src\state\utils\loader\loader.reducer.ts

export const initialLoaderState: LoaderState<any> = {

loading: false,

error: false,

success: false,

value: undefined,

};

/\*\*

\* Higher order reducer that adds generic loading flag to chunk of the state

\*

\* Utilizes "loader" meta field of actions to set specific flags for specific

\* action (LOAD, SUCCESS, FAIL, RESET)

\*/

export function loaderReducer<T>(

entityType: string,

reducer?: (state: T, action: Action) => T

): (state: LoaderState<T>, action: LoaderAction) => LoaderState<T> {

return (

state: LoaderState<T> = initialLoaderState,

action: LoaderAction

): LoaderState<T> => {

if (

action.meta &&

action.meta.loader &&

action.meta.entityType === entityType

) {

const entity = action.meta.loader;

if (entity.load) {

return {

...state,

loading: true,

value: reducer ? reducer(state.value, action) : state.value,

};

} else if (entity.error) {

return {

...state,

loading: false,

error: true,

success: false,

value: reducer ? reducer(state.value, action) : undefined,

};

} else if (entity.success) {

return {

...state,

value: reducer ? reducer(state.value, action) : action.payload,

loading: false,

error: false,

success: true,

};

} else {

// reset state action

return {

...initialLoaderState,

value: reducer

? reducer(initialLoaderState.value, action)

: initialLoaderState.value,

};

}

}

if (reducer) {

const newValue = reducer(state.value, action);

if (newValue !== state.value) {

return { ...state, value: newValue };

}

}

return state;

};

}

export const USER\_FEATURE = 'user';

export const UPDATE\_EMAIL\_PROCESS\_ID = 'updateEmail';

export const UPDATE\_PASSWORD\_PROCESS\_ID = 'updatePassword';

export const UPDATE\_USER\_DETAILS\_PROCESS\_ID = 'updateUserDetails';

export const REGISTER\_USER\_PROCESS\_ID = 'registerUser';

export const REMOVE\_USER\_PROCESS\_ID = 'removeUser';

export const GIVE\_CONSENT\_PROCESS\_ID = 'giveConsent';

export const WITHDRAW\_CONSENT\_PROCESS\_ID = 'withdrawConsent';

export const UPDATE\_NOTIFICATION\_PREFERENCES\_PROCESS\_ID =

'updateNotificationPreferences';

export const ADD\_PRODUCT\_INTEREST\_PROCESS\_ID = 'addProductInterests';

export const REMOVE\_PRODUCT\_INTERESTS\_PROCESS\_ID = 'removeProductInterests';

export const CANCEL\_ORDER\_PROCESS\_ID = 'cancelOrder';

export const CANCEL\_RETURN\_PROCESS\_ID = 'cancelReturn';

export const USER\_CONSENTS = '[User] User Consents';

export const USER\_PAYMENT\_METHODS = '[User] User Payment Methods';

export const USER\_ORDERS = '[User] User Orders';

export const USER\_ADDRESSES = '[User] User Addresses';

export const USER\_RETURN\_REQUESTS = '[User] Order Return Requests';

export const USER\_RETURN\_REQUEST\_DETAILS = '[User] Return Request Details';

export const USER\_ORDER\_DETAILS = '[User] User Order Details';

export const REGIONS = '[User] Regions';

export const CUSTOMER\_COUPONS = '[User] Customer Coupons';

export const SUBSCRIBE\_CUSTOMER\_COUPON\_PROCESS\_ID = 'subscribeCustomerCoupon';

export const UNSUBSCRIBE\_CUSTOMER\_COUPON\_PROCESS\_ID =

'unsubscribeCustomerCoupon';

export const CLAIM\_CUSTOMER\_COUPON\_PROCESS\_ID = 'claimCustomerCoupon';

export const NOTIFICATION\_PREFERENCES = '[User] Notification Preferences';

export const PRODUCT\_INTERESTS = '[User] Product Interests';

export interface StateWithUser {

[USER\_FEATURE]: UserState;

}

export interface UserState {

account: UserDetailsState;

addresses: LoaderState<Address[]>;

consents: LoaderState<ConsentTemplate[]>;

billingCountries: BillingCountriesState;

countries: DeliveryCountriesState;

payments: LoaderState<PaymentDetails[]>;

orders: LoaderState<OrderHistoryList>;

order: LoaderState<Order>;

orderReturn: LoaderState<ReturnRequest>;

orderReturnList: LoaderState<ReturnRequestList>;

titles: TitlesState;

regions: LoaderState<RegionsState>;

resetPassword: boolean;

consignmentTracking: ConsignmentTrackingState;

customerCoupons: LoaderState<CustomerCouponSearchResult>;

notificationPreferences: LoaderState<NotificationPreference[]>;

productInterests: LoaderState<ProductInterestSearchResult>;

}

export interface RegionsState {

entities: Region[];

country: string;

}

export interface BillingCountryEntities { [key: string]: Country;}

export interface BillingCountriesState { entities: BillingCountryEntities;}

export interface DeliveryCountryEntities { [key: string]: Country;}

export interface DeliveryCountriesState { entities: DeliveryCountryEntities;}

export interface TitleEntities { [key: string]: Title;}

export interface TitlesState { entities: TitleEntities;}

export interface UserDetailsState { details: User;}

export interface ConsignmentTrackingState { tracking?: ConsignmentTracking;}

export function getReducers(): ActionReducerMap<UserState> {

return {

account: combineReducers({

details: fromUserDetailsReducer.reducer,

}),

addresses: loaderReducer<Address[]>(

USER\_ADDRESSES,

fromAddressesReducer.reducer

),

billingCountries: fromBillingCountriesReducer.reducer,

consents: loaderReducer<ConsentTemplate[]>(

USER\_CONSENTS,

fromUserConsentsReducer.reducer

),

payments: loaderReducer<PaymentDetails[]>(

USER\_PAYMENT\_METHODS,

fromPaymentReducer.reducer

),

orders: loaderReducer<OrderHistoryList>(

USER\_ORDERS,

fromUserOrdersReducer.reducer

),

order: loaderReducer<Order>(

USER\_ORDER\_DETAILS,

fromOrderDetailsReducer.reducer

),

orderReturn: loaderReducer<ReturnRequest>(USER\_RETURN\_REQUEST\_DETAILS),

orderReturnList: loaderReducer<ReturnRequestList>(

USER\_RETURN\_REQUESTS,

fromOrderReturnRequestReducer.reducer

),

countries: fromDeliveryCountries.reducer,

titles: fromTitlesReducer.reducer,

regions: loaderReducer<RegionsState>(REGIONS, fromRegionsReducer.reducer),

resetPassword: fromResetPasswordReducer.reducer,

consignmentTracking: fromConsignmentTrackingReducer.reducer,

customerCoupons: loaderReducer<CustomerCouponSearchResult>(

CUSTOMER\_COUPONS,

fromCustomerCouponReducer.reducer

),

notificationPreferences: loaderReducer<NotificationPreference[]>(

NOTIFICATION\_PREFERENCES,

fromNotificationPreferenceReducer.reducer

),

productInterests: loaderReducer<ProductInterestSearchResult>(

PRODUCT\_INTERESTS,

fromInterestsReducer.reducer

),

};

}

export const reducerToken: InjectionToken<ActionReducerMap<

UserState

>> = new InjectionToken<ActionReducerMap<UserState>>('UserReducers');

export const reducerProvider: Provider = {

provide: reducerToken,

useFactory: getReducers,

};

export function clearUserState(

reducer: ActionReducer<any>

): ActionReducer<any> {

return function (state, action) {

if (action.type === AuthActions.LOGOUT) {

state = undefined;

}

return reducer(state, action);

};

}

export const metaReducers: MetaReducer<any>[] = [clearUserState];