Data driven testing

Testing the feature with multiple set of data is called as **data driven testing**.

Or

Data-driven testing (DDT) is taking a test, parameterizing it and then running that test with varying data.

Note:

- This allows you to run the same test case with many varying inputs, therefore increasing coverage from a single test. In addition to increasing test coverage, data driven testing allows the ability to build both positive and negative test cases into a single test.
- Often data is stored either in a text file and are separated by commas or in Excel files and are presented as a table. If you need to add more data, you simply modify the file either in any text editor or in Microsoft Excel (in case of hard-coded values, you should modify both data and code).

Data driven testing using webdriver

It is the responsibility of test script developer to create and script data driven infrastructure so that the environment is capable of supplying different set of test data repeatedly and workflows that need to be tested against a particular application.

Practical scenario:

- Let's suppose we have a scenario to test the login functionality using various set of test data. So as per automation testing standards and data driven approach used in automation testing, automation engineer will write a generic script to test the application with varying data values.
- Automation engineer will create test scripts against this scenario:
    - Using valid credentials, test sign in functionality
    - Using username and blank password to test sign in functionality
    - Using blank username and password to test sign in functionality
    - And finally using no username and no password means send both fields blank to check the behaviour of the sign in functionality.

Selenium cannot read the data from external resources. In order to read and write the data into external resources, selenium uses the python modules.

Note:

- Test data can be stored in different file systems.

Example:

1. JSON files.
2. Properties files.
3. XML files.
4. Excel files.
5. PDF files.

How to Handle Data in Property Files in Python?

Introduction:

- When you want to store the data as a key and value pair in the external resource file system. We can use the properties files.
- We use properties files to keep the common test data used by all the tests or the configuration test data, since properties file are light weighted software, reading and writing the data into properties file will be faster compared to any other files.
  Example: To read the data from Excel and PDF takes more time compared to properties files.

What is a property file?

- **. properties** files are mainly used in to maintain **project configuration data**, **database config** or **project settings** etc.
- Each parameter in properties file are stored as a pair of strings, in **key-value** pair format, where each key is on one line.
- You can easily read properties from some file using object of type **Properties**. This is a utility provided by "**PyJavaProperties**" package.

Prerequisites

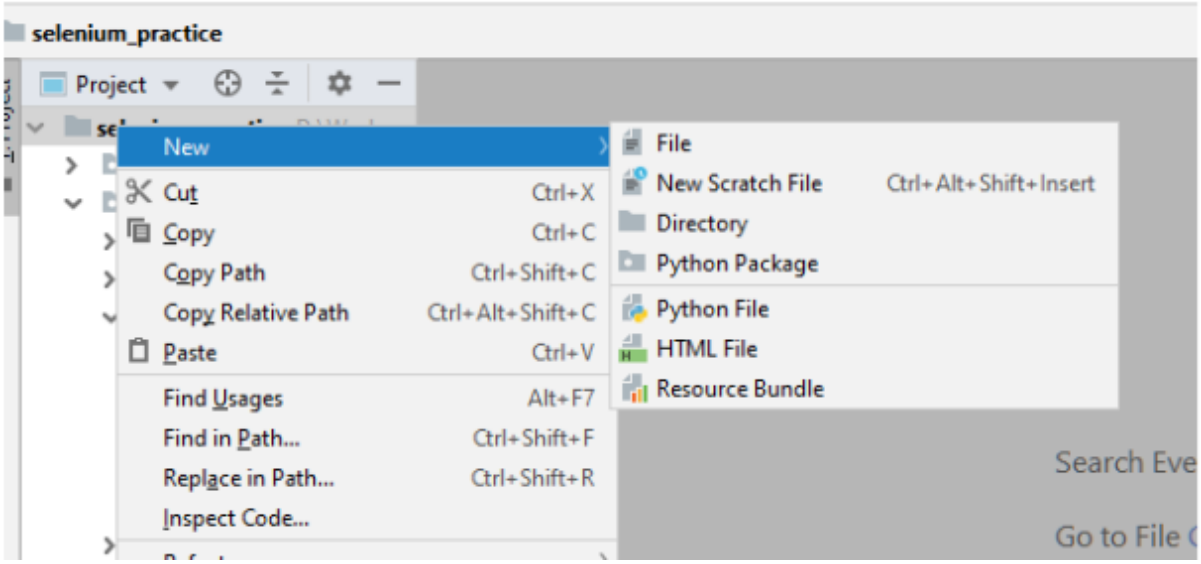
Install "**PyJavaProperties".**

```
C:\Windows\system32>pip install pyjavaproperties
Requirement already satisfied: pyjavaproperties in c:\users\jayapriyapramod\ap
pdata\local\programs\python\python37-32\lib\site-packages (0.7)
```

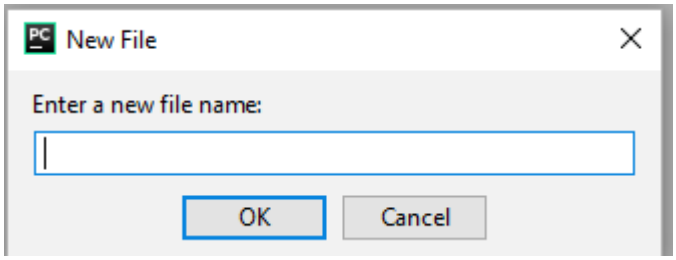
How to write/read from property files

Step 1: Create a property file.

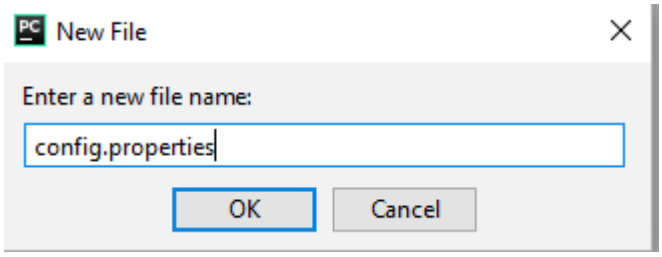
Right click on the project in PyCharm -> new -> File



Step 2: After selecting the "File" option. We will get the below window.

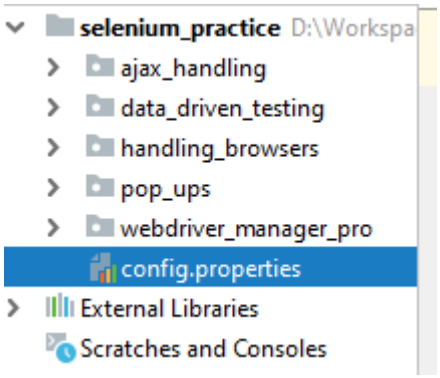


Step 3: The only thing we need to do is to give the file name and give the extension as ***.properties***. In our case we name it as ***Config.properties***.

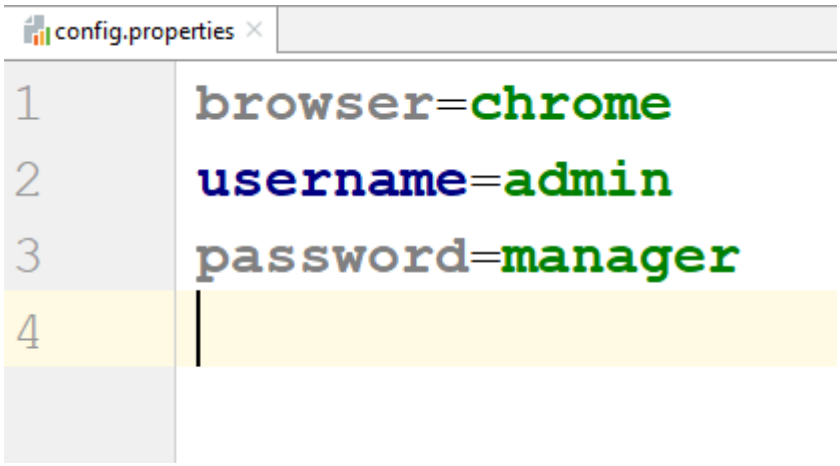


Step 4: Click Ok.

Note: Config.properties file added to your project.



Step 5: Store data onto properties file.



Note:

- Data is stored in properties file in the form of key-value pairs, with the key being unique across the file.
- Remember white space between the property name and property value is always ignored.
- Don't declare the key and value within double ("") quotation in properties file.
- Since by default all the values in properties file be saved as string.

How to read the data from properties file.

Step 1 — Opening a File

We will then use Python's "open ()" function to open our "**Config.properties**" file. The open () function requires as its first argument the file path. The function also allows for many other parameters. However, most important is the optional *mode* parameter. Mode is an optional string that specifies the mode in which the file is opened. The mode you choose will depend on what you wish to do with the file. Here are some of our mode options:

- 'r': use for reading
- 'w': use for writing
- 'x': use for creating and writing to a new file
- 'a': use for appending to a file
- 'r+': use for reading and writing to the same file

In this example, we only want to read from the file, so we will use the 'r' mode. We will use the open () function to open the "Config.properties" file and assign it to the variable "input_stream".

```
from pyjavaproperties import Properties

file_path = "D:/Workspace/config.properties"
input_stream = open(file_path, mode="r")
```

Step 2 — Reading a File

a) Since our file has been opened, we can now manipulate it (i.e. read from it). **PyJavaProperties** provides a class "**Properties**".

b) Create an object of class "**Properties**".

c) Properties class has a method "**load**", which is used to load the opened file to properties class.

d) After loading, call a method "**getProperties**" by providing "Key" name as an argument, which returns the value of the provided key.

Below is the program to read the data from properties file.

```
from pyjavaproperties import Properties

file_path = "D:/Workspace/config.properties"
input_stream = open(file_path, mode="r")

prop = Properties()
prop.load(input_stream)
browser_name = prop.getProperty("browser")
username = prop.getProperty("username")
password = prop.getProperty("password")

print("Browser name: ", browser_name)
print("Username: ", username)
print("Password: ", password)

input_stream.close()
```

Step 3 — Writing a File

a) Since we need to write data to our file, we can now need to open the file with different mode to manipulate it (i.e. write data to it).

i.e.

I. 'r': use for reading
II. 'w': use for writing
III. 'x': use for creating and writing to a new file
IV. 'a': use for appending to a file
V. 'r+': use for reading and writing to the same file

Note: Since we want to write the data to the existing file, we use "r+" mode.

b) Create an object of class "**Properties**".

c) Call the method "**setProperty**" method of properties class. This method takes two arguments**.**
Argument 1: Key.

Argument 2: Value.
Note: Key should be unique, value should be the value you want to write to properties file.

```
// set the properties value
prop.setProperty("database", "localhost");
prop.setProperty("dbuser", "pramod");
prop.setProperty("dbpassword", "password");
```

d) After writing the data, we need to save the file. In order to save, call "**store**" method. This method will save the properties file.

e) Once after saving, we need to close the output stream using the **"close"** method.

Below is the program to read the data from properties file.

write_properties.py

```
1  from pyjavaproperties import Properties
2
3  file_path = "D:/Workspace/write.properties"
4
5  prop = Properties()
6  prop.setProperty("browser", "firefox")
7  prop.setProperty("username", "admin")
8  prop.setProperty("password", "manager")
9
10 output_stream = open(file_path, mode="r+")
11 prop.store(output_stream)
12
13 output_stream.close()
14
```