Authentication and Express Middleware

It's simple to use any Express middleware in conjunction with `express-graphql`. In particular, this is a great pattern for handling authentication.

To use middleware with a GraphQL resolver, just use the middleware like you would with a normal Express app. The `request` object is then available as the second argument in any resolver.

For example, let's say we wanted our server to log the IP address of every request, and we also want to write an API that returns the IP address of the caller. We can do the former with middleware, and the latter by accessing the `request` object in a resolver. Here's server code that implements this:

```
var express = require('express');
var { graphqlHTTP } = require('express-graphql');
var { buildSchema } = require('graphql');

var schema = buildSchema(`
  type Query {
    ip: String
  }
`);

const loggingMiddleware = (req, res, next) => {
  console.log('ip:', req.ip);
  next();
}

var root = {
  ip: function (args, request) {
    return request.ip;
  }
};

var app = express();
app.use(loggingMiddleware);
app.use('/graphql', graphqlHTTP({
  schema: schema,
  rootValue: root,
  graphiql: true,
}));
app.listen(4000);
console.log('Running a GraphQL API server at localhost:4000/graphql');
```

In a REST API, authentication is often handled with a header, that contains an auth token which proves what user is making this request. Express middleware processes these headers and puts authentication data on the Express `request` object. Some middleware modules that handle authentication like this are Passport, express-jwt, and express-session. Each of these modules works with `express-graphql`.

If you aren't familiar with any of these authentication mechanisms, we recommend using `express-jwt` because it's simple without sacrificing any future flexibility.

If you've read through the docs linearly to get to this point, congratulations! You now know everything you need to build a practical GraphQL API server.

Continue Reading →
Constructing Types

Constructing Types

GraphQLSchema
GraphQLString
GraphQLUnionType
isAbstractType
isCompositeType
isInputType
isLeafType
isOutputType

graphql/utilities

astFromValue
buildASTSchema
buildClientSchema
buildSchema
introspectionQuery
isValidJSValue
isValidLiteralValue
printIntrospectionSchema
printSchema
typeFromAST
TypeInfo

graphql/validation

specifiedRules
validate

Copyright © 2021 The GraphQL Foundation. All rights reserved. The Linux Foundation has registered trademarks and uses trademarks. For a list of trademarks of The Linux Foundation, please see our Trademark Usage page. Linux is a registered trademark of Linus Torvalds. Privacy Policy and Terms of Use.