Types of Dynamic Requests

Cookie Correlation

Example. A public website uses a session cookie to maintain state integrity for every visitor. The server issues a unique cookie to every user on the first visit. A cookie is stored in the user's browser and is included in requests on subsequent visits. If cookie handling is disabled in the browser, the website might not work correctly.

Load testing requirements. To avoid request failures, a recorded cookie must be replaced by a unique value generated by the server for every VU. Cookie persistence should be provided for the duration of the test for every VU. StresStimulus handles cookie correlation automatically, regardless of the webserver platform, to ensure that every VU adheres to its client session. For example, it automatically feels the JSESSIONID cookie in Java applications and ASP. NET_SessionId cookie in ASP.net applications.

Autocorrelation

Example. A stateless Webserver sends the client a dynamic value representing the application state. This value is included in the response body as a hidden web form field. On the subsequent request, the server expects to receive the same value to maintain application integrity from this client. The application state can change from request to request.

Load testing requirements. During load test execution, reissuing requests with recorded application states can result in server errors. To avoid these errors, every request from a virtual user must include the dynamic application state that the server provided. To achieve that, the dynamic value must be identified, extracted from the responses, and then used to replace the recorded value in the subsequent requests. This process is called correlation. StresStimulus handles correlation automatically for all web application platforms. For example, it automatically correlates __VIEWSTATE, and __REQUESTDIGEST hidden web form fields for ASP.NET and SharePoint applications, respectively. Autocorrelation supports various types of web sessions, such as HTML and AJAX pages. During test execution, autocorrelation rules automatically modify requests to make VUs compatible with dynamic web applications.

Note: Internally, each autocorrelation rule is implemented as a hidden unnamed response extractor and a parameter that uses the extractor to substitute a recorded value in the subsequent request. The extractor is hidden, but the parameter appears on the Test Case Tree. Though it is not recommended, you can delete autocorrelation rules.

Parameterization with Server Data

Example 1. A secure Web application requires users to login. The Webserver creates an authenticated session for every user, generates a unique token, and sends it back to the browser in the response body. On every subsequent request, the browser sends the token to identify the user and its session.

Load testing requirements. Reissuing requests with the recorded session ID during the load test execution will result in server errors. To avoid these errors, every request from a virtual user must include a dynamic token that this virtual user received from the server. To achieve that, the dynamic tokens must be identified, extracted from the responses, and used to replace the recorded values in the subsequent requests.

Example 2. A web application maintains application-specific dynamic values that are generated on the server and are included in some of the requests. Parameterizing such requests is more difficult than in the previous example.

Several factors listed below complicate finding the dynamic values and the dynamic requests:

- A field name in the response and request are different.
- HTTP messages are not in the name/ value format, and the dynamic value is "buried" inside JSON, XML, and proprietary or even binary data stream.
- A value is received in one part of the response (e.g., the header) and submitted in another part of the request (e.g., the query string).
- A value is received from the server, stored in the browser, and after a substantial delay during which many unrelated HTTP roundtrips are performed, it is submitted back to the server.

Load testing requirements. The process of finding and matching dynamic fields in the recorded requests to responses and creating parameterization rules by injecting proper dynamic values in the proper request, is called parameterization. StresStimulus supports manual and automatic parameterization:

- **Manual parameterization:** the Extractors and Parameters are created manually.
- **Automatic parameterization.** Three tools are used to automatically create Extractors and Parameters: Parameter Finder, Parameter Creator, and Auto-Configurator.

Parameterization with User Data

Example. A test scenario includes a search page which submits a user's search text. When running a load test, the server receives the same recorded search text for all VUs. Due to server caching, the server's response will be substantially accelerated because a search result will be retrieved from memory without accessing the database. As a result, performance testing will be inaccurate.

Load testing requirements. For realistic performance testing, multiple sets of user data should be used during the replay to properly emulate multiuser data entry. Such dynamic sets of data can be either predefined or dynamically created on-the-fly.

- To maintain predefined business data StresStimulus uses Datasets. Any external tabular data stored in .csv format can be added to a test. A dataset field can be bound to the appropriate request parameter.

- To generate data in real-time during test execution, StresStimulus uses Data Generators and Functions. All major data types can be generated.

StresStimulus supports all listed types of dynamic requests and provides increased performance testing accuracy as well as saves time on test configuration.