Injecting Reducers

要将根reducers注入到应用程序中，请使用一个InjectionToken和一个Provider来通过依赖注入注册reducers。

import { SomeService } from './some.service';

import \* as fromRoot from './reducers';

export const REDUCER\_TOKEN =

new InjectionToken<ActionReducerMap<fromRoot.State>>('Registered Reducers', {

factory: () => {

const serv = inject(SomeService);

// return reducers synchronously

return serv.getReducers();

}

});

@NgModule({

imports: [StoreModule.forRoot(REDUCER\_TOKEN)]

})

export class AppModule {}

当通过特征模块组合状态时，Reducers也被注入。

import \* as fromFeature from './reducers';

export const FEATURE\_REDUCER\_TOKEN = new InjectionToken<

ActionReducerMap<fromFeature.State>

>('Feature Reducers');

export function getReducers(): ActionReducerMap<fromFeature.State> {

// map of reducers

return {};

}

@NgModule({

imports: [StoreModule.forFeature(fromFeature.featureKey, FEATURE\_REDUCER\_TOKEN)],

providers: [

{

provide: FEATURE\_REDUCER\_TOKEN,

useFactory: getReducers,

},

],

})

export class FeatureModule {}

Injecting Meta-Reducers

要注入“middleware(中间件)”meta reducers，使用在Store API中导出的META\_REDUCERS注入令牌和提供者通过依赖注入注册reducers。

import { MetaReducer, META\_REDUCERS } from '@ngrx/store';

import { SomeService } from './some.service';

import \* as fromRoot from './reducers';

export function metaReducerFactory(): MetaReducer<fromRoot.State> {

return (reducer: ActionReducer<any>) => (state, action) => {

console.log('state', state);

console.log('action', action);

return reducer(state, action);

};

}

@NgModule({

providers: [

{

provide: META\_REDUCERS,

deps: [SomeService],

useFactory: metaReducerFactory,

multi: true,

},

],

})

export class AppModule {}

应该注意在提供程序中使用META\_REDUCERS的multi属性。由于这个注入令牌可能被许多库并发使用，所以指定multi: true对于确保所有库元简化程序都应用于任何使用多个NgRx库并注册了元简化程序的项目是非常关键的。

Injecting Feature Config

注入的功能配置

要将特性存储配置注入到模块中，请使用一个InjectionToken和一个Provider来通过依赖注入注册特性配置对象

import \* as fromFeature from './reducers';

export const FEATURE\_CONFIG\_TOKEN =

new InjectionToken<StoreConfig<fromFeature.State>>('Feature Config');

export function getConfig(someService: SomeService): StoreConfig<fromFeature.State> {

// return the config synchronously.

return {

initialState: someService.getInitialState(),

metaReducers: [

fromFeature.loggerFactory(someService.loggerConfig())

]

};

}

@NgModule({

imports: [StoreModule.forFeature(fromFeature.featureKey,

fromFeature.reducers, FEATURE\_CONFIG\_TOKEN)],

providers: [

{

provide: FEATURE\_CONFIG\_TOKEN,

deps: [SomeService],

useFactory: getConfig,

},

],

})

export class FeatureModule {}