

Using GraphQL to write awesome golang APIs




Risha Mars

Software Engineer

 @marzipan

 @rmars

 @marzipan



Things we will learn

What is GraphQL?

Querying with GraphQL

Adding GraphQL to your app

Using GraphQL to connect different data types

Choosing a GraphQL library

What is GraphQL?

GraphQL is a **query language** for your API, and a **server-side runtime** for executing queries by using a **type system** you define for your data.

GraphQL isn't tied to any specific database or storage engine and is instead backed by your existing code and data.

source: <https://graphql.org/learn/>

Querying

- Specify only the data you need!
- Make one query for all the data you need!
- Get related data, without having to manually match it
- GraphQL playground

Adding GraphQL to your app

- Choose a GraphQL library
- Define your schema
- Add resolver functions to satisfy the schema

Demo app: <https://github.com/rmars/emojivoto>

Implementing the hello query

```
const Schema = `
schema {
  query: Query
}

type Query {
  hello: String!
}
`

type Resolver struct{}

func (r *Resolver) Hello() string {
  return fmt.Sprintf("HELLO WWG PEEPS!")
}
```

Connecting data

- Using resolvers, we can stitch together data from different sources!
- This is really cool because once you add the resolver, it's available anywhere!

```
func (r *userResolver) FavEmoji(...) (*emojiResolver, error) {  
    emojiRsp, err := r.emojiServiceClient.  
        FindByShortcode(Shortcode: r.u.FavEmoji)  
  
    return &emojiResolver{r.Resolver, emojiRsp.Emoji}, err  
}
```


Things I don't have time to tell you about

- Parameterized Queries
 - Add parameters to requests (e.g. `users(id: 1)`)
- Mutations
 - For adding / updating data
- Subscriptions
 - The server can send updates back to the client when data changes!
- Introspection
 - Lets you explore the GraphQL schema to see what queries it supports!
 - <https://graphql.org/learn/introspection/>

Choosing a GraphQL library

- <https://graphql.org/code/>
- Available for many languages!
- General advice:
 - Look at commit velocity
 - Look at open / resolved issues
 - Documentation
 - Help

GraphQL pros

Get exactly the data you request (small!)

Get all the data in a single request (fast!)

API exploration and documentation

Great for prototyping

Search together all the data!

GraphQL cons

It's easy to request a LOT of data

Watch your query depth!

Client caching (HTTP caching per URL not possible)

Cache your backend queries / db lookups!

Learning curve (not too bad though)

Learn GraphQL

and the REST is history!





bit.ly/wwg-graphql-talk