

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.

GRAPHQL.JS TUTORIAL

[Getting Started](#)[Running Express + GraphQL](#)[GraphQL Clients](#)[Basic Types](#)[Passing Arguments](#)[Object Types](#)[Mutations and Input Types](#)[Authentication & Middleware](#)

ADVANCED GUIDES

[Constructing Types](#)

API REFERENCE

[express-graphql](#)[graphqlHTTP](#)[graphql](#)[graphql](#)[graphql/error](#)[formatError](#)[GraphQLError](#)[locatedError](#)[syntaxError](#)[graphql/execution](#)[execute](#)[graphql/language](#)[BREAK](#)[getLocation](#)[Kind](#)[lex](#)[parse](#)[parseValue](#)[printSource](#)[visit](#)[graphql/type](#)[getNamedType](#)[getNullableType](#)[GraphQLBoolean](#)[GraphQLEnumType](#)[GraphQLFloat](#)[GraphQLID](#)[GraphQLInputObjectType](#)[GraphQLInt](#)[GraphQLInterfaceType](#)[GraphQLList](#)[GraphQLNonNull](#)[GraphQLObjectType](#)[GraphQLScalarType](#)[Continue Reading →](#)[Constructing Types](#)

Constructing Types

[GraphQLScalarType](#)
[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](#)



Learn

[Introduction](#)
[Query Language](#)
[Type System](#)
[Execution](#)
[Best Practices](#)

Code

[Languages](#)
[Tools](#)
[Services](#)

Community

[Upcoming Events](#)
[Stack Overflow](#)
[Facebook Group](#)
[Twitter](#)

More

[GraphQL Specification](#)
[GraphQL Foundation](#)
[GraphQL GitHub](#)
[Edit this page](#)