应用服务的输入输出参数均为DTO，理想情况下，展示层应调用应用服务，不应直接调用领域层

**IApplicationService 接口**

一个应用服务应当实现IApplicationService接口

public interface IPersonAppService : IApplicationService

{

    void CreatePerson(CreatePersonInput input);

}

public class PersonAppService : IPersonAppService

{

private readonly IRepository<Person> \_personRepository;

    public PersonAppService(IRepository<Person> personRepository)

    {

        \_personRepository = personRepository;

}

    public void CreatePerson(CreatePersonInput input)

    {

    }

}

**ApplicationService 类**

ApplicationService提供了基本的功能，使得日志、本地化等更容易。

public class TaskAppService : ApplicationService, ITaskAppService

{

    public TaskAppService()

    {

        LocalizationSourceName = "SimpleTaskSystem";

    }

    public void CreateTask(CreateTaskInput input)

    {

        Logger.Info("Creating a new task with description: " + input.Description);

        var text = L("SampleLocalizableTextKey");

    }

}

**CrudAppService 和 AsyncCrueAppService 类**

**CrudAppService是一个包含增删改查的AppService基类**

public class TaskAppService : AsyncCrudAppService<Task, TaskDto>

{

    public TaskAppService(IRepository<Task> repository)

        : base(repository)

    {

    }

}

**定制CRUD应用服务**

**获取列表**

Crud应用服务的GetAll方法默认以PagedAndSortedResultRequestInput为参数

如下，我们自定义一个参数：

public class GetAllTasksInput : PagedAndSortedResultRequestInput

{

    public TaskState? State { get; set; }

}

重载基类：

public class TaskAppService : AsyncCrudAppService<Task, TaskDto, int, GetAllTasksInput>

{

    public TaskAppService(IRepository<Task> repository)

        : base(repository)

    {

}、

    protected override IQueryable<Task> CreateFilteredQuery(GetAllTasksInput input)

    {

        return base.CreateFilteredQuery(input)

            .WhereIf(input.State.HasValue, t => t.State == input.State.Value);

    }

}

**创建和更新**

如下，我们想定制创建和更新的DTO

[AutoMapTo(typeof(Task))]

public class CreateTaskInput

{

    [Required]

    [MaxLength(Task.MaxTitleLength)]

    public string Title { get; set; }

    [MaxLength(Task.MaxDescriptionLength)]

public string Description { get; set; }

    public Guid? AssignedPersonId { get; set; }

}

接着创建一个UpdateTaskInput DTO：

[AutoMapTo(typeof(Task))]

public class UpdateTaskInput : CreateTaskInput, IEntityDto

{

    public int Id { get; set; }

    public TaskState State { get; set; }

}

接下来，我们可以使用这些DTO类作为AsyncCrudAppService类的泛型接口，如下所示：

public class TaskAppService : AsyncCrudAppService<Task, TaskDto, int, GetAllTasksInput, CreateTaskInput, UpdateTaskInput>

{

    public TaskAppService(IRepository<Task> repository)

        : base(repository)

    {

}

    protected override IQueryable<Task> CreateFilteredQuery(GetAllTasksInput input)

    {

        return base.CreateFilteredQuery(input)

            .WhereIf(input.State.HasValue, t => t.State == input.State.Value);

    }

}

**不公开AppService为WebApi**

如果从Abp官网下载模板，Web模块会包含如下语句

    Configuration.Modules.AbpAspNetCore()

                 .CreateControllersForAppServices(

                     typeof(IEManageSystemApplicationModule).GetAssembly()

                 )

该语句将AppService公开为Webapi

AppService会自动公开为Webapi，如果不想公开当前AppService，则可以使用特性

[RemoteService(false)]

public class AccountAppService : IEManageSystemAppServiceBase, IAccountAppService