DATAMATICS®
Next-Generation Solutions

WHITEPAPER

API Testing with GreenPepper – Challenges and Best Practices

A White Paper by: Dhaval Koradia / Ashwini Khaladkar
September, 2013

Table of Contents

Introduction

Specifications based testing, in general terms, could be understood as any testing performed against a given set of rules, conditions or specifications that define how a product or a software program is supposed to work. The true value of this approach lies in the collaboration it brings between the business experts, developers and testers by enabling them to work on one common platform, the executable specifications, that are rules written in business language and also drive the system for instantly verifying its compliance.

Greenpepper is a tool that helps implement specifications based testing against a system under test. This paper will give the readers a general outlook on the various components involved in creating executable specifications using Greenpepper, how various components interact and work together to create an end-to-end testing framework, some of the challenges that Datamatics have encountered while implementing the same along with providing alternative workarounds and best practices that have benefitted Datamatics in writing, executing and maintaining the tests. The outcome of having well-built and well-structured tests, workarounds for customized requirements and efficient use of the tool have helped the customer gain confidence of the robust test framework supporting their software products.

Scope

The purpose of the paper is to bring out some of the challenges and best practices that Datamatics has experienced while working on projects that have implemented automated API testing using GreenPepper. Being a specific and a less widely-used testing tool and in view of providing a complete picture to all those who are new to the same, general chapters including the components and working of Greenpepper have also been included. However, details such as how to create each component, detailed explanations and examples of flavours of each component and configurations involved in setting up Greenpepper are not discussed.

Specifications based approach for automated API testing using Greenpepper

"Executable Specifications" is a concept in software development that tries to have human written specification documents automatically run onto the targeted system in order to instantaneously verify their compliance.[1]

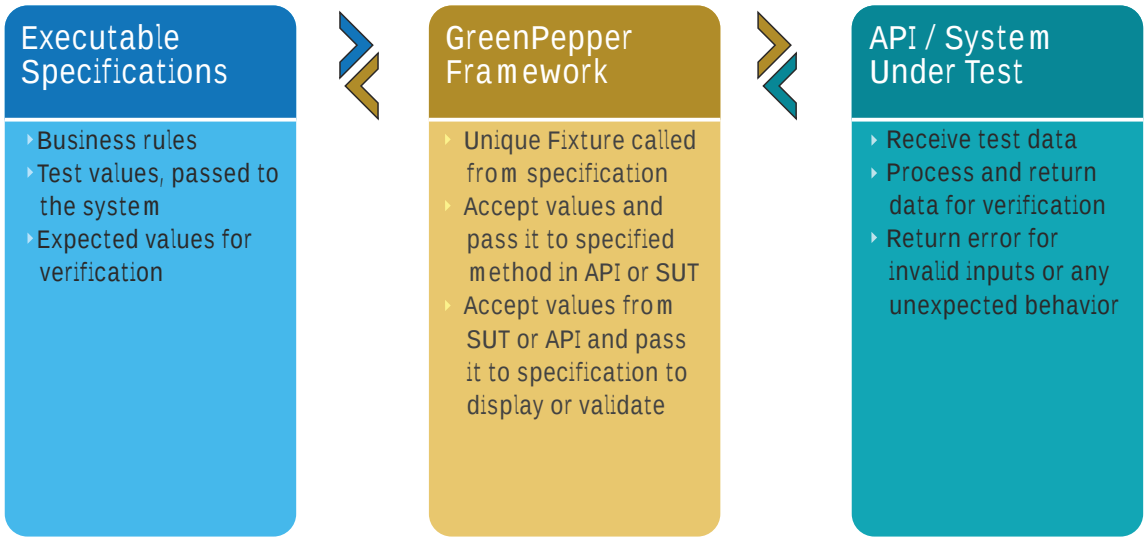
Specifications can be a set of rules, conditions or business logic that defines how a system is supposed to work. This is usually noted in business language.

Transforming this to an "Executable" specification means to somehow pass inputs from the specification itself to the system, collect the output and display it on the specification verifying if it matches the desired or the defined expectation.

This process has various benefits:

- It eliminates the need for a business expert to look at or deal with any technical code
- It enables a business expert to pass numerous inputs for any given specification, define the expected output and instantly check its compliance – all without any technical know-how or help
- Check logical validity and relationship between various business rules

The diagram below indicates how Greenpepper (a specifications-based test framework) functions at a high level.



Components of Greenpepper and how they work together to build an executable specification

Greenpepper combines its own framework along with Confluence, an enterprise WiKi, to provide a strong testing framework. Let us look at the various components involved and how they work together to build a testing framework.

Executable specifications: Greenpepper utilizes Confluence for the purpose of writing, storing, managing, and executing the executable specifications. Confluence is an enterprise Wiki enabling team collaboration.

- Confluence provides a framework to write specifications in WiKi Mark-up language. Standard language statements are written to define rules or specifications
- Confluence allows users to create a hierarchy between pages and an easy method to create, edit and delete parent-child relationships thus enabling better management of specifications
- It has a rich text editor helping the collaborators who are not familiar with WiKi Mark-up language
- Greenpepper provides custom macros and options that help execute a specific set of parent/child pages as well as execute pages against multiple System Under Test

Interpreters: Interpreters are Greenpepper implementations that define how to collect information from a specification, pass it to the fixture and display the received output. These are specified in the executable specifications. There are various Interpreters as depicted below:

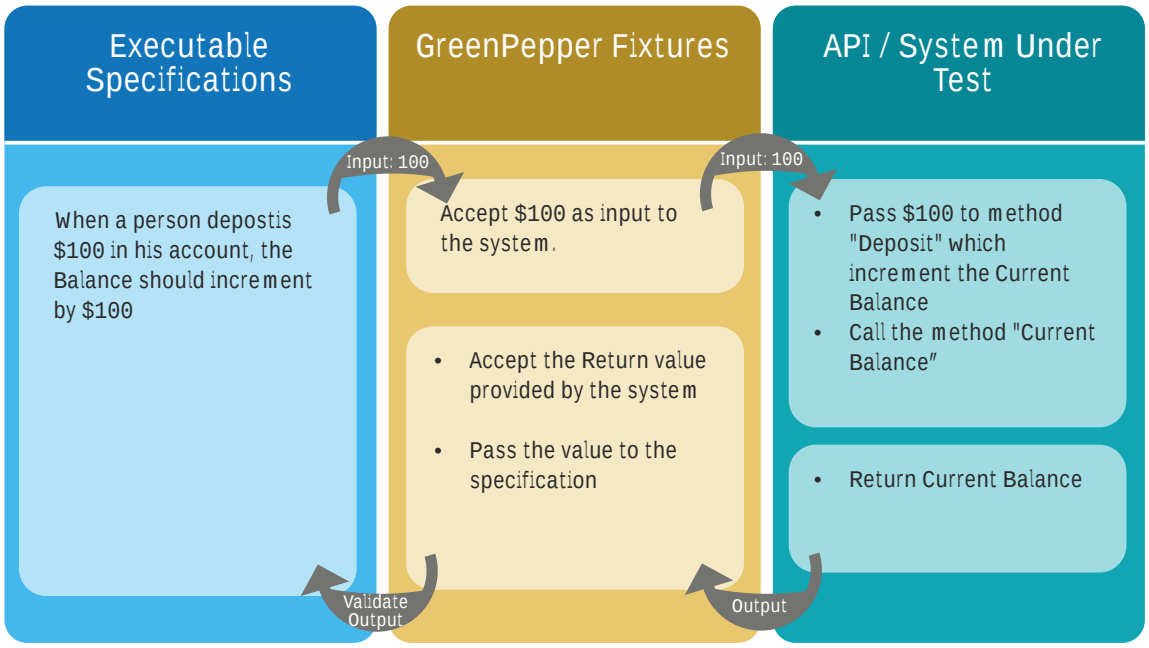
Rule for	Used to express concrete and measurable business rules. Example: Sum of x and y should be x+y
Collection Interpreters	Used to express any kind of groups, list or set of values. Example: List of saving accounts in a bank.
Workflow Validation	"Do with" or "Scenario" used to express interactions with the SUT that must be performed in a specific order thus providing information about business flows. Example: Deposit $100 in a bank account and check if the balance has increased by $100
Context Definition	Used to simplify the creation of a particular state for the SUT Example: Given the details of Bank balances across Mumbai (which serves as the required state for the SUT), generate a list of savings accounts.

Fixtures: Fixtures are the glue between the business expert examples or specifications and the software being tested. The goal of the fixture is to translate from business language to programming language and vice versa.

- Each of the above Interpreters has a corresponding Fixture
- Fixture is any class written in C Sharp or Java, as currently supported by Greenpepper
- Fixtures are a map that understand and accept data from the specifications, pass the inputs to the corresponding methods or attributes of the System under test, accept the output values received from the system and pass it on to the specification for the purpose of being displayed or verified

How all of it works together

To get a high-level understanding of how all of it works together, let us go through a simple real life scenario. Consider a banking system being tested. Let's take the example that involves deposit of a certain amount in a bank account and verifying the balance, once the amount is successfully deposited.



1. A business rule laid out by a business expert in the executable specification says "When a person deposits $100 in his account, the balance should increase by $100"
2. For simplicity, let us assume that the details required for identifying "which" account, has been provided
3. The $100 deposit mentioned in the input condition, "When a person deposits $100 in his account" is identified by the fixture as input for the method "Deposit" defined in the system
4. The system accepts this variable, as per the logic, adds it to the "Current Balance" and returns the value (output) of "Current Balance" back to the Fixture
5. The Fixture send this value to the specification where it is verified with the $100 mentioned in the validation condition "the balance should increase by $100"
6. If both values match, the success is indicated by green colour on the specification and in case of a miss-match, the specification indicates the same by red colour while also displaying the actual value returned

Challenges

Challenge # 1: Which Interpreter to use?

Greenpepper provides us with a variety of Interpreters that can be implemented. Although each interpreter has a specific purpose, it sometimes becomes a challenge to select the right one for a particular test.

Suggestion # 1:

Listed below, are some helpful hints that can help decide the right one

- Have a clear focus on the purpose of the specification that needs to be created. If the purpose is to create Unit test for attributes and methods accepting and returning simple objects, the specification will most likely be designed to test the validity with multiple input sets. In such cases, the best choice becomes a "Rule For" interpreter. Example: Input x and y and return their sum
- Input Data format is also one of the key factors in deciding the interpreter. For simple data objects, a "Rule For" or "Do With" may suffice, but for methods that require a collection of Data values to be passed on to a method, the List of the collection interpreter becomes the best choice
- Business rules requiring inputs to various methods in specific order before obtaining a value to be validated are typical examples of how Functional testing would be performed. The "Do with" or the "Scenario" interpreter becomes the best option.Functional testing would be performed. The "Do with" or the "Scenario" interpreter becomes the best option.

Challenge # 2: Data to validate is a set or an array.

Each datum to be verified is associated with a single cell in the GreenPepper specification. This is easy and convenient so long as there is one value returned by a method or an attribute of the system. However, it becomes tricky when the returned data is in the form of a Data array. There is no standard method provided by Greenpepper to deal with this and the entire content of the array would be returned in one cell. One way of dealing with this is described in short below.

Suggestion # 2:

- An additional Support fixture can be created that collects the data array returned by the system
- This fixture, if defined 'static', will retain its value for multiple rows of the "Rule For" interpreter instead of destroying it at the end of each specification row
- Each data item of the array can then be passed individually to a defined cell of the specification

Challenge # 3: Dealing with special characters that are interpreted differently by the specification

Greenpepper specifications interpret many characters differently which might either alter the returned value (or string) or may not match the expected output defined on the specification. Some known cases are highlighted below.

Suggestion # 3:

- Blank space or White space: Many a time, the system code returns values with leading or trailing spaces. However, any additional space added in the defined cell of the specification, are ignored by Greenpepper thus ensuring a mismatch between the expected and the actual result. The best way to handle this is to implement a "Trim()" method in the fixture while returning the output to Greenpepper specification. This way, without logically altering the actual value (or string) returned by the system, we can eliminate the extra spaces
- Strings containing html tags: Since Greenpepper specifications identify WiKi Mark-up language, any output string from the system containing html tags will automatically be interpreted differently. In such cases, we can either replace such elements by parenthesis, thus not altering the output logically or use escape character "/" of the markup language appropriately

Best Practices

Fixtures

Naming conventions (Fixture and method names)

- Fixture names: A good naming convention is to name a Fixture based on the functionality being tested. In the banking example used above, we could name the fixture something like CheckDeposit
- Method names: Method name is what links the executable specification to the fixture. For simple interpreters like 'Rule for', method names could be a word Sum() for a specification written to check addition or something like "theSumOfXAndYIs()" if we want to write a more expressive statement on the specification like The Sum of x and y is. The key to self-explanatory method names is to use appropriate Camel case

"Keep it simple"

- The best practice while coding the Fixtures is to keep the code within the Fixture-methods as simple as possible. What this means is to add none or the least amount of logic and let the Fixture-methods simply function as givers and takers of values to and from the Greenpepper specifications and the SUT
- The idea behind this is to build a WYSIWYG (What you see is what you get) test system which does not alter the input or output in any way capturing the actual behaviour of the SUT.

Using constructors / destructors

Using constructors and destructors in the Fixtures is an efficient way to create/destroy default objects. In GreenPepper, the object created in constructor is not destroyed and remains the same until the interpreter is ended on the specification. While for a "Rule For" interpreter, an empty line indicates end of the interpreter section, a "Do With" requires a deliberate "End" syntax

Using constructors and destructors helps control the point in the specification when the objects are to be created/destroyed.

Setup Fixture

These Fixtures are used to create a particular state for the system. The Setup Fixtures define the test data to be used, or provides data required by the objects used in the test. Once this has been created, the focus can be shifted to business processes. Implementing Setup Fixtures is a good way to ensure the system, irrelevant of its current state, is brought back to the state required to execute a given specification.

Greenpepper specifications

Use Macros

Greenpepper offers a range of macros for easier grouping and execution of specifications. The {greenpepper-children} macro can execute a batch of specifications based on space, SUT and hierarchical order among others. The {greenpepper-historic} macro creates a Chart image of the latest historic data of a page execution. The image created provides a clickable area to display the specific execution result. The {greenpepper-labels} macro can execute a batch of documents resulting from a label search.

In addition to the above three, there are many other macros offered by Greenpepper. Using the provided macro is a very convenient and a quick way to execute specifications based on some criteria and see important statistics at a glance.

Visual studio or Eclipse plugins for easier execution

Greenpepper provides a plug-in for Eclipse as well as Visual studio. Once installed and configured, one can get the entire repository view right from the IDE being worked on. The plug-in also provides the capability to execute the specifications directly from within the IDE. In addition, the Visual Studio plug-in offers an additional feature of creating a Fixture skeleton from a specification.

Using the plug-ins for the IDE being worked with, gives enhanced control on viewing and executing specifications without having to open the space on a browser.

Backups

Confluence provides a daily backup setting that backs-up the entire confluence in a zipped format. This includes Greenpepper spaces and every single specification while preserving the hierarchy. On production servers or projects where activity of test development is very high, this option can prove to be a life saver.

About the Author

Dhaval Koradia
Senior consultant for Automated testing

Dhaval S. Koradia works as a senior consultant for Automated Testing at Datamatics Global Services Limited. Dhaval has been successfully leading a testing project providing manual and automated testing solutions. With over 6 years of experience in the IT and Telecommunications industry, Dhaval has worked with various automations tools such as Selenium, QTO, GreenPepper and Watir.

Ashwini Khaladkar
Associate Consultant

Ashwini Khaladkar is an Associate Consultant in Datamatics Global Service Ltd. She is working as an Automation tester and has close to 3 years of experience in the IT industry. Ashwini is involved in various testing activities such as manual testing, regression testing, automation testing and creation of test plans among other. Having worked on Greenpepper since her initiation into the professional world, she has developed an expertise in the same. Ashwini is an ISTQB certified tester.

About Datamatics Global Services

- Global Information Technology (IT) & Knowledge Process Outsourcing (KPO) organization
- Delivers smart, next-generation business solutions
- Trusted partner to several Fortune 500 companies
- Capabilities built around technology, domain expertise & knowledge of business processes
- Featured amongst the Global Services 100 List in 2010 & 2011
- Decades of global experience having executed projects across 60 countries
- Alliances with global technology leaders such as Microsoft, IBM & EMC^2
- Certified for SEI CMMI Level 3 V1.3, ISO 27001:2005 ISO 9001:2008
- SSAE 16 compliant processes
- Global presence: U.S, UK, Germany, Switzerland, Bosnia, Australia, Singapore & India

To know more, connect with us on business@datamatics.com