Epion-Tropic-Test-Tool (ETTT)

Table of Contents

What’s This

概要

このツールは、次世代回帰試験ツールです。
テストの効率化を目的とした扱いやすい(自分にとって)テストツールです。
簡単に言うとテストシナリオをYAMLに記載し、ツールに与えることでテストを走行させるというものです。
また、このツールはキーワード駆動テストを行うための手段として用いれるようにすることを目標としています。
最終的には「ISO/IEC/IEEE 29119-5:2016」に準拠したソフトウェアを作成したいと考えています。

機能

このツールが保有する機能の大まかな一覧は以下です。

1. YAMLで定義されたシナリオに従って処理を自動実行する
2. ファイルシステムの操作を行う
3. リモートで稼働しているSSHサーバと対話形式の操作を行う
4. SCPによるファイル操作を行う
5. FTPによるファイル操作を行う
6. SeleniumのWebDriverを用いてローカルのブラウザの自動操作を行う
7. AWSのS3に対してAPI発行を行う（一部APIのみ）
8. AWSのSQSに対してAPI発行を行う（一部APIのみ）
9. Redisに対して操作を行う（AWSのElastiCache for Redisを想定）
10. RDBMSに対してSQLを発行する
11. 任意のローカルコマンドの実行を行う

How To Use

概要

前提

このツールを利用するにあたり、以下の環境が整っていることを確認してください。

ソフトウェア	内容
Java	Javaの1.8以上がインストールされており、パスが通っていることを確認してください。

YAMLの作成

YAMLはテキストエディタでの完全手作業による作成となっています。

YAMLの基本構成

```
# ツールバージョン
t3 : 1.0
type : {scenario|parts|config}

# 情報
info :
   version : {scenario_version}
   id : {scenario_id|parts_id}
   summary: {scenario_summary|scenario_summary}
   description: {scenario_description|scenario_description}

# 処理フロー
flows :
  - ref : {reference scenario_process_id}
    type : {scenario|process}

# 処理定義（前後処理・実行・エビデンス取得・アサート）
processes :
  - id : {scenario_process_id}
    summary : {scenario_process_summary}
    description : {scenario_process_description}
    command : {command}
    # 以降はコマンドによって異なる

#################################################################
#    # 外部コマンド参照を行う場合
#    ref :
#      id : {reference scenario_component_id}
#      # 引数定義
#      args :
#        # キー：値として定義する
#        - {name} : {value}
#      # 結果定義（変数としてストアする）
#      results :
#        # キー：変数名として定義する
#        - {name} : {variable_name}
#################################################################

# 変数
variables :
  # 全体変数
  global :
    {name} : {initial_value}
  # メインシナリオ内変数
  scenario :
    {name} : {initial_value}
  # ファイル内変数
  local :
```

```
    {name} : {initial_value}

# プロファイル定義
profiles :
  # プロファイル名
  {profile_name} :
    # キー：値として定義する
    {name} : {value}

# カスタム定義
customs :
  command :
    # カスタム機能名とカスタム機能を作成したパッケージをキー：値として定義する
    {custom_name} : {package}

#
definitions :
```

上記構造は、ファイルに記載できる全ての要素となっている。
実際はファイルを分割し、用途毎に作成する。

ディレクトリ構成のベストプラクティス

ETTTは様々な１つのYAMLで全てを表すこともできますが、
シナリオのメンテナンス性や複数人で作成する場合の運用を考慮する場合、
細かく用途によってファイルを分割・配置することを推奨します。
また、これらのシナリオはGitやSubversionによるバージョン管理を行うことが重要でしょう。

```
root
 |
 |-- parts
 |      |
 |      `-- {parts_id}
 |            |-- parts.yaml (ex: {parts_id}.yaml)
 |            |-- data
 |            |     |-- excel_data.xlsx
 |            |     `-- etc...
 |            `-- assert
 |                  |-- excel_data.xlsx
 |                  `-- etc...
 |
 |-- scenarios
 |      |
 |      `-- {scenario_id}
 |            |-- scenario.yaml (ex: {scenario_id}.yaml)
 |            |-- data
 |            |     |-- excel_data.xlsx
 |            |     `-- etc...
 |            `-- assert
 |                  |-- excel_data.xlsx
 |                  `-- etc...
 |-- profiles
 |      |
 |      `-- profile.yaml or {profile_name}.yaml
 |
 |
 `-- definitions
       |
       `-- {definitions}
```

ツールの起動

エンジンの構成は以下の通りです。

実行結果

```
root
 |
 |-- YYYYMMDD_HHMMSS_{scenario_id}
 |      |
 |      |-- result.html 結果となるHTMLレポート
 |      `-- evidences
 |            |
 |            `-- {reference scenario_process_id}
 |                  |-- *.log
 |                  `-- etc...
```

Customize

概要

ETTTは具備している機能では足りない場合や、振る舞いを変更したい場合に
自分自身でカスタマイズすることができるようになっています。

前提

ETTTをカスタマイズするには以下の環境が整っている必要があります。

ソフトウェア	内容
Java	ETTTのエンジンはJavaで作成されています。Javaの1.8以上がインストールされており、パスが通っていることを確認してください。
Gradle	ETTTのビルドに利用します。バージョン4以降を推奨しています。

独自コマンドの作成方法

独自コマンドを作成するには、コマンド定義（Process）と実行クラス（Runner）を作成する必要があります。
コマンド定義は、YAMLに定義する内容を決めるものです。
実行クラスはコマンド定義でユーザが指定した内容に基づき処理を実行するものです。
ETTTでは、誰でも独自に作成ができるようにインターフェースやユーティリティを提供しています。