# Document Smart Highlights

## 1 Version

0.2.0-SNAPSHOT - 20190408-011917

## 2 Introduction

Welcome to **DSH - Document Smart Highlights**. Document Smart Hightlights aims to provide a set of web services to allow uploading of PDF and HTML files and return a list of keywords and most relevant sentences. This system applies state of the art algorithms, using NLP techniques to produce both the list of keywords and most relevant sentences. This last one, using automatic document summarization techniques. More details at the wiki.

This project is distributed as source code. In order to generate the application to run it is needed to install the pre-requisites and compile. No binaries distribution is provided for now. The final application is a java `.WAR` file which can be] dropped in a standard java servlet container.

The goals for this project are basically two:

1. Implement a NLP based system to extract relevant information from PDF files in a potentially scalable fashion by using NoSql databases and Queues. This infrastructure stores requests and chain a worflow of operations (workers) which will do the operations of keyword and relevant sentences extraction. All provided as a web service REST API.
2. Learn concepts of NLP associated with scaleable cloud based REST API building. The technology used is java based, so as another goal here we can mention the build learning process using Spring Framework in order to build the API and infrastructure.

As part of the goals is learning about technologies, comments and contributions are welcome. However, this is not a final product, application or concept. Just a point for experimentation and proofing.

If anyone out there is interested in contribute or apply this project on a more product-oriented environment, please get in touch through the email: marcelo.riss@gmail.com.

Wiki: https://github.com/MRISS-Projects/dsh/wiki

Project Development Documentation: https://mriss-projects.github.io/dsh-docs/

## 3 Package/Folders Description

- **DSH-data**: data models definition for a **Document**, **Keyword** and **RelevantSentence**. Additionally this module defines a workflow to map the status progress of a document processing request.
- **DSH-doc-analyzer**: This is a container module to have the keyword and relevant sentences extractor modules. One extra module to dequeue documents to be analyzed and call the extractors.
- **DSH-doc-processor-worker**: The keyword extraction is executed by setting scores to each term in the document's text.
- **DSH-keyword-extractor**: The keyword extraction is executed by setting scores to each term in the document's text.
- **DSH-top-sentences-extractor**: The extraction of top sentences is achieved applying typical automatic summarization techniques like extractive summarization or key phrase extraction.
- **DSH-doc-indexer-worker**: dequeues a document id from a queue, gets the document from the database and send it for indexing at SOLR. Besides indexing, this module will extract the text, paragraphs, sentences on each paragraph and terms on each sentence.

- **DSH-rest-api**: this module is the real application to be deployed or installed on a servlet container. It is the entry point for document submission, document processing status querying and document processing results (keywords and relevant sentences) querying.
- **dsh-test-dataset**: default and common data set of files used for automated testing.

# 4 Installation

### 4.1 Pre-requisites

- Java 1.8
- Maven 3.3.9
- MongoDB 3.4 (windows 10)
- MongoDB 4.0.6 (Ubuntu 18.04 LTS)
- RabbitMQ 3.6.14 (windows 10)
- RabbitMQ 3.7.14 (Ubuntu 18.04 LTS)
- Tomcat 8.0.X

### 4.2 Installing/Building the Application

#### 4.2.1 Java

Download and Installation

1. Download a J2SE JDK 1.8 platform from http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html.
    1. **IMPORTANT NOTE**: Download and install **JDK, not a JRE**. Avoid downloading packages with J2EE and/or net beans. Search the download page for Java SE Development Kit (JDK) or JDK 8 Update XX.
2. Windows
    1. There should be a .exe windows installer. Just follow the instructions.
3. Linux
    1. Download the .tar.gz file. After downloading it, uncompress it at a folder of your preference.
    2. Create a link. Open a command prompt, go to the JDK parent folder (the folder where you extract JDK into), and type:

```
ln -s jdk1.8.0_XX java (where XX is the update number of your download)
```

Setting environment variables

*Linux*

1. Open the file `/home/[YOUR_USER]/.profile`. This file might be hidden. If it does not appear at your home folder, using the file explorer, type `Ctrl+h`. Go to the end of the file and add:

```
export JAVA_HOME=/your/jdk/parent/folder export PATH=$JAVA_HOME/bin:
$PATH
```

*Windows*

1. Open Control Panel go to System, Advanced system settings, `Environment Variables` button.

2. At the System Variables section, click New.
3. Set JAVA_HOME and point to the root of JDK folder.
4. Search for the variable named «Path» in the list, click on it and press Edit.
5. Prepend the value with:

```
%JAVA_HOME%\bin;
```

Verifying the installation

1. Open a command prompt and type:

`java -version` 2. The result should be something like:

```
```

java version “1.8.0_45” Java(TM) SE Runtime Environment (build 1.8.0_45-b14) Java HotSpot(TM) 64-Bit Server VM (build 25.45-b02, mixed mode) ```

4.2.2 Maven

1. Dowload maven **3.3.9** from http://archive.apache.org/dist/maven/maven-3/3.3.9/binaries/apache-maven-3.3.9-bin.zip
2. Unzip it on a folder of your preference
3. Set environment variables.
4. Linux
    1. Put it at your `$HOME/.profile` file

    ```
    ```
    export M2_HOME=/path/to/where/you/extracted/maven/apache-maven-3.3.9
    export PATH=$M2_HOME/bin:$PATH
    export MAVEN_OPTS='-Xmx1024m -XX:MaxPermSize=256m'
    ```
    ```

    1. If you already have java set up, your `.profile`, it should look like this:

    ```
    ```
    export JAVA_HOME=/your/jdk/parent/folder/java
    export M2_HOME=/path/to/where/you/extracted/maven/apache-maven-3.3.9
    export PATH=$JAVA_HOME/bin:$M2_HOME/bin:$PATH
    export MAVEN_OPTS='-Xmx1024m'
    ```
    ```

    1. Logout and login again.
    2. Test by opening a terminal and typing:

    ```
    ```
    mvn -version
    ```
    ```

    1. The result should be similar to:

```
Java HotSpot(TM) 64-Bit Server VM warning: ignoring option MaxPermSize=256m; su
Apache Maven 3.3.1 (cab6659f9874fa96462afef40fcf6bc033d58c1c; 2015-03-13T17:10
Maven home: /home/riss/apps/maven
Java version: 1.8.0_45, vendor: Oracle Corporation
Java home: /home/riss/apps/jdk1.8.0_45/jre
Default locale: en_US, platform encoding: UTF-8
OS name: "linux", version: "3.13.0-55-generic", arch: "amd64", family: "unix"
```

1. Windows
2. Open Control Panel go to System, Advanced system settings, **Environment Variables** button.
3. At the System Variables section, click New.
4. Set `M2_HOME` and point to the root of maven folder.
5. Search for the variable `Path` in the list, click on it and press Edit.
6. Put the maven bin folder right after java home:

```
%JAVA_HOME%\bin;%M2_HOME%\bin;
```

1. Set the variable `MAVEN_OPTS`.

```
MAVEN_OPTS=-Xmx1024m
```

1. The result should be similar to:

```
Apache Maven 3.3.9 (0728685237757ffbf44136acec0402957f723d9a; 2013-09-17 12:22
Maven home: C:\data\apache-maven-3.3.9
Java version: 1.8.0_45, vendor: Oracle Corporation
Java home: C:\Program Files\Java\jdk1.8.0_45\jre
Default locale: en_US, platform encoding: Cp1252
OS name: "windows 8.1", version: "6.3", arch: "amd64", family: "dos"
```

#### 4.2.3 MongoDB

Windows

1. Install MongoDB using the instructions at this link
2. Enable security following general guidelines at this link
3. Start MongoDB:

```
"C:\Program Files\MongoDB\Server\3.4\bin\mongod.exe"
```

4. In another prompt connect to MongoDB

`"C:\Program Files\MongoDB\Server\3.4\bin\mongo.exe"` 5. Create super user

`$ use admin $ db.createUser( { user: "superAdmin", pwd: "[your admin password]", roles: [ { role: "root", db: "admin" } ] })` 6. Disconnect and re-connect at MongoDB as super user:

`connect-mongo-super-user.bat [your admin password]` 7. Create user access (readWrite) for specific dsh database

`$ use dsh $ db.createUser( { user: "dshuser", pwd: "[your password]", roles: [ "readWrite"] })` 8. Disconnect and re-connect at MongoDB as specific user:

`connect-mongo.bat [your dshuser passoword]`

Linux Ubuntu 18.04 LTS

1. Install MongoDB following the instructions at https://docs.mongodb.com/manual/tutorial/install-mongodb-on-ubuntu/
2. Enable security following general guidelines at this link
3. Start MongoDB service

`sudo service mongod start` 4. In another prompt connect to MongoDB

`mongo --host 127.0.0.1:27017` 5. Create super user

`$ use admin $ db.createUser( { user: "superAdmin", pwd: "[your admin password]", roles: [ { role: "root", db: "admin" } ] })` 6. Disconnect and re-connect at MongoDB as super user:

`./connect-mongo-super-user.sh [your admin password]` 7. Create user access (readWrite) for specific dsh database

`$ use dsh $ db.createUser( { user: "dshuser", pwd: "[your password]", roles: [ "readWrite"] })`

1. Disconnect and re-connect at MongoDB as specific user:

`./connect-mongo.sh [your dshuser passoword]`

#### 4.2.4 RabitMQ

Windows

1. Follow the instructions at http://www.rabbitmq.com/install-windows.html
2. Enable the ports mentioned at the link above at the firewall.
3. Enable the management plugin:

`rabbitmq-plugins.bat enable rabbitmq_management rabbitmq-service.bat stop rabbitmq-service.bat remove rabbitmq-service.bat install rabbitmq-service.bat start` 4. Test it with `http://localhost:15672/mgmt`. User: guest. Password: guest.

Linux Ubuntu 18.04 LTS

1. Follow the instructions at https://www.rabbitmq.com/install-debian.html
    1. As Ubunt has a 3.5.x version it is better to download the .deb for version 3.7.x from link above

2. Or follow the instructions at the link and add RabbitMQ Ubuntu repositories before to run the `apt-get install`.

2. Enable the management plugin:

`sudo rabbitmq-plugins enable rabbitmq_management service rabbitmq-server stop service rabbitmq-server start` 3. Test it with `http://localhost:15672/mgmt`. User: guest. Password: guest.

#### 4.2.5 Building From Sources

1. In order to build, both MongoDB and RabbitMQ services should be running.
2. Maven development user settings should be correctly configured (see configuration section below)
3. At the root dsh folder type:

```
./install-parent-pom.sh
mvn clean install
```

#### 4.2.6 Tomcat

##### Windows

1. Download Tomcat 8.0.X 32-bit/64-bit Windows Service Installer at https://tomcat.apache.org/download-80.cgi. This will install Tomcat as a windows service.
2. Start Tomcat windows service using windows services application.
3. Look at the address: http://localhost:8080

##### Linux Ubuntu 19.04 LSTS

1. Download Tomcat 8.0.x .zip or .tar.gz file at https://tomcat.apache.org/download-80.cgi.
2. Unpack the contents on a folder.
3. Go to the `bin` folder and type `./startup.sh`.
4. Look at the address: http://localhost:8080

## 5 Configuration

### 5.1 MongoDB Access Properties

Edit or create the maven user `settings.xml` file typically at `$HOME/.m2` (or `%HOMEPATH%\.m2` at windows) folder and add a default activated profile similar to this:

```
<profile>
<id>development-properties</id>
<activation>
<activeByDefault>true</activeByDefault>
</activation>
<properties>
<mongo.host>localhost</mongo.host>
<mongo.port>27017</mongo.port>
<mongo.user>dshuser</mongo.user>
<mongo.password>[password you have configured in the steps above when installing mo
</properties>
</profile>
```

Or add the `properties` section at any default activated profile already present at `settings.xml` file.

#### 5.2 Tomcat Admin User Configuration

Stop Tomcat if it is already started, and edit the file `TOMCAT_HOME/conf/tomcat-users.xml`. If the `tomcat-users` tag is empty or with all elements commented, add the following content inside the `<tomcat-users>` tag.

```
<role rolename="tomcat" />
<role rolename="manager-gui" />
<role rolename="manager-script" />
<role rolename="admin-gui" />

<user username="admin" password="[your admin password]" roles="tomcat,manager-gui,m
<user username="tomcat" password="[your tomcat user password]" roles="tomcat,manage
```

Replace the admin and tomcat's password with any desired password.

## 6 Usage

#### 6.1 Running Application from Eclipse Embedded Tomcat

The module `DSH-rest-api` is a web application. The type tag in pom.xml file is .war. Thus the first step is to install a Tomcat (8.0.X) at https://tomcat.apache.org/download-80.cgi. After that, if you have Eclipse Oxygen JEE version correctly installed and configured, then is just a matter of showing the Servers view and adding a new server. At eclipse menu, follow the path: `Window -> Show View -> Other -> Servers -> Server`. When the `Servers` view opens, add a new Tomcat Server. You will need to have a Tomcat already installed at your system, since eclipse will ask for an installed Tomcat root directory. When creating a new server inside eclipse, it will show the DSH-rest-api as a potential project to be installed in that server.

After having the DSH-rest-api inside the server, configure the server `startup` and `shutdown` timeouts to something like 120s each.

Start the server and access the application swagger UI at: `http://localhost:8080/DSH-rest-api/swagger-ui.html`.

### 6.2 Using Application .war File on a Servlet Container

After the build, the folder DSH-rest-api/target should have a file named DSH-rest-api-[version].war. That war file can be dropped to a servlet container to be used as a web application. At this moment the server having the servlet container should be the same having MongoDB and RabbitMQ installed, up and running.

#### 6.2.1 Using Tomcat Application Manager

Access the Tomcat's manager usually at the address `http://localhost:8080/manager/html`. The browser will ask for user and password. Enter the user and password configured at the `TOMCAT_HOME/conf/tomcat-users.xml` (see the configuration section above).

After login, at the Deploy section, fulfill the fields:

```
Context Path:  DSH-rest-api
WAR or Directory URL: [absolute path to the generated DSH-rest-api-<version>.war fi
```

You can also upload the war file from `DSH-rest-api/target` folder, using the `Choose File` button at the Tomcat's manager application. However, in this case, it is recommenDed to rename the file `DSH-rest-api-<version>.war` to just `DSH-rest-api.war` just to not have the version name associated with the web application, which will then be used to access the application at the web browser.

Start the server and access the application swagger UI at: `http://localhost:8080/DSH-rest-api/swagger-ui.html`.

#### 6.2.2 Just Dropping Application .war File

Rename the file `DSH-rest-api/target/DSH-rest-api-<version>.war` to `DSH-rest-api.war` and drop it at Tomcat's `webapps` folder. Restart Tomcat if needed.

Start the server and access the application swagger UI at: `http://localhost:8080/DSH-rest-api/swagger-ui.html`.

### 6.3 Running Application Using Spring Boot Maven Plugin

Go to `DSH-rest-api` module root folder project, by using `cd DSH-rest-api` at the sources root, and run:

```
mvn spring-boot:run
```

Wait until the application boots up. Typically when the following output is present:

```
.
.
.
08:37:47.665 [main] INFO  o.s.c.s.DefaultLifecycleProcessor - Starting beans in pha
08:37:47.665 [main] INFO  s.d.s.w.p.DocumentationPluginsBootstrapper - Context refr
08:37:47.702 [main] INFO  s.d.s.w.p.DocumentationPluginsBootstrapper - Found 1 cust
08:37:47.746 [main] INFO  s.d.s.w.s.ApiListingReferenceScanner - Scanning for api l
08:37:47.962 [main] INFO  o.a.coyote.http11.Http11NioProtocol - Initializing Protoc
08:37:47.979 [main] INFO  o.a.coyote.http11.Http11NioProtocol - Starting ProtocolHa
08:37:47.984 [main] INFO  o.a.tomcat.util.net.NioSelectorPool - Using a shared sele
08:37:48.016 [main] INFO  o.s.b.w.e.tomcat.TomcatWebServer - Tomcat started on port
08:37:48.022 [main] INFO  c.m.dsh.restapi.DshRestApplication - Started DshRestAppli
08:37:48.025 [main] INFO  c.m.dsh.restapi.DshRestApplication - Main application run
```

Start the server and access the application swagger UI at: `http://localhost:8080/swagger-ui.html`.

#### 6.4 Swagger User Interface

Swagger UI has two methods for a document resource:

- submit: uploads a PDF file and returns a token.
- status: use the token returned in the first method to ask for the document processing status. At the moment the only status would be `QUEUED_FOR_INDEXING_SUCCESS`.

In order to use the methods, click on the method and after in the `Try it out` button (firstly for the submit method).

A new form will open with the fields to fulfill. In case of submit you will need to choose a file to upload and inform its title. In case of `status`, you just needs to enter the the token returned by the previous `submit` method call.

## 7 Release Notes

#### 7.1 Version 0.2.0-SNAPSHOT

| # | Type | Summary | Assignee | Reporter | Updated |
|---|---|---|---|---|---|
| 39 | task | Publish dsh site on gh-pages branch instead of another repo. | null | mriss | 3/18/19 |
| 41 | task | Move project from organization to the git project dsh. Change next milestone to 0.2.0 | mriss | mriss | 3/18/19 |

### 7.2 Version 0.0.1

| # | Type | Summary | Assignee | Reporter | Updated |
|---|---|---|---|---|---|
| 32 | task | Configure DSH to use git as scm tool and proceed to release. | mriss | mriss | 3/15/19 |
| 38 | task | Configure distribution management to local nexus and test snapshot deploy with deployment profile. | mriss | mriss | 3/1/19 |
| 37 | task | Replace release notes and release history properties using deployment profile. | mriss | mriss | 10/14/18 |
| 36 | task | Replace version property ad readme and commit using deployment profile | mriss | mriss | 2/19/18 |
| 35 | task | Test maven site publication using github using deployment profile. | mriss | mriss | 2/15/18 |
| 34 | task | Configure changes plugin and changes report to use github issues and test maven site generation. | mriss | mriss | 2/12/18 |
| 33 | task | Configure maven scm to use git | mriss | mriss | 2/12/18 |
| 10 | task | Test rest API module inside tomcat server inside eclipse as a war distribution. | mriss | mriss | 2/11/18 |
| 30 | task | Configure Swagger | mriss | mriss | 2/4/18 |
| 11 | task | Create and test rest service layer: | mriss | mriss | 1/16/18 |

| | | | | | |
|---|---|---|---|---|---|
| 23 | task | Add message and error handling for RabbitMQ queue submission. | mriss | mriss | 1/9/18 |
| 22 | task | Create document submission workflow | mriss | mriss | 1/6/18 |
| 28 | task | Create document status enumeration and define workflow transition and validation class. | mriss | mriss | 1/6/18 |
| 27 | task | Add extra columns at the Document model class for status description and status message. | mriss | mriss | 1/6/18 |
| 25 | task | Test message sending exception. | mriss | mriss | 1/4/18 |
| 17 | task | Create web service logic to generate token and return it while starting the document storage at mongo asynchronously. | mriss | mriss | 1/3/18 |
| 9 | task | Create services: | mriss | mriss | 12/19/17 |
| 21 | task | Create mongodb storage service. | mriss | mriss | 12/19/17 |
| 20 | task | Update documentation with RabbitMQ installation. | mriss | mriss | 12/19/17 |
| 18 | task | Create logic to enqueue the mongo document id to RabbitMQ using Spring integration example app. | mriss | mriss | 12/18/17 |
| 19 | task | Feature/mongo dao | mriss | mriss | 12/10/17 |

| | | | | | |
|---|---|---|---|---|---|
| 15 | task | Create and test MongoDAO | null | mriss | 12/10/17 |
| 16 | task | Test models | mriss | mriss | 12/9/17 |
| 12 | task | Create dsh-test-dataset module having all PDF and HTML files used for testing. | mriss | mriss | 12/8/17 |
| 8 | task | * Create model for the documents with following columns: | mriss | mriss | 12/7/17 |
| 7 | task | * Create package structure | null | mriss | 12/7/17 |
| 3 | task | Create parent pom | mriss | mriss | 12/7/17 |
| 5 | task | Create a model module to have the model classes of keywords, sentences and documents. | mriss | mriss | 12/7/17 |
| 6 | task | Organize dependency management among modules. | mriss | mriss | 12/7/17 |
| 2 | task | Create project structure using spring boot | mriss | mriss | 12/7/17 |
| 4 | task | Complete project structure | mriss | mriss | 12/7/17 |
| 1 | task | Install RabbitMQ | mriss | mriss | 11/29/17 |