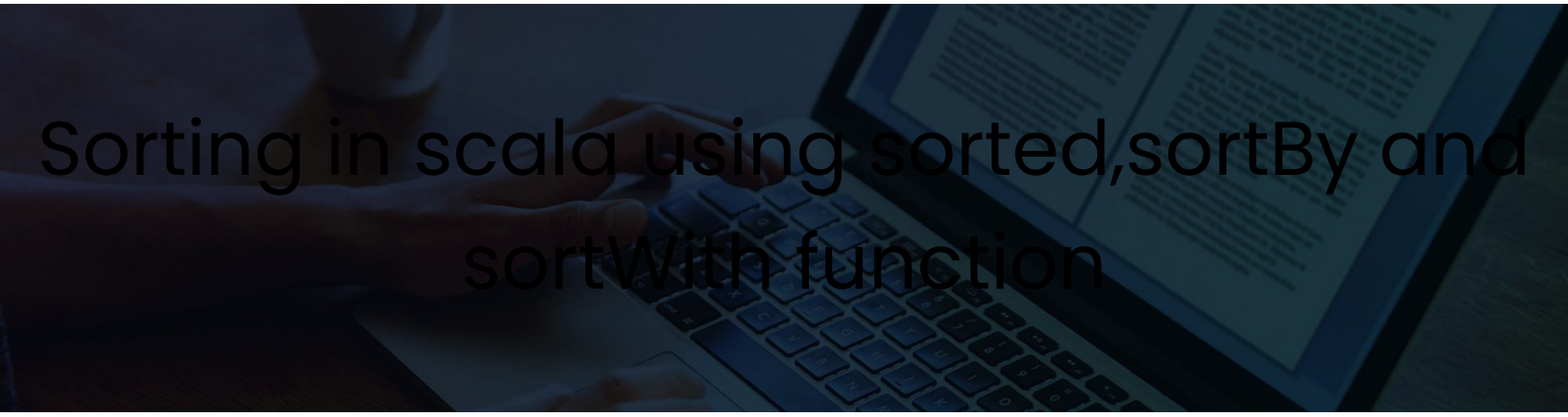
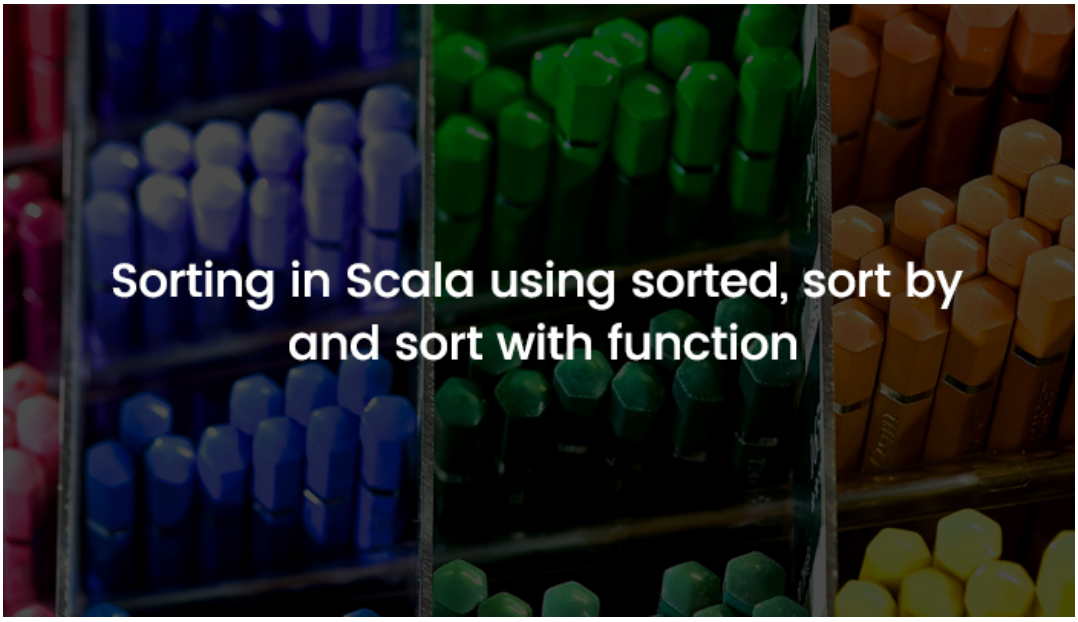
knóldus
knols . commitment . results
(https://www.knoldus.com/home)



Sorting in scala using sorted,sortBy and sortWith function



August 1, 2018 (https://blog.knoldus.com/sorting-in-scala-using-sortedsortby-and-sortwith-function/)

Randhir kumar (https://blog.knoldus.com/author/randhir1910/)

Scala (https://blog.knoldus.com/category/tech-blogs/functional-programming/scala/)

5 Comments (https://blog.knoldus.com/sorting-in-scala-using-sortedsortby-and-sortwith-function/#comments)

Reading Time: 4 minutes

Sorting is arranging the data in ascending or descending order. **Sorted** data helps us searching easily. In mobile phone contacts are sorted in the alphabetic order it helps us to find easily a mobile number of any people.
Scala uses**TimSort**, which is a hybrid of **Merge Sort and Insertion Sort**.
Here is three sorting method of Scala.

sorted

Here is signature

```
def sorted[B >: A](implicit ord: Ordering[B]): Repr
```

The **sorted** function is used to sort the **sequence** in Scala like (List, Array, Vector, Seq). The **sorted** function returns new Collection which is sorted by their **natural order.**

Now, Here is a small example Sorted with Seq

```
scala> val seq = Seq(12,3,78,90,1)
seq: Seq[Int] = List(12, 3, 78, 90, 1)

scala> seq.sorted

res4: Seq[Int] = List(1, 3, 12, 78, 90)
```

If you want to sort in descending order then, use this signature
Seq.sorted(Ordering.DataType.reverse)

```
scala> val seq = Seq(12,3,78,90,1)
seq: Seq[Int] = List(12, 3, 78, 90, 1)

scala> seq.sorted(Ordering.Int.reverse)
res6: Seq[Int] = List(90, 78, 12, 3, 1)
```

If you want to sort on the basis of an attribute of a case class using `sorted` method, then you need to extend **`Ordered trait`** and override abstract method `compare`. In `compare` method, we define on which attribute we want to sort the objects of case class.
here is an example
sort on the basis of the name attribute of the case class.

```
scala> case class Emp(id: Int,name: String,salary: Double) ext
| def compare(that: Emp) = this.name compare that.name
| }
defined class Emp

scala> val firstEmp = Emp(1, "michael", 12000.00)
firstEmp: Emp = Emp(1,michael,12000.0)

scala> val secondEmp = Emp(2, "james", 12000.00)
secondEmp: Emp = Emp(2,james,12000.0)

scala> val thirdEmp = Emp(3, "shaun", 12000.00)
thirdEmp: Emp = Emp(3,shaun,12000.0)

scala> val empList = List(firstEmp,secondEmp,thirdEmp)
empList: List[Emp] = List(Emp(1,michael,12000.0), Emp(2,james,

scala> empList.sorted
res0: List[Emp] = List(Emp(2,james,12000.0), Emp(1,michael,120
```

↑

If you do not extend **`Ordered trait`** and want to sort a case class attribute then the compiler does not know in which attribute basis it will sort so it will give a compile-time error.
here is an example.

```
scala> case class Emp(id: Int, name: String, salary: Double)
defined class Emp

scala> val firstEmp = Emp(1,"james",12000.00)
firstEmp: Emp = Emp(1,james,12000.0)

scala> val secondEmp = Emp(2,"shaun",12000.00)
secondEmp: Emp = Emp(2,shaun,12000.0)

scala> val thirdEmp = Emp(3,"michael",12000.00)
thirdEmp: Emp = Emp(3,michael,12000.0)

scala> val empList = List(firstEmp,secondEmp,thirdEmp)
empList: List[Emp] = List(Emp(1,james,12000.0), Emp(2,shaun,12

scala> empList.sorted
:13: error: No implicit Ordering defined for Emp.
empList.sorted
```

sortBy(attribute)

Here is signature

```
def sortBy[B](f: A => B)(implicit ord: Ordering[B]): Repr
```

The **`sortBy`** function is used to sort one or more **`attributes.`**
Here is a small example.
sort based on a single attribute of the case class.

↑

```
scala> case class Emp(id: Int, name: String, salary: Double)
defined class Emp

scala> val firstEmp = Emp(1,"james",12000.00)
firstEmp: Emp = Emp(1,james,12000.0)

scala> val secondEmp = Emp(2,"shaun",12000.00)
secondEmp: Emp = Emp(2,shaun,12000.0)

scala> val thirdEmp = Emp(3,"michael",12000.00)
thirdEmp: Emp = Emp(3,michael,12000.0)

scala> val forthEmp = Emp(4,"michael",11000.00)
forthEmp: Emp = Emp(4,michael,11000.0)

scala> val fifthEmp = Emp(5,"michael",15000.00)
fifthEmp: Emp = Emp(5,michael,15000.0)

scala> val empList = List(firstEmp,secondEmp,thirdEmp,forthEmp
empList: List[Emp] = List(Emp(1,james,12000.0), Emp(2,shaun,12

scala> empList.sortBy(_.name)
res0: List[Emp] = List(Emp(1,james,12000.0), Emp(3,michael,120
```

sort in `descending` by salary

```
scala> empList.sortBy(_.salary)(Ordering[Double].reverse)
res1: List[Emp] = List(Emp(5,michael,15000.0), Emp(1,james,120
```

sort is based on **`multiple attributes`**, it will sort based on the first attribute if more than one value in the first attribute is same then it will sort on the basis of the second attribute and so on.

```
scala> empList.sortBy(empList =>(empList.name,empList.salary))
res2: List[Emp] = List(Emp(1,james,12000.0), Emp(4,michael,110
```

sort list of a **`tuple`** by their second element using sortBy

↑

```
scala> val list = List(('b',30),('c',10),('a',20))
list: List[(Char, Int)] = List((b,30), (c,10), (a,20))

scala> list.sortBy(_._2)
res3: List[(Char, Int)] = List((c,10), (a,20), (b,30))
```

similarly, we can sort list of a tuple by their first element

```

scala> list.sortBy(_._1)
res4: List[(Char, Int)] = List((a,20), (b,30), (c,10))
```

sortWith(function)

Here is signature

```
 def sortWith(lt: (A, A) => Boolean): Repr
```

The **`sortWith`** function Sorts this sequence according to a **`comparison function`**. it takes a comparator function and sort according to it.you can provide your own custom comparison function.

Here is a small example

```
scala> case class Emp(id: Int, name: String, salary: Double)
defined class Emp

scala> val emp1 = Emp(1, "james", 13000.00)
emp1: Emp = Emp(1,james,13000.0)

scala> val emp2 = Emp(2, "michael", 12000.00)
emp2: Emp = Emp(2,michael,12000.0)

scala> val emp3 = Emp(3, "shaun", 15000.00)
emp3: Emp = Emp(3,shaun,15000.0)

scala> val empList = List(emp1,emp2,emp3)
empList: List[Emp] = List(Emp(1,james,13000.0), Emp(2,michael,

// sort in descending order on the basis of salary.
scala> empList.sortWith(_.salary > _.salary)
res5: List[Emp] = List(Emp(3,shaun,15000.0), Emp(1,james,13000
```

you can also sort by using own function.

```
scala> case class Emp(id: Int, name: String, salary: Double)
defined class Emp

scala> def sortBySalary(emp1 :Emp,emp2:Emp): Boolean =
| {
| emp1.salary < emp2.salary | } sortBySalary: (emp1: Emp, emp2
firstEmp: Emp = Emp(1,james,13000.0)

scala> val secondEmp = Emp(2, "michael", 12000.00)
secondEmp: Emp = Emp(2,michael,12000.0)

scala> val thirdEmp = Emp(3, "shaun", 15000.00)
thirdEmp: Emp = Emp(3,shaun,15000.0)

scala> val empList = List(firstEmp,secondEmp,thirdEmp)
empList: List[Emp] = List(Emp(1,james,13000.0), Emp(2,michael,

scala> val result = empList.sortWith((emp1,emp2) => sortBySala
result: List[Emp] = List(Emp(2,michael,12000.0), Emp(1,james,1
```

Please comment, if you have any doubt or suggestion.
thank you 🙂

(http://www.knoldus.com/connect/contactus.knol)

5 thoughts on “Sorting in scala using sorted,sortBy and sortWith function 5 min read ”

Ram Ghadiyaram (https://plus.google.com/+RamGhadiyaram) says:

August 2, 2018 at 1:05 AM (https://blog.knoldus.com/sorting-in-scala-using-sortedsortby-and-sortwith-function/#comment-38458)

Good explanation buddy. To compare, java Comparable and Comparator and Sorted, SortBy, SortWith…

SortWith is just like Comparator in this case AgeComparator or SalaryComparator etc… remaining to are Sorted and SortBy looks like Comparable(Ordering in case of scala)

Liked by 1 person

Ram Ghadiyaram (https://stackoverflow.com/users/647053/ram-ghadiyaram?tab=profile) says:

August 2, 2018 at 1:15 AM (https://blog.knoldus.com/sorting-in-scala-using-sortedsortby-and-sortwith-function/#comment-38459)

According to Java 7 API docs

Arrays#Sort() for object arrays now uses TimSort, which is a hybrid of MergeSort and InsertionSort.

On the other hand, Arrays#sort() for primitive arrays now uses Dual-PivotQuickSort.

Loading...

randhir1910 (http://gravatar.com/randhir1910) says:

August 2, 2018 at 7:39 AM (https://blog.knoldus.com/sorting-in-scala-using-sortedsortby-and-sortwith-function/#comment-38461)

Thanks for the comment, I agree with your points.

Loading...

Pingback: Sorting Data In Scala – Curated SQL (https://curatedsql.com/2018/08/03/sorting-data-in-scala/)

Shreesha S says:

March 8, 2019 at 11:13 AM (https://blog.knoldus.com/sorting-in-scala-using-sortedsortby-and-sortwith-function/#comment-39225)

Nice explaination, Thank you

Loading...

Comments are closed.

OUR OFFERINGS

High Performace System (https://www.knoldus.com/services/high-performance-systems)

Data Strategy & Analytics (https://www.knoldus.com/services/data-strategy-and-analytics)

Intelligence Driven Decisioning (https://www.knoldus.com/services/intelligence-driven-decisioning)

Platform Strategy (https://www.knoldus.com/services/platform-strategy)

Blockchain (https://www.knoldus.com/services/blockchain)

Expert Services (https://www.knoldus.com/services/expert-services)

Academy, Audit & Consulting (https://www.knoldus.com/services/academy)

KDP (https://www.knoldus.com/work/platforms-and-products/digital-platform)

KDSP (https://www.knoldus.com/work/platforms-and-products/data-science-platform)

PremonR (https://www.knoldus.com/work/platforms-and-products/premonr)

COMPANY

About Knoldus (https://www.knoldus.com/about-us)

Knolway (https://www.knoldus.com/about/process)

Case Studies (https://www.knoldus.com/work/case-studies)

Testimonials (https://www.knoldus.com/about/testimonials)

Careers (https://www.knoldus.com/company/careers)

Tech Expertise (https://www.knoldus.com/work/technologies)

Conferences and Events (https://www.knoldus.com/about/events/conferences)

News Room (https://www.knoldus.com/news)

Our Partners (https://www.knoldus.com/partners)

CSR (https://www.knoldus.com/about/corporate-social-responsibility)

LEARN

Why Knoldus? (https://www.knoldus.com/connect/why-knoldus)

Blog (https://blog.knoldus.com/)

Resources (https://www.knoldus.com/learn/resources)

KnolX (http://knolx.knoldus.com/session)

Webinars and Videos (https://www.knoldus.com/learn/webinars)

Podcast (https://www.knoldus.com/learn/podcasts)

Rust Times (https://rusttimes.com/)

Innovation Labs (https://www.knoldus.com/about/innovation-labs)

Knoldus Tech Hub (https://techhub.knoldus.com/)

Open Source Contributions (https://www.knoldus.com/work/open-source)

CONNECT

Contact Us (https://www.knoldus.com/connect/contact-us)

Engagement Model (https://www.knoldus.com/connect/engagement-models)

Follow us Here:

(https://in.linkedin.com/company/knoldus)
(https://www.youtube.com/channel/UCP4g5qGeUSY7OokXfim1QCQ)
(https://twitter.com/Knolspeak?ref_src=twsrc%5Egoogle%7Ctwcamp%5Eserp%7Ctwgr%5Eauthor)
(https://www.facebook.com/KnoldusSoftware/)
(https://www.slideshare.net/knoldus)
(https://www.instagram.com/knoldus_inc/?hl=en)

Subscribe to our newsletter

SUBSCRIBE

(https://share.hsforms.com/1oRAE_GdlQsi3niRYzwv65g2u6gi)

Partners

(https://www.lightbend.com/partners/system-integrators)

(https://databricks.com/company/partners)

(https://www.confluent.io/partners/)

(https://www.docker.com/)

(https://www.hashicorp.com/ecosystem/find-a-partner/)

(https://www.ibm.com/partnerworld/public)