# Introduction to Apache Solr

PRI 23/24 · Information Processing and Retrieval
M.EIC · Master in Informatics Engineering and Computation

Sérgio Nunes
Dept. Informatics Engineering
FEUP · U.Porto

# Outline

- Apache Solr Overview
- Key Solr Concepts
- Indexing: schemas, field types.
- Text Analysis
- Querying: query parsing, filters, ranking.
- Features: nested documents, faceted search, highlighting, query suggestion.

# Apache Solr

# Apache Solr

- “Solr is the popular, blazing-fast, open source enterprise search platform built on Apache Lucene™.” [ solr.apache.org ]

- “Solr is a scalable, ready-to-deploy enterprise search engine that’s optimized to search large volumes of text-centric data and return results sorted by relevance.” [ Solr in Action (2014) ]

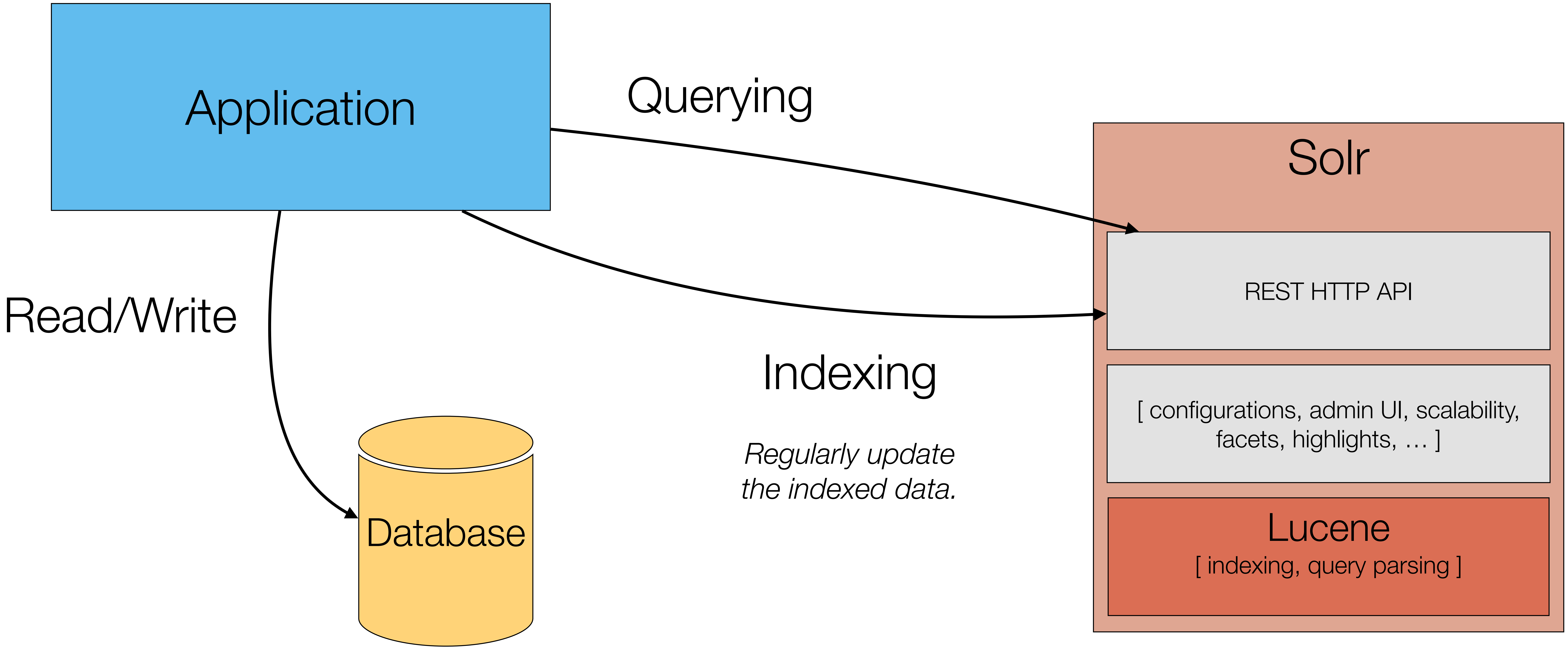
- **Apache Solr** (pronounced “solar”)
  - is an open-source text centric search platform written in Java.
  - uses Apache Lucene for full-text indexing.
  - interaction is based on a REST-like HTTP XML/JSON API.

- Current version is Solr 9.3 — https://solr.apache.org

# Lucene and Solr

- **Apache Lucene** is search library written in Java that provides the fundamental building blocks for implementing indexing and search capabilities.
    - Key features: indexing, searching, ranking.
    - Use cases: directly manage and embed the indexing and integrate search within software’s logic.
- **Apache Solr** is a search platform that uses Lucene and is prepared to be deployed and used as a standalone server in large-scale scenarios with large volumes of data.
    - Key features: scalability, REST API, user admin interface, search features (faceting, highlighting, autocomplete).
    - Use cases: deploy scalable, ready-to-use, search platform that works as a standalone server and is integrated with other services through an API.

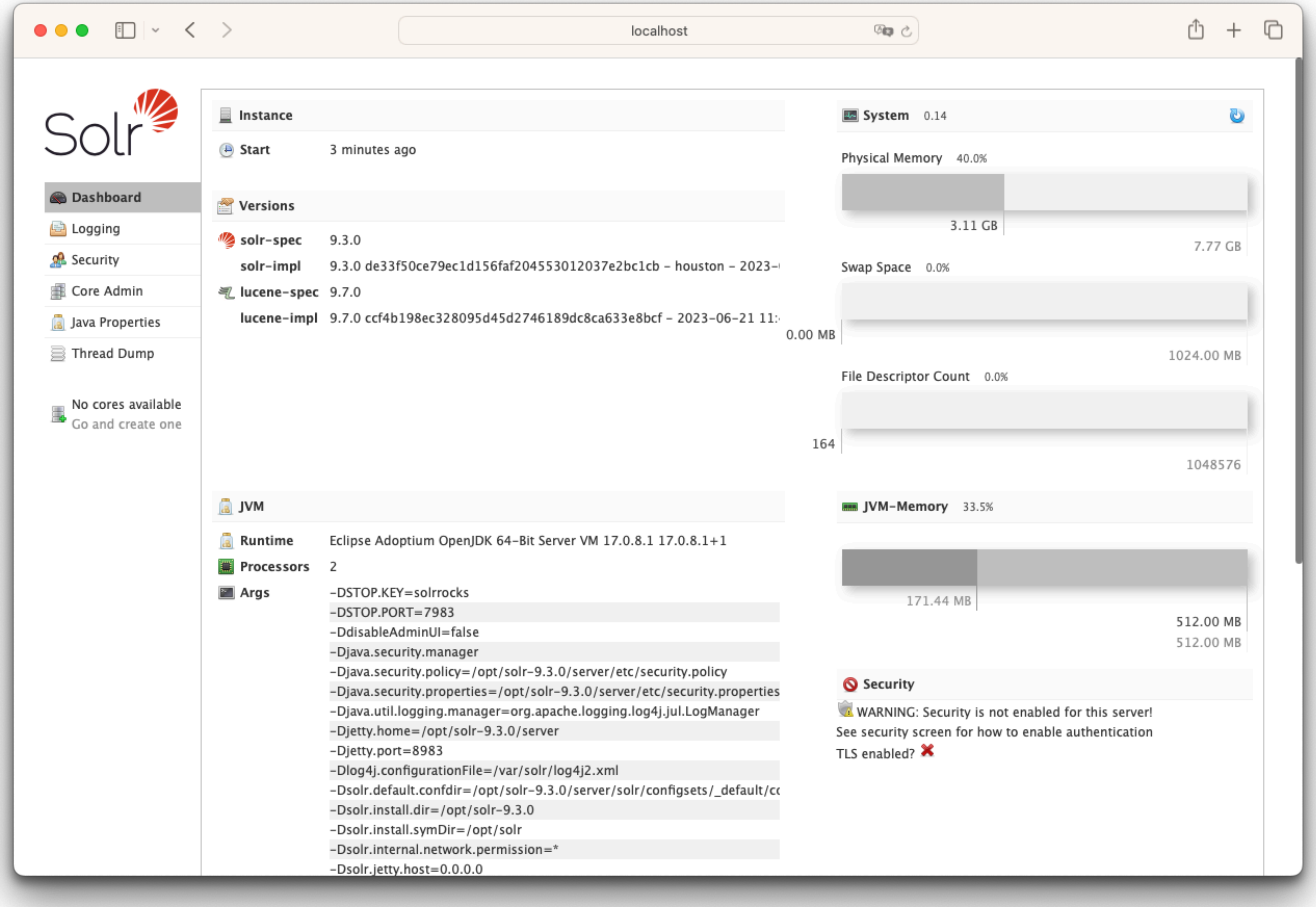
# High-level Overview


Application
Querying
Solr
REST HTTP API
Read/Write
Indexing
Regularly update the indexed data.
[ configurations, admin UI, scalability, facets, highlights, … ]
Database
Lucene
[ indexing, query parsing ]

# Solr Installation

- Official Apache Solr containers are available on Docker Hub.
  - hub.docker.com/_/solr
- You can start a Solr server with:
  - docker run --name my_solr -d -p 8983:8983 solr:9
  - --name, defines a name for the container
  - -d, starts the container in detached mode to free up the terminal
  - -p, maps Solr’s default port from the container to the host machine.
  - Different versions can be selected with ‘solr:*version*’. Omitting the version defaults to the latest.
- Head to http://localhost:8983 to access Solr Admin user interface.

# Solr Admin



**Instance**

**Start** 3 minutes ago

**Versions**

**solr-spec** 9.3.0

**solr-impl** 9.3.0 de33f50ce79ec1d156faf204553012037e2bc1cb - houston - 2023-

**lucene-spec** 9.7.0

**lucene-impl** 9.7.0 ccf4b198ec328095d45d2746189dc8ca633e8bcf - 2023-06-21 11:

**JVM**

**Runtime** Eclipse Adoptium OpenJDK 64-Bit Server VM 17.0.8.1 17.0.8.1+1

**Processors** 2

**Args**

```
-DSTOP.KEY=solrrocks
-DSTOP.PORT=7983
-DdisableAdminUI=false
-Djava.security.manager
-Djava.security.policy=/opt/solr-9.3.0/server/etc/security.policy
-Djava.security.properties=/opt/solr-9.3.0/server/etc/security.properties
-Djava.util.logging.manager=org.apache.logging.log4j.jul.LogManager
-Djetty.home=/opt/solr-9.3.0/server
-Djetty.port=8983
-Dlog4j.configurationFile=/var/solr/log4j2.xml
-Dsolr.default.confdir=/opt/solr-9.3.0/server/solr/configsets/_default/cc
-Dsolr.install.dir=/opt/solr-9.3.0
-Dsolr.install.symDir=/opt/solr
-Dsolr.internal.network.permission=*
-Dsolr.jetty.host=0.0.0.0
```

**System** 0.14

Physical Memory 40.0%

3.11 GB

7.77 GB

Swap Space 0.0%

0.00 MB

1024.00 MB

File Descriptor Count 0.0%

164

1048576

**JVM-Memory** 33.5%

171.44 MB

512.00 MB

512.00 MB

**Security**

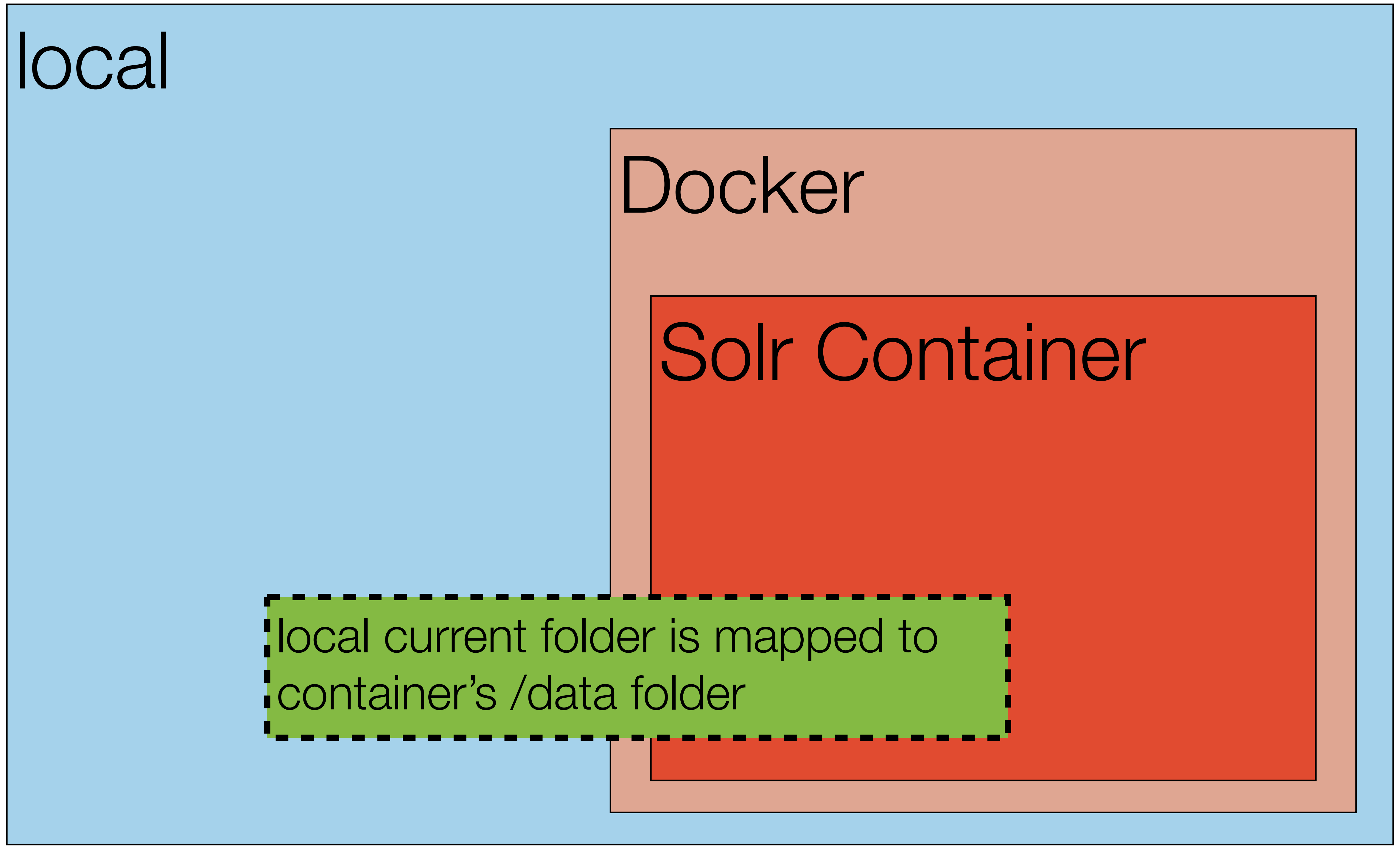
WARNING: Security is not enabled for this server!
See security screen for how to enable authentication
TLS enabled? ✗

# Working with a Solr Docker Container

- It is important to note that when working with a Docker container, you will lose all information and progress when the container is stopped.
- An option to keep a consistent workspace also to facilitate collaboration within your group, is to use configuration scripts.
- The configuration script can perform a set of custom operations on boot, e.g.:
  - Load schema configurations into the container.
  - Load data (e.g., JSON files) into the container.
  - Typical tasks for the script: create cores, configure schemas, load documents.

# Solr Tutorial Setup

- This is how the PRI Solr tutorial is set up.
- docker run -p 8983:8983 --name meic_solr -v ${PWD}:/data -d solr:9.3 solr-precreate courses



# Solr Key Concepts

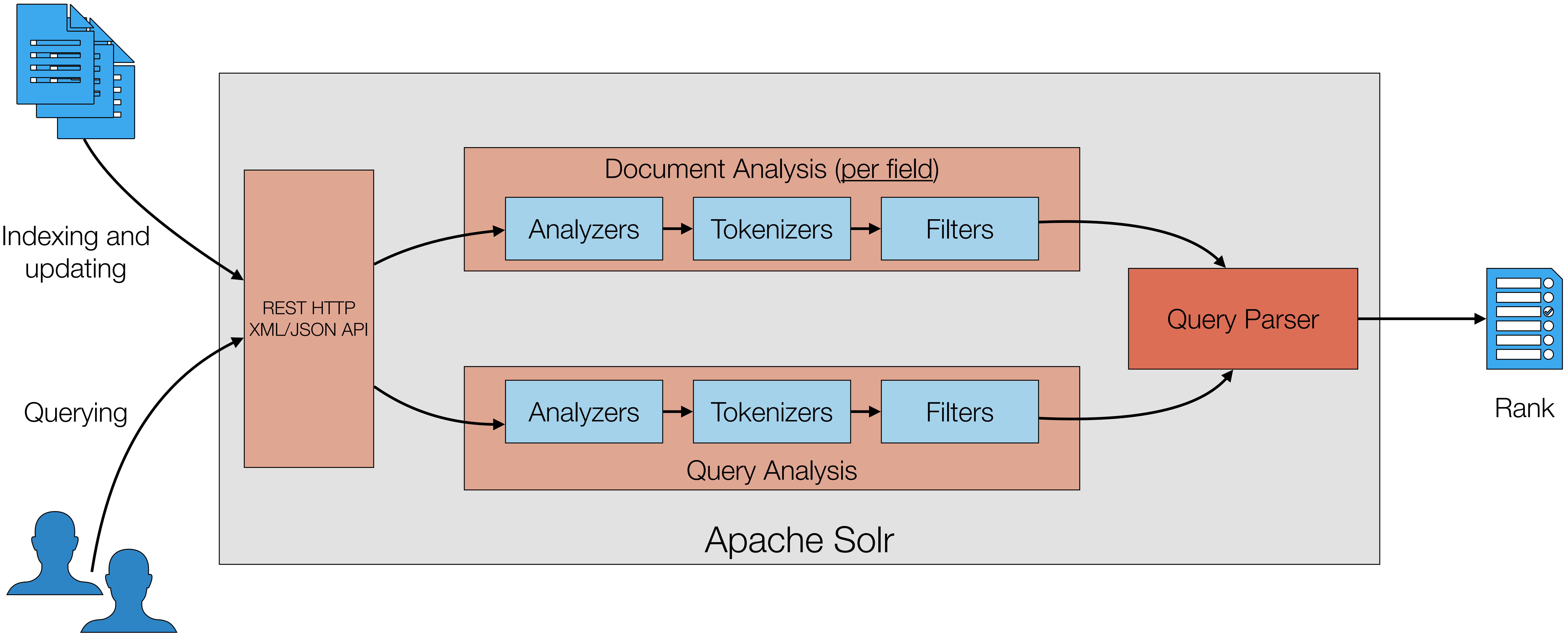
# Solr Key Concepts (1)

- Solr is a search framework that indexes data and then enables retrieval of that data.
- The basic information unit in Solr is a **document** composed of **fields.**
- Document fields can be of specific **types** (date, number, currency, text, uuid, …).
- **Textual fields** go through a pipeline of analyzers, tokenizers, and filters.
  - **Analyzers**, receive a a textual field as input and generates a token stream.
  - **Tokenizers**, receive a character stream and produce a sequence of token objects.
  - **Filters**, examine tokens and transform them (keep, discard, create, modify).
- These pipelines are applied per field in the indexing process and also to the query in the querying process.
- Definitions for field types and field configurations are defined in the **schema file**.

# Solr Key Concepts (2)

- A Solr instance can have multiple cores (or indexes).
- A Solr **core** stores information for the indexed documents.
- Solr supports a REST-like HTTP XML/JSON **API** for both indexing and querying.
- Queries, just like documents being indexed, also go through a pipeline of analyzers, tokenizer, and filters.
  - But the pipelines for documents and and queries can be different.
- **Query parsers** convert a search string into a Lucene query and find the matching documents.
  - Different query parsers exist to support different search requirements.
  - E.g. the standard query parser, eDisMax (Extended DisMax) query parser.

# Solr Pipeline

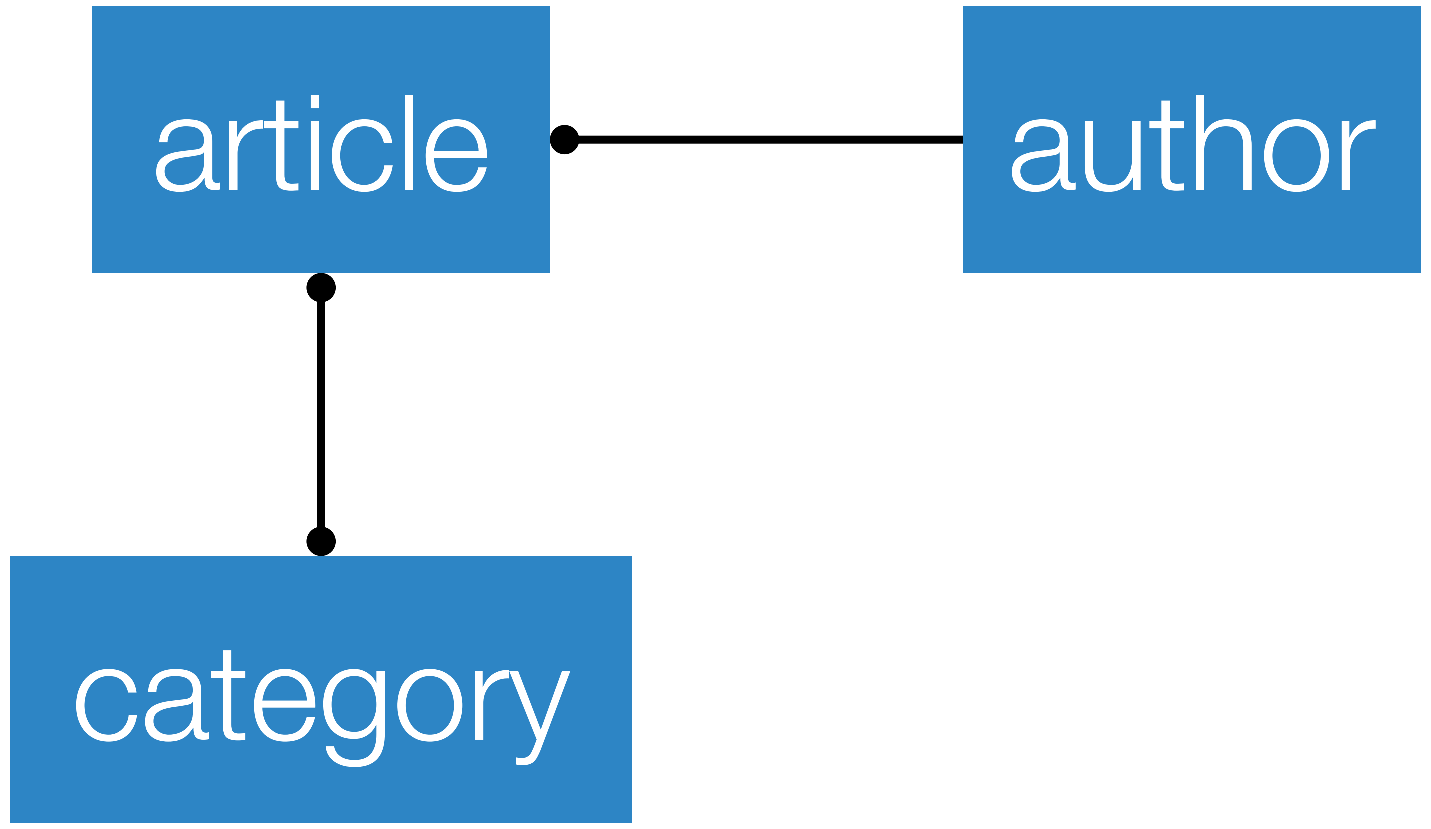


# Example: News Articles

# Setup

# Document Model

- ➤ Recall that we are indexing documents.
- ➤ Relational representations need to be mapped to documents.



```
[
…
  {
    "title": "Governo disponível para reforço de meios na Madeira",
    "pubDate": "2023-10-13 14:58:00",
    "link": "https://www.tsf.pt/portugal/sociedade/governo-disponivel-para-
reforco-de-meios-na-madeira-17162298.html",
    "author": "Lusa",
    "content": "O ministro da Administração Interna deixa uma palavra …”,
    "categories": [
      "Sociedade",
      "incêndio na madeira",
      "nacional",
      "Portugal",
      "MAI",
      "madeira"
    ]
  },
…
]
```

# Start Apache Solr and Create a Core

- **Start Solr** using Docker.
  - docker run --name pri_solr -d -p 8983:8983 solr
  - Parameters: run, named, detached, port mapping, latest Solr version.
- **Check if Solr is running** going to Solr Admin
  - http://localhost:8983
- **Create a new core** executing the command inside the container
  - docker exec pri_solr solr create_core -c ptnews
- You can verify if the new core was created using Solr Admin.

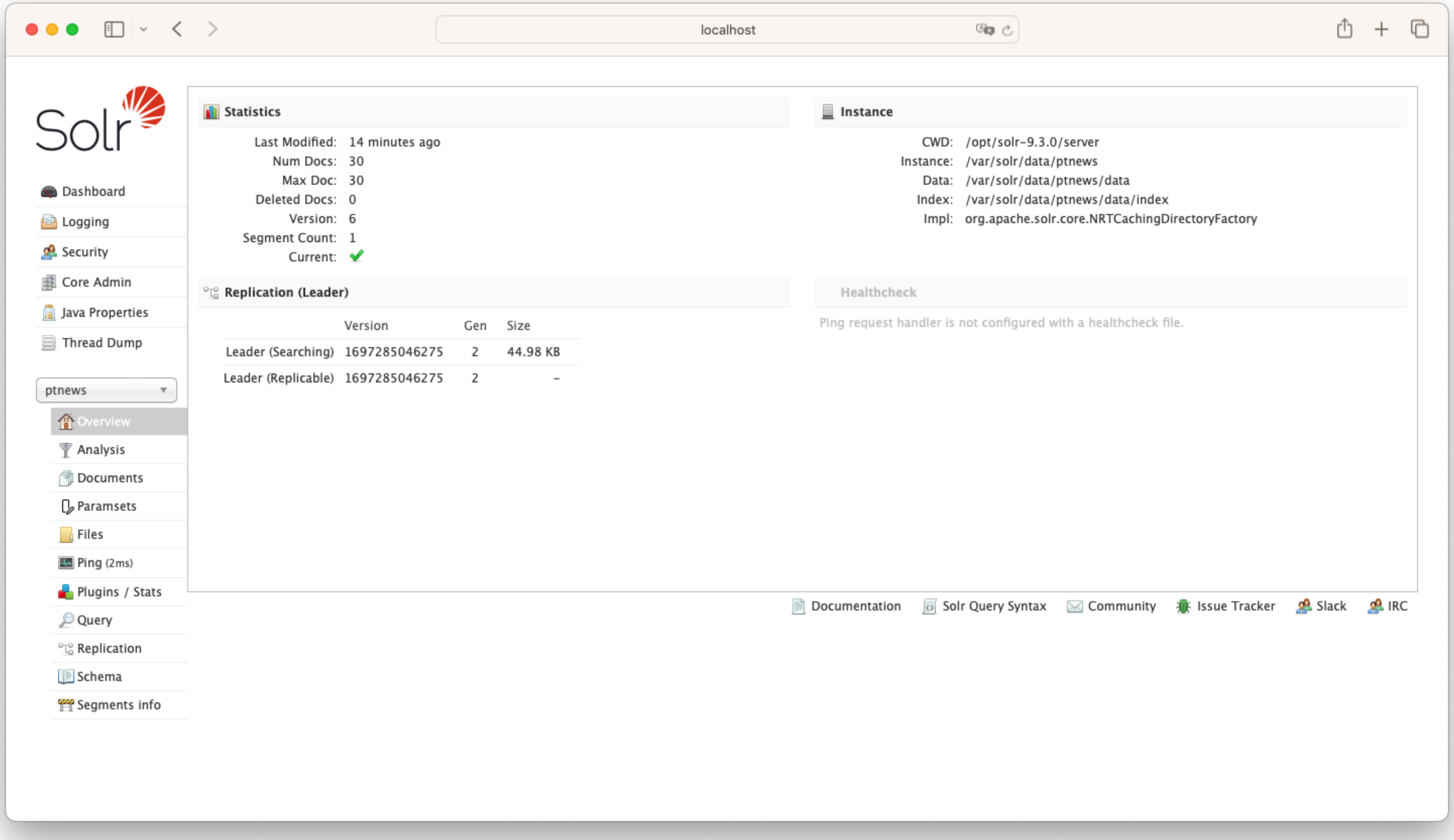
# Indexing

# Index Documents (Schemaless)

- Documents can be posted (indexed) into a newly created core without further definitions. This is referred to as “schemaless mode”.
- The are two main ways to post document to a core, using the API or the post tool.
- **Using the post tool** requires that the data is available to the container.
  - One option is to map a local folder to a container folder, as shown in the tutorial.
  - Another option is to use “docker cp …” to copy the data into the container.
- **Using the API** requires using curl to submit an HTTP POST command.

  -
    ```
    curl -X POST -H 'Content-type:application/json' \
    --data-binary "@./pt-news.json" \
    http://localhost:8983/solr/ptnews/update\?commit\=true
    ```

# Core Overview



**Statistics**

Last Modified: 14 minutes ago
Num Docs: 30
Max Doc: 30
Deleted Docs: 0
Version: 6
Segment Count: 1
Current: ✓

**Instance**

CWD: /opt/solr-9.3.0/server
Instance: /var/solr/data/ptnews
Data: /var/solr/data/ptnews/data
Index: /var/solr/data/ptnews/data/index
Impl: org.apache.solr.core.NRTCachingDirectoryFactory

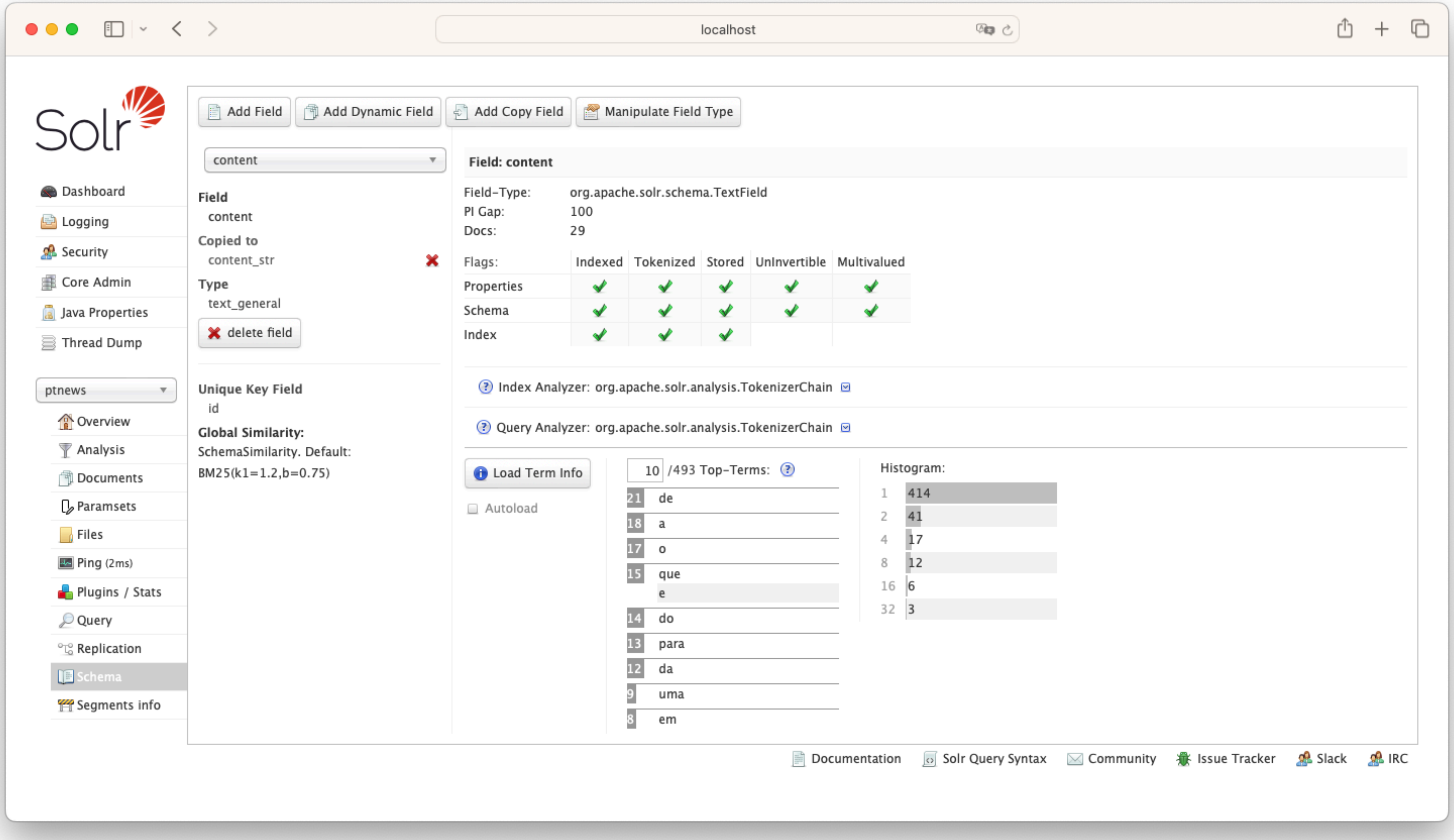
**Replication (Leader)**

| | Version | Gen | Size |
|---|---|---|---|
| Leader (Searching) | 1697285046275 | 2 | 44.98 KB |
| Leader (Replicable) | 1697285046275 | 2 | - |

**Healthcheck**

Ping request handler is not configured with a healthcheck file.

# Schema Statistics



**Field**
content

**Copied to**
content_str

**Type**
text_general

**Unique Key Field**
id

**Global Similarity:**
SchemaSimilarity. Default: BM25(k1=1.2,b=0.75)

**Field: content**

Field-Type: org.apache.solr.schema.TextField
PI Gap: 100
Docs: 29

| Flags: | Indexed | Tokenized | Stored | UnInvertible | Multivalued |
|---|---|---|---|---|---|
| Properties | ✓ | ✓ | ✓ | ✓ | ✓ |
| Schema | ✓ | ✓ | ✓ | ✓ | ✓ |
| Index | ✓ | ✓ | ✓ | | |

Index Analyzer: org.apache.solr.analysis.TokenizerChain

Query Analyzer: org.apache.solr.analysis.TokenizerChain

10 /493 Top-Terms:

| | |
|---|---|
| 21 | de |
| 18 | a |
| 17 | o |
| 15 | que |
| | e |
| 14 | do |
| 13 | para |
| 12 | da |
| 9 | uma |
| 8 | em |

Histogram:

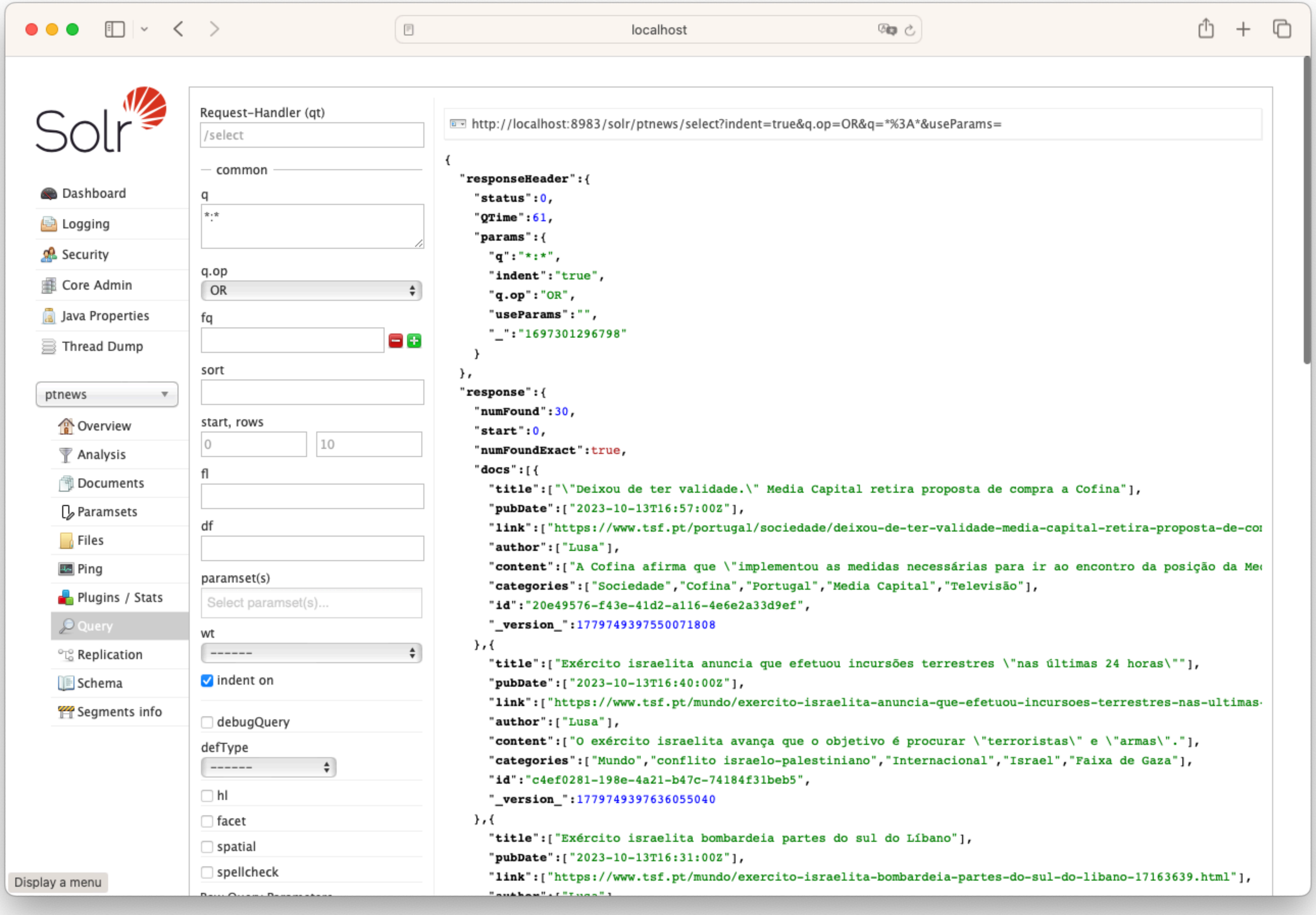
| | |
|---|---|
| 1 | 414 |
| 2 | 41 |
| 4 | 17 |
| 8 | 12 |
| 16 | 6 |
| 32 | 3 |

# Querying

# Schemaless

# Performing Queries (1)

- Queries can be performed **directly using Solr Admin** > Query
  - Example: q: “content:portugal”
- Or **submitting an HTTP GET JSON request** using curl (or other tool for HTTP requests).
  - curl http://localhost:8983/solr/ptnews/query -d '{ "query": "*:*"}'
  - curl http://localhost:8983/solr/ptnews/query -d '{ "query": "content:portugal"}'
  - See documentation Solr: JSON Request API (also includes query, update, delete)



Request-Handler (qt): /select

common

q: *:*

q.op: OR

fq:

sort:

start, rows: 0 10

fl:

df:

paramset(s): Select paramset(s)...

wt: ------

☑ indent on

□ debugQuery

defType: ------

□ hl

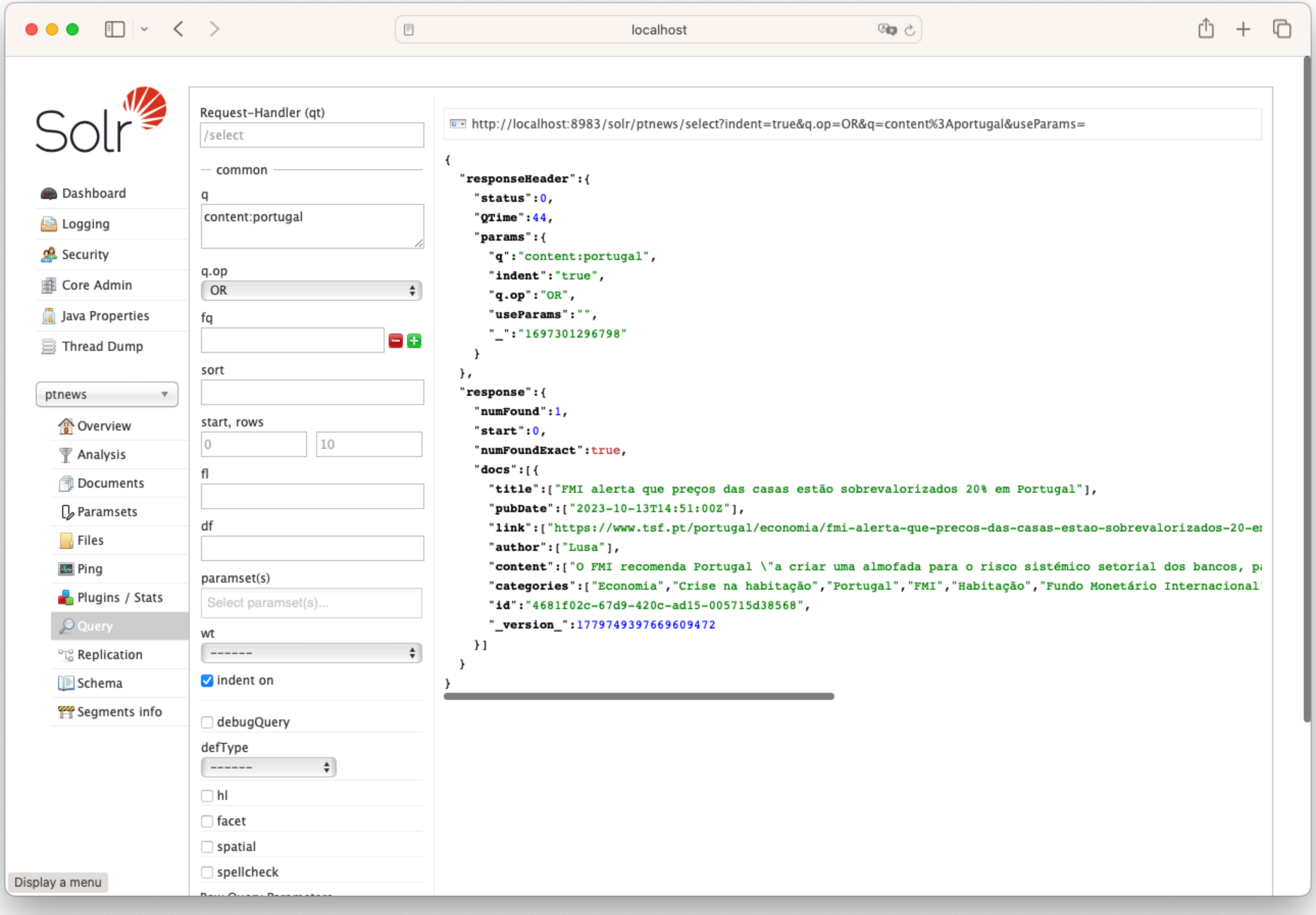
□ facet

□ spatial

□ spellcheck

http://localhost:8983/solr/ptnews/select?indent=true&q.op=OR&q=*%3A*&useParams=

```
{
  "responseHeader":{
    "status":0,
    "QTime":61,
    "params":{
      "q":"*:*",
      "indent":"true",
      "q.op":"OR",
      "useParams":"",
      "_":"1697301296798"
    }
  },
  "response":{
    "numFound":30,
    "start":0,
    "numFoundExact":true,
    "docs":[{
      "title":["\"Deixou de ter validade.\" Media Capital retira proposta de compra a Cofina"],
      "pubDate":["2023-10-13T16:57:00Z"],
      "link":["https://www.tsf.pt/portugal/sociedade/deixou-de-ter-validade-media-capital-retira-proposta-de-co
      "author":["Lusa"],
      "content":["A Cofina afirma que \"implementou as medidas necessárias para ir ao encontro da posição da Me
      "categories":["Sociedade","Cofina","Portugal","Media Capital","Televisão"],
      "id":"20e49576-f43e-41d2-a116-4e6e2a33d9ef",
      "_version_":1779749397550071808
    },{
      "title":["Exército israelita anuncia que efetuou incursões terrestres \"nas últimas 24 horas\""],
      "pubDate":["2023-10-13T16:40:00Z"],
      "link":["https://www.tsf.pt/mundo/exercito-israelita-anuncia-que-efetuou-incursoes-terrestres-nas-ultimas-
      "author":["Lusa"],
      "content":["O exército israelita avança que o objetivo é procurar \"terroristas\" e \"armas\"."],
      "categories":["Mundo","conflito israelo-palestiniano","Internacional","Israel","Faixa de Gaza"],
      "id":"c4ef0281-198e-4a21-b47c-74184f31beb5",
      "_version_":1779749397636055040
    },{
      "title":["Exército israelita bombardeia partes do sul do Líbano"],
      "pubDate":["2023-10-13T16:31:00Z"],
      "link":["https://www.tsf.pt/mundo/exercito-israelita-bombardeia-partes-do-sul-do-libano-17163639.html"],
```



Request-Handler (qt)
/select

common

q
content:portugal

q.op
OR

fq

sort

start, rows
0 10

fl

df

paramset(s)
Select paramset(s)...

wt
------

☑ indent on

□ debugQuery

defType
------

□ hl

□ facet

□ spatial

□ spellcheck

http://localhost:8983/solr/ptnews/select?indent=true&q.op=OR&q=content%3Aportugal&useParams=

```
{
  "responseHeader":{
    "status":0,
    "QTime":44,
    "params":{
      "q":"content:portugal",
      "indent":"true",
      "q.op":"OR",
      "useParams":"",
      "_":"1697301296798"
    }
  },
  "response":{
    "numFound":1,
    "start":0,
    "numFoundExact":true,
    "docs":[{
      "title":["FMI alerta que preços das casas estão sobrevalorizados 20% em Portugal"],
      "pubDate":["2023-10-13T14:51:00Z"],
      "link":["https://www.tsf.pt/portugal/economia/fmi-alerta-que-precos-das-casas-estao-sobrevalorizados-20-e
      "author":["Lusa"],
      "content":["O FMI recomenda Portugal \"a criar uma almofada para o risco sistémico setorial dos bancos, p
      "categories":["Economia","Crise na habitação","Portugal","FMI","Habitação","Fundo Monetário Internacional
      "id":"4681f02c-67d9-420c-ad15-005715d38568",
      "_version_":1779749397669609472
    }]
  }
}
```

# Performing Queries (2)

- Queries can also be executed **using direct HTTP requests** (i.e. no JSON):
  - curl 'http://localhost:8983/solr/ptnews/select?q=content:portugal'

- Example on how to **define the fields to retrieve**:
  - curl ‘http://localhost:8983/solr/ptnews/select?q=*&fl=id,title'

- Limitations of the **lack of schema**. Compare:
  - curl http://localhost:8983/solr/ptnews/query -d '{ "query": "content:orcamento"}'
  - curl http://localhost:8983/solr/ptnews/query -d '{ "query": "content:orçamento"}'

# Match Analysis (not match)

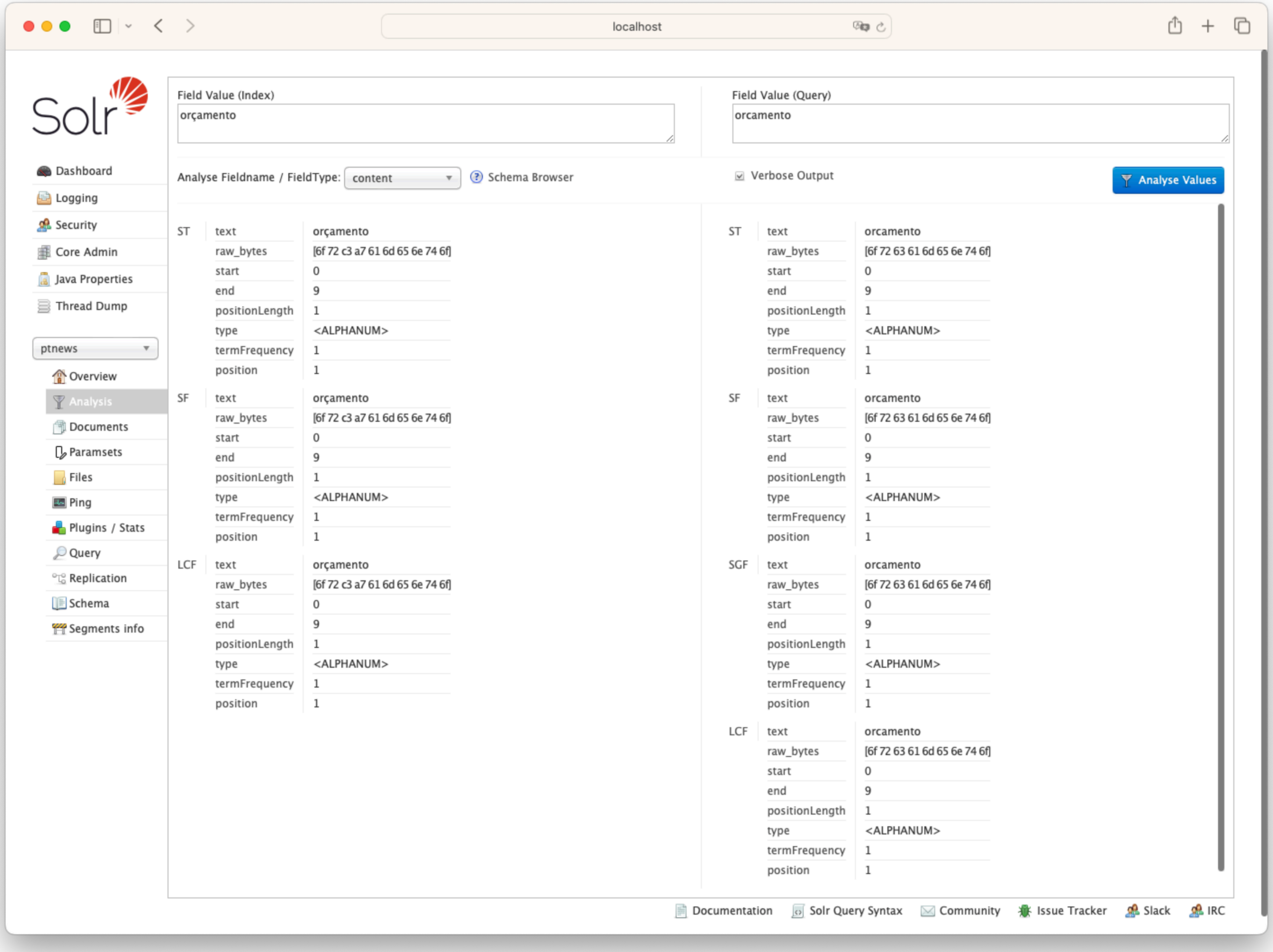
Using the word [ orcamento ] in the “Query” and [ orçamento ] in the “index” there is no match.

ST: Standard Tokenizer

SF: Stop Filter

SGF: Synonym Graph Filter

LCF: Lower Case Filter



Field Value (Index)

orçamento

Field Value (Query)

orcamento

Analyse Fieldname / FieldType: content

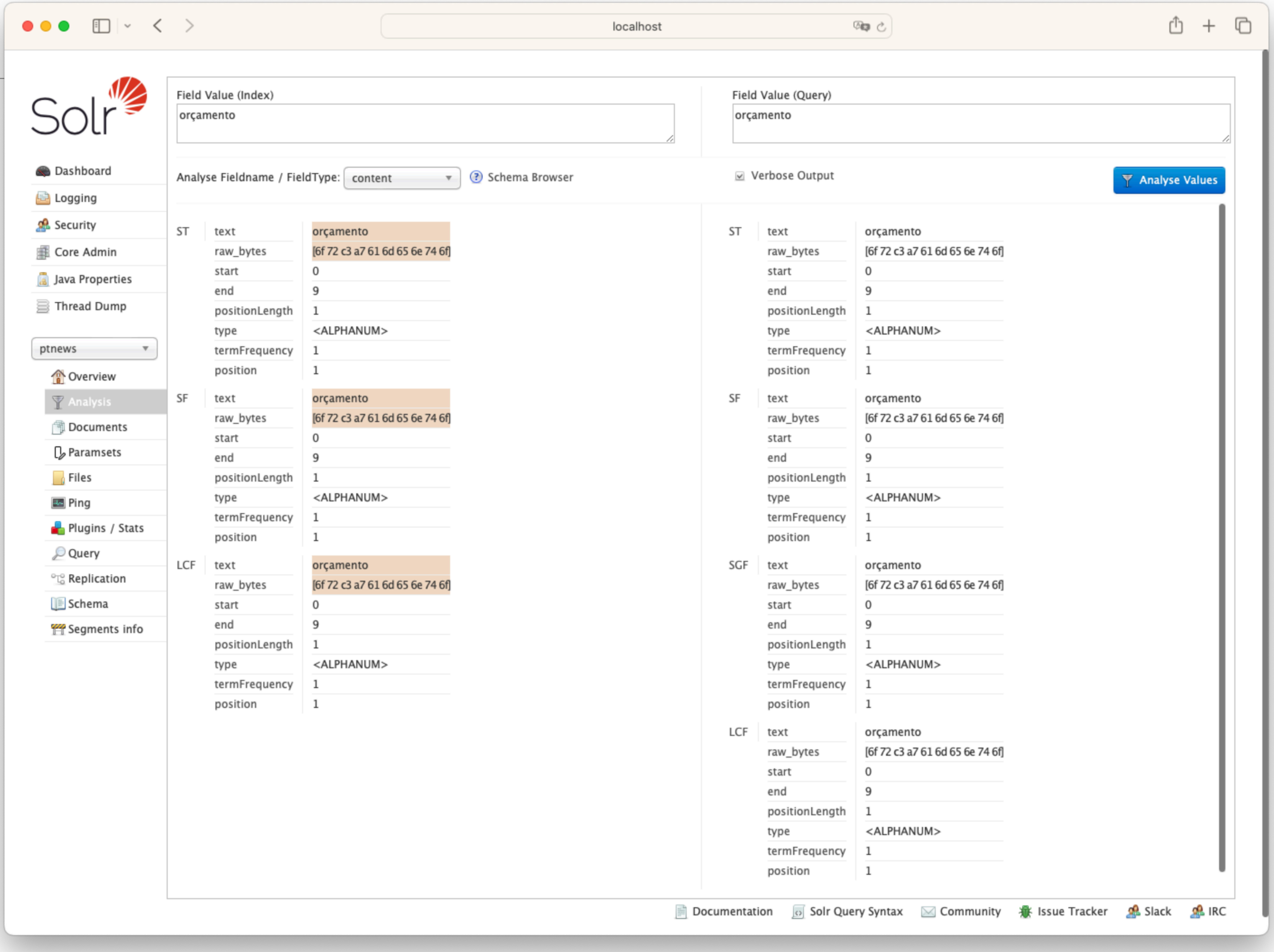
☑ Verbose Output

| | | |
|---|---|---|
| ST | text | orçamento |
| | raw_bytes | [6f 72 c3 a7 61 6d 65 6e 74 6f] |
| | start | 0 |
| | end | 9 |
| | positionLength | 1 |
| | type | <ALPHANUM> |
| | termFrequency | 1 |
| | position | 1 |
| SF | text | orçamento |
| | raw_bytes | [6f 72 c3 a7 61 6d 65 6e 74 6f] |
| | start | 0 |
| | end | 9 |
| | positionLength | 1 |
| | type | <ALPHANUM> |
| | termFrequency | 1 |
| | position | 1 |
| LCF | text | orçamento |
| | raw_bytes | [6f 72 c3 a7 61 6d 65 6e 74 6f] |
| | start | 0 |
| | end | 9 |
| | positionLength | 1 |
| | type | <ALPHANUM> |
| | termFrequency | 1 |
| | position | 1 |

| | | |
|---|---|---|
| ST | text | orcamento |
| | raw_bytes | [6f 72 63 61 6d 65 6e 74 6f] |
| | start | 0 |
| | end | 9 |
| | positionLength | 1 |
| | type | <ALPHANUM> |
| | termFrequency | 1 |
| | position | 1 |
| SF | text | orcamento |
| | raw_bytes | [6f 72 63 61 6d 65 6e 74 6f] |
| | start | 0 |
| | end | 9 |
| | positionLength | 1 |
| | type | <ALPHANUM> |
| | termFrequency | 1 |
| | position | 1 |
| SGF | text | orcamento |
| | raw_bytes | [6f 72 63 61 6d 65 6e 74 6f] |
| | start | 0 |
| | end | 9 |
| | positionLength | 1 |
| | type | <ALPHANUM> |
| | termFrequency | 1 |
| | position | 1 |
| LCF | text | orcamento |
| | raw_bytes | [6f 72 63 61 6d 65 6e 74 6f] |
| | start | 0 |
| | end | 9 |
| | positionLength | 1 |
| | type | <ALPHANUM> |
| | termFrequency | 1 |
| | position | 1 |

# Match Analysis (match)

Using the same word [ orçamento ] in both the “index” value and the “Query” value results (obviously) in a match.



Field Value (Index): orçamento

Field Value (Query): orçamento

Analyse Fieldname / FieldType: content

☑ Verbose Output

| | | |
|---|---|---|
| ST | text | orçamento |
| | raw_bytes | [6f 72 c3 a7 61 6d 65 6e 74 6f] |
| | start | 0 |
| | end | 9 |
| | positionLength | 1 |
| | type | <ALPHANUM> |
| | termFrequency | 1 |
| | position | 1 |
| SF | text | orçamento |
| | raw_bytes | [6f 72 c3 a7 61 6d 65 6e 74 6f] |
| | start | 0 |
| | end | 9 |
| | positionLength | 1 |
| | type | <ALPHANUM> |
| | termFrequency | 1 |
| | position | 1 |
| LCF | text | orçamento |
| | raw_bytes | [6f 72 c3 a7 61 6d 65 6e 74 6f] |
| | start | 0 |
| | end | 9 |
| | positionLength | 1 |
| | type | <ALPHANUM> |
| | termFrequency | 1 |
| | position | 1 |

| | | |
|---|---|---|
| ST | text | orçamento |
| | raw_bytes | [6f 72 c3 a7 61 6d 65 6e 74 6f] |
| | start | 0 |
| | end | 9 |
| | positionLength | 1 |
| | type | <ALPHANUM> |
| | termFrequency | 1 |
| | position | 1 |
| SF | text | orçamento |
| | raw_bytes | [6f 72 c3 a7 61 6d 65 6e 74 6f] |
| | start | 0 |
| | end | 9 |
| | positionLength | 1 |
| | type | <ALPHANUM> |
| | termFrequency | 1 |
| | position | 1 |
| SGF | text | orçamento |
| | raw_bytes | [6f 72 c3 a7 61 6d 65 6e 74 6f] |
| | start | 0 |
| | end | 9 |
| | positionLength | 1 |
| | type | <ALPHANUM> |
| | termFrequency | 1 |
| | position | 1 |
| LCF | text | orçamento |
| | raw_bytes | [6f 72 c3 a7 61 6d 65 6e 74 6f] |
| | start | 0 |
| | end | 9 |
| | positionLength | 1 |
| | type | <ALPHANUM> |
| | termFrequency | 1 |
| | position | 1 |

# Schemas

# Solr Schema

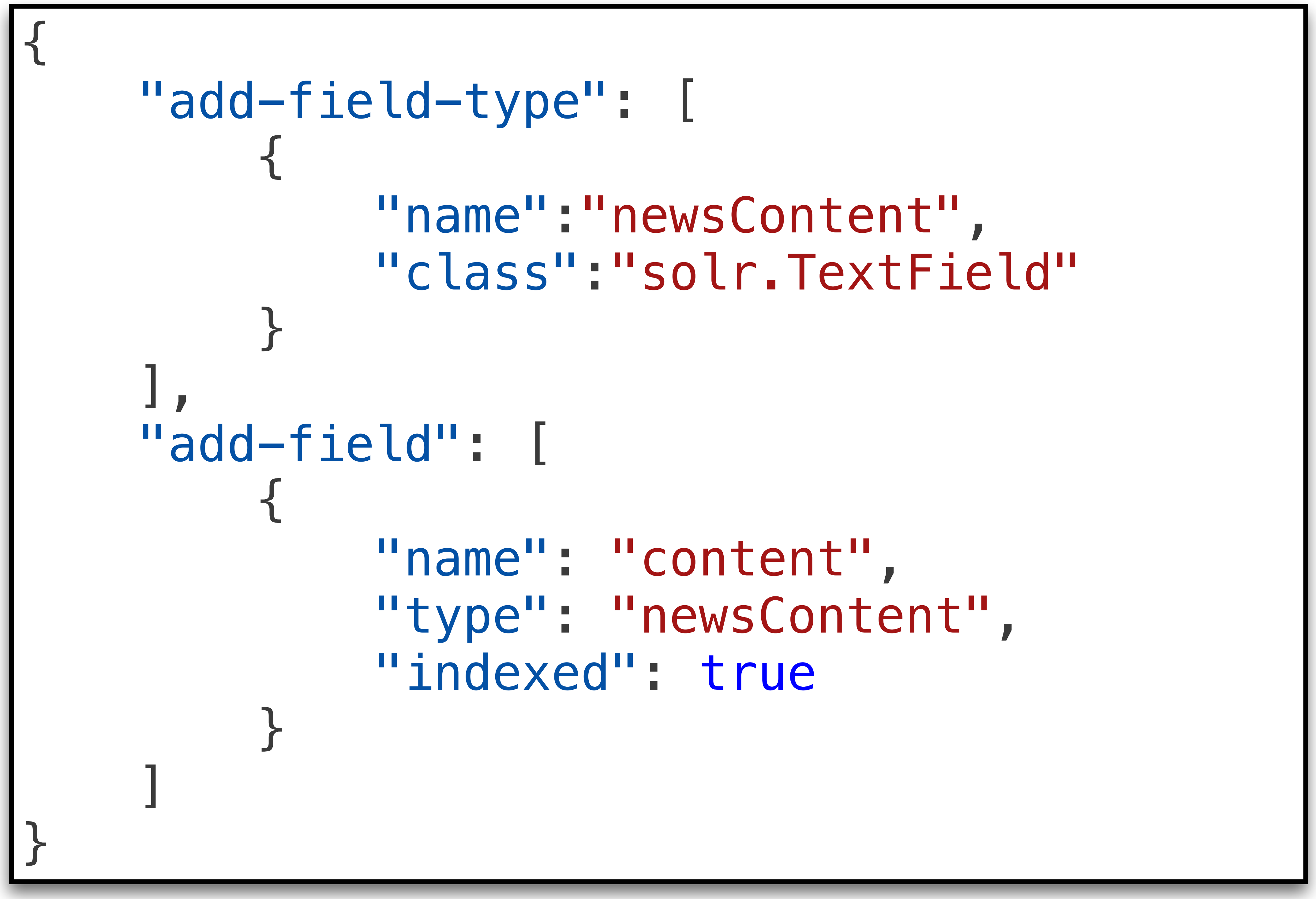
- A Solr Schema defines the configuration of fields and field types for a given core.
- Default field types include boolean, string, text_general, etc.
- Field types can also be configured based on the default ones.
- Three types of fields can be defined:
    - **Fields**, are the specific fields defined — e.g., title of type string and content of type text.
    - **dynamicFields**, are used to index fields not explicitly defined in your schema, i.e. identical to a regular field but application is based on a wildcard — e.g., define all fields ending in “_txt” as text_general.
    - **copyFields**, automatically copy the value of a given field to another. Use case: perform different transformations to ingested values, e.g., remove punctuation from a text but keep the original for displaying.

# Solr Schema Type Definition (1)

- The Schema API is used to set a core's schema definition.
- The schema definition can be provided in JSON format.
  - Reference: Solr: Schema API
- In the next example we define a new type "newsContent" of type TextField.
  - Reference: Solr: Field Types Included with Solr
- Note that to define a new schema, the previous one needs to be deleted.
  - Easiest way is to delete and create.
  - docker exec pri_solr solr delete -c ptnews

# Solr Schema Type Definition (2)

- To load a schema defined in a JSON file use:
- curl -X POST -H 'Content-type:application/json' \
  --data-binary "@./ptnews-schema.json" \
  http://localhost:8983/solr/ptnews/schema
- And then index the documents.
- Verify the new type definition in Solr Admin.



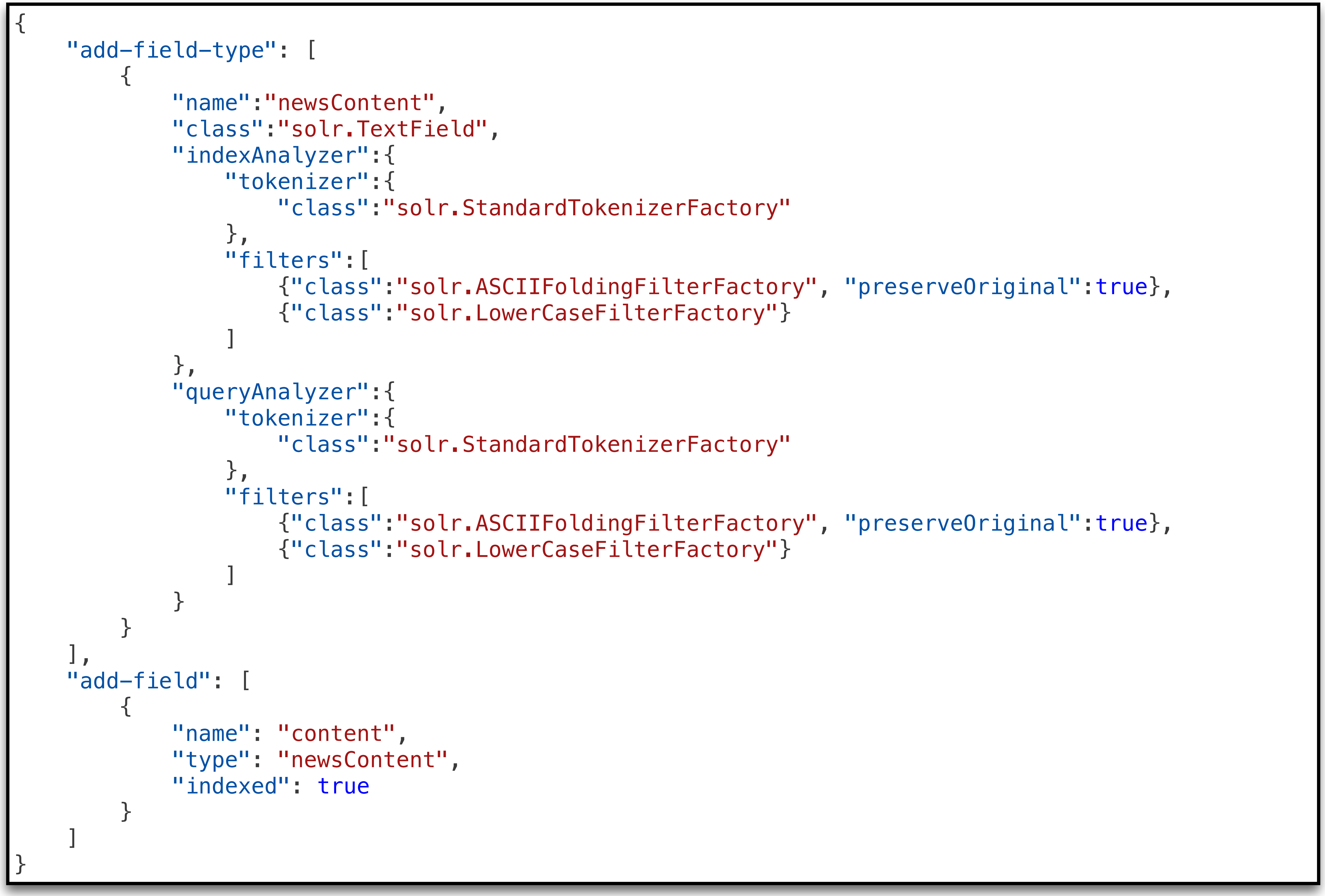
```
{
    "add-field-type": [
        {
            "name":"newsContent",
            "class":"solr.TextField"
        }
    ],
    "add-field": [
        {
            "name": "content",
            "type": "newsContent",
            "indexed": true
        }
    ]
}
```

# Analyzers

- The schema definition can include, for each field type, definitions for:
  - **indexAnalyzer**, transformations to perform as the documents are indexed. These transformations are applied to the indexed terms, not the stored values.
  - **queryAnalyzer**, transformations to perform when queries are processed.

- Analyzers can include **one tokenizer and multiple filters**.

# Analyzers Definition



```
{
    "add-field-type": [
        {
            "name":"newsContent",
            "class":"solr.TextField",
            "indexAnalyzer":{
                "tokenizer":{
                    "class":"solr.StandardTokenizerFactory"
                },
                "filters":[
                    {"class":"solr.ASCIIFoldingFilterFactory", "preserveOriginal":true},
                    {"class":"solr.LowerCaseFilterFactory"}
                ]
            },
            "queryAnalyzer":{
                "tokenizer":{
                    "class":"solr.StandardTokenizerFactory"
                },
                "filters":[
                    {"class":"solr.ASCIIFoldingFilterFactory", "preserveOriginal":true},
                    {"class":"solr.LowerCaseFilterFactory"}
                ]
            }
        }
    ],
    "add-field": [
        {
            "name": "content",
            "type": "newsContent",
            "indexed": true
        }
    ]
}
```

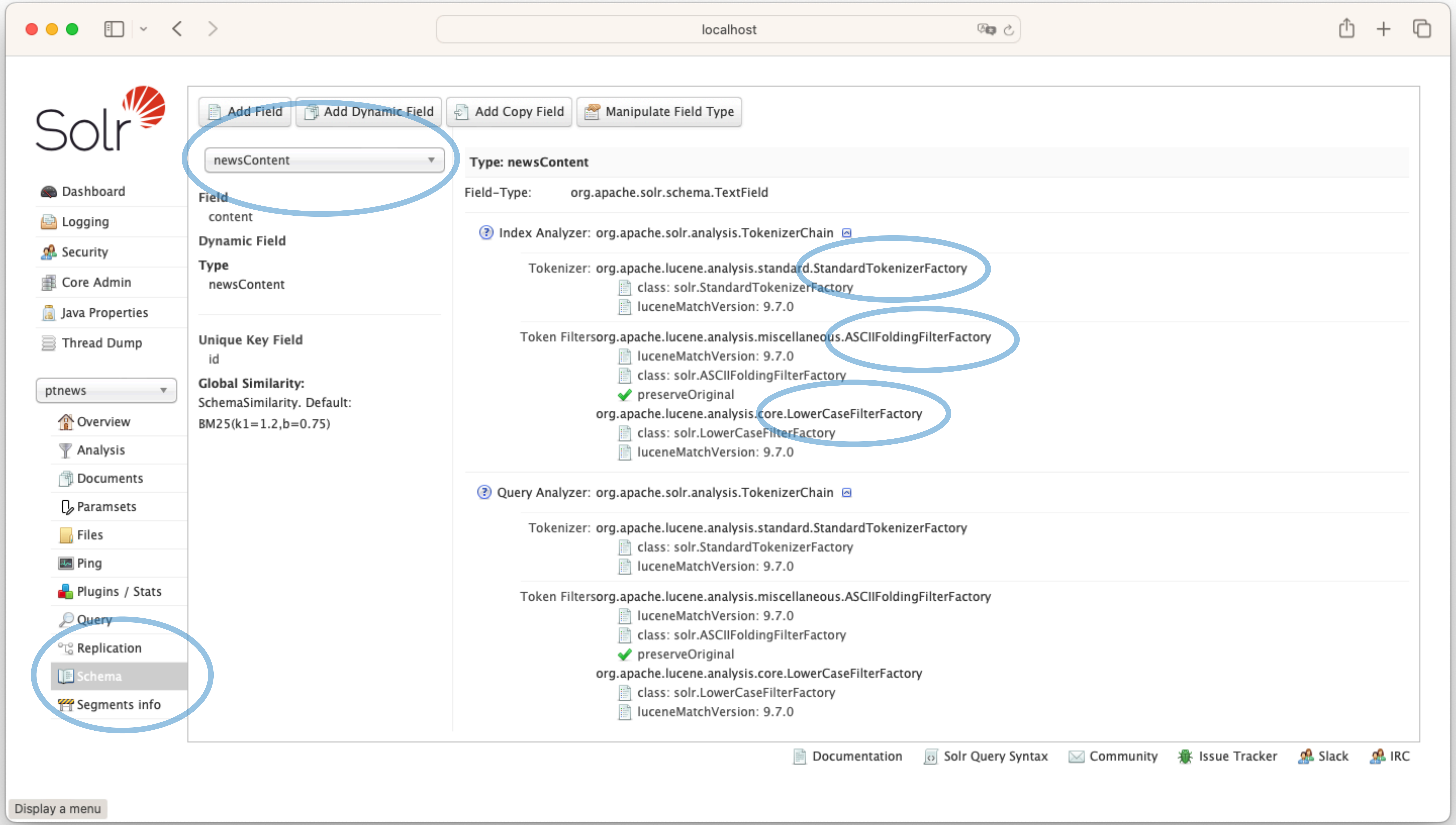
# Tokenizers

- Tokenizers break the input text stream into a stream of tokens.
- Solr built-in tokenizers: Solr: Tokenizers (see example inputs and outputs).
- Example tokenizers:
  - **Standard Tokenizer**, splits the text field into tokens, treating whitespace and punctuation as delimiters.
  - **Lower Case Tokenizer**, tokenizes the input stream by delimiting at non-letters and then converting all letters to lowercase. Whitespace and non-letters are discarded.
  - **N-Gram Tokenizer**, reads the field text and generates n-gram tokens of sizes in the given range.

# Filters

- Filters processes a stream of tokens and generates a different set of tokens.
- Solr built-in tokenizers: Solr: Filters (see in / out examples).
- Example filters:
  - **ASCII Folding Filter**, this filter converts alphabetic, numeric, and symbolic Unicode characters to their ASCII equivalents, if one exists.
  - **Lower Case Filter**, converts any uppercase letters in a token to the equivalent lowercase token. All other characters are left unchanged.
  - **Stop Filter**, this filter discards, or stops analysis of, tokens that are on the given stop words list. A standard stop words list is included in the Solr conf directory, named stopwords.txt, which is appropriate for typical English language text.
  - **Snowball Porter Stemmer Filter**, applied a language-specific stemmer generated by Snowball, a software package that generates pattern-based word stemmers. Includes built-in support for Portuguese.

# Schema Definitions



**Field**
content

**Dynamic Field**

**Type**
newsContent

**Unique Key Field**
id

**Global Similarity:**
SchemaSimilarity. Default: BM25(k1=1.2,b=0.75)

**Type: newsContent**

Field-Type: org.apache.solr.schema.TextField

Index Analyzer: org.apache.solr.analysis.TokenizerChain

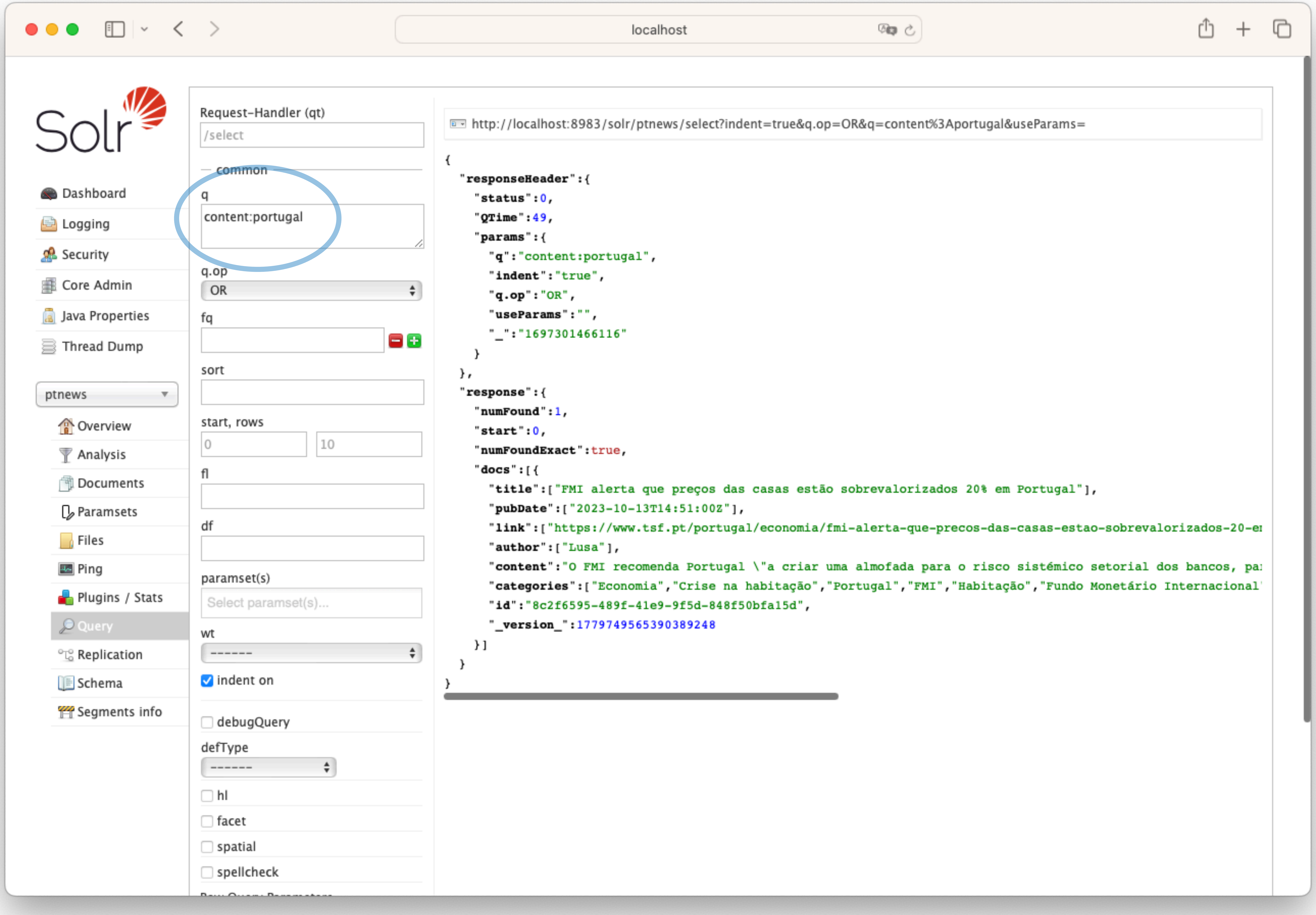
Tokenizer: org.apache.lucene.analysis.standard.StandardTokenizerFactory
class: solr.StandardTokenizerFactory
luceneMatchVersion: 9.7.0

Token Filtersorg.apache.lucene.analysis.miscellaneous.ASCIIFoldingFilterFactory
luceneMatchVersion: 9.7.0
class: solr.ASCIIFoldingFilterFactory
✓ preserveOriginal
org.apache.lucene.analysis.core.LowerCaseFilterFactory
class: solr.LowerCaseFilterFactory
luceneMatchVersion: 9.7.0

Query Analyzer: org.apache.solr.analysis.TokenizerChain

Tokenizer: org.apache.lucene.analysis.standard.StandardTokenizerFactory
class: solr.StandardTokenizerFactory
luceneMatchVersion: 9.7.0

Token Filtersorg.apache.lucene.analysis.miscellaneous.ASCIIFoldingFilterFactory
luceneMatchVersion: 9.7.0
class: solr.ASCIIFoldingFilterFactory
✓ preserveOriginal
org.apache.lucene.analysis.core.LowerCaseFilterFactory
class: solr.LowerCaseFilterFactory
luceneMatchVersion: 9.7.0



Request-Handler (qt)
/select

common

q
content:portugal

q.op
OR

fq

sort

start, rows
0 10

fl

df

paramset(s)
Select paramset(s)...

wt
------

☑ indent on

□ debugQuery

defType
------

□ hl

□ facet

□ spatial

□ spellcheck

http://localhost:8983/solr/ptnews/select?indent=true&q.op=OR&q=content%3Aportugal&useParams=

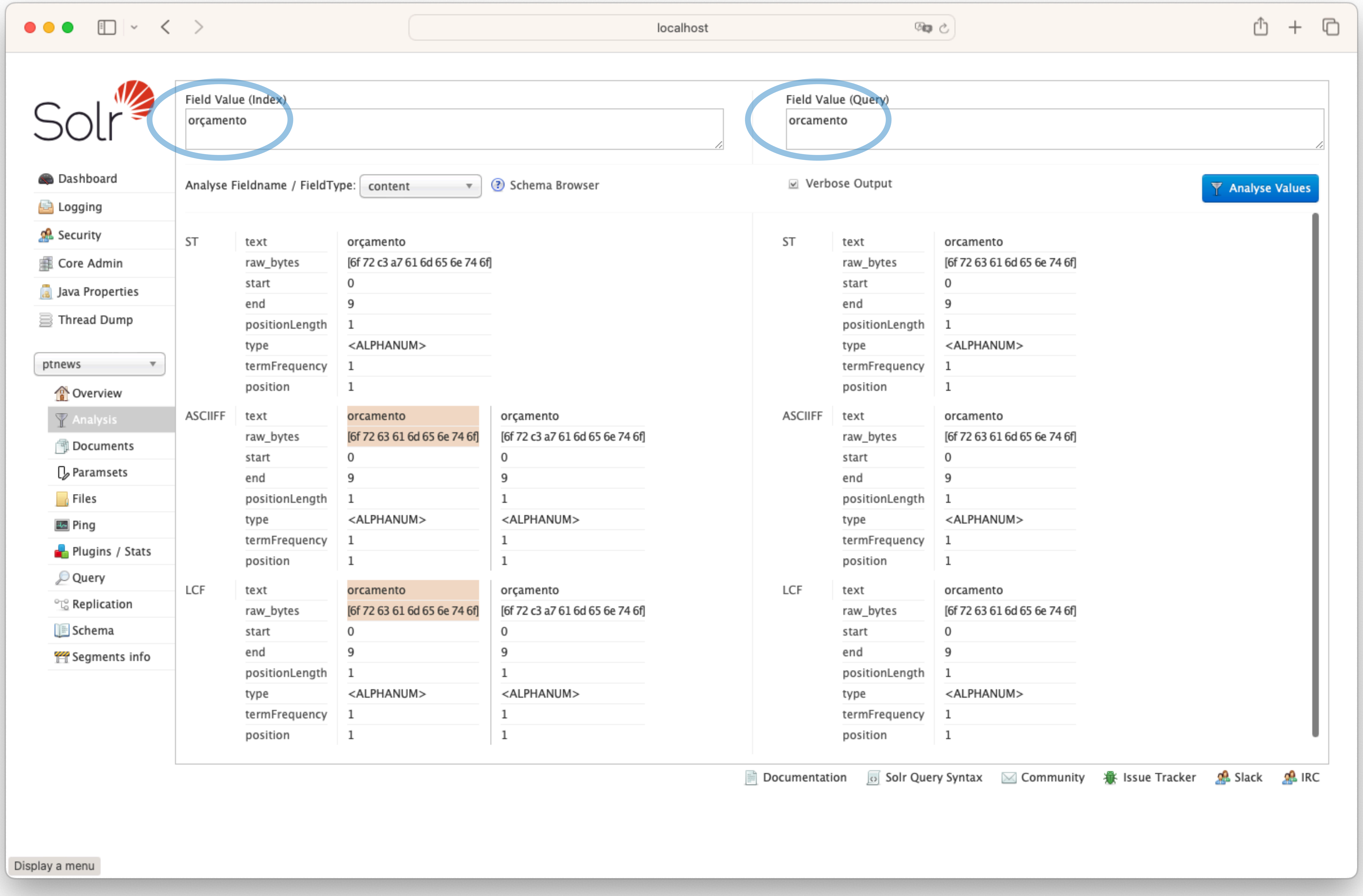
```
{
  "responseHeader":{
    "status":0,
    "QTime":49,
    "params":{
      "q":"content:portugal",
      "indent":"true",
      "q.op":"OR",
      "useParams":"",
      "_":"1697301466116"
    }
  },
  "response":{
    "numFound":1,
    "start":0,
    "numFoundExact":true,
    "docs":[{
      "title":["FMI alerta que preços das casas estão sobrevalorizados 20% em Portugal"],
      "pubDate":["2023-10-13T14:51:00Z"],
      "link":["https://www.tsf.pt/portugal/economia/fmi-alerta-que-precos-das-casas-estao-sobrevalorizados-20-e
      "author":["Lusa"],
      "content":"O FMI recomenda Portugal \"a criar uma almofada para o risco sistémico setorial dos bancos, pa
      "categories":["Economia","Crise na habitação","Portugal","FMI","Habitação","Fundo Monetário Internacional
      "id":"8c2f6595-489f-41e9-9f5d-848f50bfa15d",
      "_version_":1779749565390389248
    }]
  }
}
```

# Match Analysis

ST: Standard Tokenizer

ASCIIFF: ASCII Folder Filter

LCF: Lower Case Filter



Field Value (Index)
orçamento

Field Value (Query)
orcamento

Analyse Fieldname / FieldType: content
Schema Browser
☑ Verbose Output

| | | | |
|---|---|---|---|
| ST | text | orçamento | |
| | raw_bytes | [6f 72 c3 a7 61 6d 65 6e 74 6f] | |
| | start | 0 | |
| | end | 9 | |
| | positionLength | 1 | |
| | type | <ALPHANUM> | |
| | termFrequency | 1 | |
| | position | 1 | |
| ASCIIFF | text | orcamento | orçamento |
| | raw_bytes | [6f 72 63 61 6d 65 6e 74 6f] | [6f 72 c3 a7 61 6d 65 6e 74 6f] |
| | start | 0 | 0 |
| | end | 9 | 9 |
| | positionLength | 1 | 1 |
| | type | <ALPHANUM> | <ALPHANUM> |
| | termFrequency | 1 | 1 |
| | position | 1 | 1 |
| LCF | text | orcamento | orçamento |
| | raw_bytes | [6f 72 63 61 6d 65 6e 74 6f] | [6f 72 c3 a7 61 6d 65 6e 74 6f] |
| | start | 0 | 0 |
| | end | 9 | 9 |
| | positionLength | 1 | 1 |
| | type | <ALPHANUM> | <ALPHANUM> |
| | termFrequency | 1 | 1 |
| | position | 1 | 1 |

| | | |
|---|---|---|
| ST | text | orcamento |
| | raw_bytes | [6f 72 63 61 6d 65 6e 74 6f] |
| | start | 0 |
| | end | 9 |
| | positionLength | 1 |
| | type | <ALPHANUM> |
| | termFrequency | 1 |
| | position | 1 |
| ASCIIFF | text | orcamento |
| | raw_bytes | [6f 72 63 61 6d 65 6e 74 6f] |
| | start | 0 |
| | end | 9 |
| | positionLength | 1 |
| | type | <ALPHANUM> |
| | termFrequency | 1 |
| | position | 1 |
| LCF | text | orcamento |
| | raw_bytes | [6f 72 63 61 6d 65 6e 74 6f] |
| | start | 0 |
| | end | 9 |
| | positionLength | 1 |
| | type | <ALPHANUM> |
| | termFrequency | 1 |
| | position | 1 |

# Query Parsers

# Solr Query Parsers

- Different query parsers can be used to match documents to queries.
- The **standard query parser** offers an intuitive syntax but is very strict, i.e. is very intolerant to syntax errors.
- The **DisMax query parser** is designed to through as little errors as possible, being appropriate for consumer facing applications.
- The **Extended DisMax query parser** is an improved version that is both forgiving in the syntax and also supports complex query expressions.

# eDisMax Query Parser

- The query parser to use can be defined with the ‘defType’ parameter.
- For both the DisMax and eDisMax query parser the ‘qf’ parameter is available, defining the list of fields where the search should be executed.
  - Instead of using:
    - q=title:flower+AND+content:flower+AND+summary:flower
  - We can simply use:
    - defType=edismax&qf=title+content+summary&q=flower
- A debug parameter is available to debug query execution — debug=all
- See Solr: Common Query Parameters

# Weighting Fields

- Document fields can be weighted differently, i.e. contribute differently to estimate the relevance of a document.
- The ‘qf’ parameter can be used to specify relative field weights.
  - qf=title^5+content+summary^3
- Additional information to understand ranking decisions can be obtained with ‘debug’ and ‘debug.explain.structured’ parameters.
  - debug=all&debug.explain.structure=true

# Additional Topics

# Indexing

- **Language Detection**, identify languages and map text to language-specific fields during indexing.
  - Ref: Solr: Language Detection
- **De-Duplication**, preventing duplicate or near duplicate documents from entering an index or tagging documents with a signature/fingerprint for duplicate field collapsing.
  - Ref: Solr: De-Duplication
- Working with **nested documents**:
  - Solr: Indexing Nested Documents
  - Solr: Searching Nested Child Documents

# Enhancing Queries

- **Spell Checking**, provides inline query suggestions based on other, similar, terms
  - Ref: Solr: Spell Checking
- **Suggester** (auto complete), provides users with automatic suggestions for query terms.
  - Ref: Solr: Suggester
- **MoreLikeThis**, enables queries for documents similar to a document in their result list.
  - Ref: Solr: MoreLikeThis
- **Query Re-Ranking**, run a simple query (A) for matching documents and then re-rank the top N documents using the scores from a more complex query (B).
  - Ref: Solr: Query Re-Ranking
- **Learning to Rank** (LtoR), can configure and run machine learned ranking models.
  - Ref: Solr: Learning to Rank

# Controlling Results

- **Faceting**, arrangement of search results into categories based on indexed terms.
    - Ref: Solr: Faceting
- **Highlighting**, fragments of documents that match the user’s query to be included with the query response.
    - Ref: Solr: Highlighting

# References

**Relevant Search: With applications for Solr and Elasticsearch**
Doug Turnbull and John Berryman
Manning, 2016

**Solr in Action**
Trey Grainger and Timothy Potter
Manning, 2014