SERVER, API AND THE SDK

Producing an SDK while developing your own service APIs

Haim Kastner

Node.JS Meetup | 27/3/2024

YOU DESERVE THE BEST SECURITY

Agenda

- What is wrong with implementing a dedicated SDK?
- The solution
- Implementation - Overview
- Implementation - Server
- Implementation - SDK
- Frontend & SDK
- Advanced topics
- Resources & Questions

What is wrong with implementing a dedicated external SDK?

- Huge amount of repeated and error-prone code per API operation
- Duplicated interfaces, types and validation implementation
- No built-in documentation
- Stagnation of the SDK
- Requires significant maintenance to ensure SDK alignment.

It short, everything takes longer and you end up with duplicated code.

The solution

- Write the interfaces and operation declarations on server only
- Generate API specification during server build
- Build the SDK core
- Generate SDK operations and interfaces using the specification
- Create CI/CD pipelines to generate & publish a new SDK version on specification changes

Implementation - Overview

- Declare server operations using TSOA, a framework that manages all the API aspects (routing, permissions, validations etc.)
- During build, TSOA generates an OpenAPI specification, which will be published on SwaggerHub using the swaggerhub-cli
- Fetch specification and generate SDK operations and interfaces using OpenAPI Generator
- Generate & publish the SDK on specification changes using GitHub Actions.

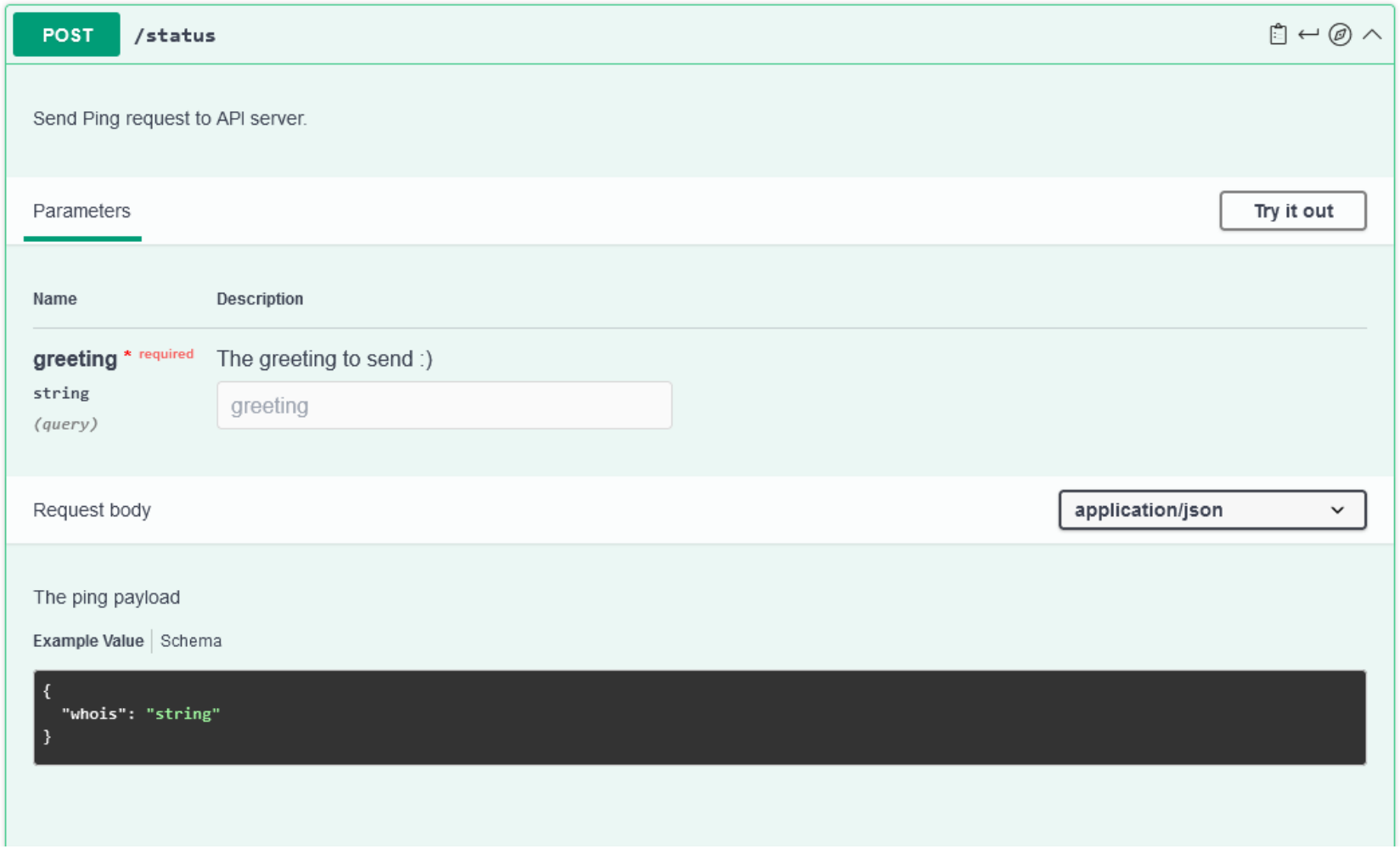
Implementation - Server

```typescript
@Tags('Status')
@Route("status")
export class PingController extends Controller {
    /**
     * Send Ping request to API server.
     * @param greeting The greeting to send :)
     * @param ping The ping payload
     * @returns A Pong object
     */
    @Post()
    public async ping(@Query() greeting: string, @Body() ping?: Ping): Promise<Pong> {
        console.log(`New ping arrived from "${ping?.whois}" who greet us with "${greeting}" :)`);
        return {
            greeting,
            time: new Date().getTime(),
        };
    }
}
```

https://github1s.com/haimkastner/node-api-spec-boilerplate

- Example of a basic, fully implemented API operation
  - TSOA auto-generates an Express route as well as OpenAPI specification for the method
  - Method, interfaces and JSDoc is propagated into the generated OpenAPI specification

The generated OpenAPI specification



- Human-readable and a standard specification
- Hosted on SwaggerHub and can be used to call the operations directly.

https://app.swaggerhub.com/apis/haimkastner/node-api-spec-boilerplate

Implementation - SDK

```
import { ServerSDK } from '@haimkastner/open-api-based-sdk-boilerplate';

async function doSomthing() {
    const serverSDK = new ServerSDK();
    const res = await serverSDK.StatusApi.ping('hello', { whois: 'me' });
    console.log(res.greeting)
}
```

```
(method) StatusApi.ping(greeting: string, ping?: Ping | undefined, options?: any): Promise<Pong>
```

Send Ping request to API server.

@param `greeting` — The greeting to send :)

@param `ping` — The ping payload

@param `options` — Override http request option.

@throws — {RequiredError}

@memberof — StatusApi

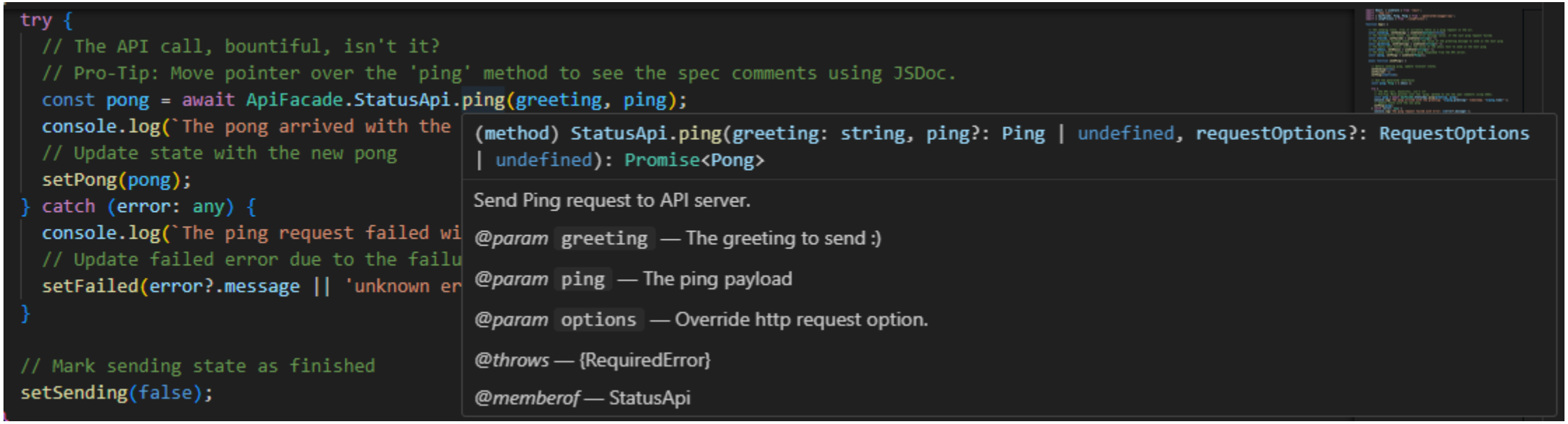
https://github1s.com/haimkastner/open-api-based-sdk-boilerplate

- Example of a basic API call (that's all it takes!)
  - Easy-to-use SDK
  - Fully generated interfaces, requests, responses, validations etc.
  - Server-side JSDoc seamlessly propagates to generated interfaces/methods

Frontend 🤝 SDK

- No more runtime API call mismatches
- No duplicated interfaces and types
- IDE hints, JSDoc and error checks during the development process
- Gets rid of decent chunk of boilerplate for each API request
- Standard, out-of-box error handling, logging, alerting etc.

Implementation - Frontend



https://github1s.com/haimkastner/react-typescript-spec-facade

- Easy-to-use API façade - zero boilerplate, call the operation ONLY.
- Fully generated interfaces, requests, responses, errors etc.
- Server-side JSDoc seamlessly propagates to generated interfaces/methods

Check Point's Open-Source SDKs

Our Harmony Endpoint Management service API infrastructure is based on such principals as demonstrated, using it, we recently released our External API specification and SDKs for TypeScript & Python.

- External API specification on SwaggerHub
- TypeScript SDK source-code on GitHub and package on NPM
- Python SDK source-code on GitHub and package on PyPi

Advanced topics

- Product-wide/cross-product ‘job’ (asynchronization) mechanisms
- Internal tools based on the standardized API, for example, generic React components utilizing useData hook.
- Serving Swagger UI on the API server itself
- Tracking requests and failure points via automatic header injections
- Automated API change log
- Automatic detection/prevention of breaking changes
- API standardizing and conformity

Go wild.

Resources & Questions

Materials

- OpenAPI - https://www.openapis.org/
- Blog's Article - Building API Server https://blog.castnet.club/en/blog/perfect-api-server-part-a
- Blog's Article - Setting Up Front Facade https://blog.castnet.club/en/blog/perfect-api-server-part-b
- Blog's Article - Long processing https://blog.castnet.club/en/blog/perfect-api-server-part-c-jobs
- Blog's Article – Setting Up SDK https://blog.castnet.club/en/blog/perfect-api-server-part-d-sdk

Examples

- Specification – https://app.swaggerhub.com/apis/haimkastner/node-api-spec-boilerplate
- Server (Node.js/TypeScript) - https://github.com/haimkastner/node-api-spec-boilerplate
- SDK (Node.JS/TypeScript) - https://github.com/haimkastner/open-api-based-sdk-boilerplate
- Front (React/TypeScript) https://github.com/haimkastner/react-typescript-spec-facade
- SDK Library - https://www.npmjs.com/package/@haimkastner/open-api-based-sdk-boilerplate
- Live App - https://react-typescript-spec-facade.castnet.club/

Tools

- SwaggerHub – https://app.swaggerhub.com/
- OpenAPI Tools - https://github.com/OpenAPITools
- TSOA - https://tsoa-community.github.io/docs/

THANK YOU

YOU DESERVE THE BEST SECURITY