28. Disable urql cache with exchanges

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

---

- urql, by default, caches the data received from GraphQL. This causes problems when we login, logout or register since the page will reload using the data in the cache and will not update properly.
- We will use @urql/exchange-graphcache (formidable.com/open-source/urql/docs/grapchcace/) package to refresh the data everytime a user logins, logouts or registers

```
yarn add @urql/exchange-graphcache
```

- Now we add the exchanges to the createClient() function
- First we need a helper function betterUpdateQuery() because graphcache's updateQuery() function is not very good with types, and is also not very readable

utils/betterUpdateQuery.tsx

```
import { Cache, QueryInput } from "@urql/exchange-graphcache";

export function betterUpdateQuery<Result, Query>( // Query will be updated when
Result mutation is executed
  cache: Cache,
  qi: QueryInput, // input type of Query
  result: any,
  updaterFn: (r: Result, q: Query) => Query
) {
  return cache.updateQuery(qi, (data) => updaterFn(result, data as any) as any);
}
```

- updaterFn() is executed everytime the mutation is executed, in order to update the data of the Query, depending on the Result.
- r is result of the mutation, q is current (cached) state of Query

- The return type of updaterFn() matches the return type of Query. If the Result has errors, the value that is already in the cache is used, if not then the updated data is used (which may or may not be based on the Result)
- betterUpdateQuery() also allows us to properly **cast** the types, as can be seen in the code

update pages/_app.tsx

```
import { Provider, createClient, dedupExchange, fetchExchange } from 'urql'
import {
  LogoutMutation,
  MeQuery,
  MeDocument,
  LoginMutation,
  RegisterMutation,
} from "../generated/graphql";
import { cacheExchange } from "@urql/exchange-graphcache";
import { betterUpdateQuery } from "../utils/betterUpdateQuery";

function MyApp({ Component, pageProps }: AppProps) {
  const client = createClient({
    url: "http://localhost:4000/graphql",
    fetchOptions: {
      credentials: "include" as const,
    },
    exchanges: [
      dedupExchange,
      cacheExchange({
        // this will update the cache everytime the defined mutations are run
        updates: {
          Mutation: {
            logout: (result, args, cache, info) => {
              betterUpdateQuery<LogoutMutation, MeQuery>(
                cache,
                { query: MeDocument }, // e.g. MeQuery's input type is MeDocument
                result,
                () => ({ me: null }) // updaterFn - clear the query
              );
            },
            login: (result, args, cache, info) => {
              // cache.updateQuery({ query: MeDocument }, (data: MeQuery) => { })
              betterUpdateQuery<LoginMutation, MeQuery>(
                cache,
                { query: MeDocument },
                result,
                (r, q) => { // updaterFn
                  if (r.login.errors) {
                    return q; // return the current query if there's error
                } else {
                  return {
                    me: r.login.user, // return the user info received from
successful login
                  };
                }
              }
            );
          },
          register: (result, args, cache, info) => {
            betterUpdateQuery<RegisterMutation, MeQuery>(
              cache,
              { query: MeDocument },
              result,
              (r, q) => { // updaterFn
                if (r.register.errors) {
                  return q; // return the current query if there's error
                } else {
                  return {
                    me: r.register.user, // return the user info received from
successful register
                  };
                }
              }
            );
          },
        },
      },
    }),
    fetchExchange,
  ],
})

return (
  <Provider value={client}>
    <ChakraProvider theme={theme}>
      <Component {...pageProps} />
    </ChakraProvider>
  </Provider>
  );
}
```