The Simple, Secure Framework Developers Trust

2020-03-01 RK

1. What is Hapi?
2. Hapi vs Express
3. Structure
4. Getting Started
5. Routing Example
6. Validation Example
7. Plugin Example

1. What is Hapi?

hapi.js (also known as hapi - short for Http API, pronounced “Happy”) is an open-source framework for web applications, created by the mobile web team at Walmart Labs .

The most common use of hapi is to build web services such as RESTful API. You can build application programming interface (API) servers, websites, and HTTP proxy applications with hapi.js.

2. Hapi vs Express

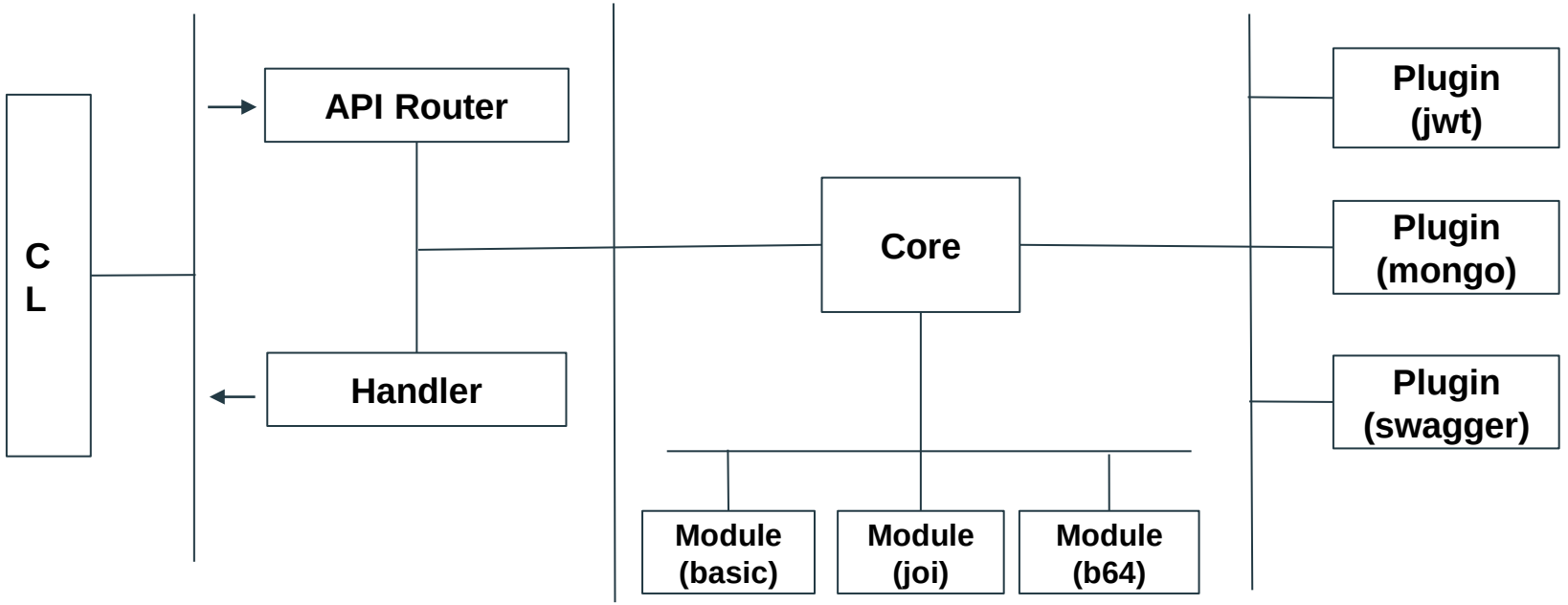
- **Plug-ins support**
- **Single framework**
- **Trustworthy and stable**

(Hapi)

- **Middleware support**
- **Worldwide**
- **Untrustworthy and unreliable**

(Express)

3. Structure


C
L
API Router
Handler
Core
Module
(basic)
Module
(joi)
Module
(b64)
Plugin
(jwt)
Plugin
(mongo)
Plugin
(swagger)

4. Getting Started

- Installing hapi

```
$ mkdir myproject

$ cd myproject

$ npm init

$ npm i @hapi/hapi
```

- Creating Server JS

```
const Hapi = require('@hapi/hapi');

const init = async () => {

    const server = Hapi.server({
        port: 3000,
        host: 'localhost'
    });

    await server.start();
};

process.on('unhandledRejection', (err) => {
    process.exit(1);
});

init();
```

- Adding Routes

```
const init = async () => {

    ...

    server.route({
        method: 'GET',
        path: '/hello/{name}',
        handler: (request, h) => {

            return `Hello ${request.params.name}`;
        }
    });

    ...
};
```

5. Routing Example

- Optional Query Param and Error Handling

```
const Boom = require("@hapi/boom");

server.route({
    method: 'GET',
    path: '/hello/{user?}',
    handler: function (request, h) {
            try {
          const user = request.params.user ? request.params.user : 'stranger';
          return h.response({data: `Hello ${user}!`}).code(200);
            } catch (err) {
               return Boom.badImplementation(err.message);
      }
    }
});
```

- Multi-Segment Parameters

```
server.route({
    method: 'GET',
    path: '/hello/{user*2}',
    handler: function (request, h) {

        const userParts = request.params.user.split('/');

        return `Hello ${userParts[0]} ${userParts[1]}!`;
    }
});
```

- Request Payload

```
server.route({
    method: 'POST',
    path: '/signup',
    handler: function (request, h) {

        const payload = request.payload;

        return `Welcome ${payload.username}!`;
    }
});
```

6. Validation Example

- Payload Validation

```
const Joi = require("@hapi/joi");

server.route({
    method: 'POST',
    path: '/signup',
    handler: function (request, h) {
        ...
    },
    options: {
        validate: {
            payload: {
                username: Joi.string().min(1).max(20),
                password: Joi.string().min(7)
            }
        }
    }
});
```

- Query Validation

```
options: {
    validate: {
        query: Joi.object({
            limit: Joi.number().integer().min(1).max(100).default(10)
        });
    }
}
```

- Params Validation

```
options: {
    validate: {
         params: Joi.object({
             name: Joi.string().min(3).max(10)
         })
    }
}
```

- Output Validation

```
const bookSchema = Joi.object({
    title: Joi.string().required(),
    author: Joi.string().required()
});

...

options: {
    response: {
        schema: Joi.array().items(bookSchema),
        failAction: 'log'
    }
}
```

7. Plugin Example

- Define Plugin

```
const myPlugin = {
  name: 'myPlugin',
  register: (server, options) =>  {
    server.route({
      method: 'GET',
      path: '/test',
      handler: (request, h) => {
          const name = options.name;
          return `Hello ${name}`;
      }
    });

    await someAsyncMethods();
  }
};
```

- Register Plugin

```
const start = async () => {

  await server.register({
     plugin: require('myplugin'),
     options: {
        name: 'Bob'
     }
  });
};
```

- Official Plugin (MongoDB)

```
server.register({
      plugin: require('hapi-mongoose'),
      options: {
            uri: 'mongodb://localhost:27017'
      }
});

… … …

const myHandler = (request, h) => {
      const db = server.plugins['hapi-mongoose'].connection;
      const mongoose = server.plugins['hapi-mongoose'].lib;
      const Schema = mongoose.Schema;
      const Tank = db.model('Tank', { size: string });
      const small = new Tank({ size: 'small' });
      return await small.save();
}
```

- hapi-jsonwebtoken
- hapi-passport-saml
- hapi-swagger
- hapi-swaggered-ui
- hapi-i18n
- hapi-plugin-websocket
- hapi-error
- hapi-geo-locate
- hapi-gate
- hapi-mysql2
- hapi-oracledb
- hapi-rate-limitor
- hapi-response-time
- hapi-router
- hapi-plugin-graphiql
- hapi-react-views
- hapi-server-session
- hapi-wechat
- hapi-pino
- hapi-plugin-traffic
- hapi-ending
- inert
- nes
- qs
- blankie
- crumb
- susie

...

Thank you for your time!