Overview

In Odyssey Lift-off III, we learned that we can define arguments for a field in our schema, which the field's resolver can use. In that course, we only defined a single argument for a field, so our schema stayed pretty simple and clean.

But in Airlock, some of our operations require several arguments all at once. How can we keep our schema easy to read while also providing our resolvers the data they need? The answer: with `input` types!

In this lesson, we will:

- Discover the benefits of using an `input` type for mutations
- See an example of how the `input` type is used in the Airlock codebase

What is the `input` type?

The `input` type in a GraphQL schema is a special object type that groups a set of arguments together, and can then be used as an argument to another field.

Using `input` types helps us group and understand arguments, especially for mutations. For example, when creating a listing in Airlock, we know the operation needs to include a bunch of data about the listing (title, description, photo thumbnail, number of beds, cost per night, location type, and amenities). We *could* list out all those arguments inside the `createListing` mutation, but that can get unwieldy and hard to understand.

Instead, we can create an input type called `CreateListingInput` and list all the necessary fields there. That way, we can keep the mutation definition clear, and we can also reuse that input type in other mutations that require the same set of arguments.

Defining an `input`

To define an input type, use the `input` keyword followed by the name and curly braces (`{}`). Inside the curly braces, we list the fields and types as usual. Note that fields of an input type can be only a scalar, an enum, or another input type.

Here's an example of another input type from the Airlock schema:

server/schema.graphql

```graphql
input SearchListingsInput {
  checkInDate: String!
  checkOutDate: String!
  numOfBeds: Int
  page: Int
  limit: Int
  sortBy: SortByCriteria # this is an enum type
}
```

Using the `input`

To use an `input` type in the schema, we can set it as the type of a field argument. For example, the `searchListings` query uses the `SearchListingsInput` type like so:

server/schema.graphql

```
type Query {
  # ...
  "Search results for listings that fit the criteria provided"
  searchListings(criteria: SearchListingsInput): [Listing]!
  # ...
}
```

> **Note:** Input types can be reused in multiple operations, but be careful when using the *same* input type for both queries and mutations! Some fields that may be required in mutations may not be required for queries.
>
> For more details, check out the Apollo docs on input types.

See it in the Airlock codebase

There are multiple inputs defined in the Airlock schema. Check them out in the `server/schema.graphql` file.

- What do you notice about the existing inputs?
- How are the inputs used in the client code?

Practice

How can we use the `input` type in our schema?

- [ ] To create form inputs in client applications
- [ ] For reuse in multiple operations

☐ To group and more easily understand argument fields

☐ As field arguments

Submit

Key takeaways

- The `input` type is a special object type used as arguments to fields. This allows us to group and understand all of our arguments together, especially for mutations.

Up next

In the next lesson, we'll learn another tool for grouping related attributes: **interfaces**.

← **Previous** **1 task remaining** ↑