DDE: Data Drive Engine

1. Workflow
2. Instructions
3. Additional configuration
4. Pre-requisites
5. FAQ
    a) What syntax is used to create the text template?
    b) I'm unsure of starting to create my rule-set. Are there sample templates I could use?
    c) What is meant by forward and backward references in an excel template?
    d) How can DDE be re-run in multiple iterations?
    e) How does variable storage work?
    f) What are the pre-requisites for the tool?
    g) The tool doesn't appear to work, and doesn't create the populated excel file. What could be wrong?
    h) What makes DDE a framework?
    i) How to create a new user-defined function?
    j) Are there any known limitations of the tool?
    k) I want to know more about the tool's architecture? Is the tool open-source?
    l) While extracting from HTML, the tool gives Exception: ReactorNotRestartable. How can this be solved?
    m) Where can I get additional help with the tool?

Appendix A

DDE: Data Drive Engine

DDE is a python-based tool that helps extend typical Excel functionalities, like its formulae and macros. On top of that, the tool allows to create massive amounts of data that follows a specified template, characterized by a set of rules. This will greatly enhance the effectiveness of testing, specifically automated and performance testing.

At the heart of the program is the concept of a data template, that, simply put, is a textual representation of an excel sheet. In the present version of DDE (DDE-Lite), you must manually populate the template with custom rules, which in turn can be used to create huge amounts of customized data, to easily fill up the web page during a performance test run.

The rules could vary in complexity and can handle simple data validation (as seen in Excel) rules, to very complicated user-defined functions (as defined from Python). It also supports the cross-references between excel cells, INDIRECT references (as supported in Excel), and has built-in concatenation methods, both of which are available out-of-the-box.

For more insights into workflow, usage instructions, and prerequisites of the tool, scroll down.

1. Workflow

Data Driven with Excel (DDE) tool is meant to help with populating pre-defined Excel templates with test data that must satisfy certain rules. The rules are typically specified in a text file, using a proprietary language. This text file can be used to generate test data.

This solution is implemented in two steps.

1. Pre-processor reads the Excel template (ExcelReader.exe), and reverse-engineers it to create a text template. This is an optional step in the tool workflow. This feature is not available in DDE-Lite.

2. Populate data in template_latest.xls, according to the rules contained in the text template. Before imitating this step, user can add/modify/delete the rules containing in the text template.

2. Instructions

1. Execute main.py, in order to populate the data, based on the rules laid out in the template (Template.txt), into Template_latest.xls

   python ExcelWriter\main.py --config "Template.txt" --rowcount 10 --colcount 10 --startrow 1 --output "Template_latest.xls

   NOTE: Only rowcount is configurable. For example, if rowcount is set to 1000, the above run would create 1000 data rows. Other parameters are not configurable in DDE-Lite.

5. Rename Template_latest.xls to Template.xls, before another run is made. Multiple runs will be necessary to correctly populate all test data, if the rules contain back-references or used to extract data from HTML pages. For details, see this FAQ entry.

6. All messages, including stack-traces for exceptions are captured in log.txt in the main folder.

3. Additional configuration

If the excel template contains any data in it, it'll be re-used when the test-data is generated. By default, it re-uses the data from the first 10 columns, starting from the 1st row. Configuration of this isn't available in DDE-Lite.

Supported rules include

- Forward references among Excel cells. For instance, value from column2 (C2) is equal to column1 (C1).
- Indirect references among Excel cells. For instance, possible values from column2 (C2) depends on the value from column1 (C1)
- Using forward references in python based user-defined functions. For a complete list of built-in functions, available out-of-the-box in the current version of the tool, refer to Appendix A.
- Back-references among Excel cells (value from C1 is equal to C2). In such cases, the program must be run in more than one iterations.

NOTE: Return value from the user-defined functions is used to populate the test data. The return value can be in the form of strings, lists, and dictionaries (hashes).

4. Pre-requisites

1. Tool works only on Windows systems, since there's a dependency on Excel files. This will changed in a future version.

2. Microsoft Excel must be present on the system. As of now, the tool supports only xls format. New excel file formats (xlsx) are not supported.

3. Python3.x must be present on the system.

4. Tool uses a few python3.x modules that aren't built-in. In order to install these modules, open a command prompt and run the following: "pip install -r .\Help\requirements.txt" from the main folder.

5. If you want the Youtube related functions (VIDEO_SEARCH, VIDEO_SIMILAR) to work, you must provide your own Youtube API key in api_keys.txt. Refer to this page for more details.

5. FAQ

a) What syntax is used to create the text template?

Rules specified using text templates follow a proprietary language, which have its own syntax. While it's simple to use, you'll also find it to be very strict about syntax checks and are incapable of raising compile-time errors, unlike other programming languages. In most cases, the run-time exceptions are logged into log.txt. Follow these guidelines to avoid such errors.

- Each line in the text template should contains the following four attributes and separated by pipe symbols: the row/column no. in the Excel to write the function's return string to (or a variable name to store the return value, in case the function returns a list or a dictionary), description of the

function, number of return values from the function, followed by the function call syntax. Note that the second and third items above are used only for informative purposes, in the present implementation.

- Pay attention to the difference in the syntax of a function call, when compared to regular programming languages. Refer to Appendix A for example usage.
- Template uses forward slash (/) instead of the standard quote character. This means that instead of writing 'first name' you will write /first name/.
- It follows from above that if you want to use any of the special characters (#, whitespace, or an empty string are all treated as special characters) as literals, bound them within two forward slashes.
- If you want to use the forward slash (/) itself as a literal, escape using backward slash (\). However, note that this works only in non-Windows systems. Use of escapes will also depend on the context. See demo templates that could offer a practical explanation.
- Use comments wherever required. Anything that appears after # (unless quoted) is treated as a comment.

b) I'm unsure of starting to create my rule-set. Are there sample templates I could use?

Yes, of course.

Use the templates contained in "Demo_templates" folder (within the main folder). Copy a sample into the root folder, and rename to 'Template.txt'. Among its text templates, they demonstrate the usage of all available built-in functions, in various combinations. Alternatively, see Appendix A for example usage of all built-in functions.

c) What is meant by forward and backward references in an excel template?

The concept is best explained with an example. Consider a template comprising of 4 columns, where the column-2 refers to value in the corresponding row of the column-1, and column-3 refers to the value in the corresponding row of the column-4. The former case is an example of a forward reference and the latter that of backward reference. In the latter case, while the data is being populated into the column-3, column-4 is not available yet. Hence, you'll want to first populate the column-4 with data, and re-run to populate the column-3.

d) How can DDE be re-run in multiple iterations?

You might want to re-run DDE in multiple iterations in two separate cases – in order to handle backward references, and to work around ReactorNotRestartable exception while extracting data from HTML pages.

Follow these steps.

- After the script is run, save Template_latest.xls to Template.xls.
- Comment out all lines that were run during earlier iteration(s) in Template.txt.
- When the tool is re-run this time, only the lines that were not run during the last time will be used.
- Repeat the above steps, to iterate again.

Note that the Template_latest.xls will contain the data from Template.xls, in addition to the new data.

e) How does variable storage work?

Variable storage is a feature available in the tool, to store data of any type. Thus, instead of storing the output into an excel cell, it's stored into variable instead. The purpose of variable storage should be evident. Typically, when a list is stored into variable, each item in the list can be accessed using the NEXT build-in function.

Variable names must start with $. Note that references doesn't apply, when data storage is used. This is because, references require specification of location inside an excel sheet (row), and are interpreted to start from the specified row.

f) What are the pre-requisites for the tool?

Refer to the section on pre-requisites.

g) The tool doesn't appear to work, and doesn't create the populated excel file. What could be wrong?

Following are a few reasons this could happen.

- During the last execution of the tool, some Excel processes were left hanging in your system, thus preventing the tool from working.

  Solution: Kill all currently running excel processes and retry.

- The tool utilizes a few python modules to do its tasks. Some of the modules were probably not available at the time of running the tool.

  Solution: Check if all the required modules have been installed, as mentioned in the pre-requisites section.

- Script ran into an exception.

  Solution: See log.txt for the last run's details, and troubleshoot code. This is a rare situation, unless you've user-defined functions of your own.

- Inadvertent modification of the source code.

  Solution: See answer to last question in this FAQ.

h) What makes DDE a framework?

DDE supports most of the required functionality required in a test-generation tool, by way of its built-in functions. However, there might be situations where you want to populate custom-created or custom-formatted data into the template. In such cases, the tool provides option to create user-defined functions, very much like with a framework. This feature helps extend the tool, well beyond its present capabilities.

i) How to create a new user-defined function?

Define them in functions.py, in *ExcelWriter* folder. Follow these steps, to create a user-defined function.

a. Copy one of the built-in functions (eg.STR_REPLACE) available in functions.py and rename appropriately.
b. Do not change the input arguments of the function. It must be function_args.
c. Return value must one of the following types – string, list or dictionary.
d. Write up your code before the return statement, and test it in a small unit before incorporating into the framework.

j) Are there any known limitations of the tool?

As of now, the tool supports only xls format. This is only because it presently uses a set of python modules (*xlrd* and *xlwt*) that are incapable of handling newer (xlsx) excel formats. However, there're other similar python modules that support the required formats. Thus, the tool can be modified to use these modules.

As mentioned in one of the above answers, template uses forward slash (/) instead of the standard quote character. Use of forward slash as a literal, however, is supported only in non-Windows environments. This is due to a limitation of the python module used for the purpose (*shlex)*

k) I want to know more about the tool's architecture? Is the tool open-source?

As of now, an architecture diagram for the tool doesn't exist. However, in future, there might be. Please stay connected. However, following can be said about the tool at a high-level.

The tool's core (including data creation, populating the data) is built using Python. There are more than one file under ExcelWriter folder.

CAUTION: If you're unfamiliar with the language or are unsure of the outcome of altering code, please don't go there. Even if you're familiar with the language, it's a good idea not to "mess" with the code, unless of course you want to create/modify user-defined functions.

Further, the tool is available as open-source, in GitHub.

l) While extracting from HTML, the tool gives Exception: ReactorNotRestartable. How can this be solved?

This error is due to a limitation of the scrapy module used to extract data from HTML. To solve the issue, do not use built-in functions more than once in a text template. If they must be used more than once, re-run DDE in multiple iterations. For an example template, see Template_demo13.txt.

m) Where can I get additional help with the tool?

Contact creator of the tool at ananddotiyer@gmail.com

Appendix A

Following the list of all built-in functions supported by the tool. All these functions are present in functions.py.

Function	Example usage	Return type	Comments
CHOICE	(CHOICE:Anand,Aravind,Sudha,Nitesh,Sadhna)	String	Makes a random choice from a list of many items.
CHOICE_RANGE	(CHOICE_RANGE:500,5000,100)	String	Makes a random choice from within a range of numbers, and optionally round it to the nearest multiple of given number.
INDIRECT	(INDIRECT:C2,/{'Mars':['Phobos','Deimos'],'Pluto':['Nix','Hydra','Kerberos','Styx'],'Earth':['Moon','ERS-1','ERS-2']}/)	String	Makes a random choice from one of the many lists, depending on another value.

DATE_RANDOM	(DATE_RANDOM:1-1-2008,1-1-2009)	String	Returns a random date between the given start and end dates.
IF	(IF:C1,C2,Equal,Not equal)	String	Returns 3rd input, if 1st input matches 2nd input, else 4th input.
FAKE	(FAKE:address,en_US)	String	Generate fake data for multiple categories, including name, company, address etc. Data can be generated in 35 different languages. Refer https://faker.readthedocs.io/en/master/ for more information.
XEGER	(XEGER:/[a-z]{3}[!@#$%][a-z]{3}\d/)	String	Generate data according to specified regex patterns. Refer https://pypi.python.org/pypi/rstr/2.2.5 for more information.
STR_LENGTH	(STR_LENGTH:C1)	String	Length of the specified string, formatted as a string.
STR_SPLIT	(STR_SPLIT:C1,1)	String	Splits string by single space, and returns the string indexed by given number.
STR_REPLACE	(STR_REPLACE:/[-_,]/,//,C2)	String	Replace part of a string by another, according to

			specified regex patterns.
STR_SUB	(STR_SUB:C1,-1,C2)	String	Extract sub-string from given string. Works in forward/reverse mode, depending on sign of start_index.
STR_CONCAT	(STR_CONCAT:C4,@,C3,.com)	String	Concatenates multiple strings provided as its arguments.
LIST	(LIST:One,Two,Three,Four,Five)	List	Creates a list, with specified string values.
NEXT	(NEXT:$store)	String	Gets next item from a list. Supports only list of strings.
LIST_FILES	(LIST_FILES:/C:\Python27/)	List	Generates a list of files (with absolute file path), by way of recursive search, from specified folder.
LIST_FILES_WITH_LINKS	(LIST_FILES_WITH_LINKS:/D:\Music/)	List	Generates a list of files (with absolute file path) with hyperlinks, by way of recursive search, from specified folder.
LIST_FROM_DICT	(LIST_FROM_DICT:$video,Title)	List	Generates separate lists from dictionaries, by its keys.
LIST_SERIAL	(LIST_SERIAL:pre_,1,1001)	List	Generates a list of serial numbers, in a given

			range. 1st argument represents a prefix (None for no prefix), 2nd and 3rd argument representing start and end of the range, and last argument representing the skip.
EVAL_IN_PYTHON	(EVAL_IN_PYTHON:/range (1,1001)/)	Any	Evaluates basic python function calls. Supports references.
VIDEO_SEARCH	(VIDEO_SEARCH:/software|testing|training/)	Dict	Generates a list of Youtube videos containing specified keywords. Refer https://developers.google.com/youtube/v3/docs/search/list for more information.
VIDEO_SIMILAR	(VIDEO_SIMILAR:cm-cSW66Isc)	Dict	Generates list of similar/recommended Youtube videos. Refer https://developers.google.com/youtube/v3/docs/search/list for more information.
LIST_EXTRACT_HTML	(LIST_EXTRACT_HTML:http://sdtimes.com/whats-store-next-generation-internet-things/,h1[@class='page-header']/text())	List (String)	Extracts data using given XPATH, from an HTML page.
LIST_LINKS_HTML	(LIST_LINKS_HTML:http://sdtimes.com/whats-store-next-generation-internet-things/)	List	Extracts embedded links from within the article, in HTML page.

EXTRACT_AMAZON	(EXTRACT_AMAZON:http://www.amazon.in/s/&page=3&keywords=laptop)	Dict	Extracts multiple attributes (including name and price) from amazon search result page.

NOTE: Unless otherwise specified, all built-in functions support references. In case the rules contain backward references, the program must be run multiple times as mentioned in the section on Instructions.