61. Invalidating the cache after createPost() - Urql Client

#urql #cache #exchange #mutation #graphql #graphcache #frontend

Invalidating part of the list

- When we create a new post it is not immediately displayed on homepage. We can invalidate cache so that data is reloaded and new the post appears
- We *could* simply add the new post to the top of the list, but that's more error-prone since there could be race conditions between different clients etc
- We use the cacheExchange in Urql Client config and add the createPost mutation so that everytime it is executed the entire posts query in the cache will be invalidated (as opposed to invalidating a single post as in 76. Invalidating the cache after deletePost() - Urql Client)
- We also get the previousLimit (should never be null anyway) from the Query to pass it back to the new posts query that will be executed, so the same number of results are displayed.

/utils/createUrqlClient.ts

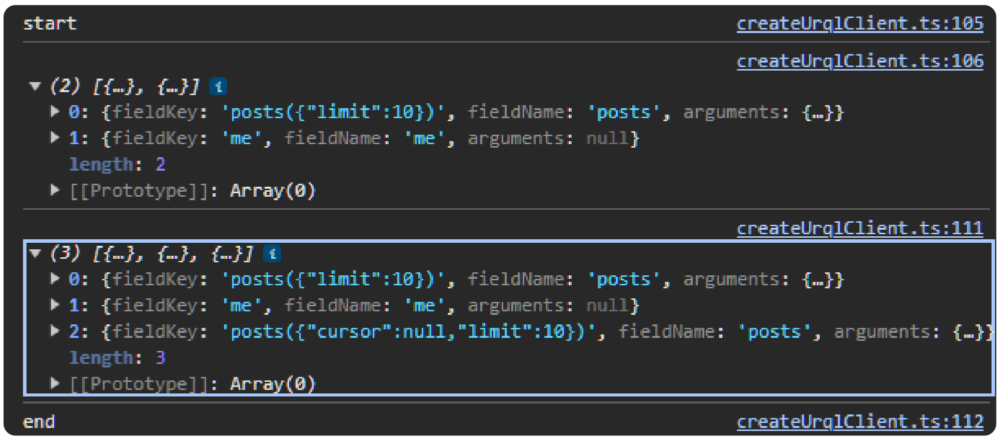
```
updates: {
  Mutation: {
    createPost: (result, args, cache, info) => {
      var previousLimit = cache
        .inspectFields("Query")
        .find((f) => f.fieldName === "posts")?.arguments?.limit as number;
      cache.invalidate("Query", "posts", {
        limit: previousLimit,
      });
    },

    logout: (result, args, cache, info) => { ...
```

- Note that adding `cursor: null` to the third parameter of cache.invalidate() results in a new query being created instead of replacing the previous query since the previous query does not have a `cursor` field in its fieldKey (see screenshot below)

⚠️ so this is wrong:

```
createPost: (result, args, cache, info) => {
  console.log("start");
  console.log(cache.inspectFields("Query"));
  cache.invalidate("Query", "posts", {
    limit: 10,
    cursor: null
  });
  console.log(cache.inspectFields("Query"));
  console.log("end");
},
```

and results in this:



```
start                                                            createUrqlClient.ts:105
                                                                 createUrqlClient.ts:106
▼ (2) [{…}, {…}]
  ▶ 0: {fieldKey: 'posts({"limit":10})', fieldName: 'posts', arguments: {…}}
  ▶ 1: {fieldKey: 'me', fieldName: 'me', arguments: null}
    length: 2
  ▶ [[Prototype]]: Array(0)
                                                                 createUrqlClient.ts:111
▼ (3) [{…}, {…}, {…}]
  ▶ 0: {fieldKey: 'posts({"limit":10})', fieldName: 'posts', arguments: {…}}
  ▶ 1: {fieldKey: 'me', fieldName: 'me', arguments: null}
  ▶ 2: {fieldKey: 'posts({"cursor":null,"limit":10})', fieldName: 'posts', arguments: {…}}
    length: 3
  ▶ [[Prototype]]: Array(0)
end                                                              createUrqlClient.ts:112
```

Invalidating the entire list

- But we could have a big paginated list created by clicking "load more" a couple of times, , how do we know which part of it to invalidate? : Answer is we don't, so we will actually have to invalidate the entire list, otherwise the other parts will remain in the cache and could cause errors / unexpected behaviour
- We will do it in a similar way we did the cursorPagination as follows
- Similar to the initial implementation, fi.arguments should never be null anyway

/utils/createUrqlClient.ts

```
createPost: (result, args, cache, info) => {
  const allFields = cache.inspectFields("Query");
  // filter allFields to get only the field infos related to the field we want to
work on
  const fieldInfos = allFields.filter(
    (info) => info.fieldName === "posts"
  );
  fieldInfos.forEach((fi) => {
    cache.invalidate("Query", "posts", fi.arguments || {});
  });
},
```