Gauge Framework

Gauge is a free and open source framework for writing and running E2E tests. Some key features of Gauge that makes it unique include:

- Simple, flexible and rich syntax based on Markdown.
- Consistent cross-platform/language support for writing test code.
- A modular architecture with plugins support
- Extensible through plugins and hackable.
- Supports data driven execution and external data sources
- Helps you create maintainable test suites
- Supports Visual Studio Code, Intellij IDEA, IDE Support

What is a Specification

Gauge specifications are written using a Markdown syntax. For example

```
# Search for the data blob

## Look for file
* Goto Azure blob
```

In this specification *Search for the data blob* is the **specification heading**, *Look for file* is a **scenario** with a step *Goto Azure blob*

What is an Implementation

You can implement the steps in a specification using a programming language, for example:

```
from getgauge.python import step
import os
from step_impl.utils.driver import Driver
@step("Goto Azure blob")
def gotoAzureStorage():
  URL = os.getenv('STORAGE_ENDPOINT')
  Driver.driver.get(URL)
```

The Gauge runner reads and runs steps and its implementation for every scenario in the specification and generates a report of passing or failing scenarios.

```
# Search for the data blob

## Look for file  ✓

Successfully generated html-report to => reports/html-report/index.html
Specifications:       1 executed      1 passed        0 failed        0 skipped
Scenarios:     1 executed      1 passed        0 failed        0 skipped
```

Re-using Steps

Gauge helps you focus on testing the flow of an application. Gauge does this by making steps as re-usable as possible. With Gauge, you don’t need to build custom frameworks using a programming language.

For example, Gauge steps can pass parameters to an implementation by using a text with quotes.

```
# Search for the data blob

## Look for file
* Goto Azure blob
* Search for "store_data.csv"
```

The implementation can now use “store_data.csv” as follows

```
from getgauge.python import step
import os
@step("Search for <query>")
def searchForQuery(query):
  write(query)
  press("Enter")

step("Search for <query>", (query) => {
  write(query);
  press("Enter");
```

You can then re-use this step within or across scenarios with different parameters:

```
# Search for the data blob

## Look for Store data #1
* Goto Azure blob
* Search for "store_1.csv"
```

```
## Look for Store data #2
* Goto Azure blob
* Search for "store_2.csv"
```

Or combine more than one step into **concepts**

```
# Search Azure Storage for <query>
* Goto Azure blob
* Search for "store_1.csv"
```

The concept, Search Azure Storage for `<query>` can be used like a step in a specification

```
# Search for the data blob

## Look for Store data #1
* Search Azure Storage for "store_1.csv"

## Look for Store data #2
* Search Azure Storage for "store_2.csv"
```

Data-Driven Testing

Gauge also supports data driven testing using Markdown tables as well as external csv files for example

```
# Search for the data blob

| query   |
|---------|
| store_1 |
| store_2 |
| store_3 |

## Look for stores data
* Search Azure Storage for <query>
```

This will execute the scenario for all rows in the table.

In the examples above, we refactored a specification to be concise and flexible without changing the implementation.

Other Features

This is brief introduction to a few Gauge features. Please refer to the Gauge documentation for additional features such as:

- Reports
- Tags
- Parallel execution
- Environments
- Screenshots
- Plugins
- And much more

Installing Gauge

This getting started guide takes you through the core features of Gauge. By the end of this guide, you'll be able to install Gauge and learn how to create your first Gauge test automation project.

Installation Instructions for Windows OS

Step 1: Installing Gauge on Windows

This section gives specific instructions on setting up Gauge in a Microsoft Windows environment. Download the following installation bundle to get the latest stable release of Gauge.

Step 2: Installing Gauge Extension for Visual Studio Code

Follow the steps to add the Gauge Visual Studio Code plugin from the IDE

1. Install the following Gauge extension for Visual Studio Code.

Troubleshooting Installation

If, when you run your first gauge spec you receive the error of missing python packages, open the command line terminal window and run this command:

```
python.exe -m pip install getgauge==0.3.7 --user
```

Installation Instructions for macOS

Step 1: Installing Gauge on macOS

This section gives specific instructions on setting up Gauge in a macOS environment.

1. Install brew if you haven't already: Go to the brew website, and follow the directions there.
2. Run the brew command to install Gauge

   ```
   > brew install gauge
   ```

   if HomeBrew is working properly, you should see something similar to the following:

```
==> Fetching gauge
==> Downloading https://ghcr.io/v2/homebrew/core/gauge/manifests/1.4.3
######################################################################### 100.0%
==> Downloading
https://ghcr.io/v2/homebrew/core/gauge/blobs/sha256:05117bb3c0b2efeafe41e817cd3ad863
==> Downloading from https://pkg-
containers.githubusercontent.com/ghcr1/blobs/sha256:05117bb3c0b2efeafe41e817cd3ad863
se=2022-12-13T12%3A35%3A00Z&sig=I78SuuwNgSMFoBTT
######################################################################### 100.0%
==> Pouring gauge--1.4.3.ventura.bottle.tar.gz
    /usr/local/Cellar/gauge/1.4.3: 6 files, 18.9MB
```

Step 2 : Installing Gauge Extension for Visual Studio Code

Follow the steps to add the Gauge Visual Studio Code plugin from the IDE

1. Install the following Gauge extension for Visual Studio Code.

Post-Installation Troubleshooting

If, when you run your first gauge spec you receive the error of missing python packages, open the command line terminal window and run this command:

```
python.exe -m pip install getgauge==0.3.7 --user
```

Last update: August 22, 2024