TASK

G.M.RISHITHA

Creating REST API using GO and MongoDB for the database:

ctx, cancel := context.WithTimeout(context.Background(), 20\*time.Second)

defer cancel()

client, err := mongo.Connect(ctx, options.Client().ApplyURI("mongodb://foo:bar@localhost:27017"))

if err != nil { return err }

This will create a new client and start monitoring the MongoDB server on localhost. The Database and Collection types can be used to access the database:

collection := client.Database("baz").Collection("qux")

A Collection can be used to query the database or insert documents:

res, err := collection.InsertOne(context.Background(), bson.M{"hello": "world"})

if err != nil { return err }

id := res.InsertedID

Several methods return a cursor, which can be used like this:

cur, err := collection.Find(context.Background(), bson.D{})

if err != nil { log.Fatal(err) }

defer cur.Close(context.Background())

for cur.Next(context.Background()) {

// To decode into a struct, use cursor.Decode()

result := struct{

Foo string

Bar int32

}{}

err := cur.Decode(&result)

if err != nil { log.Fatal(err) }

// do something with result...

// To get the raw bson bytes use cursor.Current

raw := cur.Current

// do something with raw...

}

if err := cur.Err(); err != nil {

return err

}

Cursor.All will decode all of the returned elements at once:

var results []struct{

Foo string

Bar int32

}

if err = cur.All(context.Background(), &results); err != nil {

log.Fatal(err)

}

Methods that only return a single document will return a \*SingleResult, which works like a \*sql.Row:

result := struct{

Foo string

Bar int32

}{}

filter := bson.D{{"hello", "world"}}

err := collection.FindOne(context.Background(), filter).Decode(&result)

if err != nil { return err }