

Referência de databricks russas (dbutils)

October 10, 2024

Este artigo é uma referência para Databricks russas (dbutils). dbutils russas estão disponíveis em Python, R e Scala Notebook. Você pode usar as russálias para:

- Trabalhe com arquivos e armazenamento de objetos de forma eficiente.
- Trabalhe com segredos.

Observação

dbutils O senhor só é compatível com ambientes compute que usam DBFS.

Como: listar utilidades, listar comandos, exibir ajuda de comando

utilidades: credenciais, dados, fs, Job, biblioteca, Notebook, segredos, widgets, biblioteca API de utilidades

Neste artigo:

- Listar utilidades disponíveis
- Listar os comandos disponíveis para uma utilidade
- Exibir ajuda para um comando
- Utilidade de credenciais (dbutils.credentials)

- Utilidade de dados (`dbutils.data`)
- Utilidade do sistema de arquivos (`dbutils.fs`)
- Utilitário de jobs (`dbutils.jobs`)
- Utilidades da biblioteca (`dbutils.library`)
- Utilitário de notebook (`dbutils.notebook`)
- Utilitário de segredos (`dbutils.secrets`)
- Utilitário de widgets (`dbutils.widgets`)
- Biblioteca de APIs de utilitários do Databricks
- Limitações

Listar utilidades disponíveis

Para listar as utilidades disponíveis junto com uma breve descrição de cada utilidade, execute `dbutils.help()` para Python ou Scala.

Este exemplo lista os comandos disponíveis para o Databricks Utilities.

Python Scala

```
dbutils.help()
```

 Cópia de

This module provides various utilities for users to interact with the rest of Databricks.

credentials: DatabricksCredentialUtils -> Utilities for interacting with credentials within notebooks
data: DataUtils -> Utilities for understanding and interacting with datasets (EXPERIMENTAL)
fs: DbfUtils -> Manipulates the Databricks filesystem (DBFS) from the console
jobs: JobsUtils -> Utilities for leveraging jobs features
library: LibraryUtils -> Utilities for session isolated libraries
meta: MetaUtils -> Methods to hook into the compiler (EXPERIMENTAL)
notebook: NotebookUtils -> Utilities for the control flow of a notebook (EXPERIMENTAL)
preview: Preview -> Utilities under preview category
secrets: SecretUtils -> Provides utilities for leveraging secrets within notebooks
widgets: WidgetsUtils -> Methods to create and get bound value of input widgets inside notebooks

Listar os comandos disponíveis para uma utilidade

Para listar os comandos disponíveis para uma utilidade junto com uma breve descrição de cada comando, execute `.help()` após o nome programático da utilidade.

Este exemplo lista os comandos disponíveis para a utilidade Databricks File System (DBFS).

Python R Scala

```
dbutils.fs.help()
```

Cópia de

```
dbutils.fs provides utilities for working with FileSystems. Most methods in this package can take either a DBFS path (e.g., "/foo" or "dbfs:/foo"), or another FileSystem URI. For more info about a method, use dbutils.fs.help("methodName"). In notebooks, you can also use the %fs shorthand to access DBFS. The %fs shorthand maps straightforwardly onto dbutils calls. For example, "%fs head --maxBytes=10000 /file/path" translates into "dbutils.fs.head("/file/path", maxBytes = 10000)".
```

fsutils

```
cp(from: String, to: String, recurse: boolean = false): boolean -> Copies a file or directory, possibly across FileSystems  
head(file: String, maxBytes: int = 65536): String -> Returns up to the first 'maxBytes' bytes of the given file as a String encoded in UTF-8  
ls(dir: String): Seq -> Lists the contents of a directory  
mkdirs(dir: String): boolean -> Creates the given directory if it does not exist, also creating any necessary parent directories  
mv(from: String, to: String, recurse: boolean = false): boolean -> Moves a file or directory, possibly across FileSystems  
put(file: String, contents: String, overwrite: boolean = false): boolean -> Writes the given String out to a file, encoded in UTF-8  
rm(dir: String, recurse: boolean = false): boolean -> Removes a file or directory
```

mount

```
mount(source: String, mountPoint: String, encryptionType: String = "", owner: String = null, extraConfigs: Map = Map.empty[String, String]):  
boolean -> Mounts the given source directory into DBFS at the given mount point  
mounts: Seq -> Displays information about what is mounted within DBFS  
refreshMounts: boolean -> Forces all machines in this cluster to refresh their mount cache, ensuring they receive the most recent information  
unmount(mountPoint: String): boolean -> Deletes a DBFS mount point  
updateMount(source: String, mountPoint: String = "", owner: String = null, extraConfigs: Map = Map.empty[String, String],  
String): boolean -> Similar to mount(), but updates an existing mount point instead of creating a new one
```

Exibir ajuda para um comando

Para exibir ajuda para um comando, execute `.help("<command-name>")` após o nome do comando.

Este exemplo exibe ajuda para o comando de cópia DBFS.

Python R Scala

```
dbutils.fs.help("cp")
```

Cópia de

```
/*
 * Copies a file or directory, possibly across FileSystems.
 *
 * Example: cp("/mnt/my-folder/a", "dbfs:/a/b")
 *
 * @param from FileSystem URI of the source file or directory
 * @param to FileSystem URI of the destination file or directory
 * @param recurse if true, all files and directories will be recursively copied
 * @return true if all files were successfully copied
 */
cp(from: java.lang.String, to: java.lang.String, recurse: boolean = false): boolean
```

Cópia de

Utilidade de credenciais (dbutils.credentials)

comando: `assumeRole, showCurrentRole, showRoles`

A utilidade de credenciais permite interagir com credenciais em notebooks. Essa utilidade só pode ser usada em clusters com [passagem de credenciais](#) habilitada. Para listar os comandos disponíveis, execute `dbutils.credentials.help()`.

Cópia de

```
assumeRole(role: String): boolean -> Sets the role ARN to assume when looking for credentials to authenticate with S3  
showCurrentRole: List -> Shows the currently set role  
showRoles: List -> Shows the set of possible assumed roles
```

comando AssumeRole (dbutils.credentials.assumeRole)

Define o Amazon Resource Name (ARN) para a função do AWS Identity and Access Management (IAM) a ser assumida ao procurar credenciais para autenticação no Amazon S3. Depois de executar esse comando, você pode executar comandos de acesso do S3, como sc.textFile("s3a://my-bucket/my-file.csv"), para acessar um objeto.

Para exibir ajuda para este comando, execute dbutils.credentials.help("assumeRole").

[Python](#) [R](#) [Scala](#)

```
dbutils.credentials.assumeRole("arn:aws:iam::123456789012:roles/my-role")
```

```
# Out[1]: True
```

Comando showCurrentRole (dbutils.credentials.showCurrentRole)

Lista a função atualmente definida do AWS Identity and Access Management (IAM).

Para exibir ajuda para este comando, execute dbutils.credentials.help("showCurrentRole").

[Python](#) [R](#) [Scala](#)

```
dbutils.credentials.showCurrentRole()
```

```
# Out[1]: [ 'arn:aws:iam::123456789012:role/my-role-a' ]
```

Comando showRoles (dbutils.credentials.showRoles)

lista o conjunto de possíveis funções assumidas do AWS Identity and Access Management (IAM).

Para exibir ajuda para este comando, execute `dbutils.credentials.help("showRoles")`.

[Python](#)

[R](#)

[Scala](#)

[Cópia de](#)

```
dbutils.credentials.showRoles()
```

```
# Out[1]: [ 'arn:aws:iam::123456789012:role/my-role-a' , 'arn:aws:iam::123456789012:role/my-role-b' ]
```

Utilidade de dados (dbutils.data)

Visualização

Esse recurso está na Visualização pública.

Observação

Disponível no Databricks Runtime 9.0e acima.

comando: resumir

A utilidade de dados permite a você entender e interpretar datasets. Para listar os comandos disponíveis, execute `dbutils.data.help()`.

[Cópia de](#)

`dbutils.data` provides utilities for understanding and interpreting datasets. This module is currently in preview and may be unstable. For more info about a method, use `dbutils.data.help("methodName")`.

`summarize(df: Object, precise: boolean): void -> Summarize a Spark DataFrame and visualize the statistics to get quick insights`

Comando summarize (dbutils.data.summarize)

Calcula e exibe estatísticas resumidas de um DataFrame do Apache Spark ou DataFrame do pandas. Este comando está disponível para Python, Scala e R.

Atenção

Este comando analisa o conteúdo completo do DataFrame. Executar este comando para DataFrames muito grandes pode ser muito caro.

Para exibir ajuda para este comando, execute `dbutils.data.help("summarize")`.

Em Databricks Runtime 10.4 LTS e acima, o senhor pode usar o parâmetro adicional `precise` para ajustar a precisão das estatísticas de computação.

Observação

Esse recurso está na [Visualização pública](#).

- Quando `precise` é definido como `false` (o padrão), algumas estatísticas retornadas incluem aproximações para reduzir o tempo de execução.
 - O número de valores distintos para colunas categóricas pode ter ~5% de erro relativo para colunas de alta cardinalidade.
 - As contagens de valores frequentes podem ter um erro de até 0,01% quando o número de valores distintos é maior que 10.000.
 - Os histogramas e estimativas de percentis podem apresentar um erro de até 0,01% em relação ao número total de linhas.
- Quando `precise` está configurado como `true` (verdadeiro), as estatísticas são calculadas com maior precisão. Todas as estatísticas, exceto os histogramas e percentis para colunas numéricas, agora são exatas.
- Os histogramas e as estimativas de percentis podem ter um erro de até 0,0001% em relação ao número total de linhas.

A dica de ferramenta na parte superior da saída de resumo de dados indica o modo de execução atual.

Este exemplo exibe estatísticas resumidas para um DataFrame do Apache Spark com aproximações habilitadas por default. Para ver os resultados, execute esse comando em um notebook. Este exemplo é baseado em [amostras de datasets](#).

```
df = spark.read.format('csv').load(  
    '/databricks-datasets/Rdatasets/data-001/csv/ggplot2/diamonds.csv',  
    header=True,  
    inferSchema=True  
)  
dbutils.data.summarize(df)
```

Observe que a visualização usa a **notação do SI** para renderizar de forma concisa valores numéricos menores que 0,01 ou maiores que 10.000. Por exemplo, o valor numérico 1.25e-15 será renderizado como 1.25f. Uma exceção: a visualização usa "B" para 1.0e9 (**giga**) em vez de "G".

Utilidade do sistema de arquivos (dbutils.fs)

Aviso!

A implementação do Python de todos os métodos dbutils.fs usa `snake_case` em vez de `camelCase` para formatação de palavras-chave.

Por exemplo: enquanto o `dbutils.fs.help()` exibe a opção `extraConfigs` para `dbutils.fs.mount()`, no Python você utilizaria a palavra-chave `extra_configs`.

comando: `cp, head, ls, mkdirs, mount, mounts, mv, put, refreshMounts, rm, unmount, updateMount`

O sistema de arquivos utilidades permite que o senhor acesse o **What is DBFS?**, facilitando o uso do Databricks como um sistema de arquivos.

No Notebook, o senhor também pode usar o comando mágico `%fs` para acessar DBFS. Por exemplo,
`%fs ls /Volumes/main/default/my-volume/` é o mesmo que `dbutils.fs.ls("/Volumes/main/default/my-volume/")`. Ver [comando mágico](#).

Para listar os comandos disponíveis, execute `dbutils.fs.help()`.

`dbutils.fs` provides utilities for working with `FileSystems`. Most methods in this package can take either a DBFS path (e.g., `"/foo"` or `"dbfs:/foo"`), or another `FileSystem` URI. For more info about a method, use `dbutils.fs.help("methodName")`. In notebooks, you can also use the `%fs` shorthand to access DBFS. The `%fs` shorthand maps straightforwardly onto `dbutils` calls. For example, `%fs head --maxBytes=100000 /file/path` translates into `"dbutils.fs.head("/file/path", maxBytes = 100000)"`.

```
fsutils

cp(from: String, to: String, recurse: boolean = false): boolean -> Copies a file or directory, possibly across FileSystems
head(file: String, maxBytes: int = 65536): String -> Returns up to the first 'maxBytes' bytes of the given file as a String encoded in UTF-8
ls(dir: String): Seq -> Lists the contents of a directory
mkdirs(dir: String): boolean -> Creates the given directory if it does not exist, also creating any necessary parent directories
mv(from: String, to: String, recurse: boolean = false): boolean -> Moves a file or directory, possibly across FileSystems
put(file: String, contents: String, overwrite: boolean = false): boolean -> Writes the given String out to a file, encoded in UTF-8
rm(dir: String, recurse: boolean = false): boolean -> Removes a file or directory

mount
```

```
mount(source: String, mountPoint: String, encryptionType: String = "", owner: String = null, extraConfigs: Map = Map.empty[String, String]): boolean -> Mounts the given source directory into DBFS at the given mount point
mounts: Seq -> Displays information about what is mounted within DBFS
refreshMounts: boolean -> Forces all machines in this cluster to refresh their mount cache, ensuring they receive the most recent information
unmount(mountPoint: String): boolean -> Deletes a DBFS mount point
updateMount(source: String, mountPoint: String, encryptionType: String = "", owner: String = null, extraConfigs: Map = Map.empty[String, String]): boolean -> Similar to mount(), but updates an existing mount point instead of creating a new one
```

Comando cp (dbutils.fs.cp)

Copia um arquivo ou diretório, possivelmente entre sistemas de arquivos.

Para exibir ajuda para este comando, execute `dbutils.fs.help("cp")`.

Este exemplo copia o arquivo chamado `data.csv` de `/Volumes/main/default/my-volume/` para `new-data.csv` no mesmo volume.

```
dbutils.fs.cp("/Volumes/main/default/my-volume/data.csv", "/Volumes/main/default/my-volume/new-data.csv")
```

```
# Out[4]: True
```

Comando head (dbutils.fs.head)

Retorna até o número máximo de bytes especificado do arquivo fornecido. Os bytes são retornados como uma string codificada UTF-8.

Para exibir ajuda para este comando, execute dbutils.fs.help("head").

Este exemplo exibe os primeiros 25 bytes do arquivo data.csv localizado em /Volumes/main/default/my-volume/.

[Python](#) [R](#) [Scala](#)

```
dbutils.fs.head("/Volumes/main/default/my-volume/data.csv", 25)

# [Truncated to first 25 bytes]
# Out[12]: 'Year,First Name,County,Se'
```

comando ls (dbutils.fs.ls)

Lista o conteúdo de um diretório.

Para exibir ajuda para este comando, execute dbutils.fs.help("ls").

Este exemplo exibe informações sobre o conteúdo de /Volumes/main/default/my-volume/. O campo modificationTime está disponível em Databricks Runtime 10.4 LTS e acima. No R, modificationTime é retornado como uma cadeia de caracteres.

[Python](#) [R](#) [Scala](#)

```
dbutils.fs.ls("/Volumes/main/default/my-volume/")

# Out[13]: [FileInfo(path='dbfs:/Volumes/main/default/my-volume/data.csv', name='data.csv', size=2258987, modificationTime=1711357839000)]
```

comando mkdirs (dbutils.fs.mkdirs)

Cria o diretório fornecido se ele não existir. Cria também quaisquer diretórios pai necessários.

Para exibir ajuda para este comando, execute dbutils.fs.help("mkdirs").

Este exemplo cria o diretório my-data dentro de /Volumes/main/default/my-volume/.

[Python](#) [R](#) [Scala](#)

```
dbutils.fs.mkdirs("/Volumes/main/default/my-volume/my-data")
```

```
# Out[15]: True
```

comando mount (dbutils.fs.mount)

Monta o diretório de origem especificado no DBFS no ponto de montagem especificado.

Para exibir ajuda para este comando, execute dbutils.fs.help("mount").

[Python](#) [Scala](#)

```
aws_bucket_name = "my-bucket"
mount_name = "S3-my-bucket"
```

```
dbutils.fs.mount("s3a://%s" % aws_bucket_name, "/mnt/%s" % mount_name)
```

Para obter exemplos de código adicionais, consulte [Conectar-se ao Amazon S3](#).

comando mounts (dbutils.fs.mounts)

Exibe informações sobre o que está atualmente montado no DBFS.

Para exibir ajuda para este comando, execute dbutils.fs.help("mounts").

Aviso!

Chame `dbutils.fs.refreshMounts()` em todos os outros clusters em execução para propagar a nova montagem. Consulte o comando `refreshMounts (dbutils.fs.refreshMounts)`.

Python Scala

```
dbutils.fs.mounts()

# Out[11]: [MountInfo(mountPoint='/mnt/databricks-results', source='databricks-results', encryptionType='sse-s3')]
```

Para obter exemplos de código adicionais, consulte [Conectar-se ao Amazon S3](#).

comando mv (dbutils.fs.mv)

Move um arquivo ou diretório, possivelmente entre sistemas de arquivos. Uma movimentação é uma cópia seguida de uma exclusão, mesmo para movimentações dentro de sistemas de arquivos.

Para exibir ajuda para este comando, execute `dbutils.fs.help("mv")`.

Este exemplo move o arquivo `rows.csv` de `/Volumes/main/default/my-volume/` para `/Volumes/main/default/my-volume/my-data/`.

Python R Scala

```
dbutils.fs.mv("/Volumes/main/default/my-volume/rows.csv", "/Volumes/main/default/my-volume/my-data/")

# Out[2]: True
```

comando put (dbutils.fs.put)

Escreve a string especificada em um arquivo. A string é codificada em UTF-8.

Para exibir ajuda para este comando, execute `dbutils.fs.help("put")`.

Este exemplo grava a string Hello, Databricks! em um arquivo denominado `hello.txt` no `/Volumes/main/default/my-volume/`. Se o arquivo existir, ele será substituído.

Python R Scala

```
dbutils.fs.put("/Volumes/main/default/my-volume/hello.txt", "Hello, Databricks!", True)
```

```
# Wrote 2258987 bytes.  
# Out[6]: True
```

comando refreshMounts (dbutils.fs.refreshMounts)

Força todas as máquinas no cluster a atualizar o cache de montagem, garantindo que elas recebam as informações mais recentes.

Para exibir ajuda para este comando, execute dbutils.fs.help("refreshMounts").

[Python](#) [Scala](#)

```
dbutils.fs.refreshMounts()
```

Para obter exemplos de código adicionais, consulte [Conectar-se ao Amazon S3](#).

comando rm (dbutils.fs.rm)

Remove um arquivo ou diretório e, opcionalmente, todo o seu conteúdo. Se um arquivo for especificado, o parâmetro recurse será ignorado. Se um diretório for especificado, ocorrerá um erro se a recursão estiver desabilitada e o diretório não estiver vazio.

Para exibir ajuda para este comando, execute dbutils.fs.help("rm").

Este exemplo remove o diretório /Volumes/main/default/my-volume/my-data/ incluindo o conteúdo do diretório.

[Python](#) [R](#) [Scala](#)

```
dbutils.fs.rm("/Volumes/main/default/my-volume/my-data/", True)
```

```
# Out[8]: True
```

comando unmount (dbutils.fs.unmount)

Exclui um ponto de montagem de DBFS.

Aviso!

Para evitar erros, nunca modifique um ponto de montagem enquanto outro Job estiver lendo ou gravando nele. Depois de modificar uma montagem, sempre execute `dbutils.fs.refreshMounts()` em todos os outros clusters em execução para propagar quaisquer atualizações de montagem. Consulte o comando `refreshMounts` (`dbutils.fs.refreshMounts`).

Para exibir ajuda para este comando, execute `dbutils.fs.help("umount")`.

Python

```
dbutils.fs.unmount("/mnt/<mount-name>")
```

Para obter exemplos de código adicionais, consulte [Conectar-se ao Amazon S3](#).

comando updateMount (`dbutils.fs.updateMount`)

Semelhante ao comando `dbutils.fs.mount`, mas atualiza um ponto de montagem existente em vez de criar um novo. Retorna um erro se o ponto de montagem não estiver presente.

Para exibir ajuda para este comando, execute `dbutils.fs.help("updateMount")`.

Aviso!

Para evitar erros, nunca modifique um ponto de montagem enquanto outro Job estiver lendo ou gravando nele. Depois de modificar uma montagem, sempre execute `dbutils.fs.refreshMounts()` em todos os outros clusters em execução para propagar quaisquer atualizações de montagem. Consulte o comando `refreshMounts` (`dbutils.fs.refreshMounts`).

Esse comando está disponível em Databricks Runtime 10.4 LTS e acima.

Python Scala

```
aws_bucket_name = "my-bucket"
mount_name = "S3-my-bucket"
```

```
dbutils.fs.updateMount("s3a://%s" % aws_bucket_name, "/mnt/%s" % mount_name)
```

Python CÓPIA DE

CÓPIA DE

Utilitário de jobs (dbutils.jobs)

Subutilidades: taskValues

Observação

Este utilitário está disponível somente para o Python.

O utilitário de jobs permite que você aproveite os recursos de jobs. Para exibir a ajuda deste utilitário, execute `dbutils.jobs.help()`.

 Cópia de

Provides utilities for leveraging jobs features.

`taskValues: TaskValuesUtils -> Provides utilities for leveraging job task values`

Subutilitário taskValues (dbutils.jobs.taskValues)

comando: [obter, definir](#)

Observação

Este subutilitário está disponível somente para o Python.

Fornece comandos para aproveitar os valores das tarefas de jobs.

Use este subutilitário para definir e obter valores arbitrários durante a execução de uma tarefa. Esses valores são chamados de *valores de tarefas*. Você pode acessar valores de tarefas em tarefas downstream na mesma execução de job. Por exemplo, você pode comunicar identificadores ou métricas, como informações sobre a avaliação de um modelo do machine learning, entre diferentes tarefas em uma execução de job. Cada tarefa pode definir vários valores de tarefas, obtê-los ou ambos. Cada valor de tarefa possui uma chave exclusiva dentro da mesma tarefa. Essa chave exclusiva é conhecida como chave do valor da tarefa. Um valor de tarefa é acessado com o nome da tarefa e a chave do valor da tarefa.

Para exibir ajuda para este subutilitário, execute `dbutils.jobs.taskValues.help()`.

comando get (dbutils.jobs.taskValues.get)

Observação

Este comando está disponível apenas para o Python.

No Databricks Runtime 10.4 e anteriores, se get não puder encontrar a tarefa, será gerado um `Py4JJavaError` em vez de um `ValueError`.

Obtém o conteúdo do valor da tarefa especificada para a tarefa especificada na execução do job atual.

Para exibir ajuda para este comando, execute `dbutils.jobs.taskValues.help("get")`.

Por exemplo:

Python

```
dbutils.jobs.taskValues.get(taskKey = "my-task", \
                           key = "my-key", \
                           default = 7, \
                           debugValue = 42)
```

Cópia de

No exemplo anterior:

- `taskKey` é o nome da tarefa que define o valor da tarefa. Se o comando não conseguir encontrar esta tarefa, um `ValueError` será gerado.
- `key` é o nome da `key` do valor da tarefa que você definiu com o comando `set (dbutils.Job.taskValues.set)`. Se o comando não puder localizar `key` desse valor de tarefa, um `ValueError` será levantado (a menos que `default` seja especificado).
- `default` é um valor opcional retornado se `key` não puder ser encontrado. `default` não pode ser `None`.
- `debugValue` é um valor opcional que é retornado se você tentar obter o valor da tarefa de dentro de um notebook que está sendo executado fora de um job. Isso pode ser útil durante a depuração quando você quiser executar o notebook manualmente e retornar algum valor em vez de gerar um `TypeError` por padrão. `debugValue` não pode ser `None`.

Se você tentar obter um valor de tarefa em um notebook que esteja sendo executado fora de um job, esse comando exibirá um `TypeError` por padrão. No entanto, se o argumento `debugValue` for especificado no comando, o valor de `debugValue` será retornado em vez de gerar um `TypeError`.

comando `set` (`dbutils.jobs.taskValues.set`)

Observação

Este comando está disponível apenas para o Python.

Define ou atualiza um valor de tarefa. Você pode definir até 250 valores de tarefas para uma execução de job.

Para exibir ajuda para este comando, execute `dbutils.jobs.taskValues.help("set")`.

Alguns exemplos incluem:

Python

```
dbutils.jobs.taskValues.set(key = "my-key", \
                           value = 5)
```

```
dbutils.jobs.taskValues.set(key = "my-other-key", \
                           value = "my other value")
```

Nos exemplos anteriores:

- `key` é a chave do valor da tarefa. Essa chave deve ser exclusiva da tarefa. Ou seja, se duas tarefas diferentes definirem um valor de tarefa com a chave `K`, esses são dois valores de tarefa diferentes que têm a mesma chave `K`.
- `value` é o valor para a chave desse valor de tarefa. Este comando deve ser capaz de representar o valor internamente no formato JSON. O tamanho da representação JSON do valor não pode exceder 48 KiB.

Se você tentar definir um valor de tarefa de dentro de um notebook que está sendo executado fora de um job, esse comando não fará nada.

Utilidades da biblioteca (dbutils.library)

A maioria dos métodos do submodule dbutils.library está obsoleta. Consulte [biblioteca utilidades \(dbutils.library\)](#) (legado).

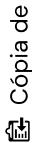
Talvez seja necessário reiniciar programmaticamente o processo Python no Databricks para garantir que a biblioteca instalada ou atualizada localmente funcione corretamente no kernel Python para sua SparkSession atual. Para fazer isso, execute o comando `dbutils.library.restartPython`. Consulte [Reiniciar o processo Python no Databricks](#).

Utilitário de notebook (dbutils.notebook)

comando: [exit](#), [execução](#)

O utilitário de notebook permite que você encadeie notebooks e atue com base em seus resultados. Consulte [Executar um notebook Databricks a partir de outro notebook](#).

Para listar os comandos disponíveis, execute `dbutils.notebook.help()`.



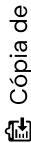
`exit(value: String): void` -> This method lets you exit a notebook with a value
`run(path: String, timeoutSeconds: int, arguments: Map): String` -> This method runs a notebook and returns its exit value.

comando exit (dbutils.notebook.exit)

Sai de um notebook com um valor.

Para exibir ajuda para este comando, execute `dbutils.notebook.help("exit")`.

Este exemplo sai do notebook com o valor Exiting from My Other Notebook.



```
dbutils.notebook.exit("Exiting from My Other Notebook")
```

```
# Notebook exited: Exiting from My Other Notebook
```

Observação

Se a execução tiver uma consulta com **transmissão estruturada** em execução em segundo plano, chamar `dbutils.notebook.exit()` não encerrará a execução. A execução continuará a ocorrer enquanto a consulta estiver sendo executada em segundo plano. Você pode interromper a execução da consulta em segundo plano clicando em **Cancelar** na célula da consulta ou executando `query.stop()`. Quando a consulta parar, você poderá encerrar a execução com `dbutils.notebook.exit()`.

comando run (dbutils.notebook.run)

Executa um notebook e retorna seu valor de saída. O notebook será executado no cluster atual por padrão.

Observação

O comprimento máximo do valor da string retornado do comando `run` é de 5 MB. Consulte [Obter a saída para uma única execução \(GET /jobs/runs/get-output\)](#).

Para exibir ajuda para este comando, execute `dbutils.notebook.help("run")`.

Este exemplo executa um notebook denominado `My Other Notebook` no mesmo local que o notebook que está chamando. O notebook chamado termina com a linha de código `dbutils.notebook.exit("Exiting from My Other Notebook")`. Se o notebook chamado não terminar a execução em 60 segundos, uma exceção será lançada.

Python Scala

```
dbutils.notebook.run("My Other Notebook", 60)
```

```
# out[14]: 'Exiting from My Other Notebook'
```

Utilitário de segredos (dbutils.secrets)

`comando: get, getBytes, list, listScopes`

Cópia de

As utilidades secretas permitem que você armazene e acesse informações de credenciais confidenciais sem torná-las visíveis no Notebook. Consulte [Gerenciamento de segredos](#) e [Use os segredos em um Notebook](#). Para listar os comandos disponíveis, execute `dbutils.secrets.help()`.

 Cópia de

```
get(scope: String, key: String): String -> Gets the string representation of a secret value with scope and key  
getBytes(scope: String, key: String): byte[] -> Gets the bytes representation of a secret value with scope and key  
list(scope: String): Seq -> Lists secret metadata for secrets within a scope  
listScopes: Seq -> Lists secret scopes
```

comando get (dbutils.secrets.get)

Obtém a representação de string de um valor secreto para o escopo e chave de segredos especificados.

Aviso!

Administradores, criadores de segredos e usuários com [permissão concedida](#) podem ler os segredos do Databricks. Embora o Databricks faça um esforço para redigir valores secretos que podem ser exibidos em notebooks, não é possível impedir que esses usuários leiam segredos. Para obter mais informações, consulte [Redação secreta](#).

Para exibir ajuda para este comando, execute `dbutils.secrets.help("get")`.

Este exemplo obtém a representação da string do valor secreto para o escopo chamado `my-scope` e a chave chamada `my-key`.

Python R Scala

```
dbutils.secrets.get(scope="my-scope", key="my-key")  
  
# Out[14]: '[REDACTED]'
```

 Cópia de

comando getBytes (dbutils.secrets.getBytes)

Obtém a representação de bytes de um valor secreto para o escopo e chave especificados.

Para exibir ajuda para este comando, execute `dbutils.secrets.help("getBytes")`.

Este exemplo obtém a representação em bytes do valor secreto (neste exemplo, `a1!b2@c3#`) para o escopo denominado `my-scope` e a chave denominada `my-key`.

[Python](#) [R](#) [Scala](#) [Cópia de](#)

```
dbutils.secrets.getBytes(scope="my-scope", key="my-key")  
  
# Out[1]: b'a1!b2@c3#'
```

comando `list (dbutils.secrets.list)`

Lista os metadados para segredos dentro do escopo especificado.

Para exibir ajuda para este comando, execute `dbutils.secrets.help("list")`.

Este exemplo lista os metadados para segredos dentro do escopo denominado `my-scope`.

[Python](#) [R](#) [Scala](#) [Cópia de](#)

```
dbutils.secrets.list("my-scope")  
  
# Out[10]: [SecretMetadata(key='my-key')]
```

comando `listScopes (dbutils.secrets.listScopes)`

Lista os escopos disponíveis.

Para exibir ajuda para este comando, execute `dbutils.secrets.help("listScopes")`.

Este exemplo lista os escopos disponíveis.

[Python](#) [R](#) [Scala](#) [Cópia de](#)

```
dbutils.secrets.listScopes()
```

```
# Out[14]: [SecretScope(name='my-scope')]
```

Utilitário de widgets (dbutils.widgets)

comando: `checkbox`, `dropdown`, `get`, `getArgument`, `multiselect`, `remove`, `removeAll`, `text`

O utilitário de widgets permite a você parameterizar notebooks. Consulte [Widgets Databricks](#).

Para listar os comandos disponíveis, execute `dbutils.widgets.help()`.

 Cópia de

```
checkbox(name: String, defaultValue: String, choices: Seq, label: String): void -> Creates a combobox input widget with a given name, default value and choices
dropdown(name: String, defaultValue: String, choices: Seq, label: String): void -> Creates a dropdown input widget a with given name, default value and choices
get(name: String): String -> Retrieves current value of an input widget
getAll: map -> Retrieves a map of all widget names and their values
getArgument(name: String, optional: String): String -> (DEPRECATED) Equivalent to get
multiselect(name: String, defaultValue: String, choices: Seq, label: String): void -> Creates a multiselect input widget with a given name, default value and choices
remove(name: String): void -> Removes an input widget from the notebook
removeAll: void -> Removes all widgets in the notebook
text(name: String, defaultValue: String, label: String): void -> Creates a text input widget with a given name and default value
```

comando `checkbox` (dbutils.widgets.checkbox)

Cria e exibe um widget de combobox com o nome programático especificado, o valor padrão, as opções e o rótulo opcional.

Para exibir ajuda para este comando, execute `dbutils.widgets.help("checkbox")`.

Este exemplo cria e exibe um widget combobox com o nome programático `fruits_combobox`. Ele oferece as opções `apple`, `banana`, `coconut`, `dragon fruit` e é definido com o valor inicial de `banana`. Este widget combobox tem um rótulo anexo. Fruits Este exemplo termina imprimindo o valor inicial do widget combobox, `banana`.

[Python](#) [R](#) [Scala](#) [SQL](#)

[Cópia de](#)

```
dbutils.widgets.combobox(  
    name='fruits_combobox',  
    defaultValue='banana',  
    choices=['apple', 'banana', 'coconut', 'dragon fruit'],  
    label='Fruits',  
)  
  
print(dbutils.widgets.get("fruits_combobox"))  
  
# banana
```

comando dropdown (dbutils.widgets.dropdown)

Cria e exibe um widget dropdown com o nome programático especificado, valor padrão, opções e rótulo opcional.

Para exibir ajuda para este comando, execute `dbutils.widgets.help("dropdown")`.

Este exemplo cria e exibe um widget dropdown com o nome programático `toys_dropdown`. Ele oferece as opções `alphabet blocks`, `basketball`, `cape`, `doll` e é definido com o valor inicial de `basketball`. Esse widget dropdown tem um rótulo anexo. Toys Este exemplo termina imprimindo o valor inicial do widget dropdown, `basketball`.

[Python](#) [R](#) [Scala](#) [SQL](#)

[Cópia de](#)

```
dbutils.widgets.dropdown(  
    name='toys_dropdown',  
    defaultValue='basketball',  
    choices=['alphabet blocks', 'basketball', 'cape', 'doll'],  
    label='Toys'  
)  
  
print(dbutils.widgets.get("toys_dropdown"))
```

```
# basketball
```

comando get (dbutils.widgets.get)

Obtém o valor atual do widget com o nome programático especificado. Esse nome programático pode ser:

- O nome de um widget personalizado no notebook, por exemplo `fruits_combobox` ou `toys_dropdown`.
- O nome de um parâmetro personalizado passado para o Notebook como parte de uma tarefa do Notebook, por exemplo, `name` ou `age`. Para obter mais informações, consulte a cobertura dos [parâmetros para a tarefa do Notebook](#) na interface do usuário do Job ou o campo `notebook_params` nas operações [Trigger a new Job execution \(POST /jobs/run-now\)](#) no Jobs API.

Para exibir ajuda para este comando, execute `dbutils.widgets.help("get")`.

Este exemplo obtém o valor do widget que tem o nome programático `fruits_combobox`.

```
Python  R  Scala  SQL  
dbutils.widgets.get('fruits_combobox')
```

```
# banana
```

Este exemplo obtém o valor do parâmetro da tarefa do notebook que tem o nome programático `age`. Este parâmetro foi configurado para 35 quando a tarefa do notebook relacionada foi executada.

```
Python  R  Scala  SQL
```

```
dbutils.widgets.get('age')
```

```
# 35
```

comando getAll (dbutils.widgets.getAll)

Obtém um mapeamento de todos os nomes e valores de widgets atuais. Isso pode ser especialmente útil para passar rapidamente valores de widget para uma consulta `spark.sql()`.

Esse comando está disponível em Databricks Runtime 13.3 LTS e acima. Ele só está disponível para Python e Scala.

Para exibir ajuda para este comando, execute `dbutils.widgets.help("getAll")`.

Este exemplo obtém o mapa de valores de `widget` e o passa como argumentos de parâmetro em uma consulta Spark SQL.

Python Scala

```
df = spark.sql("SELECT * FROM table where coll = :param", dbutils.widgets.getAll())
df.show()
```

```
# Query output
```

Cópia de

comando getArgument (dbutils.widgets.getArgument)

Obtém o valor atual do widget com o nome programático especificado. Se o widget não existir, uma mensagem opcional poderá ser retornada.

Observação

Este comando está obsoleto. Em vez disso, use `dbutils.widgets.get`.

Para exibir ajuda para este comando, execute `dbutils.widgets.help("getArgument")`.

Este exemplo obtém o valor do widget que tem o nome programático `fruits_combobox`. Se este widget não existir, a mensagem `error: Cannot find fruits_combobox` será retornada.

```
dbutils.widgets.getArgument('fruits_combobox', 'Error: Cannot find fruits combobox')

# Deprecation warning: Use dbutils.widgets.text() or dbutils.widgets.dropdown() to create a widget and dbutils.widgets.get() to get its bound
value.

# Out[3]: 'banana'
```

comando multiselect (dbutils.widgets.multiselect)

Cria e exibe um widget de seleção múltipla com o nome programático especificado, valor padrão, opções e rótulo opcional.

Para exibir ajuda para este comando, execute dbutils.widgets.help("multiselect").

Este exemplo cria e exibe um widget de seleção múltipla com o nome programático days_multiselect. Ele oferece as opções Monday a Sunday e é definido com o valor inicial de Tuesday. Este widget de seleção múltipla possui um rótulo Days of the Week. Este exemplo termina imprimindo o valor inicial do widget de seleção múltipla, Tuesday.

```
Python R Scala SQL
```

```
dbutils.widgets.multiselect(
    name='days_multiselect',
    defaultValue='Tuesday',
    choices=['Monday', 'Tuesday', 'Wednesday', 'Thursday',
             'Friday', 'Saturday', 'Sunday'],
    label='Days of the Week'
)
```

```
print(dbutils.widgets.get("days_multiselect"))
```

```
# Tuesday
```

comando remove (dbutils.widgets.remove)

Remove o widget com o nome programático especificado.

'ara exibir ajuda para este comando, execute dbutils.widgets.help("remove").

Importante!

Se você adicionar um comando para remover um widget, não poderá adicionar um comando subsequente para criar um widget na mesma célula. Você deve criar o widget em outra célula.

Este exemplo remove o widget com o nome programático `fruits_combobox`.

[Python](#) [R](#) [Scala](#) [SQL](#)

[Cópia de](#)

```
dbutils.widgets.remove('fruits_combobox')
```

comando removeAll (dbutils.widgets.removeAll)

Remove todos os widgets do notebook.

Para exibir ajuda para este comando, execute `dbutils.widgets.help("removeAll")`.

Importante!

Se você adicionar um comando para remover todos os widgets, não poderá adicionar um comando subsequente para criar quaisquer widgets na mesma célula. Você deve criar os widgets em outra célula.

Este exemplo remove todos os widgets do notebook.

[Python](#) [R](#) [Scala](#)

[Cópia de](#)

```
dbutils.widgets.removeAll()
```

comando text (dbutils.widgets.text)

Cria e exibe um widget de texto com o nome programático especificado, o valor padrão e o rótulo opcional.

Para exibir ajuda para este comando, execute `dbutils.widgets.help("text")`.

Este exemplo cria e exibe um widget de texto com o nome programático `your_name_text`. É definido com o valor inicial de `Enter your name`. Este widget de texto possui um rótulo `Your name`. Este exemplo termina imprimindo o valor inicial do widget de texto, `Enter your name`.

```
Python R Scala SQL Cópia de

dbutils.widgets.text(
    name='your_name_text',
    defaultValue='Enter your name',
    label='Your name'
)

print(dbutils.widgets.get("your_name_text"))

# Enter your name
```

Biblioteca de APIs de utilitários do Databricks

Importante!

A biblioteca Databricks utilidades API (`dbutils-api`) está obsoleta. Embora essa biblioteca ainda esteja disponível, o site Databricks não planeja nenhum novo trabalho de recurso para a biblioteca `dbutils-api`.

Databricks recomenda que o senhor use uma das seguintes bibliotecas:

- [Databricks utilidades para Scala, com Java](#)
- [Databricks utilidades para Scala, com Scala](#)

Para acelerar o desenvolvimento de aplicativos, pode ser útil compilar, criar e testar aplicativos antes de implantá-los como jobs de produção. Para que você possa compilar em relação aos utilitários do Databricks, o Databricks fornece a biblioteca `dbutils-api`. Você pode fazer o download da biblioteca `dbutils-api` na página da web da [API DBUtils](#) no site do Repositório Maven ou incluir a biblioteca adicionando uma dependência ao seu arquivo de compilação:

- [SBT](#)

```
libraryDependencies += "com.databricks" % "dbutils-api_TARGET" % "VERSION"
```

- Maven

XML

Cópia de

```
<dependency>
  <groupId>com.databricks</groupId>
  <artifactId>dbutils-api_TARGET</artifactId>
  <version>VERSION</version>
</dependency>
```

- Gradle

Bash

Cópia de

```
compile 'com.databricks:dbutils-api_TARGET:VERSION'
```

Substitua TARGET pelo destino desejado (por exemplo, 2.12) e VERSION pela versão desejada (por exemplo, 0.0.5). Para obter uma lista de destinos e versões disponíveis, consulte a página da web da [API DBUtils](#) no site do Repositório Maven.

Depois de criar seu aplicativo com base nessa biblioteca, você pode implantar o aplicativo.

Importante!

A biblioteca dbutils-api permite compilar localmente um aplicativo que usa dbutils, mas não executá-lo. Para executar o aplicativo, você deve implantá-lo no Databricks.

Limitações

\ chamada dbutils dentro dos executores pode produzir resultados inesperados ou resultar em erros.

Se o senhor precisar executar operações do sistema de arquivos em executores que usam dbutils, há várias alternativas mais rápidas e mais escaláveis disponíveis:

- Para operações de cópia ou movimentação de arquivos, você pode verificar uma opção mais rápida de executar operações do sistema de arquivos descritas em [Operações do sistema de arquivos Parallelize](#).
- Para operações de lista e exclusão do sistema de arquivos, você pode consultar métodos de listagem e exclusão paralela utilizando o Spark em [Como listar e excluir arquivos mais rapidamente no Databricks](#).

Para obter informações sobre executores, consulte [Visão geral do modo de cluster](#) no site do Apache Spark.

© Databricks 2024. Todos os direitos reservados. Apache, Apache Spark, Spark e o logotipo Spark são marcas registradas da [Apache Software Foundation](#).

[Envie-nos comentários](#) | [Política de Privacidade](#) | [Termos de uso](#)