

# Apache CXF, Tika and Lucene

## The power of search the JAX-RS way



Andriy Redko



# About myself

- Passionate Software Developer since 1999
- On Java since 2006
- Currently employed by **AppDirect** in Montreal
- Contributing to **Apache CXF** project since 2013

<http://aredko.blogspot.ca/>

<https://github.com/reta>

# What this talk is about ...

- REST web APIs are everywhere
- JSR-339 / JAX-RS 2.0 is a standard way to build RESTful web services on JVM
- Search/Filtering capabilities in one form or another are required by most of web APIs out there
- So why not to bundle search/filtering into REST apps in generic, easy to use way?

# Meet Apache CXF

- Apache CXF is very popular open source framework to develop services and web APIs on JVM platform
- The latest **3.0** release is (as complete as possible) JAX-RS 2.0 compliant implementation
- Vibrant community, complete documentation and plenty of examples make it a great choice

# Apache CXF Search Extension (I)

- Very simple concept build around customizable **\_s / \_search** query parameter
- At the moment, supports [Feed Item Query Language](#) (FIQL) expressions and [OData 2.0](#) URI filter expressions

**http://my.host:9000/api/people?\_search=  
"firstName eq 'Bob' and age gt 35"**

# FIQL

- The Feed Item Query Language
- IETF draft submitted by M. Nottingham on December 12, 2007  
<https://tools.ietf.org/html/draft-nottingham-atompub-fiql-00>
- Fully supported by Apache CXF

**`_search=firstName==Bob;age=gt=35`**

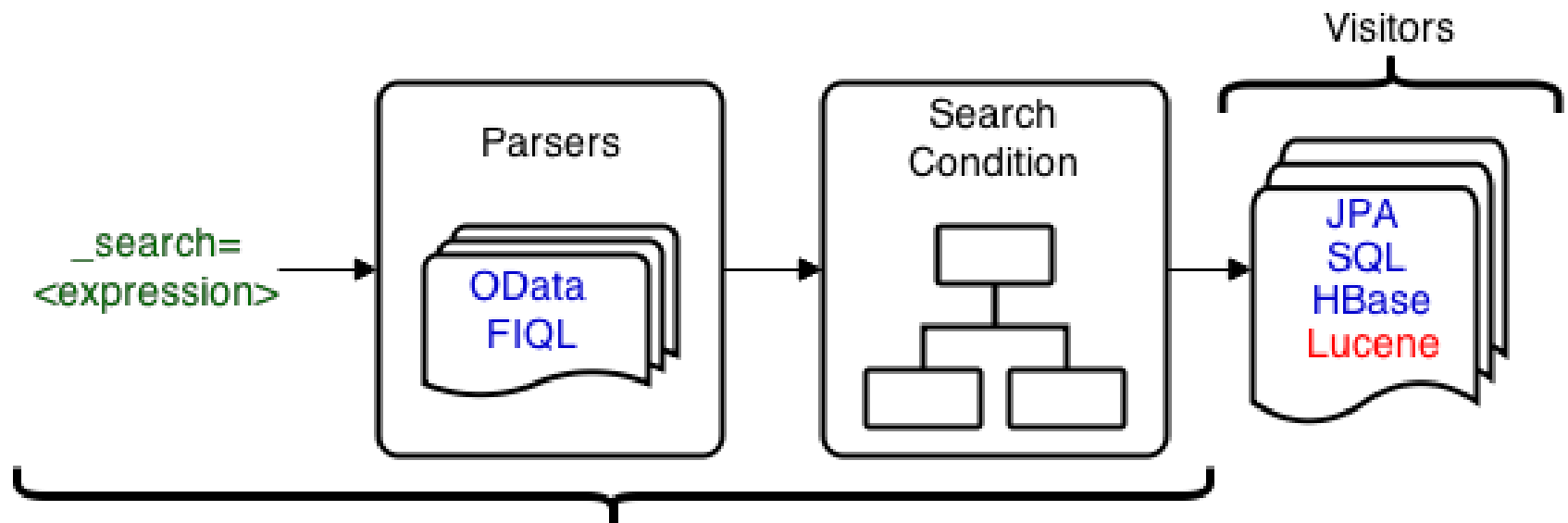
# OData 2.0

- Uses OData URI **\$filter** system query option  
<http://www.odata.org/documentation/odata-version-2-0/uri-conventions>
- Built on top of **Apache Olingo** and its **FilterParser** implementation
- Only subset of the operators is supported (matching the **FIQL** expressions set)

**\_search="firstName eq 'Bob' and age gt 35"**

# Apache CXF Search Extension (II)

- Under the hood ...



```
@GET
@Produces( { MediaType.APPLICATION_JSON } )
public Response search(@Context SearchContext context) {
    ...
}
```



# Apache Lucene In Nutshell

- Leading, battle-tested, high-performance, full-featured text search engine
- Written purely in Java
- Foundation of many specialized and general-purpose search solutions (including Solr and Elastic Search)
- Current major release branch is **5.x**

# Apache CXF Search Extension (III)

- **LuceneQueryVisitor** maps the search/filter expression into Apache Lucene query
- Uses **QueryBuilder** and is analyzer-aware (means stemming, stop words, lower case, ... apply if configured)
- Apache Lucene **4.7+** is required (**4.9+** recommended)
- **Subset** of Apache Lucene queries is supported (many improvements in upcoming **3.1** release)

# Lucene Query Visitor

- Is type-safe but supports regular key/value map (aka **SearchBean**) to simplify the usage

```
@GET
@Produces( { MediaType.APPLICATION_JSON } )
public Response search(@Context SearchContext context) {
    final LuceneQueryVisitor< SearchBean > visitor =
        new LuceneQueryVisitor< SearchBean >(analyzer);
    visitor.visit(context.getCondition(SearchBean.class));

    final IndexReader reader = ...;
    final IndexSearcher searcher = new IndexSearcher(reader);
    final Query query = visitor.getQuery();

    final TopDocs topDocs = searcher.search(query, 10);
    ...
}
```

# Supported Lucene Queries

- **TermQuery**
- **PhraseQuery**
- **WildcardQuery**
- **NumericRangeQuery** (int / long / double / float)
- **TermRangeQuery** (date)
- **BooleanQuery** (or / and)

# TermQuery Example

FIQL

`_search=firstName==Bob`

OData

`_search="firstName eq 'Bob'"`



**firstName:bob**

# PhraseQuery Example

FIQL

**\_search=content=='Lucene in Action'**

OData

**\_search="content eq 'Lucene in Action'"**



**content:"lucene ? action"**

\* **in** is typically a stopwords and is replaced by ?

# WildcardQuery Example

FIQL

`_search=firstName==Bo*`

OData

`_search="firstName eq 'Bo*'"`



**`firstName:Bo*`**

# NumericRangeQuery Example

FIQL

`_search=age=gt=35`

OData

`_search= "age gt 35"`



`age:{35 TO *}`

\* the type of **age** property should be numeric

```
visitor.setPrimitiveFieldTypeMap(singletonMap("age", Integer.class))
```



# TermRangeQuery Example

FIQL

**\_search=modified=lt=2015-10-25**

OData

**\_search= "modified lt '2015-10-25'"**



**modified:{\* TO 20151025040000000}**

\* the type of **modified** property should be date

```
visitor.setPrimitiveFieldTypeMap(singletonMap("modified", Date.class))
```

# BooleanQuery Example

FIQL

**\_search=firstName==Bob;age=gt=35**

OData

**\_search= "firstName eq 'Bob' and  
age gt 35"**



**+firstName:bob +age:{35 TO \*} }**

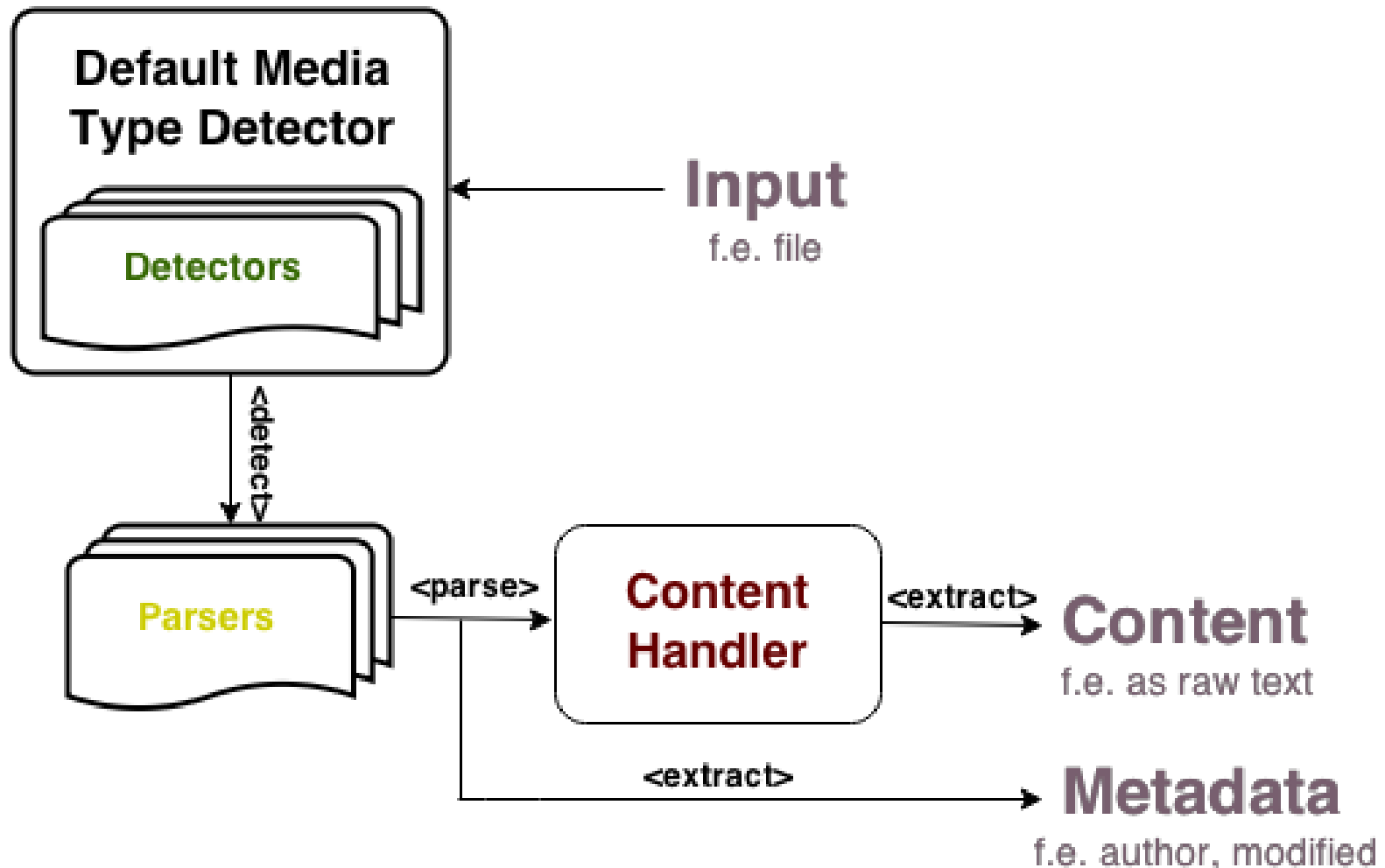
# From “How ...” to “What ...”

- Files are still the most widespread source of valuable data
- However, most of file formats are either binary (\*.pdf, \*.doc, ...) or use some kind of markup (\*.html, \*.xml, \*.md, ...)
- It makes the search a difficult problem as the raw text has to be extracted and only then indexed / searched against

# Apache Tika

- **Metadata** and **text** extraction engine
- Supports myriad of different file formats
- Pluggable modules (parsers), include only what you really need
- Extremely easy to ramp up and use
- Current release branch is **1.7**

# Apache Tika in Nutshell



# Text Extraction in Apache CXF

- Provides generic **TikaContentExtractor**

```
public class TikaContentExtractor {  
    public TikaContent extract(final InputStream in) {  
        ...  
    }  
}
```

- Also has specialization for Apache Lucene, **TikaLuceneContentExtractor**

```
public class TikaLuceneContentExtractor {  
    public Document extract(final InputStream in) {  
        ...  
    }  
}
```

# And finally, indexing ...

- The **text** and **metadata** extracted from the file could be added straight to Lucene index

```
final TikaLuceneContentExtractor extractor =  
    new TikaLuceneContentExtractor(new PDFParser());  
  
final Document document = extractor.extract(in);  
final IndexWriter writer = ...;  
  
try {  
    writer.addDocument(document);  
    writer.commit();  
} finally {  
    writer.close();  
}
```

# Demo

<https://github.com/reta/ApacheConNA2015>



# Apache CXF Search Extension (IV)

- Configuring expressions parser  
**search.parser.class=ODataParser**  
**search.parser=new ODataParser()**
- Configuring query parameter name  
**search.query.parameter.name=\$filter**
- Configuring date format  
**search.date-format=yyyy/MM/dd**

# Alternatives

- **ElasticSearch:** is a highly scalable open-source full-text search and analytics engine (<http://www.elastic.co/>)
- **Apache Solr:** highly reliable, scalable and fault tolerant open-source enterprise search platform (<http://lucene.apache.org/solr/>)

**These are dedicated, best in class solutions for solving difficult search problems.**

# Useful links

- <http://cxf.apache.org/docs/jax-rs-search.html>
- <http://lucene.apache.org/>
- <http://tika.apache.org/>
- <http://aredko.blogspot.ca/2014/12/beyond-jax-rs-spec-apache-cxf-search.html>

# Thank you!

Many thanks to **Apache Software Foundation** and **AppDirect** for the chance to be here