COMP280
Optimization Report

Morgan Penrose

December 9, 2020

Introduction

This report covers the progression of the backend application 'teams-bitbucket-hook', providing a much-needed connection between BitBucket and Discord or Teams. The webhook itself is currently designed to accept data from a provider and have it pass through to a destination, in effect acting as middleware between the two to form a makeshift bridge.

https://gamesgit.falmouth.ac.uk/projects/GAS/repos/teams-bitbucket-hook/browse
A fork is located on '2021 L2 Leaf Sheep (LSHEEP)' as a fork which I have been using to develop the new edition of the webhook.

The report detailed below is based on the changeset marked between 04cf313f449 and 3c868086e36, located on dev.

Initial baseline

Before starting modifications, the server was using @hapi/hapi which contains enough to write the initial hook to be used by teams on Discord and Teams, but provided a limited scope to expand upon due to how the application structure forces a set standard of properties to be defined and then increasing the indentation further. Although the original file (app.js@04cf313) was only 6.8 KB / 6,967 bytes, much of this was repeated code with slight variations for either of the destination platforms available.

My optimization looks at the overall coherency of the code before and after and how well the application can respond to varying requests from the provider.

Reasons for changing dependencies

The application development saw many changes in the packages it now requires to run. The first exchange is request for r2, simply because request went into maintenance mode with request/request#3142 around a year ago due to how out of date the module was becoming and with the evolving ecosystem that is the Node Package Manager (NPM). The move to r2, which is also made by mikeal – the same author behind request, would be on the principle of what if request evolved.

The second, to switch express for hapi. The main goal was to minimize the code as much as possible for ease of reading. A stretch goal was to prepare the application for another adaptation before the planning had even started. Upcoming changes would likely include separation of routes and common methods into files that would be called upon regularly.

Debugging

With the recent changes it would be quite difficult to identify possible caveats with the server or templates. The addition of 'morgan' helps by logging requests on any route, even if they are not declared. This assists in the tracking of webhook data being passed through the application by logging the endpoint used; the status code set on the header; and the time taken to complete the request.

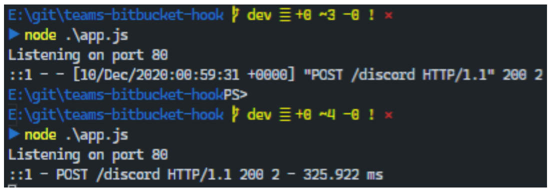
Statistics

	Before	After
Response Time [1] *For Discord Webhook*	1. 234 ms 2. 203 ms 3. 219 ms	1. 328 ms 2. 344 ms 3. 235 ms
Response Time *For New Template*	*N/A*	1. 312 ms 2. 297 ms 3. 250 ms
Project size [2] *Without dependencies*	31.2 KB 31,992 bytes	65.8 KB 67,386 bytes
Project size [3] *With dependencies*	5.19 MB 5,446,859 bytes	5.22 MB 5,478,985 bytes
Lines of code (app.js)	168	150

While the statistics here may seem quite a fair bit off from what was intended, the justification comes with the template engine and the ability to expand without having to add the excessive lines of code that would increase the time taken on the side of efficiency.

[1] Times have been recorded using Insomnia, a request sandbox for DevOps and API testing.
[2] Only includes files that were committed to version control (treated like a fresh clone), any files located on system that can also be found within the .gitignore are not part of this sum.
[3] Includes all files on system as if it were a working environment (includes .git and node_modules folders).

Offhand edits



```
E:\git\teams-bitbucket-hook ᚠ dev ≡ +0 ~3 -0 ! ×
▶ node .\app.js
Listening on port 80
::1 - - [10/Dec/2020:00:59:31 +0000] "POST /discord HTTP/1.1" 200 2
E:\git\teams-bitbucket-hookPS>
E:\git\teams-bitbucket-hook ᚠ dev ≡ +0 ~4 -0 ! ×
▶ node .\app.js
Listening on port 80
::1 - POST /discord HTTP/1.1 200 2 - 325.922 ms
```

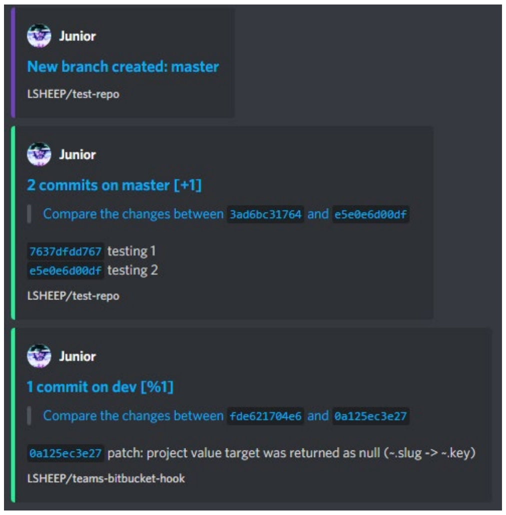
While making this report, I have since noticed some problems with one or more of the templates and patched them using the correct template formatting. The logging aspect was also not noting the time taken to run through the request, subsequently the log string was updated to use the ‘short’ format instead of ‘common’. In the near future, I hope to improve this further by adding an argument to identify which webhook was used and where it originated from.

As an additonal late find, the in-dev template had an error with one of the macro definitions which was quickly patched and rectified from status 503 to status 200 as well as the missing definitions for the loop contained inside and it’s closing tag on the end.

Feature advancement

The inclusion of ‘change summaries’ has resulted in a format I was not expecting with the symbols and allowed for an easier method for understanding how many files were modified and how at a glance. Though I must state that the symbols themselves are not my first, nor my best choice, I would most welcome anyone who would want to further contribute to this.

Future development



Junior
New branch created: master
LSHEEP/test-repo

Junior
2 commits on master [+1]
Compare the changes between `3ad6bc31764` and `e5e0e6d00df`

`7637dfdd767` testing 1
`e5e0e6d00df` testing 2
LSHEEP/test-repo

Junior
1 commit on dev [%1]
Compare the changes between `fde621704e6` and `0a125ec3e27`

`0a125ec3e27` patch: project value target was returned as null (~.slug -> ~.key)
LSHEEP/teams-bitbucket-hook

While this particular sequence of iterations is at an end, the development continues to progress to ensure that it stays up-to-date and of end-user requirements. Various forms and boards have been created and managed accordingly to indicate the next course of action, and to track feature headway before being pushed to production.

As a side project, some students requested the return of Connect 4. While this may seem like a game, this would appear to encourage constructive contributions while attempting to commit the most self-authored changes in one push. The caveat to this would be that the system can be ‘abused’ or ‘taken advantage of’ if contributors decide to not play by the rules or conventions set out by their team. I have given this warning appropriately in the message I detailed about its possible development, but also provided an opportunity to adapt this into a constructive resolution giving an option to opt-in to ‘includeMergeCommits’.

Conclusion

Over all routes and templates for the webhook, I found that the easiest solution would be to manually code the payload but that would hinder the adaptability that end-users would be able to get. The only problem that remains ever so slightly is the email conflict for domain staff. Due to how usernames were set, it is not possible to identify some contributors without querying an admin endpoint.

> Plugin architecture will be looked at for subsequent deployments, but finding an effective solution is key before attempting to write this aspect.

The goal of enhancing the webhook to be more informative towards the end user has been a resounding success for this phase. It was made possible, by the students who are currently using it or intend to be using it in the future. Their involvement on Discord, Teams and Twitch has been a contributing factor into how I develop this and how I communicate my progress back to them as those changes are pushed.

What could I have done better?

I would have started my documentation efforts sooner, to ensure that my templates and methodology can be understood by those that wish to learn from my contributions here or to contribute themselves without having to any more reading than they must. My second mistake would probably be scope. Although what you see here is very much a complete application, all be it partially, I initially set out to develop the template engine as part of a plugin which would then be imported for that environment. But that adds another dimension and complications in standardizing absolutely everything that is written for the application.