

		REST-assured	Karate	References / Comments
1	BDD Syntax	Yes	Yes	
2	True DSL	No. Fluent Interface. Also IDE formatting is a challenge	Yes	DSL vs Fluent Interface . Also see (24) and (25)
3	Runs on the JVM	Yes	Yes	
4	Implementation	Java and Groovy	Java	
5	Code-base Size	Large. 46,000 lines of code (source: OpenHub)	Large. 50,000 lines of code (source: OpenHub)	Karate has API-mocks, perf-testing and UI automation as well.
6	Mature	Yes. Inception 2010. Lots of blog posts, tutorials and StackOverflow posts. In the ThoughtWorks Tech Radar at 5 years , appeared only once.	Yes. Inception February 2017. Clear signs of wide and rapid adoption . In the ThoughtWorks Tech Radar within just 2 years and rating upgraded within 1 year .	Karate has 5000 GitHub “stars” in 4 years, which took more than 10 years for REST-assured. REST-assured does not even show up on Stack Overflow trends .
7	JsonPath Implementation	Groovy GPath	JayWay JsonPath	GPath has some limitations and updates are not possible
8	XPath implementation	Groovy GPath and “XMLSlurper”. Standard XPath is also supported, but paths that return XML nodes cannot be used in assertions. Updating an XML document is not possible.	W3C standard XPath using the Java built-in XML lib. You can even update XML documents using XPath.	
9	HTTP Client	Apache 4.X, but the code depends on deprecated APIs . There are some concerns with this design . More details in this issue .	Pluggable (future-proof), you can even implement your own. Karate also has minimal maven dependencies - and the Apache libraries are “shaded” to not cause conflicts.	Karate even has an option to mock a servlet container because of this abstraction.
10	Quick Start / Project Scaffolding	No	Yes (Maven Archetype). There’s also a standalone executable and ZIP Release .	Dev onboarding experience much better with Karate. Archetype Includes working example.
11	Test-Scripting Language	Java	Karate-Script (Cucumber / Gherkin) Java is also supported in version 1.0 onwards.	No Java knowledge needed for Karate
12	Test Scripts have to be compiled	Yes	No	Tests are plain-text. No IDE required for Karate
13	IDE Support	Yes. Intelli-Sense, Auto-Complete and Refactoring work for Java and POJO-s	Partial. Eclipse and IntelliJ have Cucumber plugins that work well and have pretty good syntax coloring. Karate has a Java API option in version 1.0 onwards.	Since you can re-use JSON payloads across tests, the “re-factorability” aspect is covered as well.
14	Step Through / Debug-ability	Yes. Java + IDE Support.	Even better ! Debug in Visual Studio Code with step-back and even hot-reload (since v0.9.5). also see the Dev-mode HTML report : steps, error diagnostics and HTTP logs in-line, and there is a Java API.	Also see (42)
15	Test Runner	Any, bring your own. TestNG or JUnit will work.	JUnit is supported and can co-exist with TestNG in the same project if needed. Even JUnit is optional	And Karate’s parallel execution capability is in “core”, independent of even Maven or any unit-testing framework.
16	Tags / Groups Built In	No (have to use TestNG or equivalent)	Yes	
17	Extend with custom routines via...	Java code	JavaScript	No need to compile, and easier for non-programmers.
18	Re-use Java code	Yes	Yes (via JavaScript interop)	
19	Validate All Payload values in one step	Not built-in. You <i>need</i> to use external libraries such as Hamcrest. <ul style="list-style-type: none"> No easy way to do a “deep equals” comparison for nested objects. No way to “ignore” fields - for e.g. id / date / time values which are dynamic 	Karate natively supports a “ <i>deep equals</i> ” and “ <i>contains in any order</i> ” assertion for JSON, JSON arrays and XML, <i>and</i> lets you ignore chosen fields at the same time. This is super-important for GraphQL .	IMO the biggest limitation of REST-Assured: <ul style="list-style-type: none"> Example 1 Example 2 Example 3 Example 4
20	Built-in data-type, conditional-logic and RegEx validations	No	Yes, includes RegEx and Macros	
21	Validate schema of all elements in a JSON array in one step	No	Yes	

22	Built-in JSON Schema and XML Schema validation support	Yes	RegEx and Macros support is sufficient (and far simpler) for most use cases. That said, users can easily add a Java lib via Karate's Java interop - if needed.	For details on how Karate's approach is simpler and more intuitive than JSON (or XML) Schema see this link .
23	Native support for expressing JSON or XML in test-scripts	No <pre>{ "name": "Billie" }</pre> <pre><cat name="Billie"></cat></pre>	Yes <pre>{ name: 'Billie' }</pre> <pre><cat name="Billie"></cat></pre>	No need to use double-quotes or "escape" characters. You can also read from files and re-use.
24	Example – JSON assertions	<pre>@Test public void lotto_resource_returns_200_with_expected_id_and_winners() { when(). get("/lotto/{id}", 5). then(). statusCode(200). body("lotto.lottoId", equalTo(5), "lotto.winners.winnerId", containsOnly(23, 54)); }</pre>	Scenario: lotto resource returns 200 with expected id and winners <pre>Given path 'lotto', 5 When method get Then status 200 And match \$.lotto.lottoId == 5 And match \$.lotto.winners[*].winnerId contains only [23, 54]</pre>	Matching built-in, and more readable syntax. Note the simpler way to specify path parameters without placeholders. For REST-assured, IDE formatting is a known challenge .
25	Example - GET with params	<pre>given(). param("key1", "value1"). param("key2", "value2"). when(). get("/somewhere"). then(). body(containsString("OK"));</pre>	<pre>Given param key1 = 'value1' And param key2 = 'value2' And path 'somewhere' When method get Then match response contains 'OK'</pre>	Karate is a true DSL . No syntax "noise", no unnecessary symbols or punctuation. No need to worry about indenting a giant "one liner" of Java code.
26	Extracting multiple data-elements for reuse in subsequent HTTP calls	Convolved. The Fluent Interface which is supposed to be the main highlight of REST-Assured actually gets in the way here. More examples .	Easy. You can even use JsonPath to extract JSON chunks or arrays and save them to variables for use in later steps. For XML, XPath does the same.	Some of the quirks of the REST-assured JsonPath implementation get in the way as well.
27	Can update a given JSON or XML using a path expression	No . Especially for data-driven tests, updating nested JSON is near impossible .	Yes. There are actually multiple ways to update payloads: a) by path b) using embedded expressions and c) via a built-in string replacement keyword.	You can even modify a response and re-use it 'as-is' as the next request.
28	Data Driven Testing	No (have to use TestNG or equivalent) REST-Assured Example	Yes. Can even use dynamic JSON or even CSV as a data-source. Karate Example	
29	SOAP support	No	Yes	Plus, Karate's XML support is far more flexible and easier to use .
30	HTTPS / SSL without certificates	Although there is " relaxed " HTTPS, a certificate is needed in some cases	Yes	
31	Built-in support for switching environment config	No Also config is somewhat convoluted in REST-Assured	Yes. Adding a new variable to a test is just one step: edit karate-config.js	In REST-assured, you have to use something like a dependency-injection framework (or roll your own) to read properties files.
32	File Upload / Multipart Support	Partial / Buggy Libraries Content-Type Dependencies 'multipart/related' not supported questions on 'multipart/mixed'	Yes	
33	URL encoded HTML Form data	Yes	Yes	
34	Cookies	Some Limitations	Yes	Can be configured one-time for all subsequent requests, just like headers.
35	Auth Schemes out of the box	Yes	No (but easily pluggable via re-usable scripts or JavaScript without needing to write Java code)	
36	Custom Auth	Java code (needs compilation). Existing mechanism is not extensible .	Unified plug-in system via JavaScript (no compilation needed)	
37	Parallel Execution of Tests	No. While some teams seem to have had success running REST-assured in parallel, there are some cases in which multi-threading is not supported. Also see this thread . The creator has also confirmed that " REST Assured is not 100% safe to use in multi-threaded scenarios ".	Yes Even if you run tests in parallel, reports and logs are collected per test and HTML reporting works as you would expect.	This is a critical requirement for HTTP integration tests which typically take a much longer time than unit tests.
38	Floating-point precision	All numbers are converted to float and you shouldn't forget to use floats (not the default double) in assertions. <pre>get("/odd") .then().assertThat() .body("odd.ck", equalTo(12.2f));</pre>	Numeric assertions work just as you expect and even auto-conversion to BigDecimal happens if needed. <pre>Given path 'odd' When method get Then \$.odd contains { ck: 12.2 }</pre>	Even this works: <pre>And \$.odd.ck == 12.200000000000000</pre>
39	Lines of Code Needed to express a test	More. By nature, Java is verbose and especially if you depend on POJO	Less.	Another example of how Java "gets in the way" - the

		representations of payloads - you need more Java code in place.	This particular comparison shows a dramatic difference, 431 lines of code reduced to 67	contortions you need to do to handle JSON arrays in REST-assured.
40	Test Reports Built-in	No, you have to use JUnit, TestNG or equivalent for test reporting.	Karate has text and HTML reports out of the box and you get the option of choosing from the Cucumber ecosystem of 3rd party reports.	Here is an example of the very nice-looking reports you can get by using the cucumber-reporting library.
41	Test any Java servlet or HTTP resource without a container	REST-assured has support for “out-of-container” testing of specifically Spring-MVC but your tests will be “hard-coded” in this mode. There is no support for things like JAX-RS or custom servlets or controllers - and for these you have to deploy to an app-server.	Karate v0.5.0 onwards has support for testing any servlet by providing extension points for teams to write an adapter. The huge advantage of Karate’s approach is that the same test-script can be re-used for http-integration tests without changes.	This is possible because of Karate’s pluggable abstraction of the HTTP Client. Refer to the documentation for more details. You will be able to quickly implement a custom adapter for any Java server-side stack in a similar way.
42	Report includes HTTP request and response logs in-line	No.	Karate includes HTTP request and response logs in the JSON report output. If you use the print keyword, the console output appears in the report as well, which is great for troubleshooting. All this works even when tests are run in parallel.	
43	Construct JSON or XML from scratch using just path expressions	No.	Best explained via some examples .	
44	Test Doubles or Mocks built-in	No. You have to use 3rd party frameworks such as Wiremock	Karate 0.7.0 onwards has support for creating API mocks that can fully simulate stateful CRUD. Having both a client and server in the same unified framework keeps things simple and you can move fast.	Karate is a superior alternative to Wiremock, here’s a comparison . Also see this perf benchmark .
45	Performance Testing	No. Also see [37]	You can re-use Karate tests as Gatling performance tests . You can compose multiple Karate feature files or “work-flows” into a single performance-test and use Gatling to define the load-model (ramp-up, concurrent users, etc)	This alone is reason to choose Karate and no other open-source test-automation framework has this option. In 0.9.0 onwards you can test any Java code , not just HTTP
46	Websockets support	No .	Karate 0.9.0 onwards has support for websockets and even generic async / await support (see below)	
47	Async Support	No, you have to use Java code or a library like Awaitility	Karate 0.9.0 has built-in support via a very elegant API .	Also see https://twitter.com/KarateDSL/status/1417023536082812935
48	Retry Support	No, you have to write custom Java logic.	Karate 0.9.0 onwards has this built-in - where you only specify the condition to be polled for.	
49	CSV file support	No, you have to use Java code or a library.	In Karate 0.9.0 you can read a CSV file and it will be auto-converted to JSON	Karate also has YAML file support out of the box.
50	Web-UI Automation	No.	In Karate 0.9.5 onwards Web-UI automation is possible.	Chrome / CDP, WebDriver, Appium and Playwright supported.
51	Distributed Testing	No.	In Karate 0.9.5 onwards, there is built-in support to split a test-suite and orchestrate the execution across multiple cloud or hardware / cluster nodes	