Content Extraction with Apache Tika

Jukka Zitting | Tika committer, co-author of Tika in Action

Content Extraction with Apache Tika

- Introduction to Apache Tika
- Full text extraction with Tika
- Tika and Solr – the ExtractingRequestHandler
- Tika and Lucene – direct feeding of the index
  - forked parsing
  - link extraction

Introduction to Apache Tika

section 1 / 4

Introduction to Apache Tika

The Apache Tika™ toolkit
- detects and extracts
- metadata and structured text content
- from various documents
- using existing parser libraries.

Problem domain

Adobe

The Tika solution

Document

It is a truth universally acknowledged, that a single man in possession of a good fortune, must be in want of a wife...

Content

dc:title=
Pride and Prejudice
dc:creator=
Jane Austen
dc:date=1813

Metadata

Project background

- Brief history
  - 2007 Tika started in the Apache Incubator
  - 2008 Tika graduates into a Lucene subproject
  - 2010 Tika becomes a standalone TLP
  - 2011 Tika 1.0 released
  - 2011 Tika in Action published
- Latest release is Apache Tika 1.2
  - thousands of known media types
    - most with associated type detection patterns
  - dozens of supported document formats
    - including all major office formats
  - basic language detection
  - etc.
- For more information http://tika.apache.org/

Ohloh summary (http://www.ohloh.net/p/tika)

In a Nutshell, Apache Tika...

...has had 1,729 commits made by 18 contributors
representing 56,625 lines of code

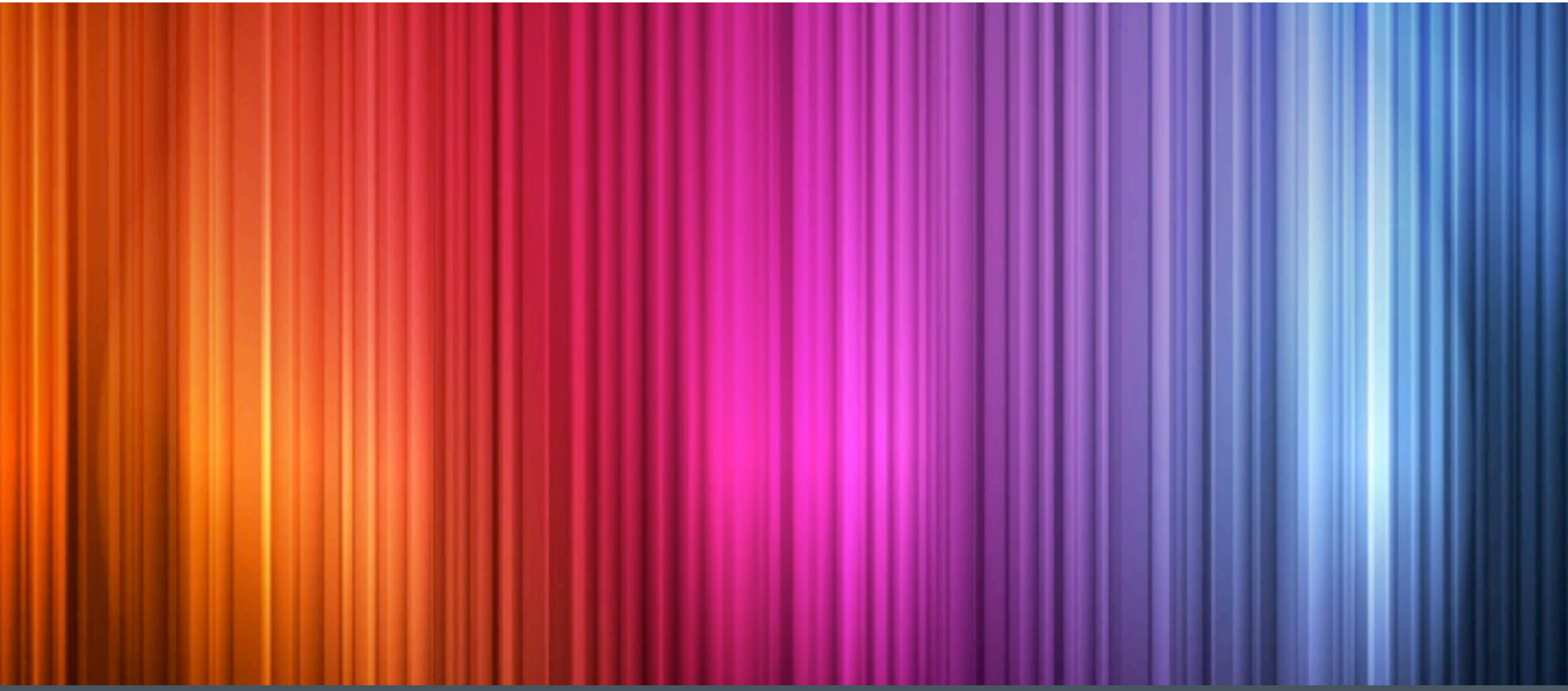
...is mostly written in Java
with an average number of source code comments

...has a well established, mature codebase
maintained by a large development team
with stable year-over-year commits

...took an estimated 14 years of effort (COCOMO model)
starting with its first commit in March, 2007
ending with its most recent commit 6 days ago

Full text extraction with Tika

section 2 / 4



Demo: tika-app-1.2.jar

- https://github.com/jukka/tika-demo
- java -jar tika-app-1.2.jar

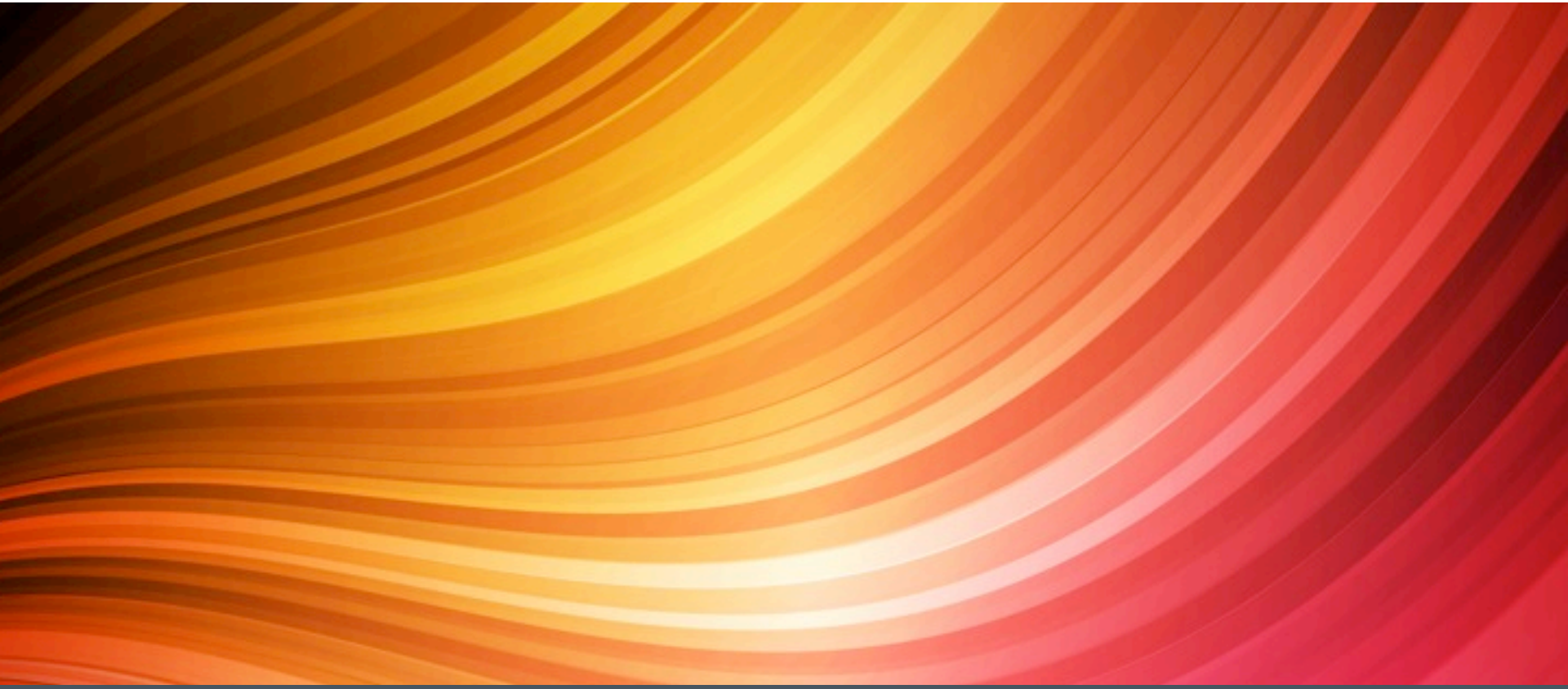
tika-app as a command line tool

```
$ java -jar tika-app-1.2.jar --xhtml /path/to/
document.doc

$ java -jar tika-app-1.2.jar --text http://example.com/
  document

$ java -jar tika-app-1.2.jar --metadata < document.doc

$ cat document.doc | java -jar tika-app-1.2.jar --text |
  grep foo

$ java -jar tika-app-1.2.jar --help
```

Tika's Java API

- Divided in two layers
  - The Tika facade: org.apache.tika.Tika
  - Lower-level interfaces like Parser, Detector, etc.
- Use the Tika facade by default
  - Provides simple support for most common use cases
  - Example: new Tika().parseToString("/path/to/document.doc")
- Use the lower-level interfaces for more power or flexibility
  - Allows fine-grained control of Tika functionality
  - More complicated programming model
    - Parsed content handled as XHTML SAX events
  - Not all functionality is exposed through the Tika facade

Tika and Solr - the ExtractingRequestHandler



ExtractingRequestHandler

- aka Solr Cell
- http://wiki.apache.org/solr/ExtractingRequestHandler

  "Solr's ExtractingRequestHandler uses Tika to allow users to upload binary files to Solr and have Solr extract text from it and then index it."

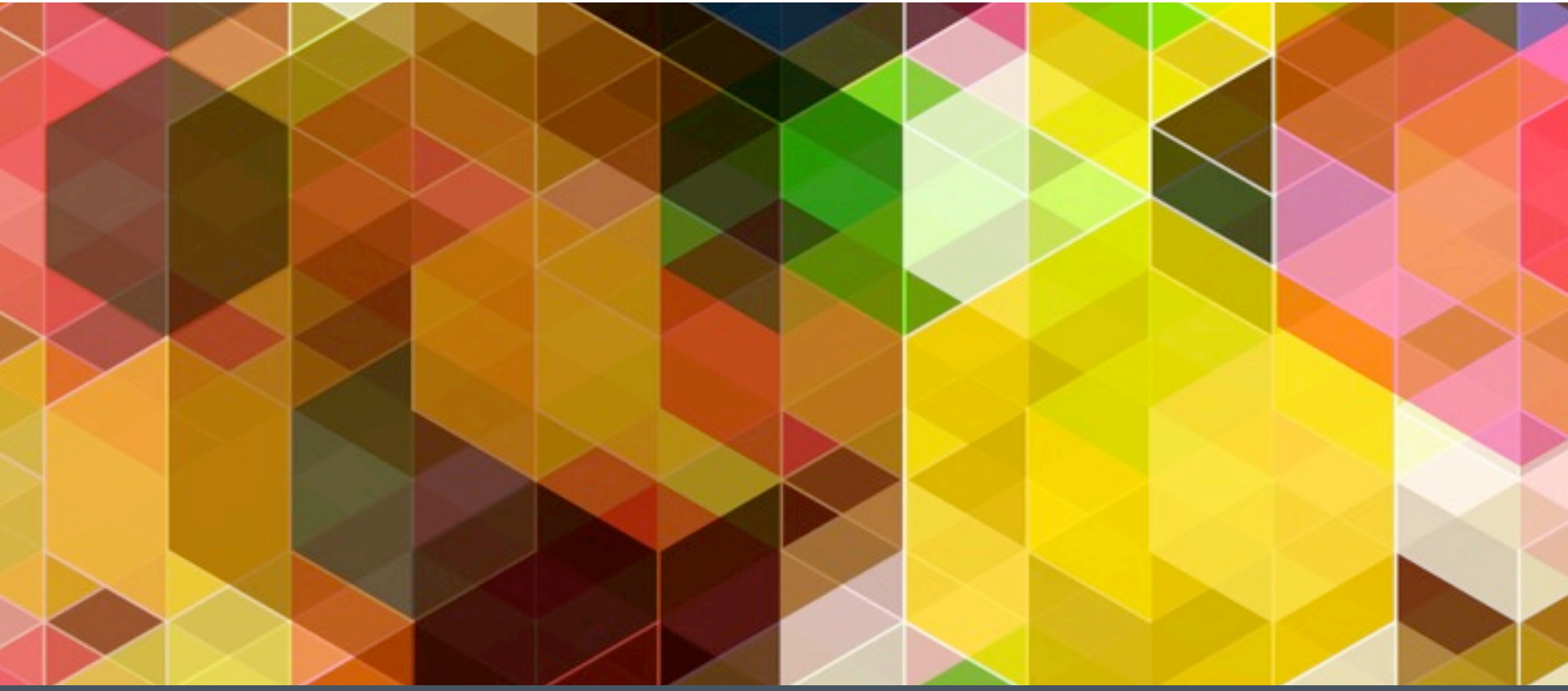
- For example:
  $ curl "http://localhost:8983/solr/update/extract?literal.id=document&commit=true" -F "file=@document.doc"

- Supports both text and metadata extraction
  - with plenty of configurable options

ExtractingRequestHandler parameters

- Helping Tika do it's job
  - resource.name=document.doc – Helps Tika's automatic type detection
  - resource.password=secret – Allows Tika to read encrypted documents
  - passwordsFile=/path/to/password-file – Resource name to password mappings
    - for example: .*\.pdf$ = pdf-secret
- Capturing special content
  - xpath=//a – Capture only content inside elements that match the specified query
  - capture=h1 – Capture content inside specific elements to a separate field
  - captureAttr=true – Capture attributes into separate fields named after the element
- Mapping field names
  - lowernames=true – Normalize metadata field names to "content_type", etc.

Tika and Lucene - direct feeding of the index



Using the Tika facade to feed Lucene

```
// Index first part of the document
String text = new Tika().parseToString("/path/to/document.doc");
document.add(new TextField("text", text, Field.Store.NO));

// Index the full document
Reader reader = new Tika().parse("/path/to/document.doc");
document.add(new TextField("text", reader));

// Index also some metadata
Metadata metadata = new Metadata();
Reader reader = new Tika().parse(
    new FileInputStream("/path/to/document.doc"), metadata);
document.add(new TextField("text", reader));
document.add(new StringField("type",
metadata.get(Metadata.CONTENT_TYPE));
```

Things to consider

- What if the document is larger than your memory?
  - Index only first N bytes/characters?
  - WriteOutContentHandler supports an explicit write limit
    - Enabled by default in the Tika facade, see get/setMaxStringLength()
- What if the document is malformed or intentionally broken?
  - Could cause denial of service problems
    - Might even crash the entire JVM due to bugs in native libraries in the JDK!
  - SecureContentHandler monitors parsing and terminates it if things look bad
    - Enabled by default in the Tika facade
- Ultimate solution: forked parsing and the Tika server
  - Parse documents in separate, sandboxed JVM processes
  - A document could fail to parse, but your application won't crash
  - Code is already there, but still a bit tricky to set up

Link extraction for web crawlers

- The LinkContentHandler class can be used to extract all links from a document
  - Works also with links in things like PDF, MS Word and email documents
  - Use TeeContentHandler to combine with other ways of capturing content

```
// for example
LinkContentHandler lch = new LinkContentHandler();
BodyContentHandler bch = new BodyContentHandler();
new Tika().getParser().parse(..., new TeeContentHandler(lch, bch), ...);

System.out.println("Content: " + bch);
for (Link link : lch.getLinks()) {
    System.out.println("Link: " + link):
}
```

Questions?

http://tika.apache.org/


Adobe