The Code Barbarian

The software development blog of Valeri Karpov

The MEAN Stack: MongoDB, ExpressJS, AngularJS, and Node.js

Posted on April 29, 2013 by vkarpov15

This post was featured as a guest blog for MongoDB on April 30th 2013, which can be found here (http://blog.mongodb.org/post/49262866911/the-mean-stack-mongodb-expressjs-angularjs-and).

A few weeks ago, a friend of mine asked me for help with PostgreSQL. As someone who's been blissfully SQL-free for a year, I was quite curious to find out why he wasn't just using MongoDB instead. It turns out that he thinks Mongo is too difficult to use for a quick weekend hack, and this couldn't be farther from the truth. I just finished my second 24 hour hackathon using Mongo and NodeJS (the FinTech Hackathon (http://francescak.me/blog/2013/04/09/fintech-hackathon-recap/) co-sponsored by 10gen (http://10gen.com)), and can confidently say that there is no reason to use anything else for your next hackathon or REST API hack.

First of all, there are huge advantages to using a uniform language throughout your stack. My team uses a set of tools that we affectionately call the MEAN stack- MongoDB, ExpressJS (http://expressjs.com), AngularJS (http://angularjs.org), and NodeJS (http://nodejs.org). By coding with Javascript throughout, we are able to realize performance gains in both the software itself and in the productivity of our developers. With Mongo, we can store our documents in a JSON-like format, write JSON queries on our ExpressJS and NodeJS based server, and seamlessly pass JSON documents to our AngularJS frontend. Debugging and database administration become a lot easier when the objects stored in your database are essentially identical to the objects your client Javascript sees. Even better, somebody working on the client side can easily understand the server side code and database queries; using the same syntax and objects the whole way through frees you from having to consider multiple sets of language best practices and reduces the barrier to entry for understanding your codebase. This is especially important in a hackathon setting- the team may not have much experience working together, and with such little time to integrate all the pieces of your project, anything that makes the development process easier is gold.

Another big reason to go with MongoDB is that you can use it in the same way you would a MySQL database (at least at a high level). My team likes to describe Mongo as a "gateway drug" for NoSQL databases because it is so easy to make the transition from SQL to MongoDB. I wish someone had told me this when I first starting looking into NoSQL databases, because it would have saved me a lot of headaches. Like many people, I was under the impression that CouchDB would be easier to use, while the performance improvements from Mongo were something I could take advantage only once I had gotten my feet wet with CouchDB. Instead CouchDB ended up being much more difficult to work with than I anticipated, largely because it uses custom map-reduce functions to query data,

rather than the more traditional SQL queries I was used to. When I finally switched I was surprised to find that with MongoDB I could still write queries and build indices; the only difference is that the queries are written in JSON and query a flexible NoSQL database.

As a NoSQL database, MongoDB also allows us to define our schema entirely on the code side. With an SQL database you're faced with the inescapable fact that the objects in your database are stored in a format that is unusable by your front-end and vice versa. This wastes precious time and mental energy when you inevitably run into a data issue or need to do some database administration. For example, if you change your ActiveRecord schema in Ruby on Rails, you have to run the "rake" command to make sure your SQL columns stay in sync with your schemas. This is a clear violation of the age-old programming principle D.R.Y.- Don't Repeat Yourself. In contrast, Mongo doesn't care what format the documents in your collections take (for the most part anyway). This means that you spend a lot less time worrying about schema migrations, because adding or removing data items from your schema doesn't really require you to do anything on the database side.

At this point I should note that to get the most out of MongoDB in your MEAN stack, you're going to want to take advantage of MongooseJS (http://mongoosejs.com). Mongoose is a schema and general usability tool for Node that lets you use MongoDB while being as lazy as you want. For example, with Mongoose we can define a schema in JSON:

```
var UserSchema = new Mongoose.Schema({
  username : { type : String, validate: /\S+/, index : { unique : true } },
  profile : {
    name : {
      first : { type : String, default : "" }
      last : { type : String, default : "" }
    }
  }
});
```

We can then create a model by mapping our schema to our MongoDB collection:

```
var User = db.model('users', UserSchema);
```

For all of the Ruby on Rails + ActiveRecord fans out there, note that this User object we've created above now allows us easy access to the basic features you would normally associate with ActiveRecord. For example, we can do thing like:

```
User.findOne({ username : 'vkarpov' },
    function(error, user) {
      /* user is either undefined or a user with username vkarpov */
    });
User.findOne({ _id : req.params.id },
    function(error, user) {
      /* user with ID defined by the hex string in req.params.id */
     });
User.find({ 'profile.name.first' : 'Valeri' },
     function(error, users) {
       /* users is a list with users with first name Valeri  */
     });

var user = new User({ username : 'vkarpov' });
user.save(function(error, user) {
  /* Saves user with default values for profile.name.first and .last into '
});

var user2 = new User({ username : 'v karpov' });
user2.save(function(error, user) {
  /* Error - regular expression validation for username failed */
});
```

Another powerful tool that MongoDB and MongooseJS provide is the ability to nest schemas. You'll notice that in the User schema above we have "profile" and "name" objects that are part of a nested schema. This is a simple and useful design choice that illustrates how powerful nested schemas can be. Suppose that we want to give our user the ability to edit their first and last name, but not their username. Assuming the website has a /profile route where our user can the change first and last names, the Javascript front-end will get a JSON object as the result of a call to User.findOne on the backend. After the user modifies their profile, the front-end makes a POST request to /profile.json with the user object in JSON as the body. Now on the backend (using ExpressJS syntax) we can simply use:

```
function(req, res) {
  user.profile = req.body.profile;
  user.save(function(error, user) {
    res.redirect('/profile');
  });
}
```

That's it. Mongoose takes care of validating of the profile information, so we don't have to change the POST /user.json route if we change the User schema, and there is no way the username field can be modified. We could do the same thing when using Ruby on Rails and ActiveRecord, but this would require having a separate Profile schema in a separate table, meaning that among other things we'd incur a performance penalty because of the extra underlying join operation.

MongoDB is the bedrock of our MEAN stack, and you should strongly consider using it for your next project. You can write your entire stack in one language, have schemas for ease of use on top of a flexible and performant NoSQL database, and nest schemas when you don't truly need to have separate collections. All of this adds up to you spending more of your precious hackathon hours building the other cool aspects of your product and less time figuring out how your objects translate between different levels of the stack. By the way, MongoDB and MEAN are useful well beyond hackathons- we use this approach for all of our commercial products, most recently The Ascot Project

(http://www.ascotproject.com).

Have any questions about the code featured in this post? Want to suggest a better approach? Feel like telling me why the MEAN Stack is the worst thing that ever happened in the history of the world and how horrible I am? Go ahead and leave a comment below, or shoot me an email at valkar207@gmail.com and I'll do my best to answer any questions you might have. You can also find me on github at https://github.com/vkarpov15 (https://github.com/vkarpov15). My current venture is called The Ascot Project, and you can find that over at www.AscotProject.com (http://www.ascotproject.com). Huge thanks to my partner William Kelly (@idostartups) for helping me write this post and for getting MongoDB to republish it.

Bookmark the permalink.

18 thoughts on “The MEAN Stack: MongoDB, ExpressJS, AngularJS, and Node.js”

William Mora (@wilmor24) says:
on May 6, 2013 at 3:05 am
Good post. I have a question: how would you handle security between your Angular.js app and your node.js backend? Would you implement OAuth just for your Angular.js app?

Reply
vkarpov15 says:
on May 13, 2013 at 3:34 pm
Hi William, thanks for the comment. Security has a very broad definition, so I'm not entirely sure what you mean. If you mean how to validate users and track sessions, there are plenty of open source solutions out there for NodeJS / ExpressJS. Personally, I think that the frontend should do as little as possible with regards to security, simply because your frontend Javascript is publicly accessible and anybody with greasemonkey installed can tweak it at their whim.

Reply

3. Pingback: The Code Barbarian

Tom Utley says:
on June 7, 2013 at 8:49 pm
I think the MEAN stack is awesome and I'm going to learn Angular so I can actually use it, lol

Reply

Andreas Mueller says:
on June 17, 2013 at 1:20 pm
Reallay Nice post! Unfortunately the MEAN stack does not work on the raspberryPi. At least I cannot get installed MongoDB on ARM 32bit.

Reply
Andreas Mueller says:
on June 17, 2013 at 1:21 pm
Really Nice post! Unfortunately the MEAN stack does not work on the raspberryPi. At least I cannot get installed MongoDB on ARM 32bit.

Reply
http://www.linksdebatte.de/index.php/Garcinia_Cambogia_Choose_Assessment says:
on July 20, 2013 at 1:49 am
Hello just wanted to give you a brief heads up
and let you know a few of the pictures aren't loading properly. I'm
not sure why but I think its a linking issue. I've tried it in two different browsers and both show the same outcome.

Reply
vkarpov15 says:
on July 20, 2013 at 3:20 am
Hi,

Which pictures are you talking about?

Reply

8. Pingback: Introduction to the MEAN Stack, Part One: Setting Up Your Tools | The Code Barbarian

9. Pingback: Introduction to the MEAN Stack, Part Two: Building and Testing a To-do List | The Code Barbarian

Erica says:
on September 22, 2013 at 12:05 am
I'm excited to discover this page. I wanted to thank you for your time for this fantastic read!! I definitely loved every bit of it and I have you book-marked to see new things in your website.

Reply

11. Pingback: Definition des MEAN Stacks | senäh

12. Pingback: MEAN stack for BitNami: MongoDB, Express, AngularJS & NodeJS | 7admin

rhef=http://verticallegacy.com/dcollins/>best webinar on a home based business says:
on January 25, 2014 at 3:29 pm
You're so interesting! I don't think I've truly read through something like this before.

So nice to find another person with unique thoughts on this subject matter.
Seriously.. thanks for starting this up. This web site is something
that is required on the internet, someone with a little originality!

Reply

14. Pingback: TDD and BDD With The MEAN Stack: Introduction | attackOfZach

how to lose weight fast says:
on February 19, 2014 at 8:30 am
I was wondering if you ever considered changing the page layout of your site?
Its very well written; I love what youve got to say.

But maybe you could a little more in the way of content so people
could connect with it better. Youve got an awful
lot of text for only having 1 or two pictures. Maybe you could space it out better?

Reply
bestmicrowavereviews2014.com says:
on July 23, 2014 at 6:31 pm
Hello there! Do you know if they make any plugins to safeguard against hackers?
I'm kinda paranoid about losing everything I've worked hard on. Any suggestions?

Reply

17. Pingback: Azure - Why you gotta be so MEAN? (with apologies to Taylor Swift) - DevFish on MSDN ...><«(o>... - Site Home - MSDN Blogs