D:\应用程序案例\core\src\process\process.module.ts

@NgModule({

imports: [ProcessStoreModule],

})

export class ProcessModule {

static forRoot(): ModuleWithProviders<ProcessModule> {

return {

ngModule: ProcessModule,

providers: [],

};

}

}

D:\应用程序案例\\core\src\process\store\process-store.module.ts

mport { StateModule } from 'zyapp/core/state/state.module';

import { PROCESS\_FEATURE } from './process-state';

import { reducerProvider, reducerToken } from './reducers/index';

@NgModule({

imports: [StateModule, StoreModule.forFeature(PROCESS\_FEATURE, reducerToken)],

providers: [reducerProvider],

})

export class ProcessStoreModule {}

D:\应用程序案例\\core\src\process\store\process-state.ts

import { EntityLoaderState } from '../../state/utils/entity-loader/entity-loader-state';

export const PROCESS\_FEATURE = 'process';

export interface StateWithProcess<T> {

[PROCESS\_FEATURE]: EntityLoaderState<T>;

}

export type EntityLoaderState<T> = EntityState<LoaderState<T>>;

export interface EntityState<T> {

entities: {

[id: string]: T;

};

}

export interface LoaderState<T> {

loading?: boolean;

error?: boolean;

success?: boolean;

value?: T;

}

D:\应用程序案例\core\src\process\store\reducers\index.ts

export function getReducers<T>(): ActionReducer<EntityLoaderState<T>> {

return entityLoaderReducer(PROCESS\_FEATURE);

}

export const reducerToken: InjectionToken<ActionReducerMap<

EntityLoaderState<any>

>> = new InjectionToken<ActionReducerMap<EntityLoaderState<any>>>(

'ProcessReducers'

);

export const reducerProvider: Provider = {

provide: reducerToken,

useFactory: getReducers,

};

D:\应用程序案例\\core\src\process\store\selectors\index.ts

import \* as ProcessSelectors from './process.selectors';

export { ProcessSelectors };

D:\应用程序案例\process\store\selectors\process-group.selectors.ts

export \* from './feature.selector';

export \* from './process.selectors';

D:\应用程序案例\process\store\selectors\feature.selector.ts

import { MemoizedSelector, createFeatureSelector } from '@ngrx/store';

import { StateWithProcess, PROCESS\_FEATURE } from '../process-state';

import { EntityLoaderState } from '../../../state/utils/entity-loader/entity-loader-state';

export function getProcessState<T>(): MemoizedSelector<

StateWithProcess<T>,

EntityLoaderState<T>

> {

return createFeatureSelector<EntityLoaderState<T>>(PROCESS\_FEATURE);

}

D:\应用程序案例\process\store\selectors\process.selectors.ts

import { createSelector, MemoizedSelector } from '@ngrx/store';

import { StateEntityLoaderSelectors } from '../../../state/utils/index';

import { LoaderState } from '../../../state/utils/loader/loader-state';

import {

loaderErrorSelector,

loaderLoadingSelector,

loaderSuccessSelector,

} from '../../../state/utils/loader/loader.selectors';

import { StateWithProcess } from '../process-state';

import { getProcessState } from './feature.selector';

export function getProcessStateFactory<T>(

processId: string

): MemoizedSelector<StateWithProcess<T>, LoaderState<T>> {

return createSelector(getProcessState(), (entityState) =>

StateEntityLoaderSelectors.entityStateSelector(entityState, processId)

);

}

export function getProcessLoadingFactory<T>(

processId: string

): MemoizedSelector<StateWithProcess<T>, boolean> {

return createSelector(getProcessStateFactory(processId), (loaderState) =>

loaderLoadingSelector(loaderState)

);

}

export function getProcessSuccessFactory<T>(

processId: string

): MemoizedSelector<StateWithProcess<T>, boolean> {

return createSelector(getProcessStateFactory(processId), (loaderState) =>

loaderSuccessSelector(loaderState)

);

}

export function getProcessErrorFactory<T>(

processId: string

): MemoizedSelector<StateWithProcess<T>, boolean> {

return createSelector(getProcessStateFactory(processId), (loaderState) =>

loaderErrorSelector(loaderState)

);

}