ROOKOUT

5 biggest changes you need to know about

This article discusses 5 new UI/UX changes within Rookout that can add value to your debugging process today.

Debug Sessions

We've recently made a change to the Rookout UI which aims to improve the user experience around debugging workflows. With Rookout, if you want to debug your application, you'll of course first and foremost need to install the Rookout SDK. From there, you'll need to do two things:

1. Connect your source code repo via a Git based repo or a locally cloned repo
2. Select the labels or filters that you've defined for the services or application instances that you're interested in debugging

Once you've completed the above you're ready to start setting non-breaking breakpoints. Now, what if you want to save your debugging configuration for later use? The new style debugging sessions allow you to set bookmarks capturing the source code repo and set of filters which have been applied so that at a later time you can simply choose a saved bookmark to be set up to debug using the same configuration.

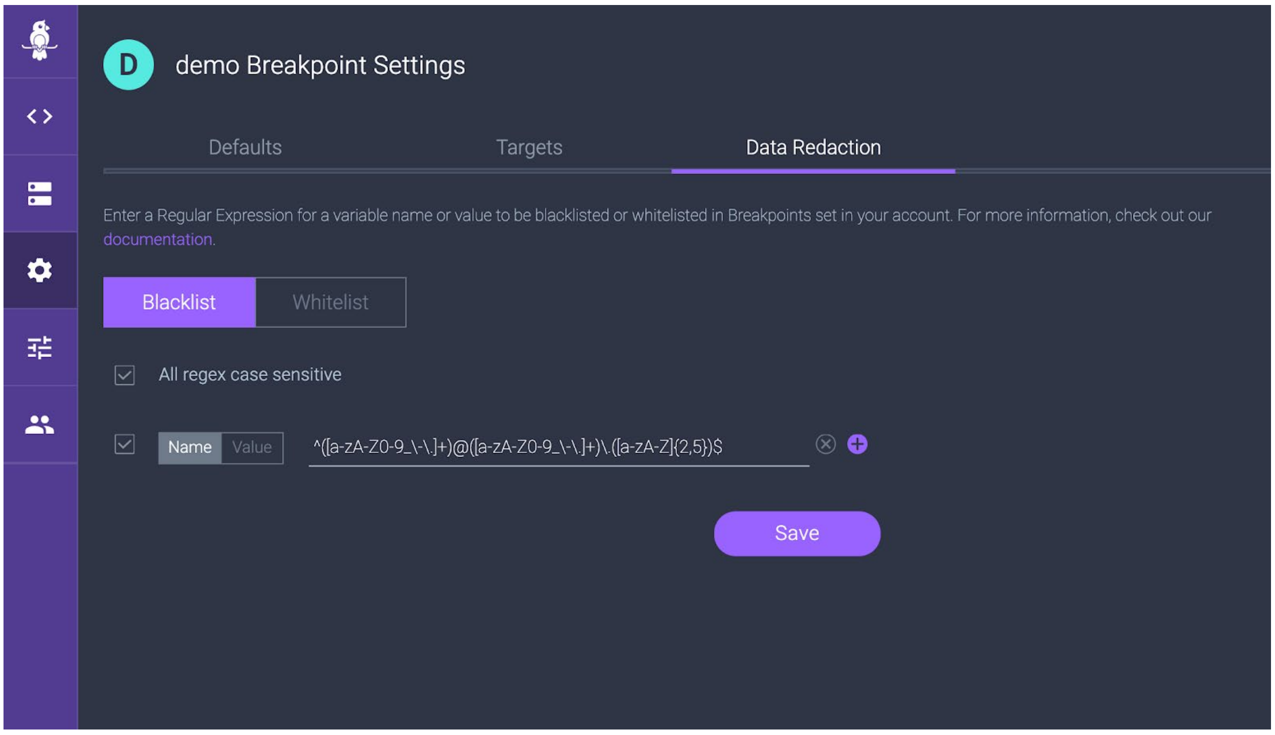
One very cool trick is that if you set ROOKOUT_COMMIT and ROOKOUT_REMOTE_ORIGIN environment variables when you install the SDK, Rookout will automatically connect to the right source code repo with the correct commit for you. You'll see the phrase 'auto loaded' in parenthesis next to these repo's in Rookout (see here for more details).

Check out this tutorial for more information: Rookout Debug Sessions Tutorial

PII Data Redaction

When you're dealing with application data today, data security including protecting things like PII (personally identifiable information) is critically important. For that reason, Rookout has invested in making data redaction / data masking as simple and easy to configure as possible.

The new Rookout data redaction UI allows users to blacklist specific variable values based on the variable name or by value using a regular expression. Alternatively, users can whitelist specific data fields which will redact all other data values other than the ones specified.



Enter a Regular Expression for a variable name or value to be blacklisted or whitelisted in Breakpoints set in your account. For more information, check out our documentation.

✓ All regex case sensitive

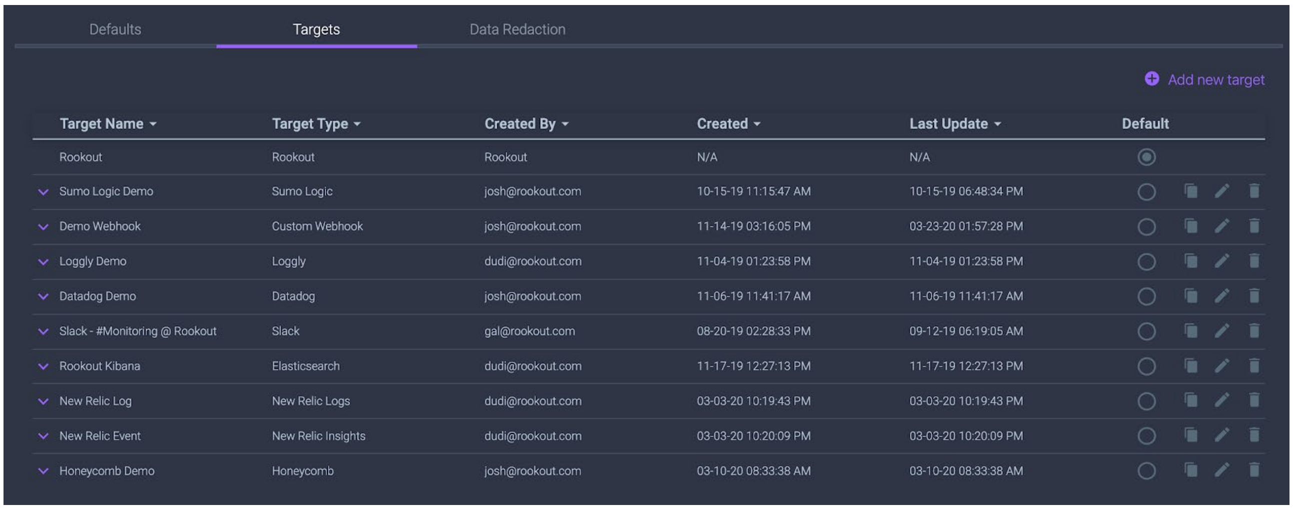
✓ Name: `^([a-zA-Z0-9_\-\.]+)@([a-zA-Z0-9_\-\.]+)\.([a-zA-Z]{2,5})$`

Data redaction capabilities combined with Rookout's hybrid on-prem deployment model allow organizations to keep all of their data on prem and ensure no sensitive data is exposed to parties who shouldn't have access.

Check out this video for more information: [Configuring PII Data Redaction with Rookout]

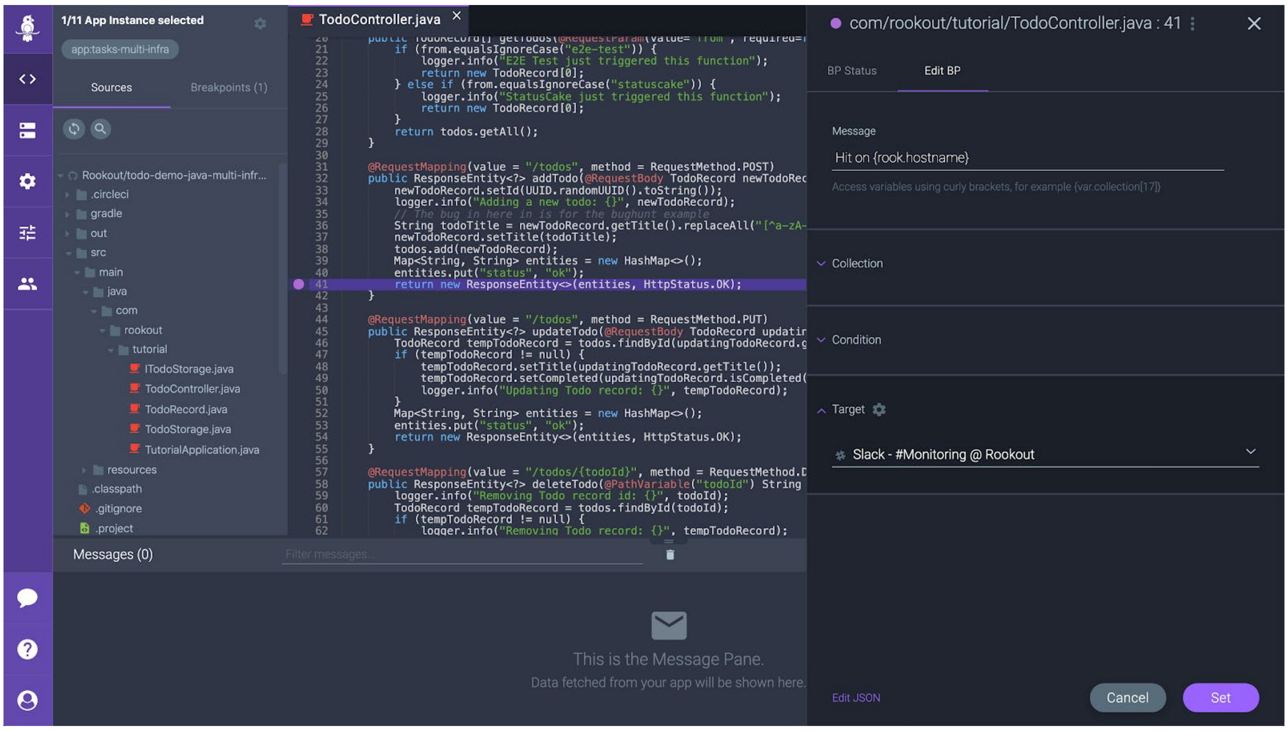
Targets

When using Rookout, many organizations have asked for a simple and easy approach to integrating with their existing logging and monitoring tool sets. Rookout's new UI makes this easy by allowing users to add new data target configurations in order to send collected snapshot data to an external system of their choice. This allows for seamless integration with external systems such as Slack, Elasticsearch, Logz.io, and many more. There is even support for a custom webhook which will send a JSON payload to a REST API endpoint chosen by the user.



Target Name	Target Type	Created By	Created	Last Update	Default
Rookout	Rookout	Rookout	N/A	N/A	◉
Sumo Logic Demo	Sumo Logic	josh@rookout.com	10-15-19 11:15:47 AM	10-15-19 06:48:34 PM	○
Demo Webhook	Custom Webhook	josh@rookout.com	11-14-19 03:16:05 PM	03-23-20 01:57:28 PM	○
Loggly Demo	Loggly	dudi@rookout.com	11-04-19 01:23:58 PM	11-04-19 01:23:58 PM	○
Datadog Demo	Datadog	josh@rookout.com	11-06-19 11:41:17 AM	11-06-19 11:41:17 AM	○
Slack - #Monitoring @ Rookout	Slack	gal@rookout.com	08-20-19 02:28:33 PM	09-12-19 06:19:05 AM	○
Rookout Kibana	Elasticsearch	dudi@rookout.com	11-17-19 12:27:13 PM	11-17-19 12:27:13 PM	○
New Relic Log	New Relic Logs	dudi@rookout.com	03-03-20 10:19:43 PM	03-03-20 10:19:43 PM	○
New Relic Event	New Relic Insights	dudi@rookout.com	03-03-20 10:20:09 PM	03-03-20 10:20:09 PM	○
Honeycomb Demo	Honeycomb	josh@rookout.com	03-10-20 08:33:38 AM	03-10-20 08:33:38 AM	○

After configuring the data target from the Targets screen, users can configure breakpoints to send data to any one of the configured targets. Simply right click on a breakpoint and select "Edit" to edit the breakpoint configuration and choose any configured target from the Target section and Rookout will automatically send collected snapshot data to that system.



```
    public TodoRecord[] getTodos(@RequestParam(value= "from", required=
        if (from.equalsIgnoreCase("e2e-test")) {
            logger.info("E2E Test just triggered this function");
            return new TodoRecord[0];
        } else if (from.equalsIgnoreCase("statuscake")) {
            logger.info("StatusCake just triggered this function");
            return new TodoRecord[0];
        }
        return todos.getAll();
    }

    @RequestMapping(value = "/todos", method = RequestMethod.POST)
    public ResponseEntity<?> addTodo(@RequestBody TodoRecord newTodoRec
        newTodoRecord.setId(UUID.randomUUID().toString());
        logger.info("Adding a new todo: {}", newTodoRecord);
        // The bug in here in is for the bughunt example
        String todoTitle = newTodoRecord.getTitle().replaceAll("[^a-zA-
        newTodoRecord.setTitle(todoTitle);
        todos.add(newTodoRecord);
        Map<String, String> entities = new HashMap<>();
        entities.put("status", "ok");
        return new ResponseEntity<>(entities, HttpStatus.OK);
    }

    @RequestMapping(value = "/todos", method = RequestMethod.PUT)
    public ResponseEntity<?> updateTodo(@RequestBody TodoRecord updatir
        TodoRecord tempTodoRecord = todos.findById(updatingTodoRecord.g
        if (tempTodoRecord != null) {
            tempTodoRecord.setTitle(updatingTodoRecord.getTitle());
            tempTodoRecord.setCompleted(updatingTodoRecord.isCompleted(
            logger.info("Updating Todo record: {}", tempTodoRecord);
        }
        Map<String, String> entities = new HashMap<>();
        entities.put("status", "ok");
        return new ResponseEntity<>(entities, HttpStatus.OK);
    }

    @RequestMapping(value = "/todos/{todoId}", method = RequestMethod.D
    public ResponseEntity<?> deleteTodo(@PathVariable("todoId") String
        logger.info("Removing Todo record id: {}", todoId);
        TodoRecord tempTodoRecord = todos.findById(todoId);
        if (tempTodoRecord != null) {
            logger.info("Removing Todo record: {}", tempTodoRecord);
```

Message: Hit on {rook.hostname}

Access variables using curly brackets, for example {var.collection[17]}

Target: Slack - #Monitoring @ Rookout

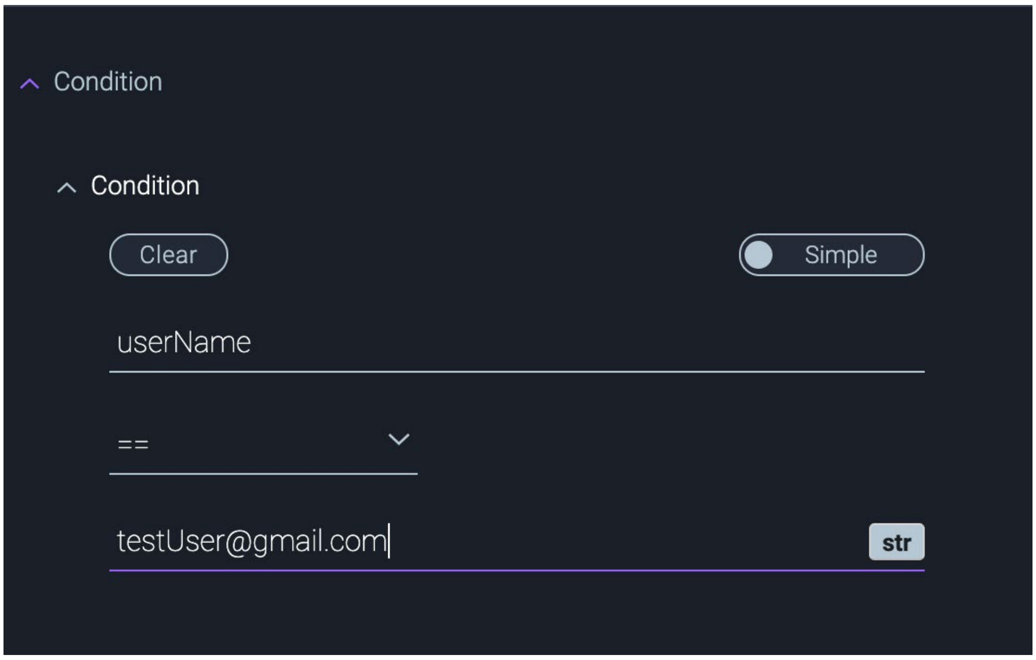
Check out this video for more information: Setting Up a Logging Target with Rookout

Conditional Breakpoints

When debugging a running application there can be large amounts of data flowing through the system at any given time. This can make it challenging to find the data that you're looking for amongst all the noise. Enter conditional breakpoints!
With conditional breakpoints, it's possible to limit the data collected by a non-breaking breakpoint by defining a condition based on the value of one or more variables. There are two types of conditions that can be defined:

1. Simple: this type compares the value of one or two variables using comparison operators
2. Advanced: this type allows you to define complex conditions that can be concatenated together using logical operations such as '&&' or '||'

This can be especially useful in cases where you are trying to track down data for a specific user or session.



userName

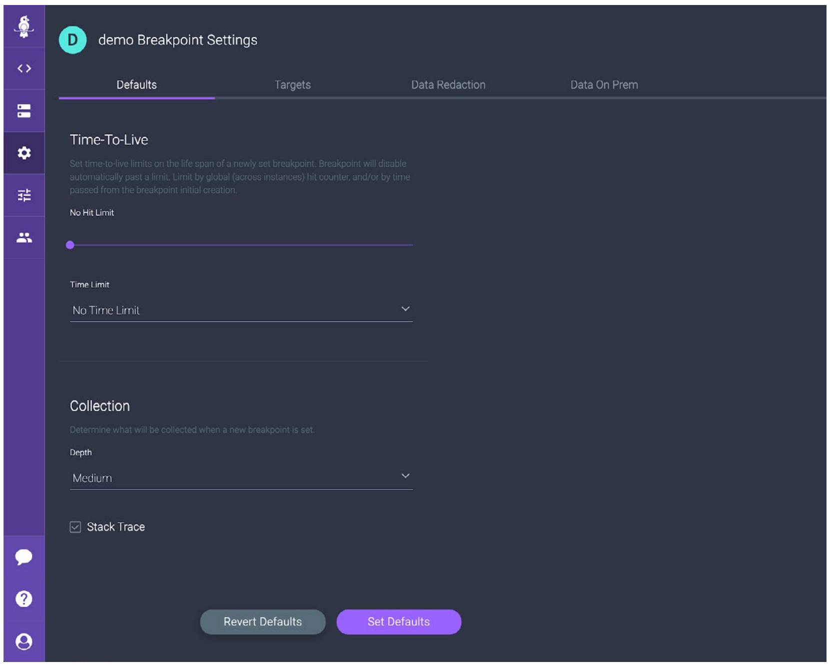
==

testUser@gmail.com

Check out this video for more information: Conditional Breakpoints

Breakpoint Defaults

When using Rookout, you have the ability to set non-breaking breakpoints in your running application and then configure properties for each of those breakpoints such as conditionals, time-to-live, as well as the collection depth of snapshots. The Rookout team has recently introduced the concept of Breakpoint Defaults, which allow you to configure default parameters for newly created breakpoints.



demo Breakpoint Settings

Time-To-Live

Set time-to-live limits on the life span of a newly set breakpoint. Breakpoint will disable automatically past a limit. Limit by global (across instances) hit counter, and/or by time passed from the breakpoint initial creation.

No Hit Limit

Time Limit: No Time Limit

Collection

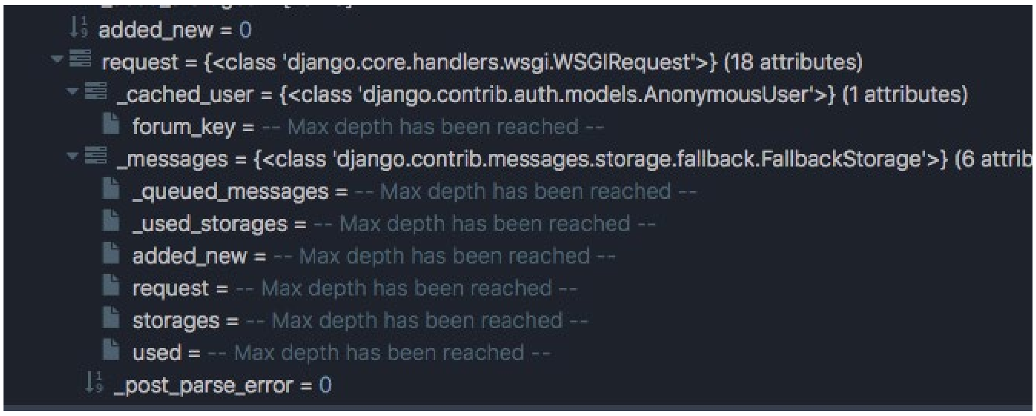
Determine what will be collected when a new breakpoint is set.

Depth: Medium

[✓] Stack Trace

For example, let's say you wanted to collect a maximum of 10 snapshots from your application for every newly created breakpoint. The time-to-live section of the breakpoints default would allow you to configure this from one place to ensure you only collect 10 snapshots before automatically disabling the breakpoint.

There is also a default section for the snapshot collection depth which allows you define how deep in the stack Rookout will go in resolving variables. This gives you finer grained control over the performance of Rookout by allowing you to collect more or less data from your application depending on your needs. The screenshot below shows what you will see from Rookout when the maximum collection depth has been reached.



```
added_new = 0
request = {<class 'django.core.handlers.wsgi.WSGIRequest'>} (18 attributes)
  _cached_user = {<class 'django.contrib.auth.models.AnonymousUser'>} (1 attributes)
    forum_key = -- Max depth has been reached --
  _messages = {<class 'django.contrib.messages.storage.fallback.FallbackStorage'>} (6 attrib
    _queued_messages = -- Max depth has been reached --
    _used_storages = -- Max depth has been reached --
    added_new = -- Max depth has been reached --
    request = -- Max depth has been reached --
    storages = -- Max depth has been reached --
    used = -- Max depth has been reached --
  _post_parse_error = 0
```

*Note: in addition to setting the collection depth, you can also right click on individual variables and choose "Collect Variable" to ask Rookout to collect specific variables if you find that they are not collected.