Hands-On Lab: Databricks DBUtils Complete Guide

Created By: Dr. Sandeep Kumar Sharma

Learning Objectives

- Understand the purpose and usage of Databricks DBUtils.
- Explore various DBUtils modules such as fs, secrets, widgets, and notebook.
- Perform hands-on operations like creating directories, listing files, copying/moving data, and working with secrets.
- Learn how to parameterize notebooks using widgets.
- Learn how to call notebooks and pass/receive parameters.

Learning Outcomes

After completing this lab, you will be able to: - Execute and utilize all major DBUtils features confidently. - Perform file system operations inside Databricks without using the storage UI. - Create and use widgets for notebook parameterization. - Access secure values using secret scopes. - Chain notebooks and pass values between them using dbutils.notebook.

---

1. Introduction to DBUtils

DBUtils is a set of utilities provided by Databricks to perform common development and administrative tasks such as: - File system operations - Notebook workflows - Secret management - Widget-based inputs

To start, run the following command to view general help about DBUtils:

```
# Display DBUtils help
dbutils.help()
```

---

2. DBUtils FS Module (File System Operations)

The `dbutils.fs` module allows you to interact with DBFS and perform operations such as: - Creating directories - Listing directories - Copying/moving files - Removing files/directories - Reading and writing file contents

2.1 List Directory

```
# List all files and folders inside a directory
files = dbutils.fs.ls("/databricks-datasets")
display(files)
```

2.2 Create Directory

```
# Create a new directory
path = "/tmp/dbutils_demo"
dbutils.fs.mkdirs(path)
print(f"Directory created: {path}")
```

2.3 Copy Files

```
# Copy a file
source = "/databricks-datasets/airlines/README.md"
destination = "/tmp/dbutils_demo/README_copy.md"
dbutils.fs.cp(source, destination)
print("File copied successfully!")
```

2.4 Move File

```
# Move file from one location to another
source = "/tmp/dbutils_demo/README_copy.md"
destination = "/tmp/dbutils_demo/moved_README.md"
dbutils.fs.mv(source, destination)
print("File moved successfully!")
```

2.5 Delete File/Directory

```
# Delete a directory or file (recursive deletion)
dbutils.fs.rm("/tmp/dbutils_demo", recurse=True)
print("Directory deleted successfully!")
```

2.6 Read File Content (Head)

```
# Show first few lines of a file
content = dbutils.fs.head("/databricks-datasets/airlines/README.md", 200)
print(content)
```

2.7 Write a File (PUT)

```
# Write content to a file
filepath = "/tmp/dbutils_demo_file.txt"
dbutils.fs.put(filepath, "This is a sample file created using DBUtils!",
overwrite=True)
print("File created successfully!")
```

---

3. DBUtils Secrets Module

The `dbutils.secrets` module allows you to securely retrieve credentials stored inside Databricks Secret Scopes.

3.1 List Secret Scopes

```
# List all available secret scopes
dbutils.secrets.listScopes()
```

3.2 List Secrets Inside a Scope

```
# List secrets inside a specific scope
# Replace "my-scope" with your actual scope name
dbutils.secrets.list("my-scope")
```

3.3 Get Secret Value

```
# Retrieve a secret value
# Replace with actual scope and key
secret_value = dbutils.secrets.get(scope="my-scope", key="my-secret-key")
print("Secret retrieved successfully (Value not printed for security)")
```

---

4. DBUtils Widgets Module

Widgets allow notebooks to accept user inputs (dropdowns, text fields, etc.).

4.1 Create a Dropdown Widget

```
dbutils.widgets.dropdown("env", "dev", ["dev", "test", "prod"], "Select
Environment")
```

4.2 Create a Text Widget

```
dbutils.widgets.text("username", "guest", "Enter Username")
```

4.3 Get Widget Value

```
env = dbutils.widgets.get("env")
username = dbutils.widgets.get("username")
print(f"Environment Selected: {env}")
print(f"Username Entered: {username}")
```

4.4 Remove Widgets

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

---

5. DBUtils Notebook Module

This module is used to call notebooks from another notebook, and pass/receive parameters.

5.1 Run Another Notebook

```
# Calling another notebook and passing parameters
result = dbutils.notebook.run("./child_notebook", 60, {"input": "Hello from
parent"})
print(result)
```

5.2 Retrieve Parameters Inside Child Notebook

```
# In child_notebook
input_val = dbutils.widgets.get("input")
print(f"Value from Parent Notebook: {input_val}")
dbutils.notebook.exit("Execution Completed Successfully!")
```

---

6. DBUtils Library Module

Used for installing and managing libraries.

6.1 Install Library

```
# Install library from PyPI
dbutils.library.installPyPI("requests")
```

6.2 Restart Python (When Needed)

```
dbutils.library.restartPython()
```

---

7. Summary

This lab covered: - DBUtils help and structure - Complete dbutils.fs operations (list, mkdirs, cp, mv, rm, put, head) - Managing secrets securely - Creating interactive widgets - Notebook chaining using dbutils.notebook - Installing Python libraries using dbutils.library

You now have a complete practical understanding of how DBUtils works in Databricks, and you can use these utilities in any real-world Databricks project.

End of Lab