1. 新增领域业务实现规则顺序
   1. BSTI.IETM.Datas
      1. Mappings
      2. Repositories
   2. BSTI.IETM.Domains
      1. Models
      2. Factories
      3. Repositories
   3. BSTI.IETM.Services
      1. Dtos
      2. Queries
      3. Abstractions
      4. Implements
      5. Configs
   4. BSTI.IETM.EDITOR
      1. Areas
      2. Apis
      3. Typings
2. 新增功能
   1. 建表 、添加字段、字段说明
   2. BSTI.IETM.Domains
      1. Models 里面添加数据库实体
         1. 注意是TreeEntityBase 还是 普通实体（AggregateRoot）
      2. 添加实体对象的Description特性或者DisplayName特性
      3. 添加字段属性
         1. 注意 必填、字符长度限制
      4. 修改 AddDescriptions 和 AddChanges
         1. 注意修改 T 类型
   3. BSTI.IETM.Datas
      1. 修改 ParaMap
         1. 修改名称
         2. 修改 T 类型
         3. builder.ToTable( "ParaMap", "Systems" ); => builder.ToTable( "Para", "Systems" );
         4. HasColumnName("ParaOptionId"); => HasColumnName("ParaId");
   4. BSTI.IETM.Domains
      1. 添加 IParaRepository
   5. BSTI.IETM.Datas
      1. 添加 ParaRepository
   6. BSTI.IETM.Domains
      1. 添加 ParaFactory
   7. BSTI.IETM.Services
      1. 添加 ParaDto
      2. 添加 Extensions
      3. 添加 Queries
      4. 添加 Abstractions
      5. 添加 Implements
   8. BSTI.IETM.EDITOR
      1. Areas
      2. Apis
      3. Typings
      4. 修改路由 和 注册 Modules
         1. system.routing.module.ts 添加新增的angular 组件
         2. system.module.ts 添加新增的angular 组件
         3. 添加菜单 Views/ Home/ Main/ SideNav.cshtml
   9. 添加选择框
      1. 在 Areas Controller 中添加视图
      2. 在 页面中添加 弹出 选择框的 js 事件代码 **on-click**="openParaDialog()"
      3. 在 angular 组件中实现点击事件
      4. 新增 选择框的 angular 组件
         1. 在 system.module.ts 添加新增的angular 选择框组件
3. 原理说明

/// <summary>

/// 获取单个实例

/// </summary>

/// <remarks>

/// 调用范例:

/// GET

/// /api/customer/1

/// </remarks>

/// <param name="id">标识</param>

[HttpGet( "{id}" )]

public virtual async Task<IActionResult> GetAsync( string id ) {

var result = await \_service.GetByIdAsync( id );

return Success( result );

}

/// <summary>

/// 删除，注意：该方法用于删除单个实体，批量删除请使用POST提交，否则可能失败

/// </summary>

/// <remarks>

/// 调用范例:

/// DELETE

/// /api/customer/1

/// </remarks>

/// <param name="id">标识</param>

[HttpDelete( "{id}" )]

public virtual async Task<IActionResult> DeleteAsync( string id ) {

await \_service.DeleteAsync( id );

return Success();

}

/// <summary>

/// 批量删除，注意：body参数需要添加引号，"'1,2,3'"而不是"1,2,3"

/// </summary>

/// <remarks>

/// 调用范例:

/// POST

/// /api/customer/delete

/// body: "'1,2,3'"

/// </remarks>

/// <param name="ids">标识列表，多个Id用逗号分隔，范例：1,2,3</param>

[HttpPost( "delete" )]

public virtual async Task<IActionResult> BatchDeleteAsync( [FromBody] string ids ) {

await \_service.DeleteAsync( ids );

return Success();

}

/// <summary>

/// 启用

/// </summary>

/// <param name="ids">标识列表</param>

[HttpPost( "enable" )]

public virtual async Task<IActionResult> Enable( [FromBody] string ids ) {

await \_service.EnableAsync( ids );

var result = await \_service.FindByIdsAsync( ids );

return Success( result );

}

/// <summary>

/// 冻结

/// </summary>

/// <param name="ids">标识列表</param>

[HttpPost( "disable" )]

public virtual async Task<IActionResult> Disable( [FromBody] string ids ) {

await \_service.DisableAsync( ids );

var result = await \_service.FindByIdsAsync( ids );

return Success( result );

}

/// <summary>

/// 交换排序

/// </summary>

/// /// <remarks>

/// 调用范例:

/// POST

/// /api/customer/SwapSort

/// body: "'1,2'"

/// </remarks>

/// <param name="ids">两个Id的标识列表，用逗号分隔，范例：1,2</param>

[HttpPost( "SwapSort" )]

public virtual async Task<IActionResult> SwapSortAsync( [FromBody] string ids ) {

var idList = ids.ToGuidList();

if( idList.Count < 2 )

return Fail( "交换排序失败" );

await \_service.SwapSortAsync( idList[0], idList[1] );

return Success();

}

/// <summary>

/// 修正排序

/// </summary>

/// <param name="parameter">查询参数</param>

[HttpPost( "fix" )]

public virtual async Task<IActionResult> FixAsync( [FromBody] TQuery parameter ) {

if ( parameter == null )

return Fail( "查询参数不能为空" );

await \_service.FixSortIdAsync( parameter );

return Success();

}

/// <summary>

/// 查询

/// </summary>

/// <remarks>

/// 调用范例:

/// GET

/// /api/role?name=a

/// </remarks>

/// <param name="query">查询参数</param>

[HttpGet]

public virtual async Task<IActionResult> QueryAsync( TQuery query ) {

if( query == null )

throw new ArgumentNullException( nameof( query ) );

QueryBefore( query );

ProcessParam( query );

PagerList<PrimeTreeNode<TDto>> result;

switch( GetOperation( query ) ) {

case LoadOperation.FirstLoad:

result = await FirstLoad( query );

break;

case LoadOperation.LoadChild:

result = await LoadChildren( query );

break;

default:

result = await Search( query );

break;

}

return Success( result );

}

/// <summary>

/// 获取单个实例

/// </summary>

/// <remarks>

/// 调用范例:

/// GET

/// /api/customer/1

/// </remarks>

/// <param name="id">标识</param>

[HttpGet( "{id}" )]

public virtual async Task<IActionResult> GetAsync( string id ) {

var result = await \_service.GetByIdAsync( id );

return Success( result );

}

/// <summary>

/// 分页查询

/// </summary>

/// <remarks>

/// 调用范例:

/// GET

/// /api/customer?name=a

/// </remarks>

/// <param name="query">查询参数</param>

[HttpGet]

public virtual async Task<IActionResult> PagerQueryAsync( TQuery query ) {

PagerQueryBefore( query );

var result = await \_service.PagerQueryAsync( query );

return Success( ToPagerQueryResult( result ) );

}

/// <summary>

/// 查询

/// </summary>

/// <remarks>

/// 调用范例:

/// GET

/// /api/customer/query?name=a

/// </remarks>

/// <param name="query">查询参数</param>

[HttpGet( "Query" )]

public virtual async Task<IActionResult> QueryAsync( TQuery query ) {

QueryBefore( query );

var result = await \_service.QueryAsync( query );

return Success( ToQueryResult( result ) );

}

/// <summary>

/// 修改

/// </summary>

/// <remarks>

/// 调用范例:

/// PUT

/// /api/customer/1

/// </remarks>

/// <param name="id">标识</param>

/// <param name="request">修改参数</param>

[HttpPut( "{id?}" )]

public virtual async Task<IActionResult> UpdateAsync( string id, [FromBody] TUpdateRequest request ) {

if( request == null )

return Fail( WebResource.UpdateRequestIsEmpty );

if( id.IsEmpty() && request.Id.IsEmpty() )

return Fail( WebResource.IdIsEmpty );

if( request.Id.IsEmpty() )

request.Id = id;

UpdateBefore( request );

await \_service.UpdateAsync( request );

var result = await \_service.GetByIdAsync( request.Id );

return Success( result );

}

/// <summary>

/// 删除，注意：该方法用于删除单个实体，批量删除请使用POST提交，否则可能失败

/// </summary>

/// <remarks>

/// 调用范例:

/// DELETE

/// /api/customer/1

/// </remarks>

/// <param name="id">标识</param>

[HttpDelete( "{id}" )]

public virtual async Task<IActionResult> DeleteAsync( string id ) {

await \_service.DeleteAsync( id );

return Success();

}

/// <summary>

/// 批量删除，注意：body参数需要添加引号，"'1,2,3'"而不是"1,2,3"

/// </summary>

/// <remarks>

/// 调用范例:

/// POST

/// /api/customer/delete

/// body: "'1,2,3'"

/// </remarks>

/// <param name="ids">标识列表，多个Id用逗号分隔，范例：1,2,3</param>

[HttpPost( "delete" )]

public virtual async Task<IActionResult> BatchDeleteAsync( [FromBody] string ids ) {

await \_service.DeleteAsync( ids );

return Success();

}