/\*\*

\* 所有实体类型的EntityCollectionServices中心注册中心的基/默认类。

\* 创建您自己的子类来添加特定于应用程序的成员，以改善开发者体验。

\* @example

\* export class EntityServices extends EntityServicesBase {

\* constructor(entityServicesElements: EntityServicesElements) {

\* super(entityServicesElements);

\* }

\* // 方便属性返回一个类型化的自定义实体收集服务

\* get companyService() {

\* return this.getEntityCollectionService<Model.Company>('Company') as CompanyService;

\* }

\* // Convenience dispatch methods

\* clearCompany(companyId: string) {

\* this.dispatch(new ClearCompanyAction(companyId));

\* }

\* }

\*/

@Injectable()

export class EntityServicesBase implements EntityServices {

Dear @ngrx/data developer: 在更改构造函数之前，请仔细考虑.

这样做会破坏派生自这个基类的应用程序，许多应用程序将派生自这个类。

不要给这个构造函数一个实现。

这样做会使mock从这个类派生的类变得困难。

使用getter属性代替。例如，请参见entityCache$

**constructor**(private entityServicesElements: EntityServicesElements) {}

// #region EntityServicesElement-based properties

/\*\* Observable of error EntityActions (e.g. QUERY\_ALL\_ERROR) for all entity types \*/

get **entityActionErrors**$(): Observable<EntityAction> {

return this.entityServicesElements.entityActionErrors$;

}

/\*\* Observable of the entire entity cache \*/

get **entityCache**$(): Observable<EntityCache> | Store<EntityCache> {

return this.entityServicesElements.entityCache$;

}

/\*\* 工厂创建EntityCollectionService的默认实例 \*/

get **entityCollectionServiceFactory**(): EntityCollectionServiceFactory {

return this.entityServicesElements.entityCollectionServiceFactory;

}

/\*\*

\* 商店用减数器处理后扫描的动作。

\*最近的动作被商店减少的回放观察。

\*/

get reducedActions$(): Observable<Action> {

return this.entityServicesElements.reducedActions$;

}

/\*\* The ngrx store, scoped to the EntityCache \*/

protected get store(): Store<EntityCache> {

return this.entityServicesElements.store;

}

// #endregion EntityServicesElement-based properties

/\*\* Dispatch any action to the store \*/

dispatch(action: Action) {

this.store.dispatch(action);

}

/\*\* Registry of EntityCollectionService instances \*/

private readonly EntityCollectionServices: EntityCollectionServiceMap = {};

/\*\*

创建一个新的EntityCollectionService默认实例。

\*首选getEntityCollectionService()，除非你真的想要一个新的默认实例。

\*这个将不会被注册到EntityServices!

\* @param entityName {string} Name of the entity type of the service

\*/

protected **createEntityCollectionService**<T,S$ extends EntitySelectors$<T> = EntitySelectors$<T> >

(entityName: string): EntityCollectionService<T> {

return this.entityCollectionServiceFactory.create<T, S$>(entityName);

}

/\*\* Get (or create) the singleton instance of an EntityCollectionService

\* @param entityName {string} 服务实体类型的名称

\*/

**getEntityCollectionService**< T, S$ extends EntitySelectors$<T> = EntitySelectors$<T>>(entityName: string): EntityCollectionService<T> {

let service = this.EntityCollectionServices[entityName];

if (!service) {

service = this.createEntityCollectionService<T, S$>(entityName);

this.EntityCollectionServices[entityName] = service;

}

return service;

}

/\*\* 在其实体类型名称下注册EntityCollectionService。

\*将替换该类型的现有服务。

\* @param service {EntityCollectionService} The entity service

\* @param serviceName {string} optional service name to use instead of the service's entityName

\*/

**registerEntityCollectionService**<T>(

service: EntityCollectionService<T>,

serviceName?: string

) {

this.EntityCollectionServices[serviceName || service.entityName] = service;

}

/\*\*

\* 一次注册多个实体类型的实体服务

\*将替换该类型的现有服务。

\* @param entityCollectionServices {EntityCollectionServiceMap | EntityCollectionService<any>[]}

\* EntityCollectionServices to register, either as a map or an array

\*/

**registerEntityCollectionServices**(

entityCollectionServices:

| EntityCollectionServiceMap

| EntityCollectionService<any>[]

): void {

if (Array.isArray(entityCollectionServices)) {

entityCollectionServices.forEach((service) =>

this.registerEntityCollectionService(service)

);

} else {

Object.keys(entityCollectionServices || {}).forEach((serviceName) => {

this.registerEntityCollectionService(

entityCollectionServices[serviceName],

serviceName

);

});

}

}

}