REACT

5. forduló

A kategória támogatója: TCS - Tata Consultancy Services

Ismertető a feladathoz

Felhasznált idő: 00:00/20:00 **Elért pontszám:** 0/8

1. feladat 0/4 pont

Az alábbi egyszerű "***logger***" middleware-t akarjuk használni a ***store***-al:

```
export default function logger({ getState }) {
    return next => action => {
        console.log('before', action);

        const returnValue = next(action);

        console.log('after', getState());

        return returnValue;
    }
}
```

Futáskor azonban bár a **store** működik, nem kapunk logot, azaz sem 'before' sem 'after' nincs a Console-ban. **Hogyan kell "regisztrálni", ha kell-e egyáltalán a "*logger*"-t?**

Válaszok

- [ ] A regisztráció automatikus:

```
export const store = configureStore({
    reducer: { counter: counterReducer },
});
```

- [ ] A regisztráció kötelező "konfigurációs" lépés:

```
export const store = configureStore({
    reducer: { counter: counterReducer },
```

```
            middle: () => [logger]
        });
```

- [ ] Nem kell regisztrálni, a logger függvény hibás. A végén nem a `***returnValue***`-t, hanem magát az `***action***`-t kell visszaadni. Azaz`***return returnValue***` -> `***return action***`

- [ ] Kell regisztrálni, és a logger függvény is hibás. A végén nem a`***returnValue***`-t, hanem magát az `***action***`-t kell visszaadni. Azaz `***return returnValue***` -> `***return action***`, illetve a regisztráció helyes kódja:

```
        export const store = configureStore({
            reducer: { counter: counterReducer },
            middleware: [logger]
        });
```

- [x]
```
    export const store = configureStore({
            reducer: { counter: counterReducer },
            middleware: [logger]
        });
```

Ez a válasz helyes, de nem jelölted meg.

- [x] Nem kell mást tenni csak:

```
        export const store = configureStore({
            reducer: { counter: counterReducer },
            middleware: () => new Array(logger)
        });
```

Ez a válasz helyes, de nem jelölted meg.

Magyarázat

A Redux Toolkit használata esetén is kötelező a middleware-ek regisztációja.

A dokumentáció alapján a **middleware** konfigurációs property értéke lehet egy tömb illetve egy függvény ami egy tömbbel tér vissza.

A logger middleware helyesen van megírva, mivel a **next(action)** értével szokás visszatérni lásd szintén a dokumentációban.

Ezért a

```
export const store = configureStore({
            reducer: { counter: counterReducer },
            middleware: [logger]
        });
```

és a

```
 export const store = configureStore({
            reducer: { counter: counterReducer },
            middleware: () => new Array(logger)
        });
```

válaszok **helyesek**.

2. feladat 0/4 pont

A Redux store működése Sync, azaz egy network kérést beleilleszteni "csak úgy" nem lehet. Sok töprengés után úgy határoztunk, hogy próbát teszünk az "RTK Query"-vel.

Az alábbi kód alapján a liberal api-t probáljuk elérni:

```
import { createApi, fetchBaseQuery } from '@reduxjs/toolkit/query/react'

export const liberalApi = createApi({
    reducerPath: 'liberalApi',
    baseQuery: fetchBaseQuery({ baseUrl: 'https://liberalapi.co/api/v2/' }),
    tagTypes: [],
    endpoints: (builder) => ({
        getLiberalByName: builder.query({
            query: name => `liberal/${name}`,
        }),
    }),
})

// Export hooks for usage in functional components
export const { useGetLiberalByNameQuery } = liberalApi
```

Minden tökéletesen működik, a **kód helyes-e**?

Válasz

- ○ Igen, bár a query-k nincsenek lokálisan "kesselve" - caching
- ● Igen, és a "kesselés" is benne van
  Ez a válasz helyes, de nem jelölted meg.
- ○ A kód hibás, mivel nincs olyan, hogy `***useGetLiberalByNameQuery***`
- ○ A kód hibás, mert a `***createApi***` csak unit teszteknél használható és lassú
- ○ A kód hibás, mert a `***baseUrl***`-ben a protocol értéke már adott, azaz a url-el: "https://https://" fog kezdődni
- ○ A kód hibás, mert a **useGetLiberalByNameQuery** egy hook

Magyarázat

A kód maga szintaktikailag helyes és **createApi** egy olyan Redux "szeletet" (slice) generál, ami már tartalmazza a megfelelő adat lekérő és "kesselő" logikát.

Ezért a

"*Igen, bár a query-k nincsenek lokálisan "kesselve" - caching*" válasz **helytelen**.

És a

"*Igen, és a "kesselés" is benne van*" válasz **helyes**.

A **useGetLiberalByNameQuery** egy autómatikusan az endpoint nevőből generált hook ezért a

"*A kód hibás, mivel nincs olyan, hogy `**useGetLiberalByNameQuery**`*" válasz **helytelen**.

A **createApi** a RTK belépési pontja ezért nem csak unit teszteknél használható,

ezért "*A kód hibás, mert a `**createApi**` csak unit teszteknél használható és lassú*" válasz **helytelen**.

A **baseUrl**-ben meg kell adni a a protocol-t is lásd dokumentációban,

ezért "*A kód hibás, mert a `**baseUrl**`-ben a protocol értéke már adott, azaz a url-el: "https://https://*" fog kezdődni" válasz **helytelen**.

3. feladat 0/0 pont

Redux-ot használunk, az alábbi egyszerű reducer függvénnyel helyes-e?

```
function counterReducer(state = { value: 0 }, action) {
        switch (action.type) {
                case 'counter/incremented':
                        state.value + 1;
                        break;
                case 'counter/decremented':
                        state.value - 1;
                        break;
                default:
                        return state
        }
}
```

Válaszok

- [x] Nem, mert a **state**-et direkt változtatjuk
  Ez a válasz helyes, de nem jelölted meg.
- [x] Igen, ha **Redux-Toolkit**-et használunk
  Ez a válasz helyes, de nem jelölted meg.
- [x] Igen, ha **Redux-Toolkit**-et használunk, mert az immer lib létrehozza az új state-et
  Ez a válasz helyes, de nem jelölted meg.
- [ ] Igen, mert a **state**-et direkt változtatjuk

Magyarázat

Kedves Versenyzők!

A feladatot 0 pontosra állítottuk, mivela state mutációja rosszul illusztrált.

Köszönjük szépen megértéseket!

Az alábbi dokumentáció megmutatja az immer-reducers használatát:

https://redux-toolkit.js.org/usage/immer-reducers