

# Java-Success.com

Prepare to fast-track, choose & go places with 800+ Java & Big Data Q&As with lots of code & diagrams.

[Home](#) [Why? ▾](#) [300+ Java FAQs ▾](#) [300+ Big Data FAQs ▾](#) [Courses ▾](#)

[Membership ▾](#) [Your Career ▾](#)

[Home](#) › [bigdata-success.com](#) › [Tutorials - Big Data](#) › [TUT - Spark Scala on Zeppelin](#) ›

06: Spark on Zeppelin – RDD operation zipWithIndex

## 06: Spark on Zeppelin – RDD operation zipWithIndex

 Posted on [August 25, 2018](#)

**Pre-requisite:** Docker is installed on your machine for Mac OS X (E.g. \$ brew cask install docker) or Windows 10. [Docker interview Q&As](#). This extends [setting up Apache Zeppelin Notebook](#).

**Q.** Why do we need zipWithIndex?

**A.** In database world there are various instances where we want to assign a unique sequence number to the records in a table. It may be a challenge in a distributed environment like Hadoop as we have to make sure we don't come across duplicate sequence numbers for data stored in multiple nodes. Before

### 300+ Java Interview FAQs

300+ Java FAQs



16+ Java Key  
Areas Q&As



150+ Java  
Architect FAQs



80+ Java Code  
Quality Q&As



150+ Java Coding  
Q&As



### 300+ Big Data Interview FAQs

300+ Big Data  
FAQs



Tutorials - Big  
Data



TUT -  Starting Big  
Data

TUT - Starting Spark &  
Scala

using `zipWithIndex` we could run `distinct()` and `sort()` functions to assign indices the way we want.

**Step 1:** Pull this from the docker hub, and build the image with the following command.

```
1 $ docker pull apache/zeppelin:0.7.3
2
```

You can verify the image with the “docker images” command.

**Step 2:** The input file to read “employees.txt” in the `$(pwd)/seed`.

```
1 00~Information About employees
2 01~1~John~USA~100000.00
3 01~2~Peter~Australia~200000.00
4 01~3~Sam~USA~76000.00
5 01~4~Daniel~France~86000.00
6 01~5~Simon~Australia~96000.00
7 01~6~Roseanne~France~156000.00
8 99~6 records
9
```

**Step 3:** Run the container with the above image.

```
1 $ docker run --rm -it -p 8080:8080 -v "$(pwd)/seed
2
```

**Note:** `$(pwd)/seed` – is the folder where the `employees.txt` input file will be placed on the host system, and will be synchronized with the container path “/zeppelin/seed”.

You can inspect the container files/logs with the following commands in a separate terminal window:

TUT - Starting with Python

TUT - Kafka

TUT - Pig

TUT - Apache Storm

TUT - Spark Scala on Zeppelin

TUT - Cloudera

TUT - Cloudera on Docker

TUT - File Formats

TUT - Spark on Docker

TUT - Flume

TUT - Hadoop (HDFS)

TUT - HBase (NoSQL)

TUT - Hive (SQL)

TUT - Hadoop & Spark

TUT - MapReduce

TUT - Spark and Scala

TUT - Spark & Java

TUT - PySpark on Databricks

TUT - Zookeeper

## 800+ Java Interview Q&As

300+ Core Java Q&As



300+ Enterprise Java Q&As



150+ Java Frameworks Q&As



120+ Companion Tech Q&As



Tutorials - Enterprise Java



Get the container id with:

```
1 $ docker ps
2
```

sh to the container with:

```
1 $ docker exec -it <container id> /bin/bash
2
```

**Step 4:** Open Zeppelin notebook via a web browser "http://localhost:8080". Create a note book with "spark" as a default interpreter.

The following code gets rid of the header & footer record types with values "00" and "99" respectively, and extracts out the employee details as shown below:

```
1 %spark
2
3 val lines_rdd = sc.textFile("file:///zeppelin/see
4
5 val cached_rdd = lines_rdd.cache()
6 val total_records = cached_rdd.count()
7
8 case class Employee (id: Integer, name: String, lo
9
10 val employee_df = cached_rdd.zipWithIndex()
11     .filter(_._2 > 0)
12     .filter(_._2 < (total_records - 1))
13     .map(_._1.split("~"))
14     .map(s => Employee(s(1).toInt, s(2).toInt, s(3))
15     .toDF()
16
17 employee_df.show(false)
18
```

**Output:**

```

1  +---+-----+-----+-----+
2  |id|name    |location|salary|
3  +---+-----+-----+-----+
4  |1 |John    |USA     |100000.0|
5  |2 |Peter   |Australia|200000.0|
6  |3 |Sam     |USA     |76000.0 |
7  |4 |Daniel  |France  |86000.0 |
8  |5 |Simon   |Australia|96000.0 |
9  |6 |Roseanne|France  |156000.0|
10 +---+-----+-----+-----+
11

```

## Getting the bottom n records

Getting the top “N” records is straightforward (E.g. `df.limit(2)`). How do we get the bottom “N” records?

```

1  %spark
2
3  val lines_rdd = sc.textFile("file:///zeppelin/see
4
5  val cached_rdd = lines_rdd.cache()
6  val total_records = cached_rdd.count()
7
8  case class Employee (id: Integer, name: String, lo
9
10 val startIndex: Long = (total_records - 2) - 2
11
12
13 val employee_df = cached_rdd.zipWithIndex()
14     .filter(_._2 > 0)
15     .filter(_._2 < (total_records - 1)
16     .filter{case (_, index) => index >
17     .map(_._1.split("~"))
18     .map(s => Employee(s(1).toInt, s(2)
19     .toDF()
20
21 employee_df.show(false)
22

```

## Output:

```

1  +---+-----+-----+-----+
2  |id|name    |location|salary|
3  +---+-----+-----+-----+
4  |5 |Simon   |Australia|96000.0 |
5  |6 |Roseanne|France  |156000.0|
6  +---+-----+-----+-----+

```

**Note:** Since Spark 1.6 there is a function called `monotonically_increasing_id()`, which generates a new column with unique 64-bit monotonic index for each row, but it isn't consequential.

◀ 5 curl vs wget interview Q&As

07: Spark on Zeppelin – window functions in Scala ▶

## Disclaimer

The contents in this Java-Success are copyrighted and from EmpoweringTech pty ltd. The EmpoweringTech pty ltd has the right to correct or enhance the current content without any prior notice. These are general advice only, and one needs to take his/her own circumstances into consideration. The EmpoweringTech pty ltd will not be held liable for any damages caused or alleged to be caused either directly or indirectly by these materials and resources. Any trademarked names or labels used in this blog remain the property of their respective trademark owners. Links to external sites do not imply endorsement of the linked-to sites. [Privacy Policy](#)