/\*\*\*对实体集合端点进行类似rest的CRUD调用的EntityCollection数据服务的注册表。

@Injectable()

**export class EntityDataService {**

protected services: { [name: string]: EntityCollectionDataService<any> } = {};

// TODO: 可选地为那些不是从BaseDataService派生的服务注入专门的实体数据服务。

**constructor**(protected defaultDataServiceFactory: DefaultDataServiceFactory) {}

/ 获取(或创建)实体类型的数据服务 @param entityName - the name of the type

\*例如： getService('Hero'); // data service for Heroes, 非类型化

\* getService<Hero>('Hero'); // data service for Heroes, 类型化

**getService**<T>(entityName: string): EntityCollectionDataService<T> {

entityName = entityName.trim();

let service = this.services[entityName];

if (!service) {

service = this.defaultDataServiceFactory.create(entityName);

this.services[entityName] = service;

}

return service;

}

/\* 为实体类型注册EntityCollectionDataService

\* @param entityName - the name of the entity type

\* @param service -用于该实体类型的数据服务

\* 例如：

\* registerService('Hero', myHeroDataService);

\* registerService('Villain', myVillainDataService);

**registerService**<T>(

entityName: string,

service: EntityCollectionDataService<T>

) {

this.services[entityName.trim()] = service;

}

/ 注册一批数据服务。

\* @param services - 将数据服务合并到现有服务中

\* 例如：

\* registerServices({

\* Hero: myHeroDataService,

\* Villain: myVillainDataService

\* });

\*/

**registerServices**(services: {

[name: string]: EntityCollectionDataService<any>;

}) {

this.services = { ...this.services, ...services };

}