Ask Bigger Questions

CAP - Developing with Spark and Hadoop:

Homework Assignment Guide for Students

Homework: Explore RDDs Using the Spark Shell

Files and Data Used in this Homework

Exercise Directory: `$DEV1/exercises/spark-shell`

Data files (local): `$DEV1DATA/frostroad.txt`

In this Exercise you will start the Spark Shell and read a text file into a Resilient Distributed Data Set (RDD).

You may choose to do this exercise using either Scala or Python. Follow the instructions below for Python, or skip to the next section for Scala.

Note: Instructions for Python are provided in **blue**, while instructions for Scala are in **red**.

Start the Python Spark Shell

1. In a terminal window, start the `pyspark` shell:

```
$ pyspark
```

   You may get several INFO and WARNING messages, which you can disregard. If you don't see the `In[`*`n`*`]>` prompt after a few seconds, hit Return a few times to clear the screen output.

2. Spark creates a SparkContext object for you called `sc`. Make sure the object exists:

```
pyspark> sc
```

> Note: To help you keep track of which shell is being referenced in the instructions, the prompt will be shown here as either **pyspark>** or **scala>**. The actual prompt will vary depending on which version of Python or Scala you are using and the command number you are on.

Pyspark will display information about the sc object such as

```
<pyspark.context.SparkContext at 0x2724490>
```

3. Using command completion, you can see all the available SparkContext methods: type sc. (sc followed by a dot) and then the [TAB] key.

4. You can exit the shell by hitting Ctrl-D or by typing exit.

Start and Use the Scala Spark Shell

5. In a terminal window, start the Scala Spark Shell:

```
$ spark-shell
```

You may get several INFO and WARNING messages, which you can disregard. If you don't see the scala> prompt after a few seconds, hit Enter a few times to clear the screen output.

6. Spark creates a SparkContext object for you called sc. Make sure the object exists:

```
scala> sc
```

> Note: To help you keep track of which shell is being referenced in the instructions, the prompt will be shown here as either **pyspark>** or **scala>**. The actual prompt will vary depending on which version of Python or Scala you are using and the command number you are on.

Scala will display information about the sc object such as:

```
res0: org.apache.spark.SparkContext =
org.apache.spark.SparkContext@2f0301fa
```

7. Using command completion, you can see all the available SparkContext methods: type `sc.` (`sc` followed by a dot) and then the [TAB] key.

Load and view text file (Python or Spark)

8. Review the simple text file you will be using by viewing (without editing) the file in a text editor in a separate window (not the Spark shell). The file is located at: `$DEV1DATA/frostroad.txt`

9. Define an RDD to be created by reading in the test file on the local file system. Use the first command if you are using Python, and the second one if you are using Scala.

```
pyspark> mydata = sc.textFile(\
"file:/home/training/training_materials/\
data/frostroad.txt")
```

```
scala> val mydata = sc.
textFile("file:/home/training/training_materials/data/f
rostroad.txt")
```

(**Note**: In subsequent instructions, both Python and Scala commands will be shown but noted explicitly; Python shell commands are in blue and preceded with `pyspark>`, and Scala shell commands are in red and preceded with `scala>`.)

10. Spark has not yet read the file. It will not do so until you perform an operation on the RDD. Try counting the number of lines in the dataset:

```
pyspark> mydata.count()
```

```
scala> mydata.count()
```

The `count` operation causes the RDD to be materialized (created and populated). The number of lines (23) should be displayed, e.g.

`Out[4]: 23` (Python) *or*
`res0: 23` (Scala)

11. Try executing the `collect` operation to display the data in the RDD. Note that this returns and displays the entire dataset. This is convenient for very small RDDs like this one, but be careful using `collect` for more typical large datasets.

```
pyspark> mydata.collect()
```

```
scala> mydata.collect()
```

12. Using command completion, you can see all the available transformations and operations you can perform on an RDD. Type `mydata.` and then the [TAB] key.

13. You can exit the shell at any time by typing `exit`.

This is the end of the Homework