/\*\*

\* Base class for a concrete EntityCollectionService<T>.

\* 可以实例化。不能被注入。使用EntityCollectionServiceFactory创建。

\* @param EntityCollectionServiceElements这个服务的成分，它是用于创建EntityCollectionService<T>实例的支持服务的来源。

\*/

export declare class EntityCollectionServiceBase<T, S$ extends EntitySelectors$<T> = EntitySelectors$<T>> implements EntityCollectionService<T> {

/\*\* 此集合服务的实体类型的名称 \*/

**readonly entityName: string;**

/\*\* EntityCommands (EntityActions)调度器 \*/

**readonly dispatcher: EntityDispatcher<T>;**

/\*\*实体集合属性的所有选择器 \*/

**readonly selectors: EntitySelectors<T>;**

/\*\* 所有选择器$(实体集合属性的可观察对象)\*/

**readonly selectors$: S$;**

**constructor**(

/\*\* 此集合服务的实体类型的名称

entityName: string,

/\*\* 为这个实体类型创建EntityCollectionService的核心元素 \*/

serviceElementsFactory: EntityCollectionServiceElementsFactory

);

/\*\*为这个实体类型创建一个{EntityAction}。

@param op {EntityOp}实体操作

\* @param [data]动作数据

\* @param [options]附加选项

@返回EntityAction

\*/

**createEntityAction<P = any>(op: EntityOp, data?: P, options?: EntityActionOptions): EntityAction<P>;**

/\*\*

为这个实体类型和创建一个{EntityAction}

立即把它送到商店。

@param op {EntityOp}实体操作

\* @param [data]动作数据

\* @param [options]附加选项

\* @返回已调度的EntityAction

\*/

**createAndDispatch<P = any>(op: EntityOp, data?: P, options?: EntityActionOptions): EntityAction<P>;**

/\*\*

\* 将任何类型的动作分派到ngrx存储区

\* @param action the Action

\* @returns 返回已分派的操作

\*/

**dispatch(action: Action): Action;**

/\*\* The NgRx Store for the {EntityCache} \*/

**get store(): Store<import("..").EntityCache>;**

/\*\* 具有验证EntityAction有效负载的方法的实用程序类。

**guard: EntityActionGuard<T>;**

/\*\*返回此实体的主键(id)\*/

**selectId: IdSelector<T>;**

/\*\*

\* 将一个实体(或部分实体)转换为' Update<T> '对象

\* `update...` and `upsert...` methods take `Update<T>` args

\*/

**toUpdate: (entity: Partial<T>) => Update<T>;**

/\*\*

\*调度动作保存一个新的实体到远程存储。

\* @param entity要添加的实体，如果悲观且服务器创建了键，则可能忽略它的键;

\*如果乐观保存必须有一个键。

\* @param [options]选项影响保存和合并行为

返回实体的可观察对象

\*服务器报告成功保存或保存错误后。

\*/

**add(entity: T, options?: EntityActionOptions): Observable<T>;**

/\*\*

\*调度操作取消持久化操作(查询或保存)与给定的correlationId。

\* @param correlationId对应EntityAction的相关id

\* @param [reason]解释为什么取消和由谁取消。

\* @param [options]选项，如标签和合并策略

\*/

**cancel(correlationId: any, reason?: string, options?: EntityActionOptions): void;**

/\*\*

\*调度行动删除实体从远程存储键。

\* @param key要删除的实体

\* @param [options]选项影响保存和合并行为

@return被删除键的可观察对象

\*服务器报告成功保存或保存错误后。

\*/

**delete(entity: T, options?: EntityActionOptions): Observable<number | string>;**

/\*\*

\*调度行动删除实体从远程存储键。

\* @param key要删除的实体的主键

\* @param [options]选项影响保存和合并行为

@return被删除键的可观察对象

\*服务器报告成功保存或保存错误后。进入翻译页面

\*/

**delete(key: number | string, options?: EntityActionOptions): Observable<number | string>;**

/\*\*

\*调度操作查询远程存储的所有实体和

将查询到的实体合并到缓存集合中。

@param [options]影响合并行为的选项

返回集合的可观察对象

\*当服务器报告查询成功或查询错误时。

\* @see负载()

\*/

**getAll(options?: EntityActionOptions): Observable<T[]>;**

/\*\*

\*调度操作查询远程存储的实体与这个主键。

\*如果服务器返回一个实体，

\*合并到缓存集合。

\* @param key要获取的实体的主键。

@param [options]影响合并行为的选项

返回集合中被查询实体的可观察对象

\*服务器报告成功或查询错误后。

\*/

**getByKey(key: any, options?: EntityActionOptions): Observable<T>;**

/\*\*

\*调度操作来查询满足查询的实体的远程存储

\*查询参数map或HTTP URL查询字符串，

\*并将结果合并到缓存集合中。

\* @param query以服务器能理解的形式对查询进行参数化

@param [options]影响合并行为的选项

\* @return被查询实体的可观察对象

\*当服务器报告查询成功或查询错误时。

\*/

**getWithQuery(queryParams: QueryParams | string, options?: EntityActionOptions): Observable<T[]>;**

/\*\*

\*调度操作查询远程存储的所有实体和

\*完全替换缓存的集合与查询的实体。

\* @param [options]影响加载行为的选项

返回集合的可观察对象

\*当服务器报告查询成功或查询错误时。

\* @see getAll

\*/

**load(options?: EntityActionOptions): Observable<T[]>;**

/\*\*

\*调度行动，以保存更新的实体(或部分实体)在远程存储。

\*更新实体可能是部分的(但必须有它的键)

\*在这种情况下，它修补现有的实体。

\* @param实体更新实体，它可能是T的一部分，但必须至少有它的键。

\* @param [options]选项影响保存和合并行为

\* @返回更新后实体的可观察对象

\*服务器报告成功保存或保存错误后。

\*/

**update(entity: Partial<T>, options?: EntityActionOptions): Observable<T>;**

/\*\*

\*调度动作保存一个新的或现有的实体到远程存储。

\*服务器支持upsert时才呼叫。

\* @param实体实体添加或升级。

\*它可能忽略它的键，如果添加，并且是悲观的，并且服务器创建键;

\*如果乐观保存必须有一个键。

\* @param [options]选项影响保存和合并行为

返回实体的可观察对象

\*服务器报告成功保存或保存错误后。

\*/

**upsert(entity: T, options?: EntityActionOptions): Observable<T>;**

/\*\*\* 不更新远程存储的仅缓存操作

/\*\*

\*替换缓存集合中的所有实体。

\*不保存到远程存储。

@param实体直接添加到缓存。

\* @param [options]选项，如mergeStrategy

\*/

**addAllToCache(entities: T[], options?: EntityActionOptions): void;**

/\*\*

\*添加一个新的实体直接到缓存。

\*不保存到远程存储。

如果缓存中有相同主键的实体，则忽略该实体。

@param实体直接添加到缓存。

\* @param [options]选项，如mergeStrategy

\*/

**addOneToCache(entity: T, options?: EntityActionOptions): void;**

/\*\*

\*添加多个新的实体直接到缓存。

\*不保存到远程存储。

在缓存中已经有主键的实体将被忽略。

@param实体直接添加到缓存。

\* @param [options]选项，如mergeStrategy

\*/

**addManyToCache(entities: T[], options?: EntityActionOptions): void;**

/\*\* 清除缓存的实体集合

**clearCache(): void;**

/\*\*

\*直接从缓存中删除一个实体。

\*不从远程存储删除该实体。

\* @param entity要删除的实体

\* @param [options]选项，如mergeStrategy

\*/

**removeOneFromCache(entity: T, options?: EntityActionOptions): void;**

/\*\*

直接从缓存中删除一个实体。

不从远程存储删除该实体。

@param key要删除的实体的主键

@param [options]选项，如mergeStrategy

\*/

**removeOneFromCache(key: number | string, options?: EntityActionOptions): void;**

\*直接从缓存中删除多个实体。

\*不从远程存储删除这些实体。

@param entity要删除的实体

\* @param [options]选项，如mergeStrategy

**removeManyFromCache(entities: T[], options?: EntityActionOptions): void;**

/\*\*

\*直接从缓存中删除多个实体。

\*不从远程存储删除这些实体。

\* @param 要删除实体的主键

\* @param [options]选项，如mergeStrategy

\*/

**removeManyFromCache(keys: (number | string)[], options?: EntityActionOptions): void;**

/\*\*

\*直接更新缓存的实体。

不更新远程存储中的实体。

如果与主键匹配的实体不在缓存中，则忽略该实体。

\*更新实体可能是部分的(但必须有它的键)

\*在这种情况下，它修补现有的实体。

\* @param实体直接在缓存中更新。

\* @param [options] mergeStrategy等选项

\*/

**updateOneInCache(entity: Partial<T>, options?: EntityActionOptions): void;**

/\*\*

\*直接更新多个缓存实体。

\*不更新远程存储中的这些实体。

\*主键不在缓存中的实体将被忽略。

\*更新实体可能是部分的，但必须至少有它们的密钥。

\*这样的部分实体修补他们的缓存对应。

\* @param 在缓存中直接更新的实体

\* @param [options] mergeStrategy等选项

\*/

**updateManyInCache(entities: Partial<T>[], options?: EntityActionOptions): void;**

/\*\*

\*直接插入或更新缓存的实体。

\*不保存到远程存储。

\* Upsert实体可能是T的一部分，但必须至少有它的键。

\*传递Update<T>结构作为有效负载。

\* @param 在缓存中直接向上插入的实体。

\* @param [options] mergeStrategy等选项

\*/

**upsertOneInCache(entity: Partial<T>, options?: EntityActionOptions): void;**

/\*\*

\*直接插入或更新多个缓存实体。

\*不保存到远程存储。

\* Upsert实体可能是部分的，但必须至少有它们的密钥。

\*传递一个Update<T>结构的数组作为有效负载。

\* @param 要在缓存中直接向上插入的实体。

\* @param [options] mergeStrategy等选项

\*/

**upsertManyInCache(entities: Partial<T>[], options?: EntityActionOptions): void;**

/\* 当使用' filteredEntities '选择器时，设置集合的筛选器应用的模式

**setFilter(pattern: any): void;**

/\*\* 设置加载标志

**setLoaded(isLoaded: boolean): void;**

/\*\*设置装载标志

**setLoading(isLoading: boolean): void;**

/\*\* 可观察的集合作为一个整体 \*/

**collection$: Observable<EntityCollection<T>> | Store<EntityCollection<T>>;**

/\*\* 可观察到的缓存集合中的实体数量. \*/

**count$: Observable<number> | Store<number>;**

/\*\* 对缓存集合中的所有实体进行观察 \*/

**entities$: Observable<T[]> | Store<T[]>;**

*/\*\*可观察到与此实体类型相关的操作 \*/*

**entityActions$: Observable<EntityAction>;**

/\*\* 实体键到实体的映射 \*/

**entityMap$: Observable<Dictionary<T>> | Store<Dictionary<T>>;**

/\*\* 可观察到与此实体类型相关的错误操作。 \*/

**errors$: Observable<EntityAction>;**

/\*\*由实体集合的filter函数应用的过滤模式的可观察对象 \*/

**filter$: Observable<any> | Store<any>;**

/\*\*缓存集合中传递filter函数的实体的可观察对象

**filteredEntities$: Observable<T[]> | Store<T[]>;**

/\*\*可观察的缓存集合的键，在集合的本机排序顺序

**keys$: Observable<string[] | number[]> | Store<string[] | number[]>;**

/\*\* 当集合被加载时，可观察为true

**loaded$: Observable<boolean> | Store<boolean>;**

/\*\* 当执行多实体查询命令时，可观察为true. \*/

**loading$: Observable<boolean> | Store<boolean>;**

/\*\* 具有未保存更改的实体的原始实体值 \*/

**changeState$: Observable<ChangeStateMap<T>> | Store<ChangeStateMap<T>>;**

}