Designing a test automation framework

Before we start coding, it is important to think carefully about the underlining structure (architecture) of the target framework. This will minimize the costs of framework development and maintenance of th **automated test assets**, so that they can easily be extended to cove new requirements.

Key design activities for a framework

The key activities we need to perform in order to design a robust test automation framework may include:

- Defining a standard project (or solution) folder hierarchy.
- Defining a standard configuration and launcher script. This is typically not required when using HP ALM/QC together with UF thanks to their interoperability.
- Designing a standard format for the automation components.
- Designing reusable modules to reduce development and maintenance costs.
- Designing a layered architecture of reusable modules.
- Designing a standard flow control module.
- Designing a standard data storage, loading, and sharing mechanism.
- Designing a standard reporting mechanism.
- Designing a standard error/exception handling mechanism.

Components of a framework

The components of a test automation framework comprise an array of modules that manage different requirements for automation. The main modules include a flow controller, a reporter/logger, and an exception handler. For simplicity, data loading is implemented here using the UFT native DataTable, but of course, other options, such as XML files or a DB, can be used.

We also use a command wrapper design pattern to implement reusable runnable components, which are direct replacements for the reusable UFT actions.

We will now list out the component design patterns.

Controller

The controller supports the following requirements:

- Loading the list of reusable components (actions)
- Importing the datasheet for each action
- Running each action for the number of iterations indicated
- Invoking the event handler to check if an error was thrown and handle it as predefined

Reusable components (actions)

A reusable component supports the following requirements:

- Implementing the runnable interface (requires a `Run` method to be implemented), which will execute the actual code

- Implementing the data loading of its parameters from the DataTable, as required for each iteration
- Reporting the results of the Action, using the generic reporting mechanism

Event handler

The event handler supports the following requirements:

- Enabling the mapping of a procedure to a key (basically, an error number)
- Invoking the corresponding procedure when a key (error number) is passed
- Ensuring that the error handling procedures should be implemented as actions

Reporter

The custom reporter supports the following requirements:

- Retrieving the data for the report from the sending Action
- Invoking the UFT and passing it to the appropriate parameters

To conclude this recipe, the basic design outlined here covers the basic needs of a test automation framework, namely, flow control, reporting, data loading and sharing, and event/exception handling. This design pattern also supports keyword-/data-driven flow control, and encourages the use of highly effective object-oriented design patterns. The next sections will describe in detail how to build each of these components.