Tech Eval Written

Group 13

Introduction

ExpressJS is an open-source web application framework that runs on Node.js. As Express is not a complete MVC framework by itself, it contains a template engine for optional use. Express was originally created in 2010, which was acquired by StrongLoop, a company working with Node.js that creates and supports mobile backend as a service named loopback. In 2015, IBM took over the project, who then announced that Express.js would be placed under the care a community led and industry backed consortium called the Node.js Foundation Incubator. Express.js is a lightweight framework for small to medium projects, which could be used in parallel with alternative framework extensions for Node.js. Express contains a generator which builds an application skeleton to give users a quick start to the project and access to middleware, which are pre-made function solutions to a variety of problems. Features of Express include routing, request and response handling, middleware add-ons, static file serving, cookie management, security and database integration. With the minimal amount of features that express offers, companies often consider alternatives such as Koa, Ruby on Rails, Sails, Hapi and Restify.

Koa

There are a variety of alternatives that are available should you decide against using ExpressJS, most of them also running on NodeJS. One of the alternatives is called Koa, and is designed by the people behind Express. The goal of Koa, however, is divergent from Express': "Philosophically, Koa aims to 'fix and replace node', whereas Express 'augments node'. Koa offers much more limited functionality as a result when compared to Express.

Koa does not offer routing or middleware like express does and lacks helpful utilities such as sending files and templating. However, Koa is much more modular than express and uses "Less Hackery" to provide a better user experience. Using a generator based flow, it avoids callback hell. It also makes error handling a lot easier to implement. You can implement a catch all handler, or go more granular in your code. The amount of work required to set up error handling in Koa is much less than if you were setting up error handling in Express and is much cleaner. Koa is modelled more as a HTTP framework rather than an complete MVC web app framework. So while koa is a lot better in some of the more granular areas it also lacks functionality that express provides. If you are looking for a lighter and cleaner HTTP alternative to express then Koa is your best bet.

Ruby on Rails

ExpressJS, like Ruby on Rails is easy to setup and get started. A similarity between the two, is that they both use a Model, View and Controller framework. Expressjs is great when it comes to building a small lightweight application and excels in making it as fast as possible. While on the other hand, Ruby on Rails allows larger structured projects that requires more power. Both are great when it comes to support: ExpressJS and Ruby on Rails are backed by big communities and make it easy to find packages. ExpressJS uses npm and Ruby on Rails uses Gems. One point where Ruby on Rails excels at is that it is easier to debug and test what you have written. In ExpressJS if a bug crash occurs, it crashes the entire project making it harder to find the bug. After using both ExpressJS and Ruby on Rails it is hard to decide which is the better and really comes down to the circumstances. Ruby on Rails is the better choice when comes to a large project with multiple people involved while ExpressJS is great if you wish to keep things simple and have everything in JavaScript.

Sails

Sails follows the same MVC framework as ExpressJS. Their level of complexity is rather low making it suitable for small to medium sized projects.Rather than depending on middleware for creating applications, Sails packages offer securities policies, pipeline management ORM and anything that you may require. Express uses an unopinionated programming approach while Sails is opinionated which may result as a barrier for additional customization. However this closely resembles a Ruby on Rails approach as it will auto-generate code for you to develop applications through a RESTful API approach. In Express, it handles HTTP and websocket servers individually through different instances while Sails incorporates a websocket wrapper called socket.io for better management of these websocket servers. This also means that Sails is better suited for real-time socket communication. Sails use a Object Role Modelling method for database modelling. Specifically, it uses a storage and retrieval engine called Waterline which can be installed through npm. This is an issue that fixes Express's shortcomings in database abstraction. However, Waterline has a lack of support for associations (relational data). Sails is a relatively new framework compared to Express and there is not much documentation for users. As mentioned, it's opinionated nature may drive away Express users from switching. All in all, Sails is still a great framework with lots of potential having features such as Grunt, a flexible asset pipeline for supporting the front-end asset workflow.

Hapi

Hapi (**H**ttp **API** server) was developed by Eran Hammer and the team at Walmart labs. They found that Express wasn't meeting their needs when it came to extensibility and support. So they went along and built hapi over parts of Portmile, something that was originally built on top of Express. It is a framework build on configuration over code. Express is minimal and gives us a small API to start up. Additional functionalities have to be configured manually by the developer by installing separate modules. On the contrary, Hapi consist of a rich set of features which can be accessed through different configuration options, additional functionalities can be carried out by just handling these configurations. Hapi consist of built-in features such as input validation called joy, authentication, cookie-parsing, Logging, Cross-Origin-Resource-Sharing support, Server-side caching that can use multiple storage options and various other features that are not included within Express. Hapi uses request lifecycle and consist of extension points unlike Express that is focused more on middleware functions. It also consist of a plugin-system, this allows the developers to freely work over their part of the application without conflicting with the overall application. These plugins can then be combined into the server and later deployed. Conclusively, Hapi is more tailored towards bigger or more complex applications and consist of too much boilerplate code to throw together to create a simple web application, that is where Express shines. Although Hapi has been open sourced for a long time the community backing it up is still smaller and doesn't consist of a lot of documentation, making it harder for beginner developers.

Restify

Restify is a node.js module built to create REST web services in Node. Restify makes a lot of hard problems when building such a service, like error handling, version control, and content negotiation, easier. It provides built in DTrace probes to quickly find out the performance problems of an application. Restify also provides built in throttling and SPDY support. It does not have unnecessary functionality like templating and rendering. Ideally, for a service that wants to move quickly but receives lots of requests from the same clients, Restify performs the best for easier service deployments. Restify is used to build strict API services that are maintainable and observable. Whereas Express is targeted at browser applications and rich in functionality. However, it is still lightweight since unused functionality does not have much impact on the overall performance. Express is well maintained, fully customizable, and has little learning curve since it is nearly a standard for Node.js web application. However, when it comes to RESTful APIs, both Restify and Express require a lot of manual labor. For example, Refactoring would be hard and painful because everything needs to be updated everywhere since there is no automatic update system. And all end points need to be created and tested manually so people end up doing the same code over and over again.

References

https://github.com/koajs/koa/blob/master/docs/koa-vs-express.md

https://www.airpair.com/node.js/posts/nodejs-framework-comparison-express-koa-hapi

http://matt-harrison.com/moving-from-express-to-hapi-js/

http://stackoverflow.com/questions/17589178/why-should-i-use-restify

http://runastartup.com/express-js-vs-sails-js-comparison/

https://nathanleclaire.com/blog/2013/12/28/the-good-the-bad-and-the-ugly-of-sails-dot-js-realtime-javascript-mvc-framework/

https://www.pluralsight.com/blog/software-development/node-web-frameworks

http://sailsjs.org/