Sentry

What is it, how do you use it, and why should you care?

Sentry is an exception tracker

whats that?

The pitch: Sentry allows find and fix bugs in production before our users report them.

or as Sentry helpfully puts it...

Error monitoring more helpful than

“Your sh*t is broken”

Sentry gives us tools for tracking and resolving uncaught exceptions

you know, these:

airbnb

Shoot!

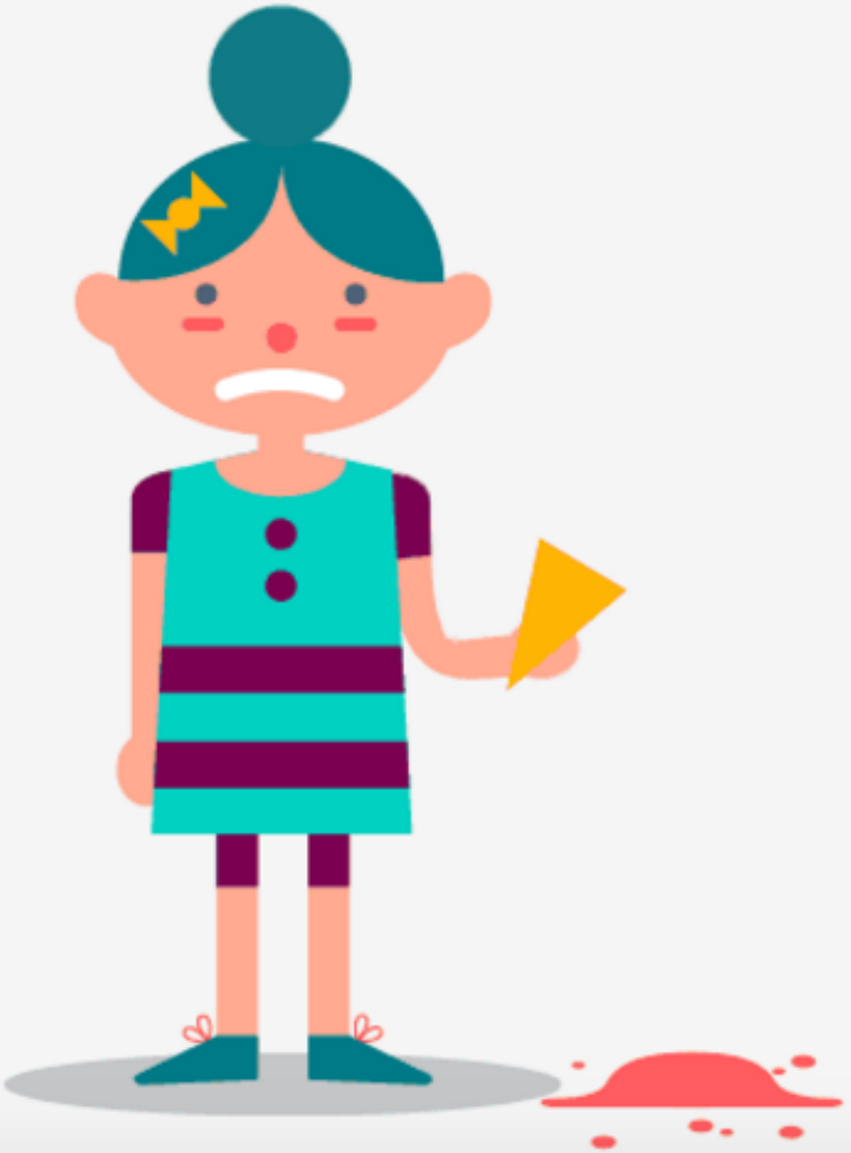
Well, this is unexpected...

Error code: 500

An error has occurred and we're working to fix the problem! We'll be up and running shortly.

If you need immediate help from our customer service team about an ongoing reservation, please call us. If it isn't an urgent matter, please visit our Help Center for additional information. Thanks for your patience!

For urgent situations please call us



Internal Server Error

The page you requested generated an unexpected error on this server.

- **Verify the address** you entered in your web browser is valid.
- If you are sure that the address is correct but this page is still displayed **contact your TAO administrator**.

Go Back | TAO Home

500

Sorry, It's not you.It's us

We're experencing an internal server problem.
Please try again later

Isn't that what the logs are for?

SENTRY

Sentry vs Logging

Two halves make one whole

Isn't Sentry just logging?

Logging provides you with a trail of events. Often those events are errors, but many times they're simply informational. Sentry is fundamentally different because we focus on exceptions and application crashes. Sentry does not replace the need for those logs, and it's also not a destination for things that aren't actionable errors or crashes.

-Sentry docs

Logs inform us about expected **events**

Sentry informs us about
unexpected **events**

Logs inform us about expected **events**

```
Logger::debug("Searching for user")

$user = User::findBy($id);

if ($user == null) {
  Logger::warning("User not found!");
}
```

Logs inform us about expected **events**

```
Logger::info("Searching for user");

try {
  $user = User::findBy($id);
} catch (NotFoundException $e) {
  Logger::error("User not found!", ["exception" => $e]);
}
```

But what about unexpected **events?**

```
try {
  $user = User::findBy($id);
  // throws ArgumentException:
  // expected $id to be of type integer, got string

} catch (NotFoundException $e) {
  Logger::error("User not found!", ["exception" => $e]);
}
```

Where are your logs now?

Easy peasy!

```
try {

  try {
    $user = User::findBy($id);
  } catch (NotFoundException $e) {
    Logger::error("User not found!", ["exception" => $e]);
  }

} catch (Exception $e) {
  Logger::error("Unexpected error!!", ["exception" => $e]);
}
```

How long will it be until you know about the error?

Not all bugs are exceptions,
but most uncaught exceptions are bugs

Uncaught exceptions are really, really bad**.**

— Users seeing 500 pages
— Data not loading
— Background processes failing to do their work
— It's basically your program giving up :(

Sentry helps us answer important questions about bugs caused by exceptions:

— When was it introduced?
— How many users have experienced it?
— What was the user doing when they experienced it?

Sentry notifies us about exceptions, and creates a workflow for resolving them:

— Actively notify after certain number of occurences/ number of users are affected.

— Ignore potentially unactionable exceptions until they become larger issues.

— Resolve exceptions, track regressions, and view new issues in a release.

Sentry Main Concepts

Projects

Project: Sentry's idea of an application. Hosted, DeepData, iOS CRM, ember-app.

Projects are identifered by a DSN:

```
https://4720d02296f74f2492978ba6bdd09929@sentry.io/133474
```

Issues

Issue: A class of exceptions within a project, specific to a single location in the code.

Events

Event: A single instance of an issue. Sentry will intelligently group events into issues.

Context

Context: Extra data associated with an event (request parameters, browser, logged in user).

Integrating Sentry into an application

demo

Install

```
# Pipfile/requirements.txt

[packages]
flask = "*"
flask_sqlalchemy = "*"
sentry-sdk = "*"
"sentry-sdk[flask]" = "*"
```

Import the SDK

```
# app.py

import sentry_sdk
sentry_sdk.init(
  "https://40c3c5d83e5a46199f063c0ac2b5d200@sentry.io/1390866"
)
```

Catch exceptions in web requests

```
@app.errorhandler(Exception)
def handle_exception(e):
    sentry_sdk.capture_exception(e)
    raise e
```

Shortcut with the Flask SDK

```
from sentry_sdk.integrations.flask import FlaskIntegration

sentry_sdk.init(
    dsn="https://40c3c5d83e5a46199f063c0ac2b5d200@sentry.io/1390866",
    integrations=[FlaskIntegration()],
)
```

Adding user context

```
@app.before_request
def setup_setry_context():
    with sentry_sdk.configure_scope() as scope:
        if 'user' in session:
            scope.user = {"id" : session['user']}
```

Environment

```
sentry_sdk.init(
    dsn="https://40c3c5d83e5a46199f063c0ac2b5d200@sentry.io/1390866",
    environment=app.config["ENV"],
    integrations=[FlaskIntegration()],
)
```

Callbacks

```
def before_sentry_send(event, hint):
    if app.config["ENV"] == "development":
        return None

    Metrics.increment("exceptions.count")
    return event
```

Callbacks

```
sentry_sdk.init(
    dsn="https://40c3c5d83e5a46199f063c0ac2b5d200@sentry.io/1390866",
    environment=app.config["ENV"],
    integrations=[FlaskIntegration()],
    before_send=before_sentry_send
)
```

Ignore Exceptions

```
import werkzeug.exceptions

sentry_sdk.init(
    dsn="https://40c3c5d83e5a46199f063c0ac2b5d200@sentry.io/1390866",
    environment=app.config["ENV"],
    integrations=[FlaskIntegration()],
    before_send=before_sentry_send
    ignore_errors=[werkzeug.exceptions.NotFound],
)
```

Capturing Exceptions outside of HTTP requests

```
# cron job that runs daily

for todo in todos:
    try:
      if todo.text == "Can't delete me!":
          raise PermissionError("Couldn't delete todo!")

      todo.delete()
    except Exception as e:
      Logger.info("Encountered error!")
```

Capture those errors!

```
for todo in todos:
  try:
      if todo.text == "Can't delete me!":
          raise Exception("Couldn't delete todo!")

      todo.delete()

  except Exception as e:
      # Capture the exception, but keep the job moving
      print("Encountered exception")
      capture_exception(e)
```

Custom Tags

```
# clear_todos.py

with sentry_sdk.configure_scope() as scope:
    scope.set_tag("cron-job", "clear_todos")

for todo in todos:
  ...
```

Searchable in Sentry as `cron-job-name:clear_todos`.
Indexed!

Custom Context

```
with sentry_sdk.configure_scope() as scope:
    scope.set_tag("cron-job", "clear_todos")
    scope.set_extra("todo.id", todo.id)
    scope.set_extra("todo.text", todo.text)
```

Sentry and you

How can you incorporate Sentry into your team's workflow?

Exception tracking should be set up for every new app before it's first production deploy.

Step 1: Create a project for your team's applications

Maybe ask an admin for help.

Step 2: Integrate the SDK, make sure it's capturing everywhere

Test it in prod and staging, disable it in development.

Step 3: Ignore Unactionable Exceptions

Otherwise they'll eat into our rate limits.

Step 4: Add user and extra context

At minimum, set `user.id` to the Hosted account.

Step 5: Set up notification rules

They're super customizable.

Example Notification Rulset

— New issues go to Slack.
— 100 events for an issue in a day sends an email.
— 1000 events/100 users in an hour Pages someone.

Step 7: Utilize Sentry's workflow

Interruptible dev should look at new issues immediately.

Workflow for new issues:

after triaging the issue, do one of the following

1. Immediately release a fix, move to Resolved.
2. Move to Ignored - circle back when impact is more clear.
3. Create a backlog item, leave a comment in Sentry.
4. Don't do nothing.

Workflow for an issue backlog:

— Prioritize by impact to users.
— Fix issues as they come up in client bugs (utilize search).
— Filter out what's not actionable.
— Utilize auto-resolve.
— Follow the same workflow for new issues!

There's a lot more to learn!

— Releases and Github integration
— Breadcrumbs
— Client-side integrations! (team mobile, where you at)

Resources

Sentry best practices on Confluence:
 - https://activecampaign.atlassian.net/wiki/spaces/DEV/pages/444498665/Sentry+for+Developers+-+Overview+and+Best+Practices
Clone the sample application (slides are also here).
  - https://github.com/akowalz/sentry-example-app/
Sentry Documentation:
 - https://docs.sentry.io/

Thank you! Questions?