



 zalando

COPING WITH THE CHALLENGE OF SORTING LARGE PRODUCT CATALOGS



ONLINE - SHOP WINDOW
ARRANGEMENT



CAGDAS SENOL

19-06-2019





TABLE OF CONTENTS

- Challenges
- Data Driven Sorting
- Three Improvements for faster iteration
- One Open Source Contribution
- Status Quo
- Q&A

ZALANDO AT A GLANCE

~ 5.4 billion EUR
revenue 2018

> 15,500
employees in
Europe

> 80%
of visits via
mobile devices

**> 300
million** visits
per
month

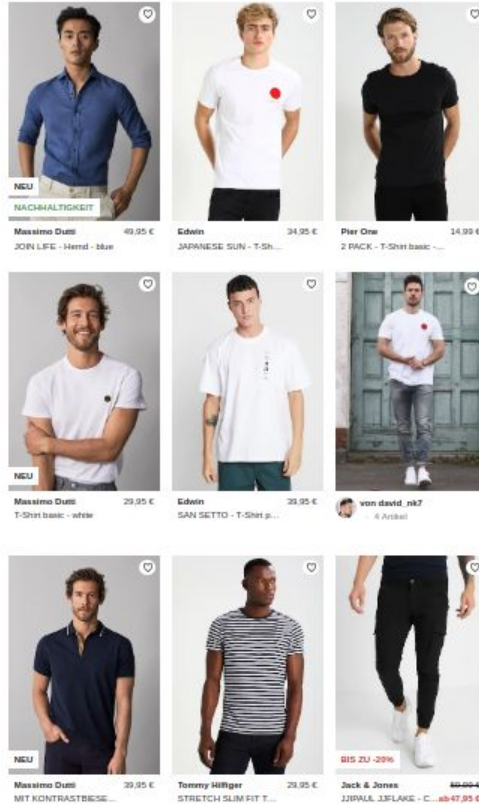
**> 27
million** active customers

> 400,000
product choices

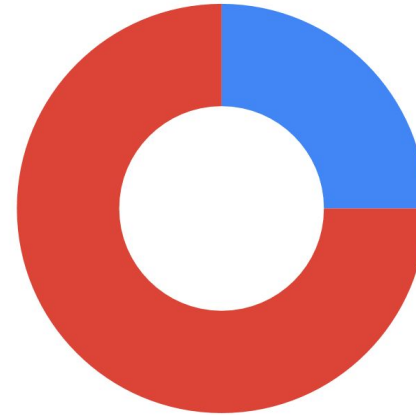
~ 2,000
brands

17
countries

DISCLAIMER



Fulltext Search vs Just Browsing



- Fulltext Search
- Just Browsing

Window Dressing





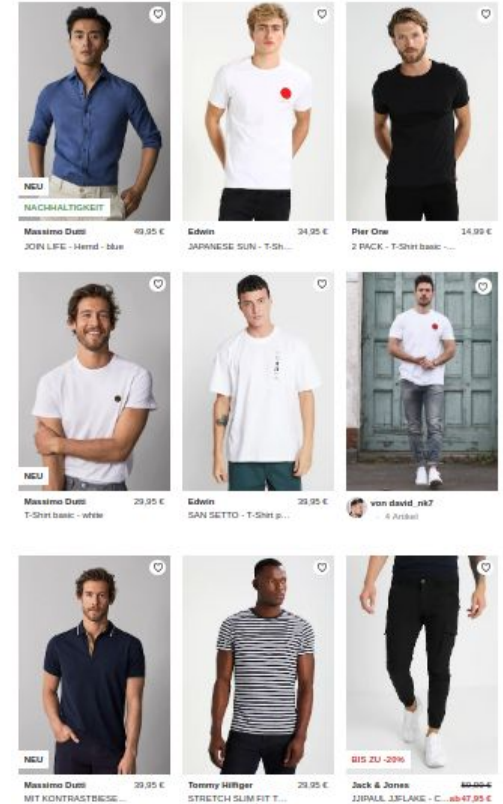
WIKIPEDIA
The Free Encyclopedia

“ Window dresser

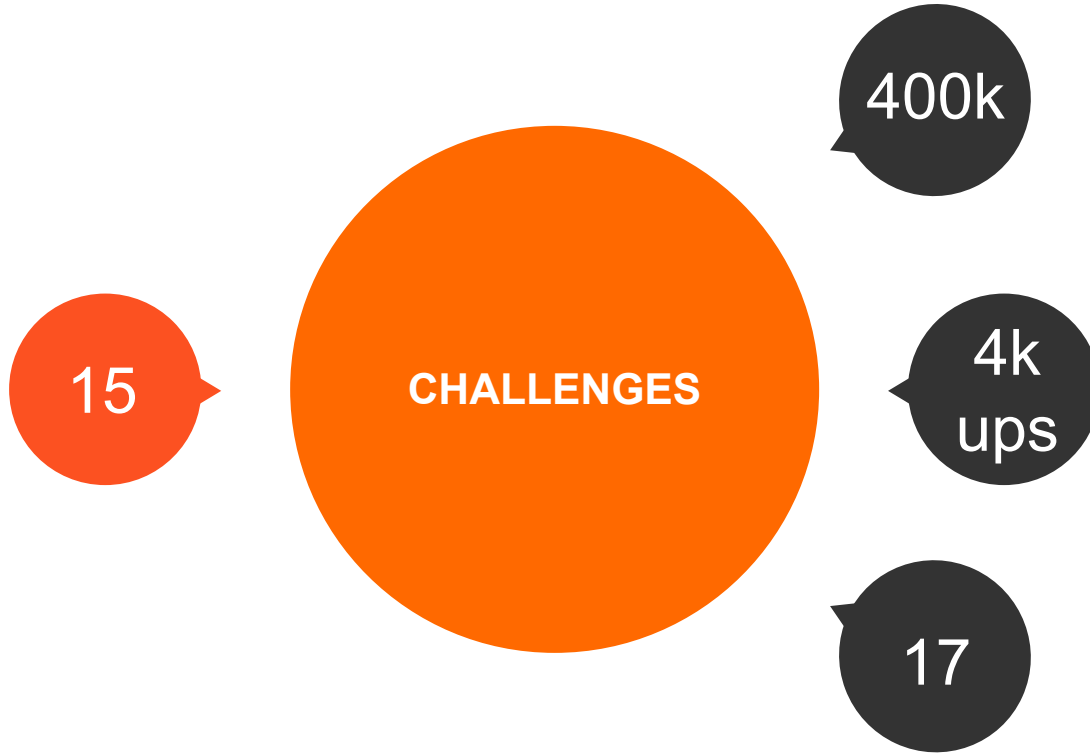
Window dressers arrange displays of goods in shop windows or within a shop itself. Such displays are themselves known as "[window dressing](#)". They may work for design companies contracted to work for clients or for department stores, independent retailers, airport or hotel shops.

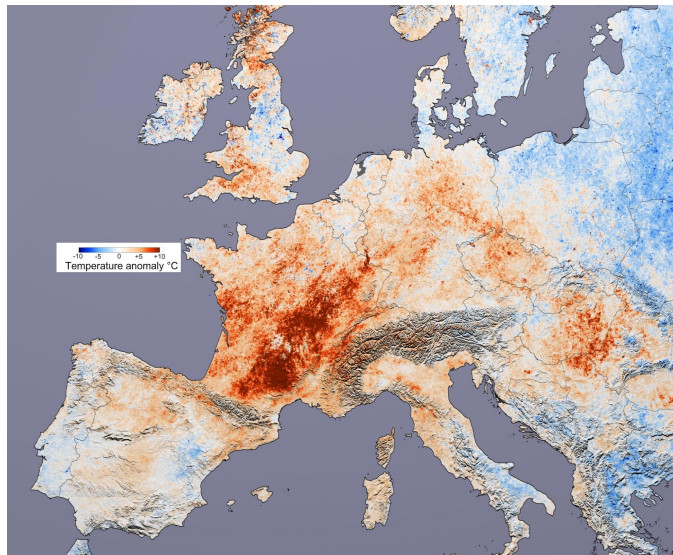
“

DATA DRIVEN SORTING



CHALLENGES

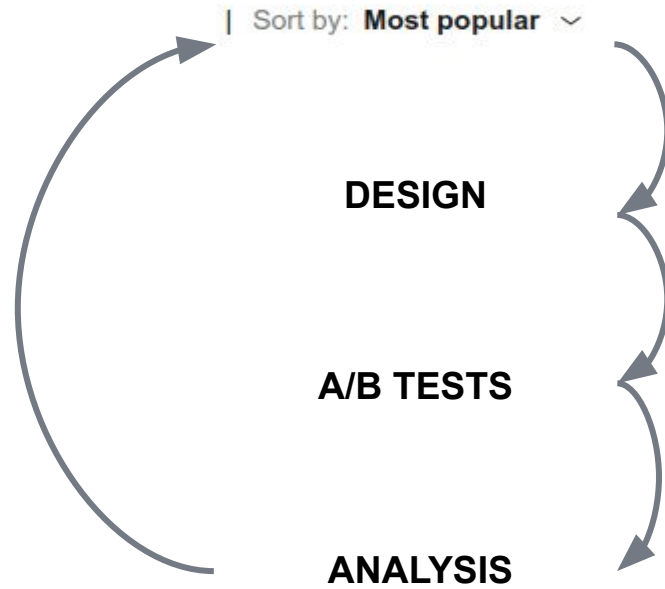




BLACK FRIDAY

Fail fast,
Iterate
faster

ITERATE FAST



Three Improvements

- Steering
- Fast Index Updates
- Sorting with functions

First Improvement: Sort Steering

Sort Steering - SQL Analogy

Id	Bucket	Popularity
sku1	1	0.2332332
sku2	2	0.123233
sku3	1	0.4533

SELECT * FROM articles ORDER BY **Bucket**
Desc, Popularity DESC

Sku2

Sku3

Sku1

Sort Steering - SQL Analogy

Id	Bucket	Popularity	Popularity_male
sku1	1	0.2332332	0.4
sku2	2	0.123233	0.6
sku3	1	0.4533	0.1

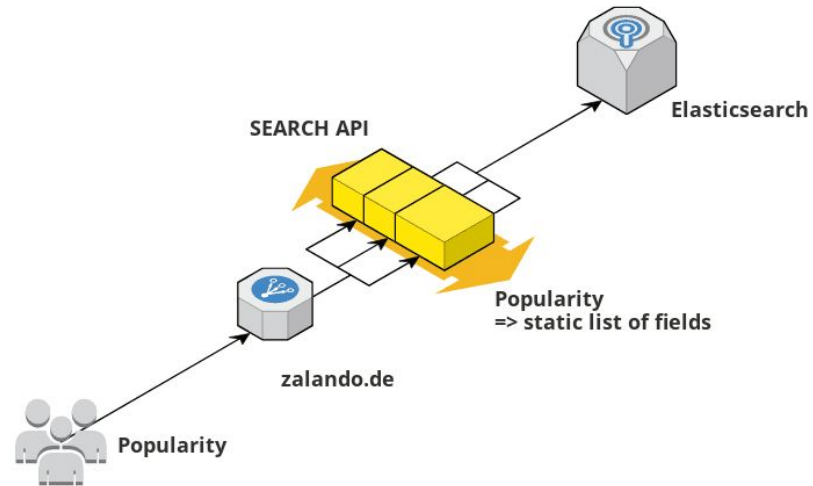
If category_gender == "men"

SELECT * FROM articles ORDER BY **Bucket** DESC, Popularity_male DESC

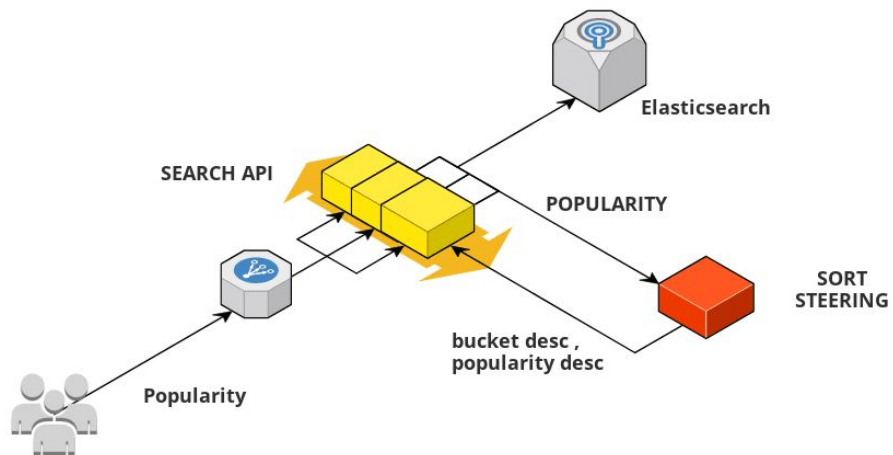
Else

SELECT * FROM articles ORDER BY **Bucket** DESC, Popularity DESC

Pre-Sort Steering Architecture



Sort Steering Added



Sort Steering - SQL Analogy

Id	Bucket	Popularity	Popularity_male
sku1	1	0.2332332	0.4
sku2	2	0.123233	0.6
sku3	1	0.4533	0.1

If category_gender == "men"

SELECT * FROM articles ORDER BY **Bucket** DESC, Popularity_male DESC

Else

SELECT * FROM articles ORDER BY **Bucket** DESC, Popularity DESC

```
1  description: Example sorting rule for MICES
2  enabled: true
3  rules:
4  - category_gender: men
5    precedence: 40
6    schema: article
7    variant: control-group
8    sorting_fields:
9      - direction: desc
10        field: boost.bucket
11      - direction: desc
12        field: boost.popularity
13      - direction: desc
14        field: first_activated
15    sorting_type: popularity
16  - category_gender: men
17    precedence: 40
18    schema: article
19    variant: treatment-group
20    sorting_fields:
21      - direction: desc
22        field: boost.bucket
23      - direction: desc
24        field: boost.popularity_male
25      - direction: desc
26        field: first_activated
27    sorting_type: popularity
```

2nd Improvement: Decoupled Data Ingestion

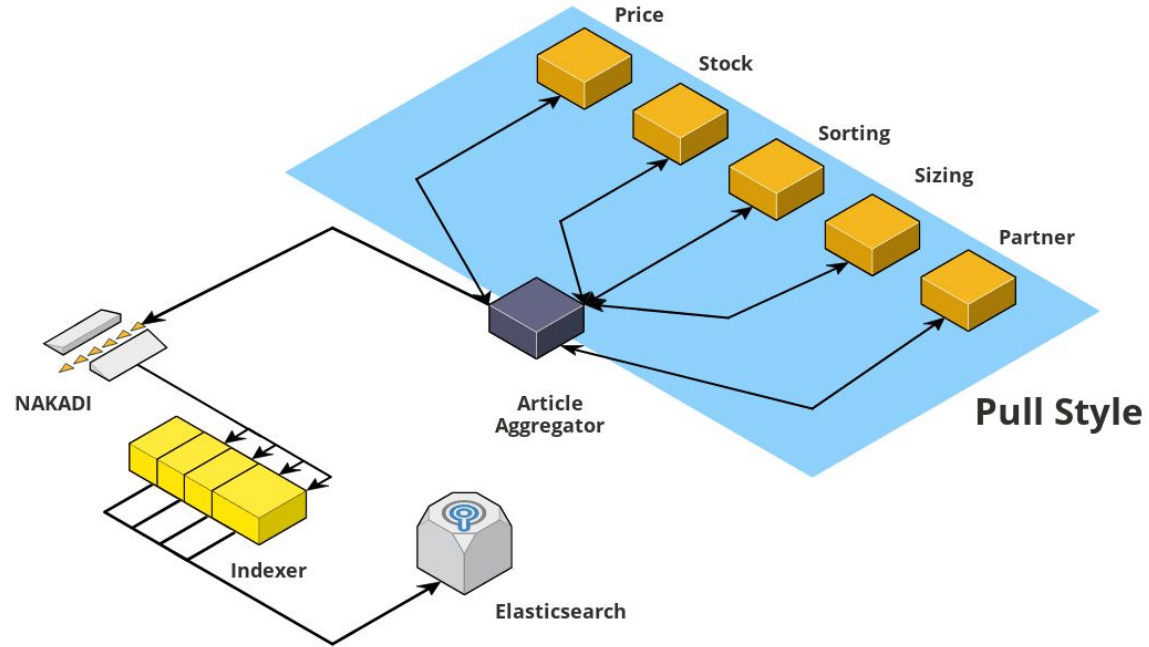
Indexing - SQL Analogy

Id	Price	Stock	Size	Partner	Performance	Performance_new_formula
sku1	9.99	100	32	false	0.5	0.4
sku1	9.99	100	32	false	0.3	0.6

INSERT INTO articles VALUES(9.99, 100, 32, false, **0.5**, **0.4**)

INSERT INTO articles VALUES(9.99, 100, 32, false, **0.3**, **0.6**)

Intake Architecture



Indexing - SQL Analogy

Id	Price	Stock	Size	Partner
sku1	9.99	100	32	false

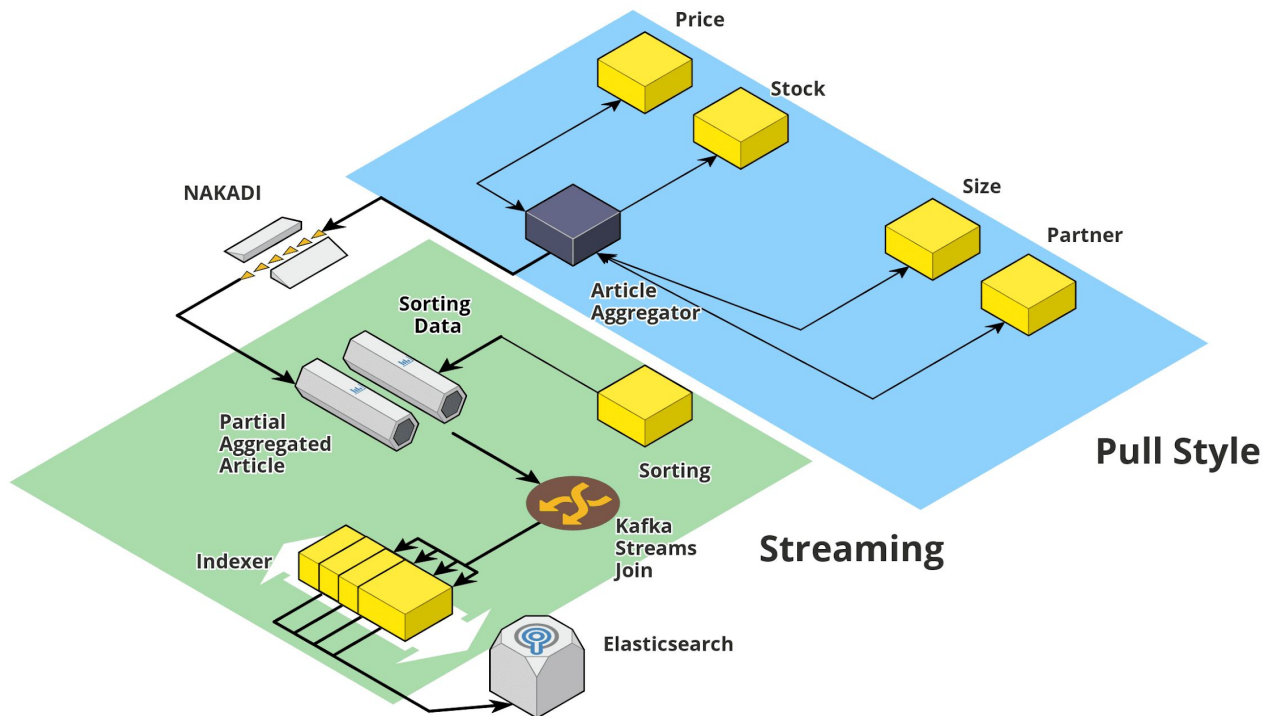
Id	Performance	Performance_new_formula
sku1	0.5	0.4
sku1	0.3	0.6



JOINS => Elasticsearch

Id	Price	Stock	Size	Partner	Performance	Performance_new_formula
sku1	9.99	100	32	false	0.3	0.6

Intake Architecture Now



3rd Improvement: Sorting with Functions

Painless Scripts

Script Based Sorting



Allow to sort based on custom scripts, here is an example:

```
GET /_search
{
  "query" : {
    "term" : { "user" : "kimchy" }
  },
  "sort" : {
    "_script" : {
      "type" : "number",
      "script" : {
        "lang": "painless",
        "source": "doc['field_name'].value * params.factor",
        "params" : {
          "factor" : 1.1
        }
      }
    },
    "order" : "asc"
  }
}
```

[COPY AS CURL](#) [VIEW IN CONSOLE](#) 

Sorting with Functions - Eliminate Reindexing

Id	Price	Stock	Size	Partner	clicks	sales
sku1	9.99	100	32	false	10000	300

SELECT * FROM articles ORDER BY **popularity(sales,clicks)**

popularity(sales, clicks) = sales/clicks

Sorting with Functions - Personalization

Id	Price	Stock	Size	Partner	popularity	article_features
sku1	9.99	100	32	false	1.2	[9.99, 100, 32]

If **known_customer** :

SELECT * FROM articles ORDER BY **dot_product(article_feature, customer_features)**

Else

SELECT * FROM articles ORDER BY **popularity**

Sorting with Functions - Fulltext Search

Id	Price	Stock	Size	Partner	clicks	sales
sku1	9.99	100	32	false	10000	300

If **fulltext_search** :

```
SELECT * FROM articles ORDER BY f(relevance_score, clicks, sales, customer_features,  
article_features)
```

Else

```
SELECT * FROM articles ORDER BY g(clicks, bucket, sales, customer_features)
```

EXAMPLE SORTING RULES

```
1  description: Rules for MICES DEMO
2  enabled: true
3  rules:
4  - precedence: 10
5    variant: mices-ab-test
6    schema: article
7    sorting_fields:
8    - direction: desc
9      script_line: >-
10        def personalizedScore = 0;
11        for(int i = 0; i < doc['article_features'].length; i++) {
12          personilzedScore = personalizedScore + doc['article_features'][i] * params['customer_features'][i];
13        }
14        return personalizedScore + _score + doc['sales'].value / doc['clicks'].value
15    test_path: test/mices.json
16    type: inline_script
17    sorting_type: popularity
```

Personalization

```
1  description: Rules for MICES DEMO
2  enabled: true
3  rules:
4  - precedence: 10
5    variant: mices-ab-test
6    schema: article
7    sorting_fields:
8    - direction: desc
9      script_line: >-
10        def personalizedScore = 0;
11        for(int i = 0; i < doc['article_features'].length; i++) {
12          personalilzedScore = personalizedScore + doc['article_features'][i] * params['customer_features'][i];
13        }
14        return personalizedScore + _score + doc['sales'].value / doc['clicks'].value
15    test_path: test/mices.json
16    type: inline_script
17  sorting_type: popularity
```

Query Relevance

```
1  description: Rules for MICES DEMO
2  enabled: true
3  rules:
4  - precedence: 10
5    variant: mices-ab-test
6    schema: article
7    sorting_fields:
8    - direction: desc
9      script_line: >-
10        def personalizedScore = 0;
11        for(int i = 0; i < doc['article_features'].length; i++) {
12          personilzedScore = personalizedScore + doc['article_features'][i] * params['customer_features'][i];
13        }
14        return personalizedScore + _score + doc['sales'].value / doc['clicks'].value
15  test_path: test/mices.json
16  type: inline_script
17  sorting_type: popularity
```

Inline Popularity Calculation

```
1  description: Rules for MICES DEMO
2  enabled: true
3  rules:
4  - precedence: 10
5    variant: mices-ab-test
6    schema: article
7    sorting_fields:
8    - direction: desc
9      script_line: >-
10        def personalizedScore = 0;
11        for(int i = 0; i < doc['article_features'].length; i++) {
12          personilzedScore = personalizedScore + doc['article_features'][i] * params['customer_features'][i];
13        }
14        return personalizedScore + _score + doc['sales'].value / doc['clicks'].value
15    test_path: test/mices.json
16    type: inline_script
17    sorting_type: popularity
```

Open Source Contribution

CODE IN CONFIG ????????

```
1  description: Rules for MICES DEMO
2  enabled: true
3  rules:
4  - precedence: 10
5    variant: mices-ab-test
6    schema: article
7    sorting_fields:
8    - direction: desc
9      script_line: >-
10      def personalizedScore = 0;
11      for(int i = 0; i < doc['article_features'].length; i++) {
12        personilzedScore = personalizedScore + doc['article_features'][i] * params['customer_features'][i];
13      }
14      return personalizedScore + score + doc['sales'].value / doc['clicks'].value
15  test_path: test/mices.json
16  type: inline_script
17  sorting_type: popularity
```

TGYHT - Thanks God You Have Tests

```
1  description: Rules for MICES DEMO
2  enabled: true
3  rules:
4  - precedence: 10
5    variant: mices-ab-test
6    schema: article
7    sorting_fields:
8    - direction: desc
9      script_line: >-
10      def personalizedScore = 0;
11      for(int i = 0; i < doc['article_features'].length; i++) {
12        personilzedScore = personalizedScore + doc['article_features'][i] * params['customer_features'][i];
13      }
14      return personalizedScore + score + doc['sales'].value / doc['clicks'].value
15  test_path: test/mices.json
16  type: inline_script
17  sorting_type: popularity
```

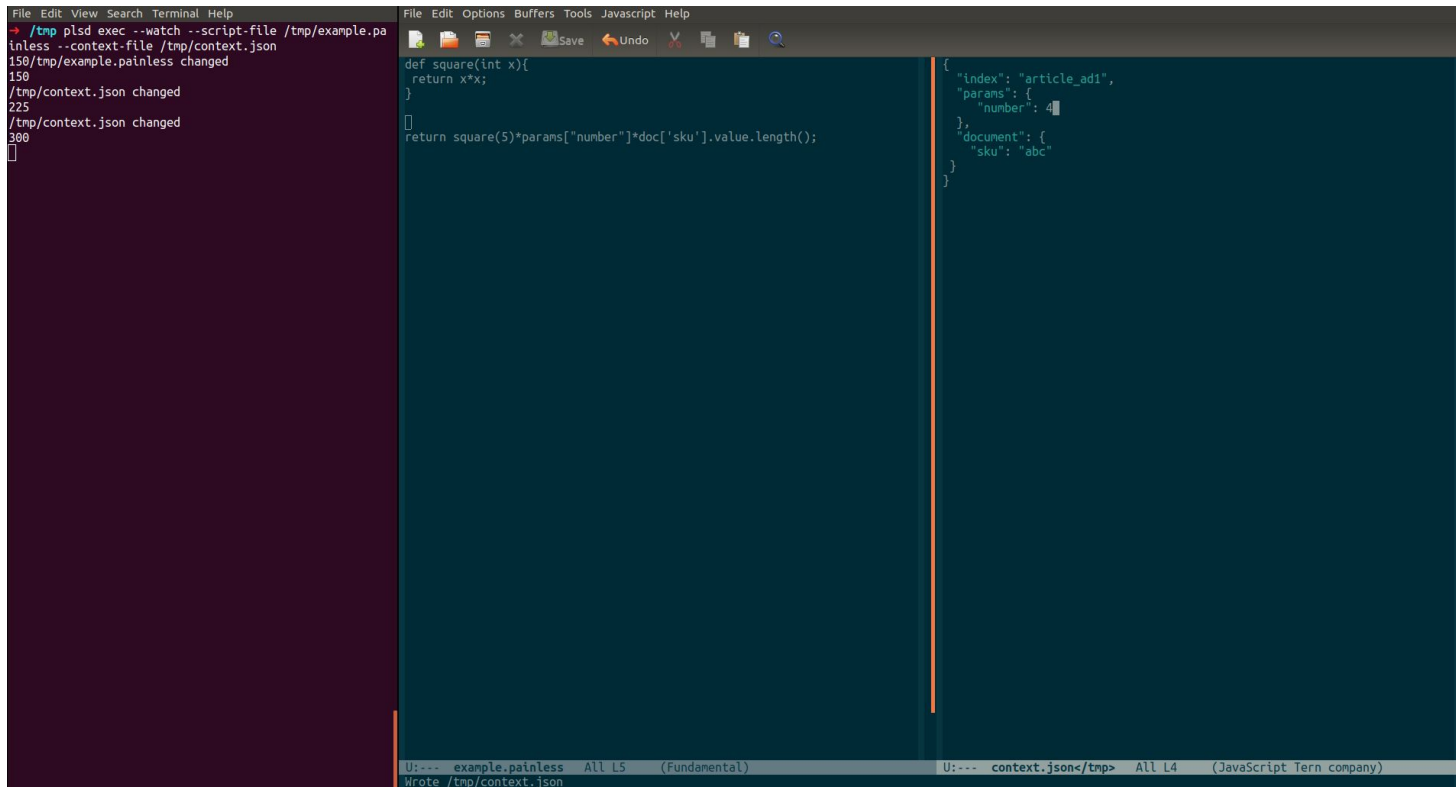
Make Painless Script Development Painless

- Painless Lacks Tooling
- Elasticsearch Painless Execute API
- <https://www.elastic.co/guide/en/elasticsearch/painless/current/painless-execute-api.html>

Painless Scripts Development Tool

- <https://github.com/csenol/plsd>
- Integrated with CI/CD Pipelines

Painless Script Development Environment



The screenshot displays the Painless Script Development Environment, which consists of a terminal window on the left and a code editor on the right.

Terminal Window:

```
File Edit View Search Terminal Help
→ /tmp plsd exec --watch --script-file /tmp/example.pa
inless --context-file /tmp/context.json
159/tmp/example.painless changed
159
/tmp/context.json changed
225
/tmp/context.json changed
300
```

Code Editor:

```
File Edit Options Buffers Tools Javascript Help
def square(int x){
  return x*x;
}
return square(5)*params["number"]*doc['sku'].value.length();

{
  "index": "article_ad1",
  "params": {
    "number": 4
  },
  "document": {
    "sku": "abc"
  }
}
```

Status Bar:

U:--- example.painless All L5 (Fundamental) U:--- context.json</tmp> All L4 (JavaScript Tern company)
Wrote /tmp/context.json

Painless Script Performance Tests

```
→ plsd git:(add-perf-support) ✗ plsd perf --query-file esquery --index article_ad1 --context-file /tmp/context.json --repeat 100 < /tmp/big-bang.script
300 samples of 300 events
Cumulative: 2.368030044s
HMean: 7.330798ms
Avg.: 7.893433ms
p50: 6.502339ms
p75: 10.083563ms
p95: 11.785152ms
p99: 17.758704ms
p999: 20.450956ms
Long 5%: 15.03373ms
Short 5%: 5.887896ms
Max: 20.450956ms
Min: 5.824649ms
Range: 14.626307ms
StdDev: 2.510904ms
Rate/sec.: 126.69
```

TESTING PAINLESS SCRIPTS/ CI-CD Integration

```
3:30:14 PM Test Passed: We don't have commodity group data in article
3:30:14 PM Test Passed: Testing Learn To Rank With Price Cluster Factor (positive learn to rank)
3:30:14 PM Test Passed: Testing Learn To Rank With Price Cluster Factor (negative learn to rank)
3:30:14 PM Test Passed: Testing Learn To Rank With Price Cluster (Price cluster factor missing)
3:30:14 PM Test Passed: Testing Learn To Rank With Price Cluster (Price cluster factor is 0 and LTR is positive)
3:30:14 PM Test Passed: Testing Learn To Rank With Price Cluster (Price cluster factor is 0 and LTR is negative)
3:30:14 PM Test Passed: Testing Learn To Rank With Price Cluster Factor (positive learn to rank)
3:30:14 PM Test Passed: Testing Learn To Rank With Price Cluster Factor (negative learn to rank)
3:30:14 PM Test Passed: Testing Learn To Rank With Price Cluster (Price cluster factor missing)
3:30:14 PM Test Passed: Testing Learn To Rank With Price Cluster (Price cluster factor is 0 and LTR is positive)
3:30:14 PM Test Passed: Testing Learn To Rank With Price Cluster (Price cluster factor is 0 and LTR is negative)
3:30:14 PM Test Passed: Testing Learn To Rank With Price Cluster Factor (positive learn to rank)
3:30:14 PM Test Passed: Testing Learn To Rank With Price Cluster Factor (negative learn to rank)
3:30:14 PM Test Passed: Testing Learn To Rank With Price Cluster (Price cluster factor missing)
3:30:14 PM Test Passed: Testing Learn To Rank With Price Cluster (Price cluster factor is 0 and LTR is positive)
3:30:14 PM Test Passed: Testing Learn To Rank With Price Cluster (Price cluster factor is 0 and LTR is negative)
3:30:14 PM Test Passed: Testing Learn To Rank With Price Cluster Factor (positive learn to rank)
3:30:14 PM Test Passed: Testing Learn To Rank With Price Cluster Factor (negative learn to rank)
3:30:14 PM Test Passed: Testing Learn To Rank With Price Cluster (Price cluster factor missing)
3:30:14 PM Test Passed: Testing Learn To Rank With Price Cluster (Price cluster factor is 0 and LTR is positive)
3:30:14 PM Test Passed: Testing Learn To Rank With Price Cluster (Price cluster factor is 0 and LTR is negative)
3:30:14 PM Test Passed: Testing Learn To Rank With Price Cluster Factor (positive learn to rank)
3:30:14 PM Test Passed: Testing Learn To Rank With Price Cluster Factor (negative learn to rank)
3:30:14 PM Test Passed: Testing Learn To Rank With Price Cluster (Price cluster factor missing)
```

```
1  [
2    {
3      "description": "Example Test Case For MICES",
4      "index": "article_ad1",
5      "params": {
6        "customer_features": [1,1,1,1,1]
7      },
8      "document": {
9        "article_features": [2,2,2,2,2],
10       "sales": 10,
11       "click": 100
12     },
13     "expected_result": 10.1
14   }
15 ]
```

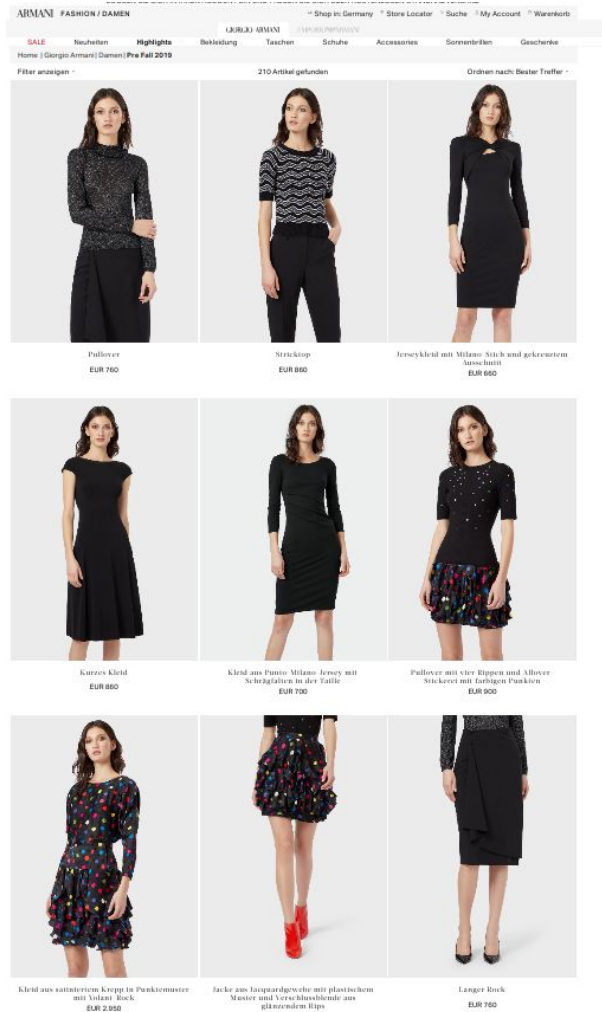
Sum up

- Sort Steering => A/B tests
- Decoupled Indexing => Data Enrichment
- Sorting With Functions => Faster Implementation + Personalization



Notable window dressers^[edit]

- [Giorgio Armani](#), the fashion designer once worked as a window dresser.^[1]





Cagdas Senol



cagdassenol@gmail.com



Q&A