Introduction to GraphQL

***Good reading material:**

https://scotch.io/@codediger/build-a-simple-graphql-api-server-with-express-and-nodejs

Make sure you have Node v6 installed: ***node -v***

To install GraphQL.js in your current directory:

npm install express –save npm install graphql –save npm install espress-graphql –save

Let's see a basic structure:

Create a server.js file in the root directory and add this code and run ***node server.js:***

```
const express = require('express');
const { buildSchema } = require('graphql');
const graphqlHTTP = require('express-graphql');
let port = 3000;

/* Here a simple schema is constructed using the GraphQL schema language
(buildSchema).
   More information can be found in the GraphQL spec release */

let schema = buildSchema(`
  type Query {
    postTitle: String,
    blogTitle: String
  }
`);

// Root provides a resolver function for each API endpoint
let root = {
    postTitle: () =&gt; {
    return 'Build a Simple GraphQL Server With Express and NodeJS';
},
blogTitle: () =&gt; {
    return 'scotch.io';
}
};

const app = express();
app.use('/', graphqlHTTP({
    schema: schema,
    rootValue: root,
    graphiql: true //Set to false if you don't want graphiql enabled
}));

app.listen(port);
console.log('GraphQL API server running at localhost:'+ port);
```

GraphQL terminology

Schema

A schema defines a GraphQL API's type system. It describes the complete set of possible data (objects, fields, relationships, everything) that a client can access. Calls from the client are validated and executed against the schema.

Field

A field is a unit of data you can retrieve from an object.***The GraphQL query language is basically about selecting fields on objects.***All GraphQL operations must specify their selections down to fields which return scalar values to ensure an unambiguously shaped response.

Argument

An argument is a set of key-value pairs attached to a specific field. Some fields require an argument. Mutations require an input object as an argument.

Implementation

A GraphQL schema may use the term *implements* to define how an object inherits from an interface.

Here's a contrived example of a schema that defines interface ***X*** and object ***Y***:

```
interface X {
  some_field: String!
  other_field: String!
}

type Y implements X {
  some_field: String!
  other_field: String!
  new_field: String! }
```

This means object ***Y*** requires the same fields/arguments/return types that interface ***X*** does, while adding new fields specific to object ***Y***. (The ***!*** means the field is required.)

- Each object lists the interface(s) *from which it inherits* under **Implements**.
- Each interface lists the objects *that inherit from it* under **Implementations**.

Connection

Connections let you query related objects as part of the same call. With connections, you can use a single GraphQL call where you would have to use multiple calls to a REST API.

It's helpful to picture a graph: dots connected by lines. The dots are the nodes, the lines are edges. A connection defines a relationship between nodes.

Edge

Edges represent connections between nodes. When you query a connection, you traverse its edges to get to its nodes. Every edges field has a node field and a cursor field. *Cursors are used for pagination.

Connections

marketplaceListings (MarketplaceListingConnection)

Argument	Type	Description
adminId	ID	Select listings that can be administered by the specified user.
after	String	Returns the elements in the list that come after the specified global ID.
allStates	Boolean	Select listings visible to the viewer even if they are not approved. If omitted or false, only approved listings will be returned.
before	String	Returns the elements in the list that come before the specified global ID.
categorySlug	String	Select only listings with the given category.
first	Int	Returns the first *n* elements from the list.
last	Int	Returns the last *n* elements from the list.
organizationId	ID	Select listings for products owned by the specified organization.
primaryCategoryOnly	Boolean	Select only listings where the primary category matches the given category slug.
slugs	[String]	Select the listings with these slugs, if they are visible to the viewer.
viewerCanAdmin	Boolean	Select listings to which user has admin access. If omitted, listings visible to the viewer are returned.
withFreeTrialsOnly	Boolean	Select only listings that offer a free trial.

search (SearchResultItemConnection)

Argument	Type	Description
after	String	Returns the elements in the list that come after the specified global ID.
before	String	Returns the elements in the list that come before the specified global ID.
first	Int	Returns the first *n* elements from the list.
last	Int	Returns the last *n* elements from the list.
query	String!	The search string to look for.
type	SearchType!	The types of search items to search within.

Fields

codeOfConduct (CodeofConduct)

Argument	Type	Description
key	String!	The code of conduct's key

codesOfConduct ([CodeofConduct])

marketplaceCategories ([MarketplaceCategory])

Argument	Type	Description
excludeEmpty	Boolean	Exclude categories with no listings.

marketplaceCategory (MarketplaceCategory

Argument	Type	Description
slug	String!	The URL slug of the category.

marketplaceListing (MarketplaceListing)

Argument	Type	Description
slug	String!	Select the listing that matches this slug. It's the short name of the listing used in its URL.

node (Node)

Argument	Type	Description
id	ID!	ID of the object.

nodes ([Node])

Argument	Type	Description
ids	[ID!]!	The list of node IDs.

organization (Organization)

Argument	Type	Description
login	String!	The organization's login.

relay (Query)

Hack to workaround https://github.com/facebook/relay/issues/112 re-exposing the root query object

resource (UniformResourceLookable)

Argument	Type	Description
url	URI!	The URL.

topic (Topic)

Argument	Type	Description
name	String!	The topic's name.

user (User)

Argument	Type	Description
login	String!	The user's login.

Mutations

The mutation type defines GraphQL operations that change data on the server:

- ***acceptTopicSuggestion***
- ***addComment***
- ***addProjectCard***
- ***addProjectColumn***
- ***addPullRequestReview***
- ***addPullRequestReviewComment***
- ***addReaction***
- ***addStar***
- ***createProject***
- ***declineTopicSuggestion***
- ***deleteProject***
- ***deleteProjectCard***

- ***deleteProjectColumn***
- ***deletePullRequestReview***
- ***dismissPullRequestReview***
- ***moveProjectCard***
- ***moveProjectColumn***
- ***removeOutsideCollaborator***
- ***removeReaction***
- ***removeStar***
- ***requestReviews***
- ***submitPullRequestReview***
- ***updateProject***
- ***updateProjectCard***
- ***updateProjectColumn***
- ***updatePullRequestReview***
- ***updatePullRequestReviewComment***
- ***updateSubscription***
- ***updateTopics***

Objects

Objects in GraphQL represent the resources you can access. An object can contain a list of fields, which are specifically typed.

- ***AddedToProjectEvent***
- ***AssignedEvent***
- ***BaseRefChangedEvent***
- ***BaseRefForcePushedEvent***
- ***Blame***
- ***BlameRange***
- ***Blob***
- ***Bot***
- ***ClosedEvent***
- ***CodeOfConduct***

- ***CommentDeletedEvent***
- ***Commit***
- ***CommitComment***
- ***CommitCommentThread***
- ***ConvertedNoteToIssueEvent***
- ***CrossReferencedEvent***
- ***DemilestonedEvent***
- ***DeployKey***
- ***DeployedEvent***
- ***Deployment***
- ***DeploymentStatus***
- ***ExternalIdentity***
- ***ExternalIdentitySamlAttributes***
- ***ExternalIdentityScimAttributes***
- ***Gist***
- ***GistComment***
- ***GitActor***
- ***GitHubMetadata***
- ***GpgSignature***
- ***HeadRefDeletedEvent***
- ***HeadRefForcePushedEvent***
- ***HeadRefRestoredEvent***
- ***Issue***
- ***IssueComment***
- ***Label***
- ***LabeledEvent***
- ***Language***
- ***License***
- ***LicenseRule***
- ***LockedEvent***
- ***MarketplaceCategory***

- ***MarketplaceListing***
- ***MentionedEvent***
- ***MergedEvent***
- ***Milestone***
- ***MilestonedEvent***
- ***MovedColumnsInProjectEvent***
- ***Organization***
- ***OrganizationIdentityProvider***
- ***OrganizationInvitation***
- ***PageInfo***
- ***Project***
- ***ProjectCard***
- ***ProjectColumn***
- ***ProtectedBranch***
- ***PublicKey***
- ***PullRequest***
- ***PullRequestCommit***
- ***PullRequestReview***
- ***PullRequestReviewComment***
- ***PullRequestReviewThread***
- ***PushAllowance***
- ***RateLimit***
- ***Reaction***
- ***ReactionGroup***
- ***Ref***
- ***ReferencedEvent***
- ***Release***
- ***ReleaseAsset***
- ***RemovedFromProjectEvent***
- ***RenamedTitleEvent***
- ***ReopenedEvent***

- ***Repository***
- ***RepositoryInvitation***
- ***RepositoryInvitationRepository***
- ***RepositoryTopic***
- ***ReviewDismissalAllowance***
- ***ReviewDismissedEvent***
- ***ReviewRequest***
- ***ReviewRequestRemovedEvent***
- ***ReviewRequestedEvent***
- ***SmimeSignature***
- ***Status***
- ***StatusContext***
- ***SubscribedEvent***
- ***SuggestedReviewer***
- ***Tag***
- ***Team***
- ***Topic***
- ***Tree***
- ***TreeEntry***
- ***UnassignedEvent***
- ***UnknownSignature***
- ***UnlabeledEvent***
- ***UnlockedEvent***
- ***UnsubscribedEvent***
- ***User***
- ***UserContentEdit***