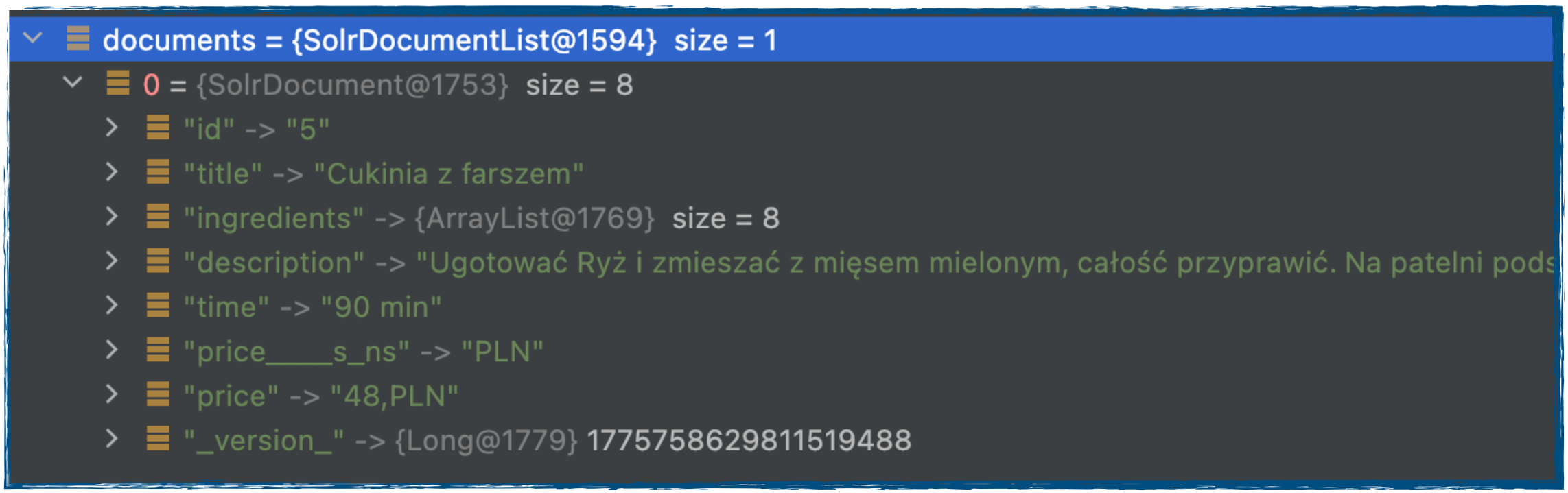

response = {QueryResponse@1593} "{responseHeader={zkConnected=true,
_header = {SimpleOrderedMap@1748} "{zkConnected=true,status=0,QTin
_results = {SolrDocumentList@1594} size = 1
0 = {SolrDocument@1753} size = 8
_sortvalues = null
_facetInfo = null
_debugInfo = null
_highlightingInfo = null
_spellInfo = null
_clusterInfo = null
_jsonFacetingInfo = null
_suggestInfo = null

Documents

http://localhost:8983/solr/recipes/select?indent=true&q.op=OR&q=ingredients%3Aprzecier


```
{
  "responseHeader":{
    "zkConnected":true,
    "status":0,
    "QTime":101,
    "params":{
      "q":"ingredients:przecier",
      "indent":"true",
      "q.op":"OR",
      "_":"1693667548734"}},
  "response":{"numFound":1,"start":0,"numFoundExact":true,"docs":[
      {
        "id":"5",
        "title":"Cukinia z farszem",
        "ingredients":["Mięso mielone wieprzowe.",
          "Ryż",
          "Papryka",
          "Cebula",
          "Przecier/Passata pomidorowa",
          "Cukinie (kilka)",
          "Przyprawy (oregano, majeranek, bazylia, pieprz, sól)",
          "Ser (ewentualnie)"],
        "description":"Ugotować Ryż i zmieszać z mięsem mielonym, całość przyprawić. Na
        "time":"90 min",
        "price___s_ns":"PLN",
        "price":"48,PLN",
        "_version_":1775758629811519488}]
  }}
```

Documents


documents = {SolrDocumentList@1594} size = 1
0 = {SolrDocument@1753} size = 8
"id" -> "5"
"title" -> "Cukinia z farszem"
"ingredients" -> {ArrayList@1769} size = 8
"description" -> "Ugotować Ryż i zmieszać z mięsem mielonym, całość przyprawić. Na patelni pods
"time" -> "90 min"
"price____s_ns" -> "PLN"
"price" -> "48,PLN"
"_version_" -> {Long@1779} 1775758629811519488

Piszemy bibliotekę do wyszukiwania

RDF RIOTECH DATA FACTORY

Cel

- Napisać prostą bibliotekę, która uprości przeszukiwanie tego co trzymamy w Solr

- Biblioteka powinna znaleźć zdrowy środek między uniwersalnością a dostosowaniem do aktualnych potrzeb

- Biblioteka powinna być dość łatwo rozbudowywalna

Funkcjonalności

- Wyszukiwanie po słowach kluczowych
- Możliwość dodania warunku cenowego
- Możliwość przeszukiwania z literówkami lub bez podczas wyszukiwania (fuzzy search)

Jak to osiągnąć?

Przykładowe zastosowanie

(Efekt końcowy)

```
List<Recipe> recipes = new RecipeSearch().turnOnTypo()
        .simpleSearch( ...keys: "przecier", "jaja", "cukinia")
        .addCondition(new CurrencyCondition( fieldName: "price").to( priceValue: "100"))
        .search();
```

Struktura

- Models -> Recipe
- Search (interfejs) -> Recipe Search
- Pomocnicze

Mapowanie na model

- Pobierając dokumenty z Solr możemy zmapować je na naszą strukturę.

- Dokładnie to zrobimy w naszej bibliotece, ponieważ znacznie łatwiej posługiwać się obiektem klasy, niż domyślnego “Document”.

Mapowanie na model

```java
public class Recipe extends SearchModel{
    @Field
    String id;
    @Field
    String title;
    @Field
    List<String> ingredients;
    @Field
    String description;
    @Field
    String time;
    @Field("price____s_ns")
    String priceCurrency;

    1 usage
    String price;

    1 usage
    String priceValue;

    @Field("price")
    protected void setPriceValue(String price){
        this.price = price;
        this.priceValue = price.split( regex: ",")[0];
    }
}
```

```java
MapSolrParams queryParams = new MapSolrParams(queryParamsString);
QueryResponse response = client.query(COLLECTION, queryParams);
List<Recipe> recipes = response.getBeans(Recipe.class);
```

Search (interfejs)

- Założenie jest następujące: klasy służące do przeszukiwania konkretnych modeli powinny umożliwić (jako minimum) przeszukiwanie po kluczowych wyrazach.

- Dodatkowo powinniśmy móc dodać konkretne warunki w query.

- Całość można zrobić w formie funkcji - “tasiemca”.

Pomocnicze

- Wewnątrz searcha pojawiają się także klasy i metody, które są “pomocnicze”.

- Pozwalają one na “uniwersalne” zbudowanie biblioteki.

Do dzieła!

RDF RIOTECH DATA FACTORY

Indeksacja

RDF RIOTECH DATA FACTORY

Podstawy

- Wykorzystujemy *SolrInputDocument*.
- Dodajemy do niego kolejne kolumny.
- Na koniec dokument (lub listę dokumentów) dodajemy do metody client.add()
- Dodatkowo możemy użyć *“commit”*, jeśli nie mamy skonfigurowanego autocommita.

Przykład

```
SolrInputDocument doc = new SolrInputDocument();
doc.addField( name: "id", UUID.randomUUID().toString());
doc.addField( name: "title",  value: "sukces");
doc.addField( name: "ingredients", Arrays.asList("pomysł", "dyscyplina", "szczescie", "konsekwencja"));
doc.addField( name: "description",  value: "Przykładowy przepis na sukces. Liczę, że komuś pomoże:-)");
doc.addField( name: "time",  value: "20y");
doc.addField( name: "price",  value: "1000000,PLN");

UpdateResponse updateResponse = client.add( collection: "recipes", doc);
client.commit( collection: "recipes");
```

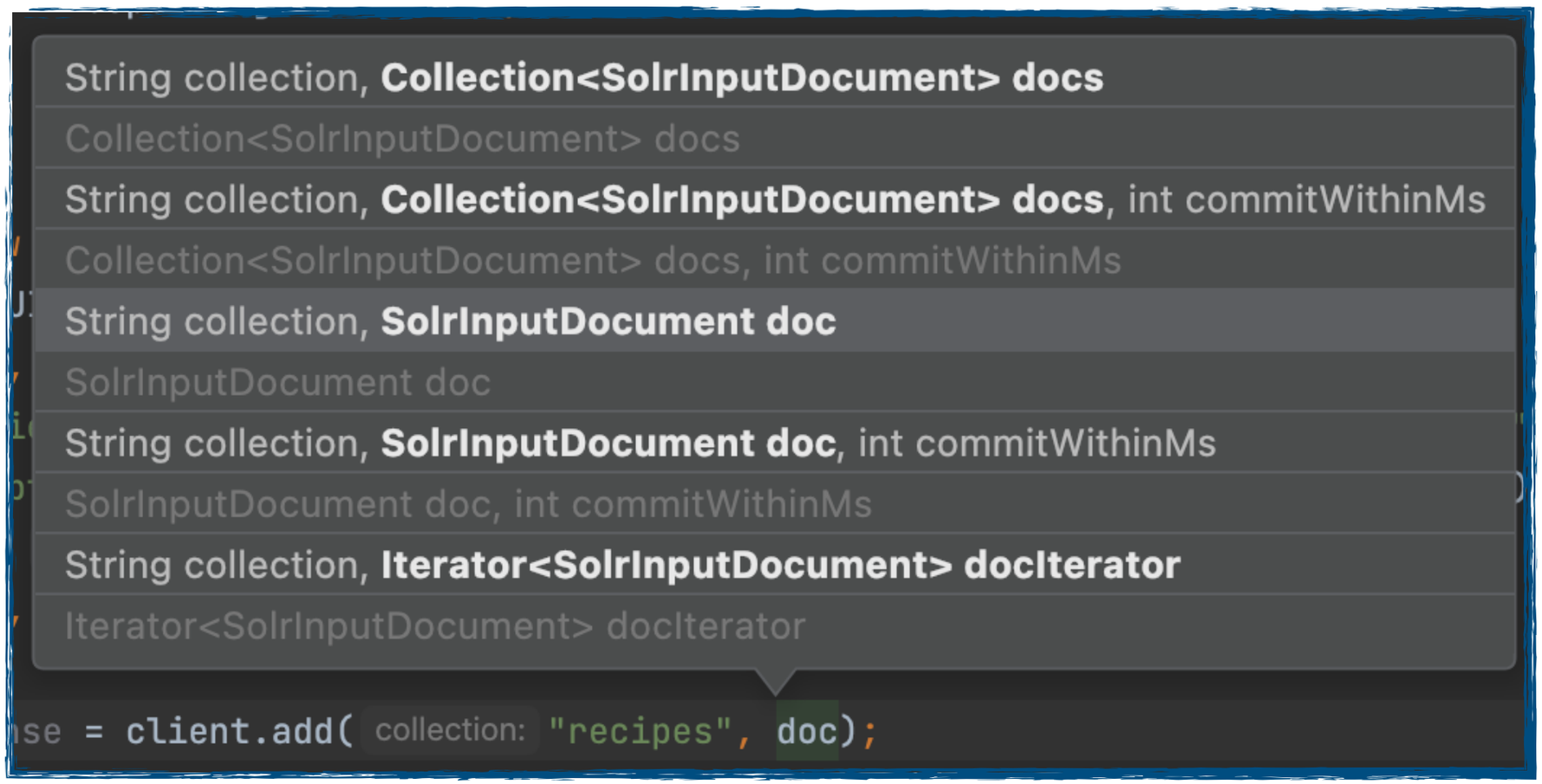
Przykład

```java
SolrInputDocument doc = new SolrInputDocument();
doc.addField( name: "id", UUID.randomUUID().toString());
doc.addField( name: "title", value: "sukces");
doc.addField( name: "ingredients", Arrays.asList("pomysł", "dyscyplina", "szczescie", "konsekwencja"));
doc.addField( name: "description", value: "Przykładowy przepis na sukces. Liczę, że komuś pomoże:-)");
doc.addField( name: "time", value: "20y");
doc.addField( name: "price", value: "1000000,PLN");

UpdateResponse updateRes
client.commit( collection: '
```


```
"response":{"numFound":1,"start":0,"numFoundExact":true,"docs":[
    {
      "id":"9f814e88-8903-4ee5-8f70-76b333a8bf95",
      "title":"sukces",
      "ingredients":["pomysł",
        "dyscyplina",
        "szczescie",
        "konsekwencja"],
      "description":"Przykładowy przepis na sukces. Liczę, że komuś pomoże:-)",
      "time":"20y",
      "price____s_ns":"PLN",
      "price":"1000000,PLN",
      "_version_":1776118872675450880}]
}}
```

Przykład


String collection, Collection<SolrInputDocument> docs
Collection<SolrInputDocument> docs
String collection, Collection<SolrInputDocument> docs, int commitWithinMs
Collection<SolrInputDocument> docs, int commitWithinMs
String collection, SolrInputDocument doc
SolrInputDocument doc
String collection, SolrInputDocument doc, int commitWithinMs
SolrInputDocument doc, int commitWithinMs
String collection, Iterator<SolrInputDocument> docIterator
Iterator<SolrInputDocument> docIterator
se = client.add(collection: "recipes", doc);

Mapowanie

```
HttpSolrClient client = SearchlibSolrClient.getClient();
List<String> ingredients = Arrays.asList("kluski", "Śląsk");

Recipe recipe = new Recipe( id: "131",  title: "Kluski po śląsku", ingredients,
UpdateResponse updateResponse = client.addBean( collection: "recipes", recipe);
client.commit( collection: "recipes");
```

Gratulacje!

Good job;-)

RDF RIOTECH DATA FACTORY