TAPAD

Alexa voice assistant

Part 1: Build a very simple search engine

Luca Venturi

Italian, from Modena
Tapad, Senior Software Engineer
Moved to Oslo in 2008


TAPAD
WE ARE HIRING!
WE'VE GOT A SWEET STACK

LEAD/ARCHITECT/DESIGN/DEVELOP

Google BigQuery

Google Dataflow

Scala

Google Dataproc

Spark

LEARN/GROW/DESIGN/DEVELOP

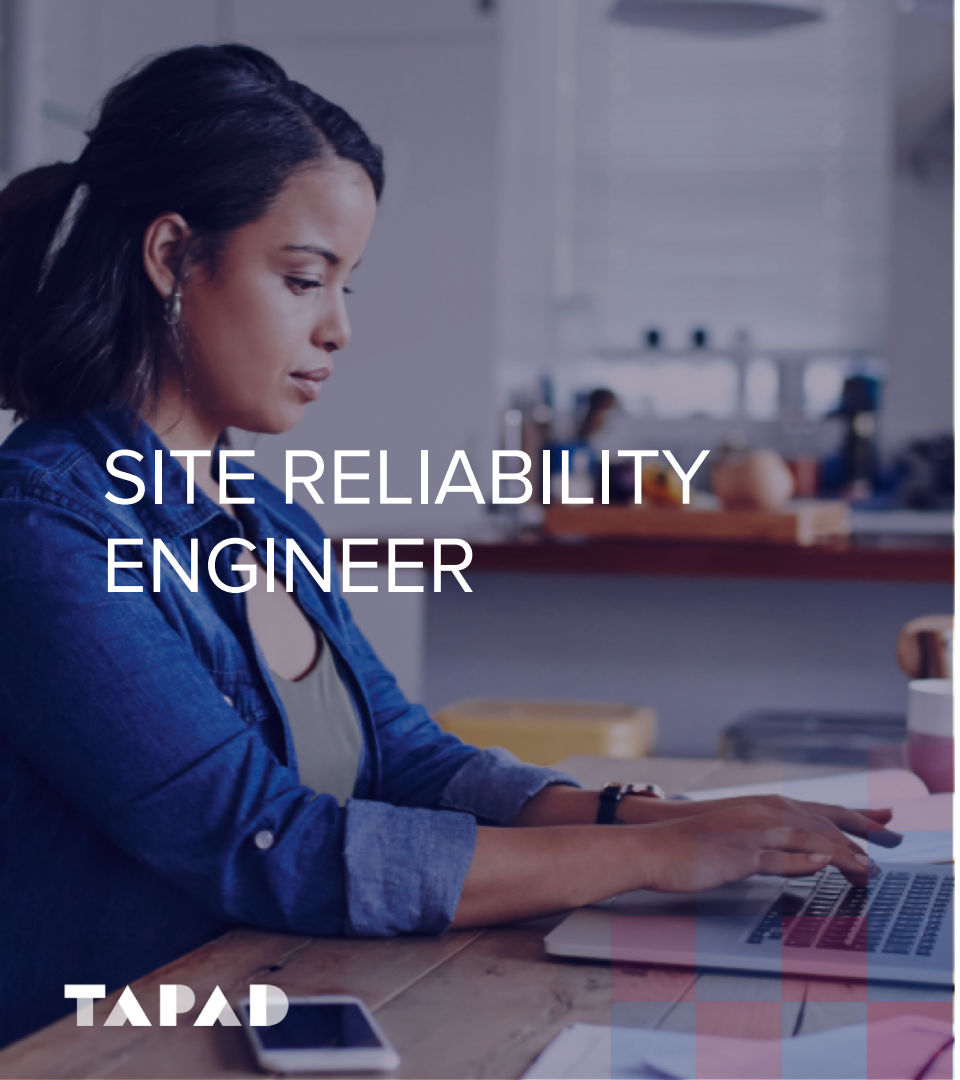
Google BigQuery

Google Dataflow

Scala

Google Dataproc

Spark



ARCHITECT/DESIGN/CI/CD/MAINTAIN

Google BigQuery

Google Dataproc

Google Dataflow

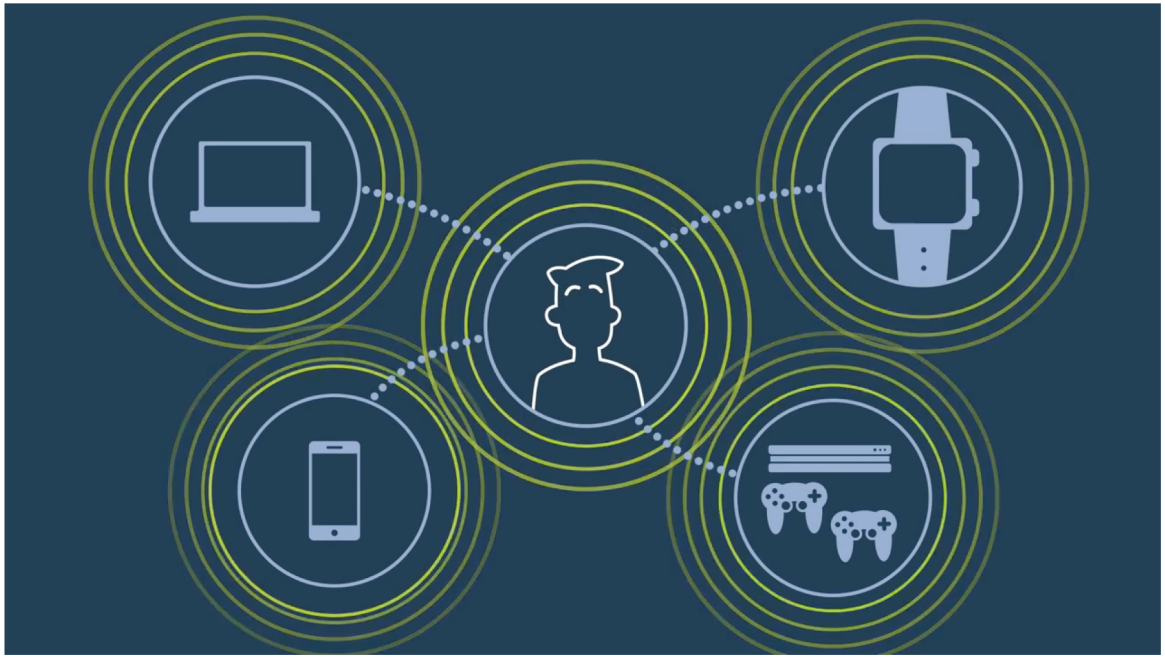
kubernetes

docker

CoreOS

Google Cloud Platform

INVENTORS OF THE DEVICE GRAPH



TAPAD

Need more reasons to join?

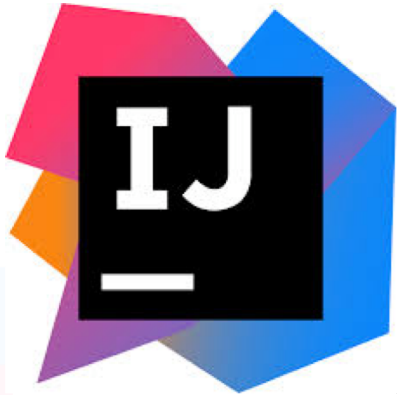
- Nice and competent colleagues (you have a selection here!)
- Startup environment + Telenor benefits
- Training in **New York City**
- Centrally located
- All the coffee that you can drink, all the ping pong that you can play

TAPAD

Just in case you haven't done so...

- Please download IntelliJ Community Edition
- Please download Java JDK 8

It might take some time, as we are many!

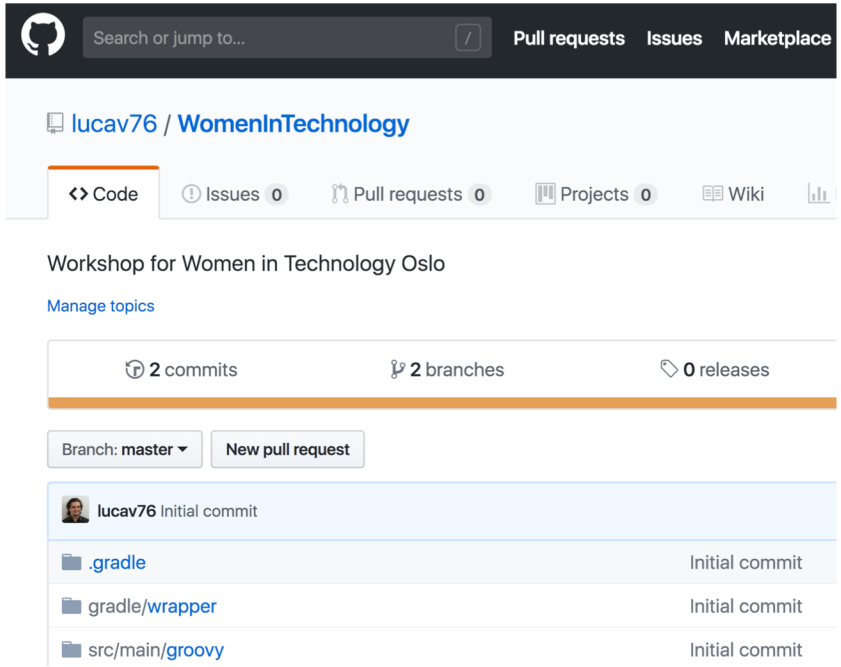


The plan

- Groovy
- 3 ways to access Wikipedia content
- Some tasks
- Find the links in a page
- Crawl some pages from Wikipedia
- Some questions
- Mini full-text search engine
- Q&A

For the impatient

https://github.com/lucav76/WomenInTechnology



Apache Groovy? Why Groovy?

- Powerful but easy
- Code compact and dense
- Runs on the JVM
- Java code is often valid Groovy code
- It’s very simple to download a URL:
  "https://www.google.com".toURL().text

What are we downloading?

```
def topic = "Pizza"
def textUrl = "https://en.wikipedia.org/wiki/$topic"
```

Pizza

From Wikipedia, the free encyclopedia

For other uses, see Pizza (disambiguation).

Pizza (Italian: [ˈpittsa], Neapolitan: [ˈpittsə]) is a savory dish of Italian origin, consisting of a usually round, flattened base of leavened wheat-based dough topped with tomatoes, cheese, and various other ingredients (anchovies, olives, meat, etc.) baked at a high

Pizza

Pizza Margherita, the archetype of Neapolitan pizza

Type Flatbread

- Groovy **$**: String interpolation, put a variable in a string

What are we downloading?

```
title="Edit this template">e</abbr></a></li></ul></div></td></tr></tbody></table>
<p><b>Pizza</b> (<small>Italian: </small><span title="Representation in the
International Phonetic Alphabet (IPA)" class="IPA"><a
href="/wiki/Help:IPA/Italian" title="Help:IPA/Italian">[ˈpittsa]</a></span>,
<small>Neapolitan: </small><span title="Representation in the International
Phonetic Alphabet (IPA)" class="IPA"><a href="/wiki/Help:IPA/Neapolitan"
title="Help:IPA/Neapolitan">[ˈpittsə]</a></span>) is a savory <a
href="/wiki/Dish_(food)" title="Dish (food)">dish</a> of <a
href="/wiki/Italian_cuisine" title="Italian cuisine">Italian</a> origin,
consisting of a usually round, flattened base of <a href="/wiki/Leavening_agent"
title="Leavening agent">leavened</a> wheat-based <a href="/wiki/Dough"
title="Dough">dough</a> topped with tomatoes, cheese, and various other
ingredients (anchovies, olives, meat, etc.) baked at a high temperature,
traditionally in a  wood-fired oven.<sup id="cite_ref-1" class="reference"><a
href="#cite_note-1">&#91;1&#93;</a></sup> In formal settings, like a restaurant,
pizza is  eaten with knife and fork, but in casual settings it is cut into wedges
to be eaten <a href="/wiki/Finger_food" title="Finger food">while held in the
hand</a>. Small pizzas are sometimes called <a href="/wiki/Pizzetta"
title="Pizzetta">pizzettas</a>.
</p><p>The term <i>pizza</i> was first recorded in the 10th century in a <a
href="/wiki/Latin" title="Latin">Latin</a> manuscript from the <a
```

What are we looking for?

```
href="//en.wikipedia.org/w/index.php?title=Template:Pizza&amp;action=edit"><abbr title="Edit
this template">e</abbr></a></li></ul></div></td></tr></tbody></table>
<p><b>Pizza</b> (<small>Italian: </small><span title="Representation in the
International Phonetic Alphabet (IPA)" class="IPA"><a href="/wiki/Help:IPA/Italian"
title="Help:IPA/Italian">[ˈpittsa]</a></span>, <small>Neapolitan: </small><span
title="Representation in the International Phonetic Alphabet (IPA)" class="IPA"><a
href="/wiki/Help:IPA/Neapolitan" title="Help:IPA/Neapolitan">[ˈpittsə]</a></span>) is a
savory <a href="/wiki/Dish_(food)" title="Dish (food)">dish</a> of <a
href="/wiki/Italian_cuisine" title="Italian cuisine">Italian</a> origin, consisting of a
usually round, flattened base of <a href="/wiki/Leavening_agent" title="Leavening
agent">leavened</a> wheat-based <a href="/wiki/Dough" title="Dough">dough</a> topped with
tomatoes, cheese, and various other ingredients (anchovies, olives, meat, etc.) baked at a
high temperature, traditionally in a  wood-fired oven.<sup id="cite_ref-1" class="reference">
<a href="#cite_note-1">&#91;1&#93;</a></sup> In formal settings, like a restaurant, pizza is
eaten with knife and fork, but in casual settings it is cut into wedges to be eaten <a
href="/wiki/Finger_food" title="Finger food">while held in the hand</a>. Small pizzas are
sometimes called <a href="/wiki/Pizzetta" title="Pizzetta">pizzettas</a>.
</p><p>The term <i>pizza</i> was first recorded in the 10th century in a <a
```

Regular expressions: patterns used to select part of a text
Our “core” Regular Expression: ** .*? ** ➔ **The whole paragraph**

- ** ** Where Wikipedia puts the real content
- **.*?** Any character, any number of times, only one paragraph

Regular expressions in Groovy

```
groups = txt =~ "(?s)<p>(.*?)</p>"
```

- **(?s)** Single line mode: matches line terminators
- **( )** Capture group: the text is kept and can be retrieved
- *** *** *Where Wikipedia puts the real content*
- ***.*?*** *Any character, any number of times, only one paragraph*
- **=~** Groovy operator to apply the regular expression to a string

Method 1: parsing a page, the hard way

```
List<String> parseRegEx(String textUrl, int numParagraphs=1) {
    def txt = textUrl.toURL().text
    def groups = txt =~ "(?s)<p>(.*?)</p>"

    // Indexes: [group][capture]; usually you want [group][0]
    return groups.collect() { it[0] }.take(numParagraphs)
}
```

- The URL is downloaded
- The ”**p**” blocks (paragraphs) are selected
- A list with the first capture (e.g. all the paragraphs found) is returned
- Some cleaning
- ***Groovy collect()***: similar to map(), iterates over collections and transform each element
- **Groovy take()**: keeps only the first elements

Method 2: Parsing a page with JSoup

```
List<String> parseJSoup(String textUrl, int numParagraphs=1, int minLenght=30) {
    def doc = Jsoup.connect(textUrl).get()
    def paragraphs = doc.getElementsByTag( tagName: "p")

    return paragraphs.findAll { it.toString().length() > minLenght }.take(numParagraphs)
            .collect { it.toString() }
}
```

- The URL is downloaded and parsed
- The ”**p**” blocks (paragraphs) are selected
- Returns the “*numParagraphs*” paragraphs longer than “*minLength*” characters
- Some cleaning
- ***Groovy findAll()***: It finds all values matching the condition

Method 3: Wikipedia API

```groovy
String wikipediaApi(String topic, int numSentences=1) {
    def apiUrl = "https://en.wikipedia.org/w/api.php?action=query&titles=$topic&prop=extracts&format=json&explaintext&ex
    def json = new JsonSlurper().parse(apiUrl.toURL())

    return json.query.pages.entrySet()[0].value.extract
}
```

- The URL of the API call
- Parse the JSON answer
- Returns the content of the whole page


```
{
  "batchcomplete": "",
  "warnings": {
    "extracts": {
      "*": "\"exlimit\" was too larg
    }
  },
  "query": {
    "pages": {
      "24768": {
        "pageid": 24768,
        "ns": 0,
        "title": "Pizza",
        "extract": "Pizza (Italian:
      }
    }
  }
}
```

Wrapping it up

```
// Part 1: Html download and Parsing
def topic = "Pizza"
def textUrl = "https://en.wikipedia.org/wiki/$topic"

println "\n * RegEx parsing *\n"
WikiCrawler.instance.parseRegEx(textUrl,  numParagraphs: 1).each { showString(it) }

println "\n * JSoup parsing *\n"
WikiCrawler.instance.parseJSoup(textUrl,  numParagraphs: 1).each { showString(it) }

println "\n * Wikipedia API *\n"
println WikiCrawler.instance.wikipediaApi(topic,  numSentences: 3)
```

- Groovy **each {}**: executes a “for loop” with the supplied code
- Groovy **it**: current value during the iteration
- **showString()** print the text removing HTML

Output

```
 * RegEx parsing *

Pizza (Italian:['pittsa], Neapolitan:['pittsə]) is a savory dish of Italian c
  dough topped with tomatoes, cheese, and various other ingredients (anchovie
  wood-fired oven.1 In formal settings, like a restaurant, pizza is  eaten wi
  while held in the hand. Small pizzas are sometimes called pizzettas.


 * JSoup parsing *

Pizza (Italian:['pittsa], Neapolitan:['pittsə]) is a savory dish of Italian c
  dough topped with tomatoes, cheese, and various other ingredients (anchovie
  wood-fired oven.[1] In formal settings, like a restaurant, pizza is eaten w
  while held in the hand. Small pizzas are sometimes called pizzettas.


 * Wikipedia API *

Pizza (Italian: ['pittsa], Neapolitan: ['pittsə]) is a savory dish of Italian
  dough topped with tomatoes, cheese, and various other ingredients (anchovie
  wood-fired oven. In formal settings, like a restaurant, pizza is  eaten wit
  while held in the hand. Small pizzas are sometimes called pizzettas.
```

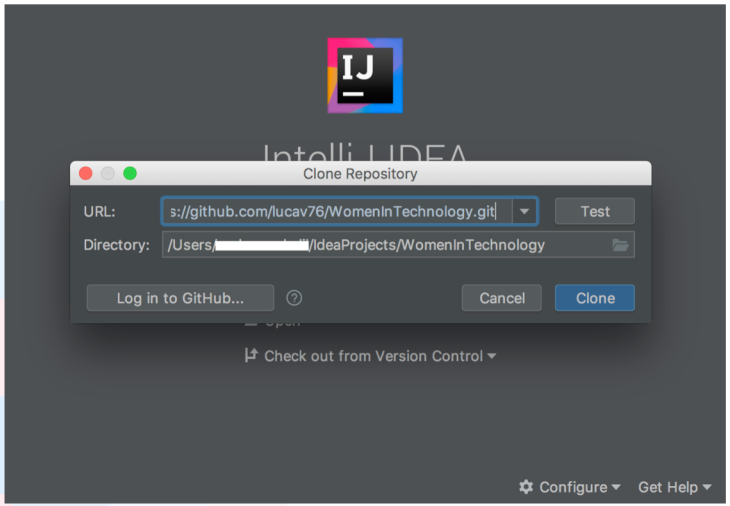
Task 0: New Project in IntelliJ Community

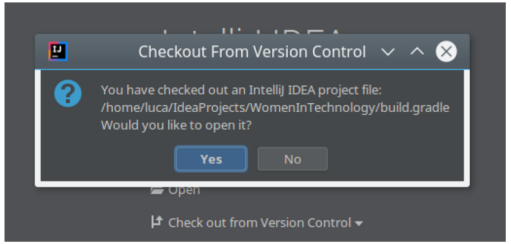
Clone Repository

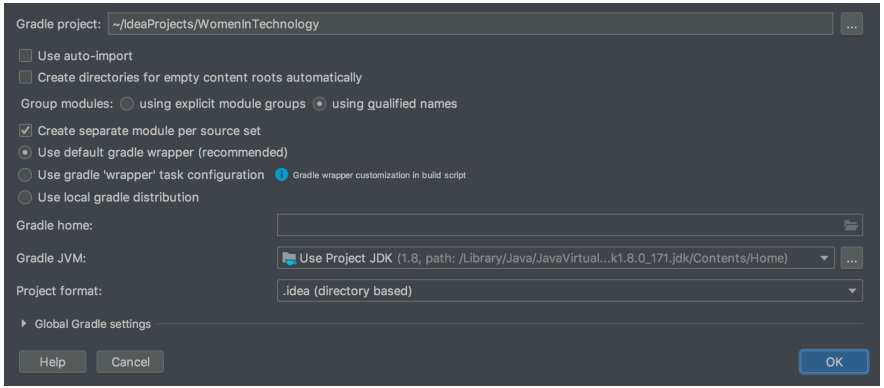
https://github.com/lucav76/WomenInTechnology.git

File/New/Project From Version Control/git

Java ***JDK*** 8

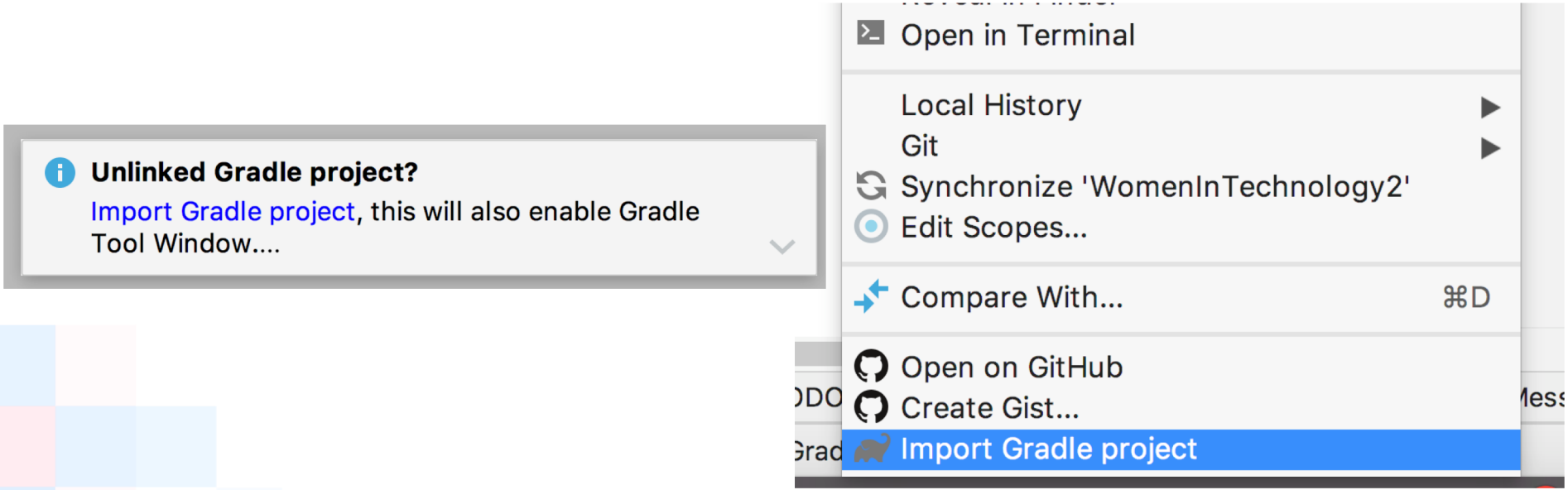






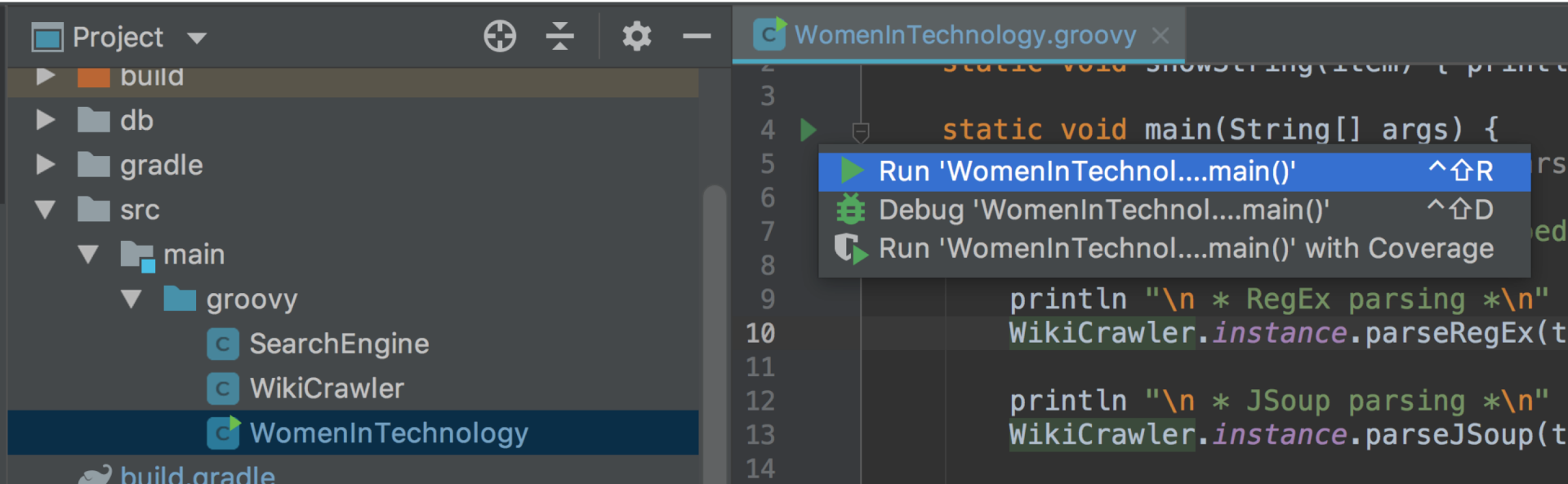
https://www.jetbrains.com/idea/

Task 0: Import Gradle



Task 0: run it!

- Run the program, see if you can get information about pizza
- **src/main/groovy/WomenInTechnology.groovy**

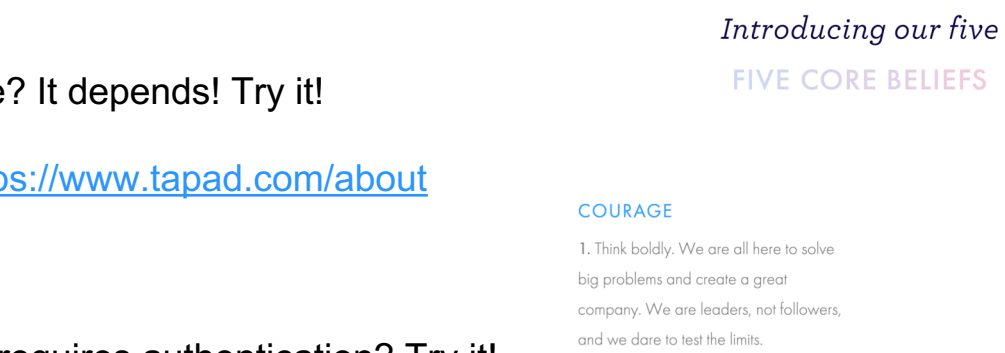


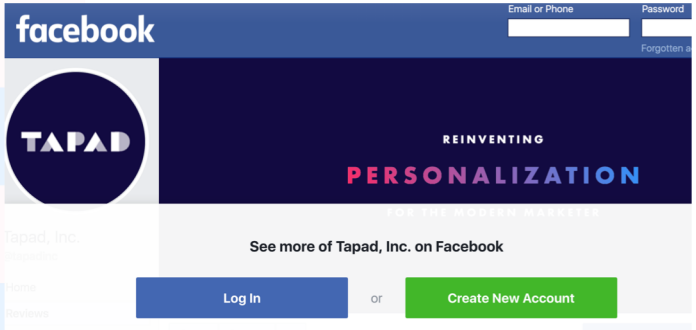
Task 1: search something else

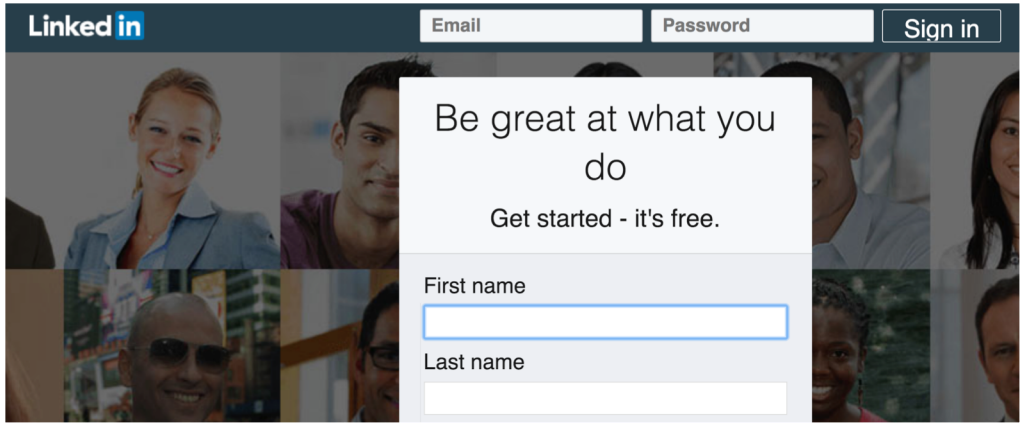
- Try to search some other words.
- Does it work?
- Can you download more than one paragraph?

Task 2 (optional): another site

- Is it easy to support another web site? It depends! Try it!
- Some ideas:
  - **Tapad** core beliefs, from https://www.tapad.com/about
  - Google News
  - Your favorite news paper
  - The menu of a restaurant
- (Not recommended) What if the site requires authentication? Try it!
  - Facebook
  - LinkedIn







Finding links in a page

```
List<String> extractLinks(String textUrl) {
    try {
        def links = Jsoup.connect(textUrl).get().select( cssQuery: "a[href]").asList()

        // Keeps only interesting links to other Wikipedia pages
        links.collect { (it.attr( attributeKey: "href") =~ /(\/wiki\/Category:.*)/) }
                .findAll()
                .collect { "https://en.wikipedia.org" + it[0][1] }
                .findAll {
            !it.contains( text: "maint") && !it.contains( text: "%") && !it.contains( text: ":Wikipedia") && !it.contains( text: "unsourced")
        }
        .unique()
    }
    catch (e) {
        return []
    }
}
```

- Mostly about selecting useful links

(Web) Crawling

Web crawlers are programs that scan the web, 'reading' everything they find and keeping tracks of the links

```
def links = WikiCrawler.instance.extractLinks(textUrl)
def links2 = links.collect { WikiCrawler.instance.extractLinks(it) + it }.flatten().unique()
```

- Groovy **flatten()**: List of lists => List
- Groovy **unique()**: Removes duplicates

Task 3: Bring your own crawler

- No code, just think about the challenges
- How would you download a complete website? 3 levels? 10? How do you know it is finished?
- How can you update a site without re-downloading it?
- How could you crawl the whole Internet?
- How do you “play nice” with websites?
- How do you make it fast?
- How do you choose the order to show the result

Google

Bing

Task 3: Possible solutions

- Download: List of unique pages downloaded, queue of unique pages to download, download and add links until the queue is empty
- Refresh content: HTTP conditional request (date, ETag)
- Crawl the whole internet: follow links to other websites, seeds sites (e.g. site directories), your previous list, buy site lists, etc…
- To play nice: Follow robots.txt (disallow paths), follow site maps (list of pages in the web site)
- To make it fast, you need an index!
- To choose the order, you can use PageRank or similar!

Google

A mini “full text” search engine

```groovy
void downloadAndSaveSingleUrl(URL url) {
    try {
        def file = new File(outputDir, url.getPath().replaceAll( regex: "[/:]",  replacement: "_") + ".json")

        if (!file.exists()) {
            file.text = JsonOutput.toJson([
                    url: url,
                    text: url.text])
            log.info("Downloaded " + url.getPath())
        }
    } catch (e) {
        log.severe("Error while downloading " + url.getPath() + ": " + e)
    }
}
```

- Download an URL and save it, if it is not on disk yet
- Groovy **try-catch**: to catch download exceptions

A mini “full text” search engine

```
List search(String text) {
    log.info( msg: "Searching for $text")
    outputDir.listFiles().collect() { new JsonSlurper().parseText(it.text) }.findAll {
        it.text.toLowerCase().contains(text.toLowerCase())
    }
}

List searchRegEx(String text) {
    log.info( msg: "Searching RegEx $text")
    outputDir.listFiles().collect() { new JsonSlurper().parseText(it.text) }.findAll {
        it.text.toLowerCase().matches("(?s).*"+text.toLowerCase()+".*")
    }
}
```

- Load all the files saved, searching for the text
- **Flexible** but **Slow** and **not scalable**: you need an index!

Wrapping it up

```
// Part 3: Search Engine
SearchEngine.instance.downloadAndSaveAllUrls(links2)

println "\n\nPages containing 'gnocco'"
SearchEngine.instance.search( text: "gnocco").each { println it.url }

println "\nPages containing 'gno..o'"
SearchEngine.instance.searchRegEx( text: "gno..o").each { println it.url }

println "\nPages containing Calzone"
SearchEngine.instance.search( text: "Calzone").each { println it.url }
```

Q&A

Useful links

- GitHub: https://github.com/lucav76/WomenInTechnology
- Groovy: http://groovy-lang.org
- jsoup: https://jsoup.org
- Wikipedia API to get the whole page: https://en.wikipedia.org/w/api.php?action=query&titles=$topic&prop=extracts&format=json&explaintext
- Wikipedia API to get some sentences: https://en.wikipedia.org/w/api.php?action=query&titles=$topic&prop=extracts&format=json&explaintext&exsentences=$numSentences