Sangria

Scala GraphQL Implementation

Get a Hotel

```
type Hotel {
  id: String!
  name: String!
  cityId: String!
}

type Query {
  hotel(id: String!): Hotel
}
```

```scala
case class Hotel(id: String, name: String, destinationId: String)

class HotelRepository {
  def get(id: String): Future[Option[Hotel]] = ???
}
```

```scala
trait HcContext {
  def hotelRepo: HotelRepository
}

object SchemaDefinition {
  val HotelType = deriveObjectType[Unit, Hotel]()

  val ID = Argument("id", StringType)

  val hotel = Field(
    "hotel",
    OptionType(HotelType),
    arguments = List(ID),
    resolve = (c: Context[HcContext, Unit]) => c.ctx.hotelRepo.get(c.arg(ID))
  )

  val Query = ObjectType(
    "Query",
    fields(
      hotel
    )
  )

  val HolidayCheckSchema = Schema(Query)
}
```

```scala
trait HcContext {
  def hotelRepo: HotelRepository
}

object SchemaDefinition {
  val HotelType = deriveObjectType[Unit, Hotel]()

  val ID = Argument("id", StringType)

  val hotel = Field(
    "hotel",
    OptionType(HotelType),
    arguments = List(ID),
    resolve = (c: Context[HcContext, Unit]) => c.ctx.hotelRepo.get(c.arg(ID))
  )

  val Query = ObjectType(
    "Query",
    fields(
      hotel
    )
  )

  val HolidayCheckSchema = Schema(Query)
}
```

```
// ... some boilerplate

val context = new HcContext {
  override val hotelRepo: HotelRepository = new HotelRepository
}

val graphQL = GraphEndpoint(SchemaDefinition.HolidayCheckSchema,
                            context,
                            graphQLPath = "graphql")

Http().bindAndHandle(graphQL.route, "0.0.0.0", 7070)
```

```graphql
{
  hotel(id: "1aa4c4ad-f9ea-3367-a163-8a3a6884d450") {
    id
    name
    cityId
  }
}

```

```json
{
  "data": {
    "hotel": {
      "id": "1aa4c4ad-f9ea-3367-a163-8a3a6884d450",
      "name": "Dana Beach Resort",
      "cityId": "94773a8c-b71d-3be6-b57e-db9d8740bb98"
    }
  },
  "extensions": {}
}
```

Get the City of a Hotel

```
type Destination {
  id: String!
  name: String!
}

type Hotel {
  id: String!
  name: String!
  city: Destination
}

type Query {
  hotel(id: String!): Hotel
}
```

```scala
case class Destination(id: String, name: String)

class DestinationRepository {
  def get(id: String): Future[Option[Destination]] = ???
}
```

```scala
trait HcContext {
  def hotelRepo: HotelRepository
  def destinationRepo: DestinationRepository
}

object SchemaDefinition {
  val ID = Argument("id", StringType)

  val DestinationType = deriveObjectType[Unit, Destination]()

  val cityOfHotel = Field(
    "city",
    OptionType(DestinationType),
    resolve = (c: Context[HcContext, Hotel]) =>
c.ctx.destinationRepo.get(c.value.id)
  )

  val HotelType = deriveObjectType[HcContext, Hotel](
    ReplaceField("cityId", cityOfHotel)
  )

  val hotel = Field(
    "hotel",
    OptionType(HotelType),
    arguments = List(ID),
    resolve = (c: Context[HcContext, Unit]) => c.ctx.hotelRepo.get(c.arg(ID))
  )

  val Query = ObjectType(
    "Query",
    fields(hotel)
  )

  val HolidayCheckSchema = Schema(Query)
}
```

Get multiple Hotels

```
type Destination {
  id: String!
  name: String!
}

type Hotel {
  id: String!
  name: String!
  city: Destination
}

type Query {
  hotel(id: String!): Hotel
  hotels(limit: Int!): [Hotel!]!
}
```

```scala
object SchemaDefinition {

  // ...

  val Limit = Argument("limit", IntType)

  val hotels = Field(
    "hotels",
    ListType(HotelType),
    arguments = List(Limit),
    resolve = (c: Context[HcContext, Unit]) => c.ctx.hotelRepo.getAll(c.arg(Limit))
  )

  val Query = ObjectType(
    "Query",
    fields(hotel, hotels)
  )

  val HolidayCheckSchema = Schema(Query)
}
```

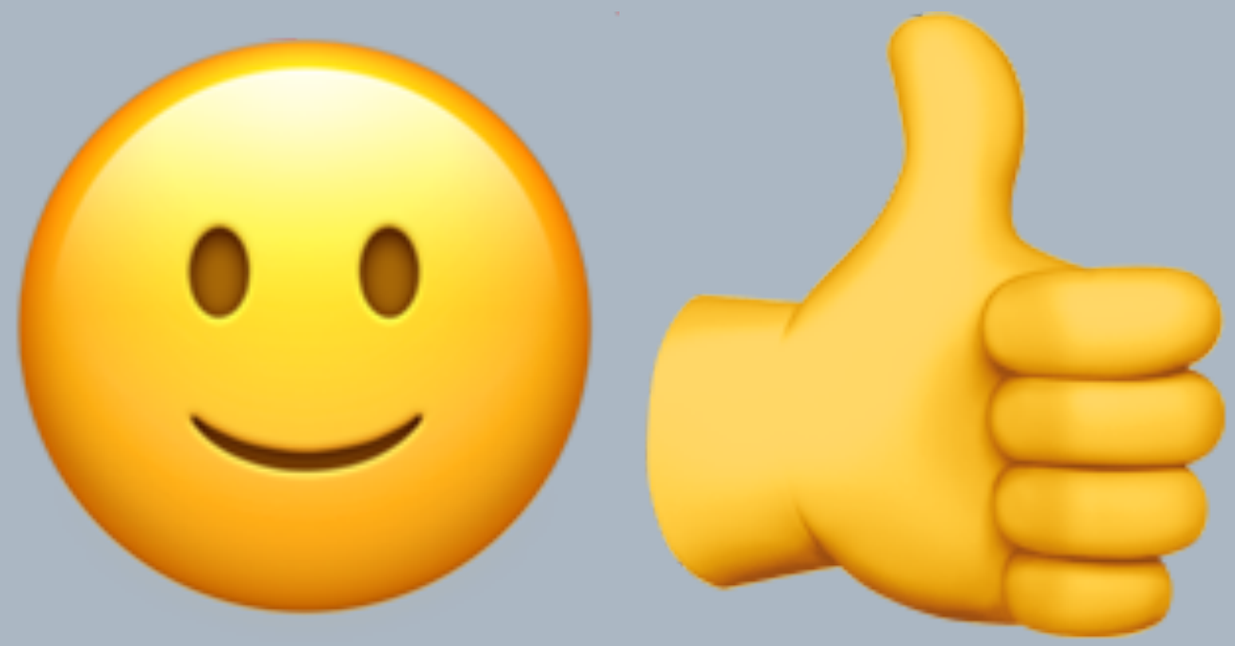
```
Fetching destination with id 94773a8c-b71d-3be6-b57e-db9d8740bb98
Fetching destination with id 94773a8c-b71d-3be6-b57e-db9d8740bb98
Fetching destination with id 07f5f656-4acc-3230-b7dd-aec3c13af37c
Fetching destination with id d84f18dd-c1fe-31e7-99e0-da0337d1deb7
Fetching destination with id d84f18dd-c1fe-31e7-99e0-da0337d1deb7
Fetching destination with id 436a3194-80d4-3fd4-8a64-af0d0f629ae6
Fetching destination with id 07f5f656-4acc-3230-b7dd-aec3c13af37c
Fetching destination with id 436a3194-80d4-3fd4-8a64-af0d0f629ae6
Fetching destination with id 94773a8c-b71d-3be6-b57e-db9d8740bb98
Fetching destination with id 94773a8c-b71d-3be6-b57e-db9d8740bb98
```

N + 1

Optimise City fetching

```scala
object Fetchers {

  val destinationFetcher =
    Fetcher.caching(
      (ctx: HcContext, ids: Seq[String]) => ctx.destinationRepo.getByIds(ids),
      FetcherConfig.maxBatchSize(30)
    )(HasId(_.id))

  val fetchers = DeferredResolver.fetchers(destinationFetcher)
}
```

```scala
object SchemaDefinition {
  // ...

  val cityOfHotel = Field(
    "city",
    OptionType(DestinationType),
    resolve = (c: Context[HcContext, Hotel]) =>
        // c.ctx.destinationRepo.get(c.value.cityId)
        destinationFetcher.defer(c.value.cityId)
  )

  // ...
}
```



Fetchers

Other interesting Features

Query Validaton

Permissions

Query Verification & Complexity Analysis

Reject too complex/expensive queries
Limit max query depth

Schema Comparison

Detect Breaking Changes

Stream-based Subscriptions

akka-streams, rxscala, monix, future

Middlewares

Profiling

Authentication & Authorisation

Deprecation tracking

🥂🍊 Cheers!