prof. Giovanni Stilo

Take to Wife Lucene

Lucene Basic Principles

Lucene ?

Name **Lucene** came from Doug Cutting's (founder) wife's middle name.

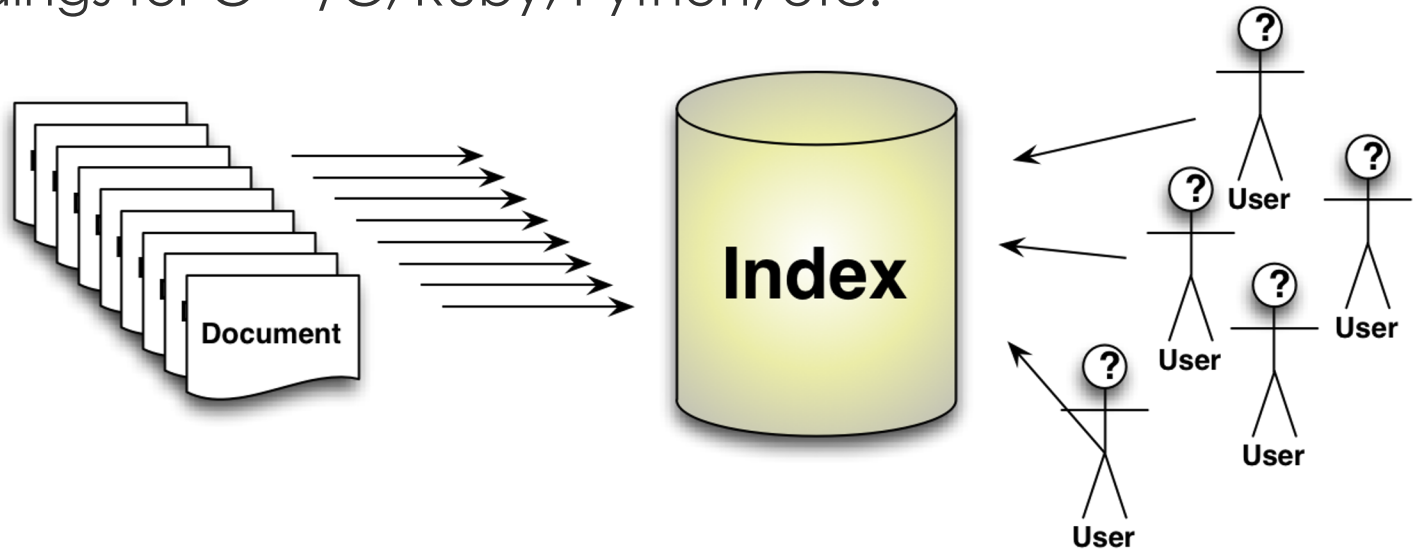
Free-text indexing library

Implements standard **IR/search** functionality

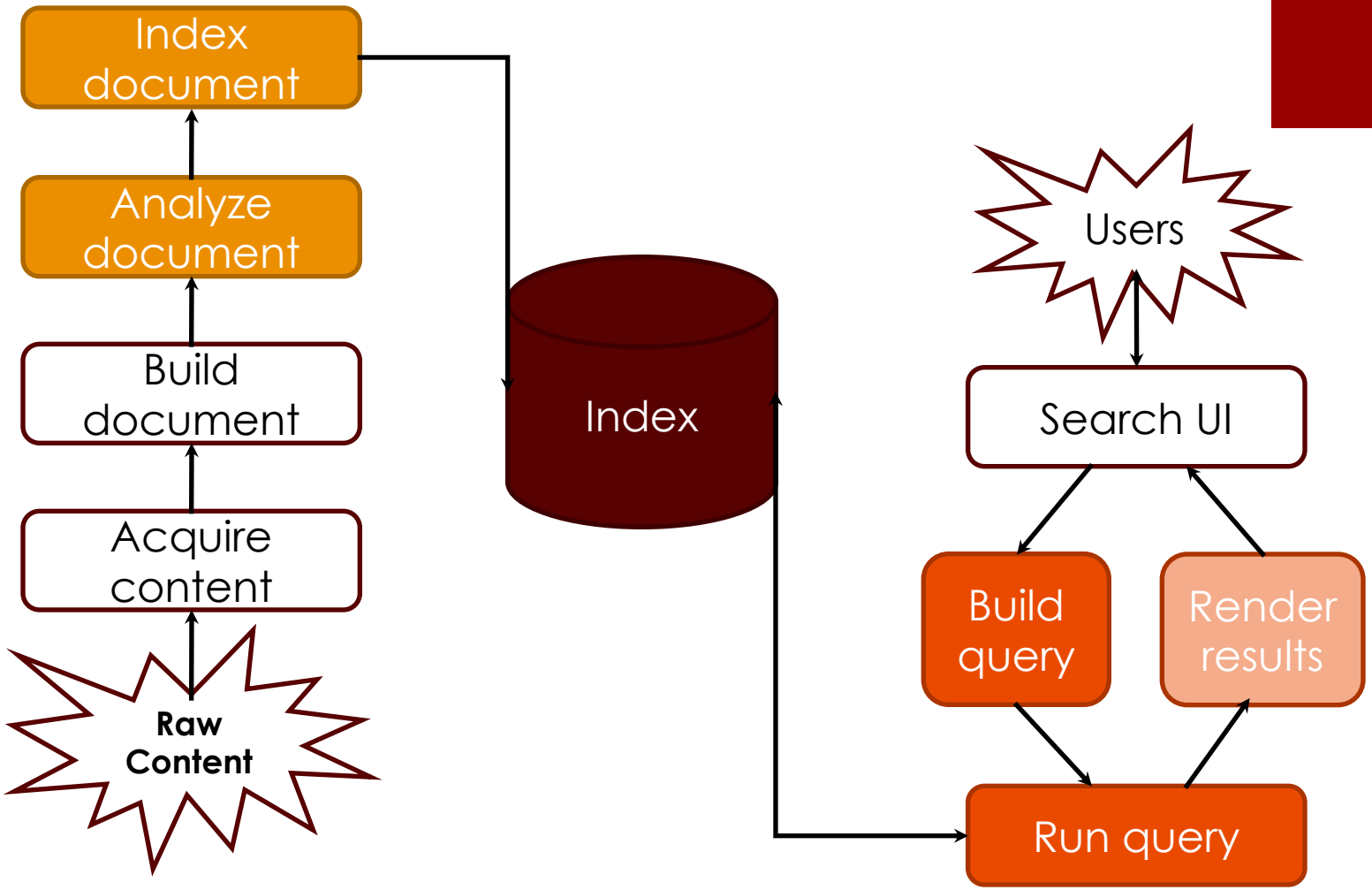
Query models, ranking, indexing

Core API is implemented in Java

Bindings for C++/C, Ruby, Python, etc.



Main Flow


Index document
Analyze document
Build document
Acquire content
Raw Content
Index
Users
Search UI
Build query
Render results
Run query

Document

A Document is the **basic unit** for *indexing* and *searching*
(**note**: it is different from the notion of document as file)

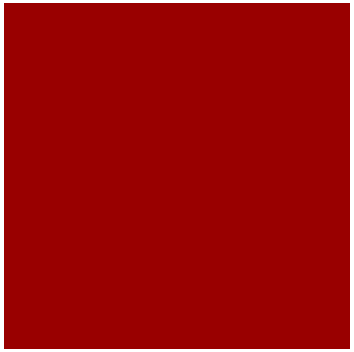
Each Document is a **list** of Field(s)

Each Field has a **name** and a text value

It is up to us to decide what to include in a Document

```
…
Document doc = new Document();
doc.add(new Field(…));
doc.add(new Field(…));
doc.add(new Field(…));
…
```

Field



A field is the **basic unit** composing *Documents* - each Document is a list of Field(s).

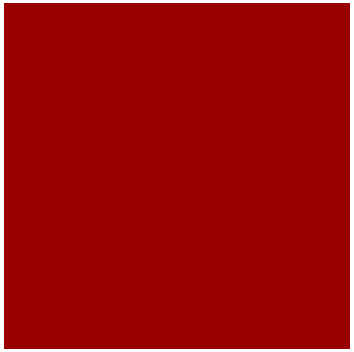
For each field, you need to **specify**

Name

Value

```
…
Document doc = new Document();
Field f1 = new Field("fieldName1", "fieldContent1");
Field f2 = new Field("fieldName2", "fieldContent2");
doc.add(new Field(…));
…
```

Field Types



Numeric types:

- FloatPoint
- DoublePoint
- IntPoint
- LongPoint

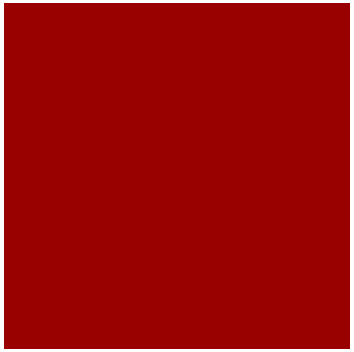
Text types:

- StringField
- TextField

Expert types:

- Field

Directory



At its core, a list of files.

ByteBuffersDirectory

in-memory volatile directory

useful for “on-the-fly” indexes

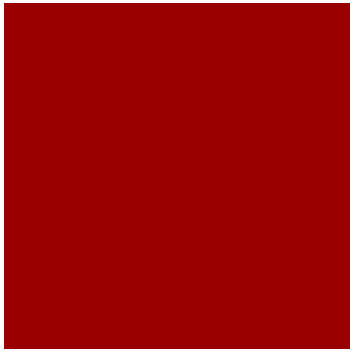
```
…
Directory dir= new ByteBuffersDirectory();
…
```

SimpleFSDirectory

file-based, persistent directory

```
…
Directory dir= Directory dir = new
SimpleFSDirectory(FileSystems.getDefault().getPath("index"));
…
```

Indexing Documents

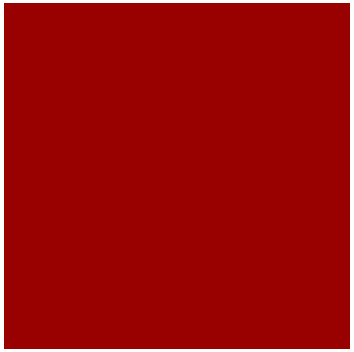


Is the process of Writing the Index

We need an *IndexWriter*

```
…
Directory dir= new ByteBuffersDirectory();
Analyzer analyzer = new StandardAnalyzer();
IndexWriterConfig cfg= new IndexWriterConfig(analyzer);
IndexWriter writer = new IndexWriter(dir, cfg);
…
Document doc = new Document();
…
for(all documents) {
        writer.addDocument(document);
}
writer.commit();
writer.close();
…
```

LongPoint



Field that indexes long values for **efficient** range *filtering* and *sorting*:

```
…
document.add(new LongPoint(name, 6L)); // no storage default
document.add(new StoredField(name, 6L)); // stored long field
…
```

For optimal performance, **re-use** the **LongPoint** and **Document** instance for more than one document:

```
LongPoint field = new LongPoint("fieldName", 0L);
Document document = new Document();
document.add(field);
for(all documents) {
  field.setLongValue(value of doc);
  writer.addDocument(document);
}
```

Textual Fields

StringField: a field that is indexed but *not tokenized*; the entire String value is indexed as a single token.

For example this might be used for a 'country' field or an 'id' field.

```
…
doc.add(new StringField("fieldName", content, Field.Store.YES));
…
```

TextField: A field that is indexed and *tokenized*, without term vectors.

For example this would be used on a 'body' field, that contains the bulk of a document's text.

```
…
doc.add(new TextField ("fieldName", content, Field.Store.YES));
…
```

Field

Expert: *directly* create a field for a document. Most users should use one of the sugar subclasses:

LongPoint, IntPoint, FloatPoint, DoublePoint, StringField, TextField.

```
FieldType MY_FLD_TYPE = new FieldType();
MY_FLD_TYPE.setIndexed(true);
MY_FLD_TYPE.setTokenized(true);
MY_FLD_TYPE.setStored(true);
MY_FLD_TYPE.setStoreTermVectors(true);
MY_FLD_TYPE.setStoreTermVectorPositions(true);
MY_FLD_TYPE.freeze();
…
doc.add(new Field("fieldName", content, MY_FLD_TYPE ));
…
```

Maven Artifacts

```
<dependency>
        <groupId>org.apache.lucene</groupId>
        <artifactId>lucene-core</artifactId>
        <version>8.0.0</version>
</dependency>
<dependency>
        <groupId>org.apache.lucene</groupId>
        <artifactId>lucene-queryparser</artifactId>
        <version>8.0.0</version>
</dependency>
<dependency>
        <groupId>org.apache.lucene</groupId>
        <artifactId>lucene-queries</artifactId>
        <version>8.0.0</version>
</dependency>
<dependency>
        <groupId>org.apache.lucene</groupId>
      <artifactId>lucene-analyzers-common</artifactId>
        <version>8.0.0</version>
</dependency>
```

Excercise

Create a class that allow to index many Identity Card Documents.

Each documents have the following fields (choosing the best type of field):

- IC Tag;
- Name;
- Surname;
- Birth Date
- State
- City
- Height (expressed in meters for example 1.75)

Field Types

Numeric types:

- FloatPoint
- DoublePoint (height)
- IntPoint
- LongPoint (date)

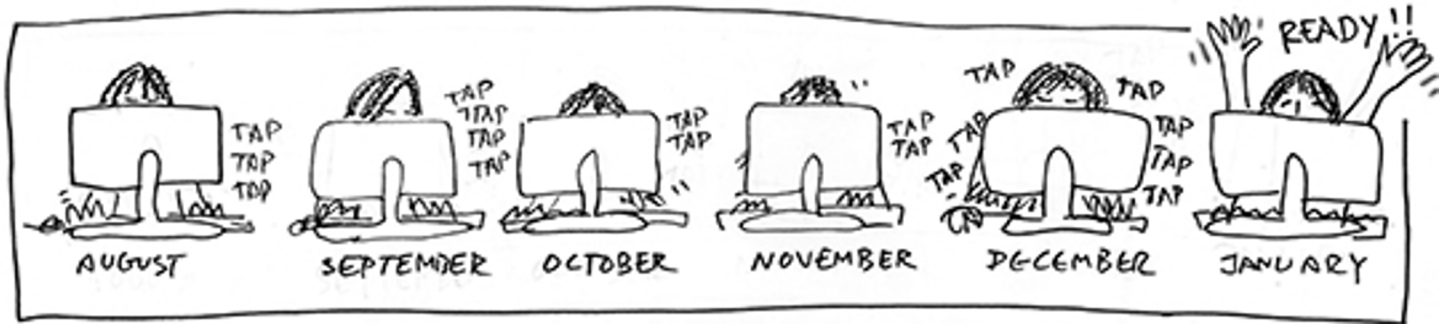
Text types:

- StringField (ID)
- TextField (name, surname, state, city)

Expert types:

- Field

Let's Try?!?!


TAP
TAP
TAP
AUGUST
TAP
TAP
TAP
TAP
SEPTEMBER
TAP
TAP
OCTOBER
TAP
TAP
NOVEMBER
TAP
TAP
TAP
TAP
TAP
TAP
TAP
DECEMBER
READY!!
JANUARY