GSOC 2024 PROPOSAL

AsyncAPI

Becoming a Maintainer of AsyncAPI Generator

Mintu Gogoi

Table of Contents

Contact Information

- **Name:** Mintu Gogoi
- **Degree**: Bachelor of Technology
- **Year**: Sophomore
- **University**: Indian Institute Of Technology, BHU(Varanasi)
- **Github Handle**: @utnim2
- **Contact Number:** (+91) 6002759470

My Contributions

I have been an active member of the AsyncAPI community for the last 4 months and have been actively engaged in discussions, creating issues, solving bugs, helping others and also reviewed a PR.

Some of my contributions to AsyncAPI community are -

PR Number	Description	Status
#583	Added scripts that on release will publish transpiled template to npm	[open]
#1165	“ts-node” is registered only if generator is imported in a “.js” file	[open]
#1089	Added a lifecycle file to the default glee example	[merged]
#473	Added code linting to the “spec-json-schemas” repository	[merged]
#483	Reduced the complexity of “validate-schema.js”	[merged]
#1687	Added test for cpp constraint	[merged]
#1695	Added test for PHP constraint	[merged]
#263	Fixes wrong format for Co-authored automerged + pagination	[open]

#274	Darken the background image of the card	[open]
#1730	Added loading animation for when playground generate models	[open]

In addition to solving the above issue i have also created 3 issues while using studio(#915), reading docs of generator(#1164) and when visiting conference-website(#285).

Also while learning the concept of CLI encountered upon an issue(#1081) which I was trying to solve but a contributor had already solved it so I went ahead and give him a review of his PR(#1091).

Currently I am working on generator(#1128) , studio(#703), studio(#754).

Project Information

In my exploration of the AsyncAPI Generator.I have delved into the github repository and documentation to gain a comprehensive understanding of its capabilities and the scope of its application.The AsyncAPI generator is a versatile tool designed to generate various outputs based on an AsyncAPI specification file.The generator supports a wide range of templates including those of node-js,node-js-ws etc. showcasing its adaptability across different programming languages and technologies.The generator follows a well defined process that includes parsing AsyncAPI document, validating it against schemas,creating a template context with helper functions and passing this context to a rendering

engine(React or Nunjucks) along with the template file. The rendering engine then injects dynamic values into the templates,resulting in the desired output.With its extensible nature and the availability of community maintained template, the AsyncAPI generator allows developers to streamline the process of generating assets for their event driven applications, promoting consistency and efficiency across different projects and teams.

Familiarity with generator & template tutorial:

I have thoroughly reviewed the Generator documentation and explored its github repository understanding it and have completed the tutorial of “Creating a template” . The link to the repository is (this). To push my understanding further, I have not only completed the tutorial but also extended the existing Python MQTT client template with more functionality. These include -

1. Retained Message support: I have modified the generated client to publish retained message using a “retain” parameter added to “ send{OperationId} ” functions.This enables the client to store messages on the MQTT broker, delivering them to new subscribers when they connect to the topic.This is particularly useful for scenarios like sending the latest device state to new clients joining the network.

2. Last Will and Testament Support: I have modified the “index.js” to introduced a new constructor “(__init)” in the generated “client.py” file.This constructor accepts parameters for configuring the Last Will and Testament message, topic, Quality of service(QoS) level and retain flag.This feature allows client to publish a predefined message when it disconnects

unexpectedly, enabling better error handling and recovery mechanism in connected system.

Beyond completing the tutorial, I am actively contributing and understanding the generator codebase. Here are some of my contributions :

- PR for “ts-node” registration issue (#1030): I have created a PR to fix the issue(#1030). The issue was “ts-node” was being registered unnecessarily even in Javascript projects. My PR implements the following changes.

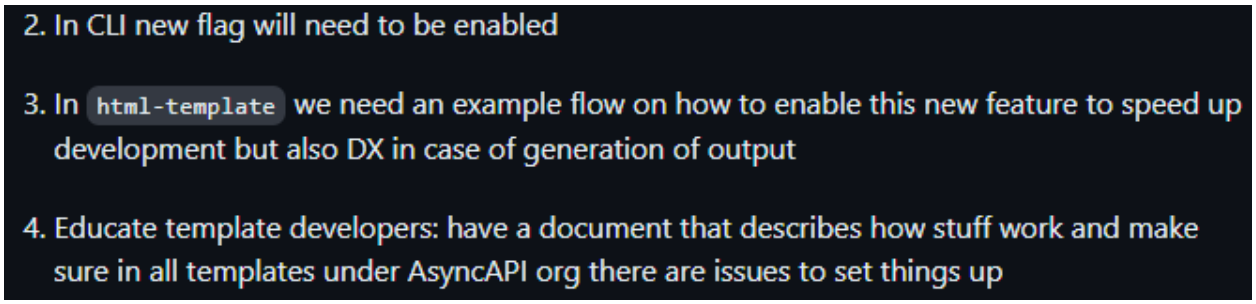
    - Remove the unconditional registration in lib/genrator.js
    - Created “checkForTypescript” and “hasTypescriptFilesDir” functions to check presence of “ts” files in the “filters” and “hooks” directories
    - Conditionally register the “ts” transpiles in the “configureTemplateWorkflow” functions using “checkForTypescriptFiles”

- Working on noOverWriteGlobs Issue(#1128): I am currently investigating a solution for the “noOverwriteGlobs” issue, which causes challenges in react template. My initial solution involves(need further testing).

    - Refactoring to a more general file handling function that accommodates both Njuncks and React templates.
    - Extending React rendering functions to return output filename metadata.
    - Updating “noOverwriteGlobs” to use output filenames for file overwrite checks

    I anticipate opening a PR soon solving the issue.

In addition to the above two issue i have also completed the issue html-template#558 and would like to proceed with @derberg solution of the below image in next(2 and 4) which will mark completion of #521(React templates make the generation process longer).



I guess this contributions and the new functionalities of the template-tutorial reflect my understanding of the generator tools.

Notes on Documentation improvement:

The AsyncAPI Generator documentation(here) is generally well written and provides valuable information. However there are few areas that could be improved:

- Template customization: The section on customizing templates could be expanded with more detailed examples and explanations .While the concept is introduced , it would be beneficial to have a step by step guide or a dedicated section that covers template customization in depth,
- Error handling and troubleshooting: When i was trying to create a project (this- A realtime notification manager) i was encountered with the error or ERROR-NET and many errors

on generating the template with asyncapi/cli when i had search in the docs i have not found many resources or common errors troubleshooting guides.I think this is very much needed as developers will better understand how to diagnose and resolve problems they may encounter during code or documentation generation.

- I will also vow for more visual aids, such as diagrams of “excalidraw” or images to better understand the generator flow.
- Include more examples/tutorial like that one of “template-tutoria”l it generally helps a beginner to understand how things work with a hands on learning experience.

My Projects:

I have created many projects but notably this one is one of my fav. It is a JSON Schema playground that users can edit and validate a JSON schema against the JSON Schema Draft 2020-12 specification.

Involvement in the Community:

I have been an active contributor/learner of AsyncAPI for nearly 4 months and I absolutely love the community engagement and the kindness of the community. I have attended many community meetings be it community meetings,contributor introduction session, gsoc faq session etc.

I plan to be a **tsc-member** of the asyncapi community and would love to maintain a project or two in near future.

I am really interested in the parser-go project and would love to be a maintainer of that project.

About me

Hii ,my name is Mintu Gogoi, a sophomore from IIT-BHU, India. I am a developer that loves learning and exploring new things and have worked on multiple projects as part of my hobby. I love open source development and building scalable and dependable software that affects a number of people.

I plan to become an integral part of JSON Schema Community over a long period of time