Namespace Solstice.Applications.Attributes

Classes

ServiceAttribute

The ServiceAttribute is a custom attribute used for marking classes within the Solstice.Service namespace. This attribute is sealed, implying that it cannot be inherited from.

ServiceInterfaceAttribute

The ServiceInterfaceAttribute is a custom attribute used for marking interface within the Solstice.Service namespace. This attribute is sealed, implying that it cannot be inherited from.

Class ServiceAttribute

Namespace: Solstice.Applications.Attributes

Assembly: Solstice.Applications.dll

The ServiceAttribute is a custom attribute used for marking classes within the Solstice.Service namespace. This attribute is sealed, implying that it cannot be inherited from.

```
[AttributeUsage(AttributeTargets.Class)]
public sealed class ServiceAttribute : Attribute
```

Inheritance

object ← Attribute ← ServiceAttribute

Inherited Members

Attribute.Equals(object) , Attribute.GetCustomAttribute(Assembly, Type) ,
Attribute.GetCustomAttribute(Assembly, Type, bool) ,
Attribute.GetCustomAttribute(MemberInfo, Type) ,
Attribute.GetCustomAttribute(MemberInfo, Type, bool) ,
Attribute.GetCustomAttribute(Module, Type) ,
Attribute.GetCustomAttribute(Module, Type, bool) ,
Attribute.GetCustomAttribute(ParameterInfo, Type) ,
Attribute.GetCustomAttribute(ParameterInfo, Type, bool) ,
Attribute.GetCustomAttributes(Assembly) ,
Attribute.GetCustomAttributes(Assembly, bool) ,
Attribute.GetCustomAttributes(Assembly, Type) ,
Attribute.GetCustomAttributes(Assembly, Type, bool) ,
Attribute.GetCustomAttributes(MemberInfo) ,
Attribute.GetCustomAttributes(MemberInfo, bool) ,
Attribute.GetCustomAttributes(MemberInfo, Type) ,
Attribute.GetCustomAttributes(MemberInfo, Type, bool) ,
Attribute.GetCustomAttributes(Module) , Attribute.GetCustomAttributes(Module, bool) ,
Attribute.GetCustomAttributes(Module, Type) ,
Attribute.GetCustomAttributes(Module, Type, bool) ,
Attribute.GetCustomAttributes(ParameterInfo) ,
Attribute.GetCustomAttributes(ParameterInfo, bool) ,
Attribute.GetCustomAttributes(ParameterInfo, Type) ,
Attribute.GetCustomAttributes(ParameterInfo, Type, bool) , Attribute.GetHashCode() ,

Attribute.IsDefaultAttribute() , Attribute.IsDefined(Assembly, Type) , Attribute.IsDefined(Assembly, Type, bool) , Attribute.IsDefined(MemberInfo, Type) , Attribute.IsDefined(MemberInfo, Type, bool) , Attribute.IsDefined(Module, Type) , Attribute.IsDefined(Module, Type, bool) , Attribute.IsDefined(ParameterInfo, Type) , Attribute.IsDefined(ParameterInfo, Type, bool) , Attribute.Match(object) , Attribute.TypeId , object.Equals(object, object) , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

Class ServiceInterfaceAttribute

Namespace: Solstice.Applications.Attributes

Assembly: Solstice.Applications.dll

The ServiceInterfaceAttribute is a custom attribute used for marking interface within the Solstice.Service namespace. This attribute is sealed, implying that it cannot be inherited from.

```
[AttributeUsage(AttributeTargets.Interface)]
public sealed class ServiceInterfaceAttribute : Attribute
```

Inheritance

object ← Attribute ← ServiceInterfaceAttribute

Inherited Members

Attribute.Equals(object) , Attribute.GetCustomAttribute(Assembly, Type) ,
Attribute.GetCustomAttribute(Assembly, Type, bool) ,
Attribute.GetCustomAttribute(MemberInfo, Type) ,
Attribute.GetCustomAttribute(MemberInfo, Type, bool) ,
Attribute.GetCustomAttribute(Module, Type) ,
Attribute.GetCustomAttribute(Module, Type, bool) ,
Attribute.GetCustomAttribute(ParameterInfo, Type) ,
Attribute.GetCustomAttribute(ParameterInfo, Type, bool) ,
Attribute.GetCustomAttributes(Assembly) ,
Attribute.GetCustomAttributes(Assembly, bool) ,
Attribute.GetCustomAttributes(Assembly, Type) ,
Attribute.GetCustomAttributes(Assembly, Type, bool) ,
Attribute.GetCustomAttributes(MemberInfo) ,
Attribute.GetCustomAttributes(MemberInfo, bool) ,
Attribute.GetCustomAttributes(MemberInfo, Type) ,
Attribute.GetCustomAttributes(MemberInfo, Type, bool) ,
Attribute.GetCustomAttributes(Module) , Attribute.GetCustomAttributes(Module, bool) ,
Attribute.GetCustomAttributes(Module, Type) ,
Attribute.GetCustomAttributes(Module, Type, bool) ,
Attribute.GetCustomAttributes(ParameterInfo) ,
Attribute.GetCustomAttributes(ParameterInfo, bool) ,
Attribute.GetCustomAttributes(ParameterInfo, Type) ,
Attribute.GetCustomAttributes(ParameterInfo, Type, bool) , Attribute.GetHashCode() ,

Attribute.IsDefaultAttribute() , Attribute.IsDefined(Assembly, Type) , Attribute.IsDefined(Assembly, Type, bool) , Attribute.IsDefined(MemberInfo, Type) , Attribute.IsDefined(MemberInfo, Type, bool) , Attribute.IsDefined(Module, Type) , Attribute.IsDefined(Module, Type, bool) , Attribute.IsDefined(ParameterInfo, Type) , Attribute.IsDefined(ParameterInfo, Type, bool) , Attribute.Match(object) , Attribute.TypeId , object.Equals(object, object) , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

Namespace Solstice.Applications.Core

Classes

CoreService<TRepository, T>

Abstract class 'CoreService' defines the operations for managing entities of type T in the database.

Interfaces

ICoreService<T>

This interface describes an abstraction for a data manipulation service on a given type of CoreModel. It includes methods for CRUD operations, queries for specific entities, counting entities, starting transactions, and more.

Class CoreService<TRepository, T>

Namespace: Solstice.Applications.Core

Assembly: Solstice.Applications.dll

Abstract class 'CoreService' defines the operations for managing entities of type T in the database.

```
public class CoreService<TRepository, T> : ICoreService<T> where TRepository :
ICoreRepository<T> where T : class
```

Type Parameters

`TRepository`

Specifies the repository type handling the operations. TRepository must be an implementation of "ICoreRepository{T}".

`T`

Specifies the entity type to be handled by this service.

Inheritance

object ← CoreService<TRepository, T>

Implements

ICoreService<T>

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Remarks

This class provides the ability to perform CRUD operations and various queries on entities of type T. Each instance of this class will be associated with a repository which interacts with the database.

Constructors

CoreService(IUnitOfWork)

Constructor for the Solstice service.

```
protected CoreService(IUnitOfWork unitOfWork)
```

Parameters

`unitOfWork` IUnitOfWork

Fields

_repository

```
protected readonly TRepository _repository
```

Field Value

TRepository

Methods

AddAndSaveAsync(T)

```
public Task AddAndSaveAsync(T entity)
```

Parameters

`entity` T

Returns

Task

AddAndSaveAsync<TEntity>(TEntity)

```
public Task AddAndSaveAsync<TEntity>(TEntity entity) where TEntity : class
```

Parameters

`entity` TEntity

Returns

Task

Type Parameters

`TEntity`

AddAsync(T)

Method to add new entity to the repository.

```
public Task AddAsync(T entity)
```

Parameters

`entity` T

Entity of type T which need to be added.

Returns

Task

Awaitable task.

AddAsync<TEntity>(TEntity)

```
public Task AddAsync<TEntity>(TEntity entity) where TEntity : class
```

Parameters

`entity` TEntity

Returns

Task

Type Parameters

`TEntity`

AddRangeAndSaveAsync(ICollection<T>)

```
public Task AddRangeAndSaveAsync(ICollection<T> entities)
```

Parameters

`entities` ICollection<T>

Returns

Task

AddRangeAndSaveAsync<TEntity>(ICollection<TEntity>)

```
public Task AddRangeAndSaveAsync<TEntity>(ICollection<TEntity> entities) where
TEntity : class
```

Parameters

`entities` ICollection<TEntity>

Returns

Task

Type Parameters

TEntity

AddRangeAsync(ICollection<T>)

Method to add new range of entities to the repository.

```
public Task AddRangeAsync(ICollection<T> entities)
```

Parameters

`entities` ICollection<T>

Collection of entities of type T which need to be added.

Returns

Task

Awaitable task.

AddRangeAsync<TEntity>(ICollection<TEntity>)

```
public Task AddRangeAsync<TEntity>(ICollection<TEntity> entities) where TEntity
: class
```

Parameters

`entities` ICollection<TEntity>

Returns

Task

Type Parameters

`TEntity`

AnyAsyncBy(Expression<Func<T, bool>>)

Checks if there are any entities that satisfy the provided condition.

```
public Task<bool> AnyAsyncBy(Expression<Func<T, bool>> where)
```

Parameters

`where` Expression<Func<T, bool>>

A delegate defining the condition for the entities to meet,

Returns

Task<bool>

Bool value indicating whether any entities meet the specified condition.

AnyAsyncBy<TEntity>(Expression<Func<TEntity, bool>>)

```
public Task<bool> AnyAsyncBy<TEntity>(Expression<Func<TEntity, bool>> where) where
TEntity : class
```

Parameters

`where` Expression<Func<TEntity, bool>>

Returns

Task<bool>

Type Parameters

`TEntity`

AnyAsyncBy<TEntity>(IQueryable<TEntity>)

```
public Task<bool> AnyAsyncBy<TEntity>(IQueryable<TEntity> queryable) where TEntity
: class
```

Parameters

`queryable` IQueryable<TEntity>

Returns

Task<bool>

Type Parameters

`TEntity`

BeginTransactionAsync()

Begins a new transaction in the context of the database.

```
public Task<IDbContextTransaction> BeginTransactionAsync()
```

Returns

Task<IDbContextTransaction>

A task representing the asynchronous operation, containing the started database transaction.

CountAllAsync()

Counts the total number of all entities present in the database.

```
public Task<decimal> CountAllAsync()
```

Returns

Task<decimal>

Task with the total count of entities available.

CountAllAsyncBy(Expression<Func<T, bool>>)

Counts the number of entities that satisfy the provided condition.

```
public Task<decimal> CountAllAsyncBy(Expression<Func<T, bool>> where)
```

Parameters

`where` Expression<Func<T, bool>>

A delegate defining the condition for the entities to meet.

Returns

Task<decimal>

Task with the total count of entities that meet the specified condition.

CountAllAsyncBy<TEntity>(Expression<Func<TEntity, bool>>)

```
public Task<decimal> CountAllAsyncBy<TEntity>(Expression<Func<TEntity, bool>> where)
where TEntity : class
```

Parameters

`where` Expression<Func<TEntity, bool>>

Returns

Task<decimal>

Type Parameters

`TEntity`

FindAsync(int)

```
public Task<T> FindAsync(int id)
```

Parameters

`id` int

Returns

Task<T>

FindAsync<TEntity>(int)

```
public Task<TEntity> FindAsync<TEntity>(int id) where TEntity : class
```

Parameters

`id` int

Returns

Task<TEntity>

Type Parameters

`TEntity`

GetAllAsync(ICoreSpecifications<T>?)

Gets all entities satisfying the provided specification in the database.

```
public Task<ICollection<T>> GetAllAsync(ICoreSpecifications<T>? coreSpecifications)
```

Parameters

`coreSpecifications` ICoreSpecifications<T>

Specifications to be met by the entities.

Returns

Task<ICollection<T>>

Task representing the asynchronous operation, containing the list of entities that meet the provided specification.

GetAllAsync(Expression<Func<T, bool>>)

Gets all entities that satisfy the provided condition.

```
public Task<ICollection<T>> GetAllAsync(Expression<Func<T, bool>> where)
```

Parameters

`where` Expression<Func<T, bool>>

A delegate defining the condition for the entities to meet.

Returns

Task<ICollection<T>>

A task representing the asynchronous operation, containing the entities that meet the specified condition.

GetAllAsync(string, ICoreSpecifications<T>?)

Gets all entities based on provided raw SQL query and specifications.

```
public Task<ICollection<T>> GetAllAsync(string query, ICoreSpecifications<T>?
coreSpecifications)
```

Parameters

`query` string

Raw SQL query to be executed.

`coreSpecifications` ICoreSpecifications<T>

Specifications to be met by the entities.

Returns

Task<ICollection<T>>

A task representing the asynchronous operation, containing the list of entities that meet the provided query and specification.

GetAllAsync(string, ICollection<DbParameter>, ICoreSpecifications<T>?)

Gets all entities based on provided raw SQL query, parameters and specifications.

```
public Task<ICollection<T>> GetAllAsync(string query, ICollection<DbParameter>
parameters, ICoreSpecifications<T>? coreSpecifications)
```

Parameters

`query` string

Raw SQL query to be executed.

`parameters` ICollection<DbParameter>

List of parameters to be used in query.

`coreSpecifications` ICoreSpecifications<T>

Specifications to be met by the entities.

Returns

Task<ICollection<T>>

A task representing the asynchronous operation, containing the list of entities that meet the provided query, parameters and specification.

GetAllAsync<TEntity>(ICoreSpecifications<TEntity>)

```
public Task<ICollection<TEntity>> GetAllAsync<TEntity>(ICoreSpecifications<TEntity>
coreSpecifications) where TEntity : class
```

Parameters

`coreSpecifications` ICoreSpecifications<TEntity>

Returns

Task<ICollection<TEntity>>

Type Parameters

`TEntity`

GetAllAsync<TEntity>(Expression<Func<TEntity, bool>>)

```
public Task<ICollection<TEntity>> GetAllAsync<TEntity>(Expression<Func<TEntity,
bool>> where) where TEntity : class
```

Parameters

`where` Expression<Func<TEntity, bool>>

Returns

Task<ICollection<TEntity>>

Type Parameters

`TEntity`

GetAllAsync<TEntity>(string)

```
public Task<ICollection<TEntity>> GetAllAsync<TEntity>(string query) where TEntity
: class
```

Parameters

`query` string

Returns

Task<ICollection<TEntity>>

Type Parameters

`TEntity`

GetAllAsync<TEntity>(string, ICoreSpecifications<TEntity>)

```
public Task<ICollection<TEntity>> GetAllAsync<TEntity>(string query,
ICoreSpecifications<TEntity> coreSpecifications) where TEntity : class
```

Parameters

`query` string

`coreSpecifications` ICoreSpecifications<TEntity>

Returns

Task<ICollection<TEntity>>

Type Parameters

`TEntity`

GetAllAsync<TEntity>(string, ICollection<DbParameter>)

```
public Task<ICollection<TEntity>> GetAllAsync<TEntity>(string query,
ICollection<DbParameter> parameters) where TEntity : class
```

Parameters

`query` string

`parameters` ICollection<DbParameter>

Returns

Task<ICollection<TEntity>>

Type Parameters

`TEntity`

GetAllAsync<TEntity>(string, ICollection<DbParameter>, ICoreSpecifications<TEntity>)

```
public Task<ICollection<TEntity>> GetAllAsync<TEntity>(string query,
ICollection<DbParameter> parameters, ICoreSpecifications<TEntity>
coreSpecifications) where TEntity : class
```

Parameters

`query` string

`parameters` ICollection<DbParameter>

`coreSpecifications` ICoreSpecifications<TEntity>

Returns

Task<ICollection<TEntity>>

Type Parameters

`TEntity`

GetAllByQueryable<TEntity>(IQueryable<TEntity>)

Gets all entities based on provided IQueryable query.

```
public Task<ICollection<TEntity>> GetAllByQueryable<TEntity>
(IQueryable<TEntity> query)
```

Parameters

`query` IQueryable<TEntity>

IQueryable query to be executed.

Returns

Task<ICollection<TEntity>>

A task representing the asynchronous operation, containing the list of entities that meet the provided query.

Type Parameters

`TEntity`

GetAllQueryable()

Gets an IQueryable of all entities.

```
public IQueryable<T> GetAllQueryable()
```

Returns

IQueryable<T>

IQueryable of all entities.

GetAllQueryable(ICoreSpecifications<T>?)

Gets entities based on provided specifications.

```
public IQueryable<T> GetAllQueryable(ICoreSpecifications<T>? coreSpecifications)
```

Parameters

`coreSpecifications` ICoreSpecifications<T>

Specifications to be met by the entities.

Returns

IQueryable<T>

IQueryable of entities that meet the provided specifications.

GetAllQueryable(Expression<Func<T, bool>>)

Gets entities by their specifications and provides a queryable list.

```
public IQueryable<T> GetAllQueryable(Expression<Func<T, bool>> where)
```

Parameters

`where` Expression<Func<T, bool>>

A delegate defining the condition for the entities to meet.

Returns

IQueryable<T>

IQueryable of entities that meet the specified condition.

GetAllQueryable(string, ICoreSpecifications<T>?)

Gets entities based on provided raw SQL query and specifications.

```
public IQueryable<T> GetAllQueryable(string query, ICoreSpecifications<T>?
coreSpecifications)
```

Parameters

`query` string

Raw SQL query to be executed.

`coreSpecifications` ICoreSpecifications<T>

Specifications to be met by the entities.

Returns

IQueryable<T>

A IQueryable representation of entities that meet the provided query and specification.

GetAllQueryable(string, ICollection<DbParameter>, ICoreSpecifications<T>?)

Gets entities based on provided raw SQL query, parameters and specifications.

```
public IQueryable<T> GetAllQueryable(string query, ICollection<DbParameter>
parameters, ICoreSpecifications<T>? coreSpecifications)
```

Parameters

`query` string

Raw SQL query to be executed.

`parameters` ICollection<DbParameter>

List of parameters to be used in the query.

`coreSpecifications` ICoreSpecifications<T>

Specifications to be met by the entities.

Returns

IQueryable<T>

A IQueryable representation of entities that meet the provided query, parameters and specification.

GetBy(ICoreSpecifications<T>)

Get an entity by the given specifications.

```
public Task<T?> GetBy(ICoreSpecifications<T> coreSpecifications)
```

Parameters

`coreSpecifications` ICoreSpecifications<T>

Specifications to be met by the entity.

Returns

Task<T>

A task representing the asynchronous operation, containing an entity that meets the provided specification.

GetBy(Expression<Func<T, bool>>)

Get an entity by the given specifications.

```
public Task<T?> GetBy(Expression<Func<T, bool>> where)
```

Parameters

`where` Expression<Func<T, bool>>

A delegate defining the condition for the entity to meet.

Returns

Task<T>

A task representing the asynchronous operation, containing an entity that meets the provided condition.

GetBy<TEntity>(ICoreSpecifications<TEntity>)

```
public Task<TEntity?> GetBy<TEntity>(ICoreSpecifications<TEntity>
coreSpecifications) where TEntity : class
```

Parameters

`coreSpecifications` ICoreSpecifications<TEntity>

Returns

Task<TEntity>

Type Parameters

`TEntity`

GetBy<TEntity>(Expression<Func<TEntity, bool>>)

```
public Task<TEntity?> GetBy<TEntity>(Expression<Func<TEntity, bool>> where) where
TEntity : class
```

Parameters

`where` Expression<Func<TEntity, bool>>

Returns

Task<TEntity>

Type Parameters

`TEntity`

GetBy<TEntity>(string, ICollection<DbParameter>, ICoreSpecifications<TEntity>)

```
public Task<TEntity?> GetBy<TEntity>(string query, ICollection<DbParameter>
parameters, ICoreSpecifications<TEntity> coreSpecifications) where TEntity : class
```

Parameters

`query` string

`parameters` ICollection<DbParameter>

`coreSpecifications` ICoreSpecifications<TEntity>

Returns

Task<TEntity>

Type Parameters

`TEntity`

GetPagedResult(Page)

Retrieves a paginated list of entities from the repository that conforms to the specified page.

```
public Task<Paged<T>> GetPagedResult(Page page)
```

Parameters

`page` Page

The page number and size of the entities to be retrieved from the repository.

Returns

Task<Paged<T>>

A task that represents the asynchronous operation. The task result is a collection of entities that meet The core specifications, paginated based on the given page object. If no entities meet the specifications, the task result is an empty collection.

GetPagedResult(Page, ICoreSpecifications<T>)

```
public Task<Paged<T>> GetPagedResult(Page page, ICoreSpecifications<T>
coreSpecifications)
```

Parameters

`page` Page

`coreSpecifications` ICoreSpecifications<T>

Returns

Task<Paged<T>>

GetPagedResult(Page, string, ICollection<DbParameter>)

```
public Task<Paged<T>> GetPagedResult(Page page, string query,
ICollection<DbParameter> parameters)
```

Parameters

`page` Page

`query` string

`parameters` ICollection<DbParameter>

Returns

Task<Paged<T>>

GetPagedResult(Page, string, ICollection<DbParameter>, ICoreSpecifications<T>)

```
public Task<Paged<T>> GetPagedResult(Page page, string query,
ICollection<DbParameter> parameters, ICoreSpecifications<T> coreSpecifications)
```

Parameters

`page` Page

`query` string

`parameters` ICollection<DbParameter>

`coreSpecifications` ICoreSpecifications<T>

Returns

Task<Paged<T>>

GetPagedResult<TEntity>(Page)

```
public Task<Paged<TEntity>> GetPagedResult<TEntity>(Page page) where TEntity : class
```

Parameters

`page` Page

Returns

Task<Paged<TEntity>>

Type Parameters

`TEntity`

GetPagedResult<TEntity>(Page, ICoreSpecifications<TEntity>)

```
public Task<Paged<TEntity>> GetPagedResult<TEntity>(Page page,
ICoreSpecifications<TEntity> coreSpecifications) where TEntity : class
```

Parameters

`page` Page

`coreSpecifications` ICoreSpecifications<TEntity>

Returns

Task<Paged<TEntity>>

Type Parameters

`TEntity`

GetPagedResult<TEntity>(Page, string, ICollection<DbParameter>)

```
public Task<Paged<TEntity>> GetPagedResult<TEntity>(Page page, string query,
ICollection<DbParameter> parameters) where TEntity : class
```

Parameters

`page` Page

`query` string

`parameters` ICollection<DbParameter>

Returns

Task<Paged<TEntity>>

Type Parameters

TEntity

GetPagedResult<TEntity>(Page, string, ICollection<DbParameter>, ICoreSpecifications<TEntity>)

```
public Task<Paged<TEntity>> GetPagedResult<TEntity>(Page page, string query,
ICollection<DbParameter> parameters, ICoreSpecifications<TEntity>
coreSpecifications) where TEntity : class
```

Parameters

page Page

query string

parameters ICollection<DbParameter>

coreSpecifications ICoreSpecifications<TEntity>

Returns

Task<Paged<TEntity>>

Type Parameters

TEntity

PageAllAsync(Page, ICoreSpecifications<T>?)

Gets entities based on provided page and specifications.

```
public Task<ICollection<T>> PageAllAsync(Page page, ICoreSpecifications<T>?
coreSpecifications)
```

Parameters

page Page

Defines the page boundaries.

`coreSpecifications` ICoreSpecifications<T>

Specifications to be met by the entities.

Returns

Task<ICollection<T>>

A task with a collection of all entities meeting the provided page and specifications.

PageAllQueryable(Page, ICoreSpecifications<T>?)

Enables paging to the queryable list of represented entities based on provided page and specifications.

```
public IQueryable<T> PageAllQueryable(Page page, ICoreSpecifications<T>?
coreSpecifications)
```

Parameters

`page` Page

Defines the page boundaries.

`coreSpecifications` ICoreSpecifications<T>

Specifications to be met by the entities.

Returns

IQueryable<T>

A IQueryable sublist of all entities meeting the provided page and specifications.

Remove(T)

Deletes the entity from the database.

```
public void Remove(T entity)
```

Parameters

`entity` T

The entity to be removed.

RemoveAndSaveAsync(T)

```
public Task RemoveAndSaveAsync(T entity)
```

Parameters

`entity` T

Returns

Task

RemoveAndSaveAsync<TEntity>(TEntity)

```
public Task RemoveAndSaveAsync<TEntity>(TEntity entity) where TEntity : class
```

Parameters

`entity` TEntity

Returns

Task

Type Parameters

`TEntity`

RemoveRange(ICollection<T>)

Deletes a range of entities from the database.

```
public void RemoveRange(ICollection<T> entities)
```

Parameters

`entities` ICollection<T>

The entities to be removed.

RemoveRangeAndSaveAsync(ICollection<T>)

```
public Task RemoveRangeAndSaveAsync(ICollection<T> entities)
```

Parameters

`entities` ICollection<T>

Returns

Task

RemoveRangeAndSaveAsync<TEntity>(ICollection<TEntity>)

```
public Task RemoveRangeAndSaveAsync<TEntity>(ICollection<TEntity> entities) where
TEntity : class
```

Parameters

`entities` ICollection<TEntity>

Returns

Task

Type Parameters

`TEntity`

RemoveRange<TEntity>(ICollection<TEntity>)

```
public void RemoveRange<TEntity>(ICollection<TEntity> entities) where TEntity
: class
```

Parameters

`entities` ICollection<TEntity>

Type Parameters

`TEntity`

Remove<TEntity>(TEntity)

```
public void Remove<TEntity>(TEntity entity) where TEntity : class
```

Parameters

`entity` TEntity

Type Parameters

`TEntity`

SaveAsync()

Save any changes in the context to the database.

```
public Task SaveAsync()
```

Returns

Task

Awaitable task.

Update(T)

Updates an entity in the database.

```
public void Update(T entity)
```

Parameters

`entity` T

Entity to be updated.

UpdateAndSaveAsync(T)

```
public Task UpdateAndSaveAsync(T entity)
```

Parameters

`entity` T

Returns

Task

UpdateAndSaveAsync<TEntity>(TEntity)

```
public Task UpdateAndSaveAsync<TEntity>(TEntity entity) where TEntity : class
```

Parameters

`entity` TEntity

Returns

Task

Type Parameters

`TEntity`

UpdateRange(ICollection<T>)

Method to update a range of entities in the repository.

```
public void UpdateRange(ICollection<T> entities)
```

Parameters

`entities` ICollection<T>

Collection of entities which need to be updated.

UpdateRangeAndSaveAsync(ICollection<T>)

```
public Task UpdateRangeAndSaveAsync(ICollection<T> entities)
```

Parameters

`entities` ICollection<T>

Returns

Task

UpdateRangeAndSaveAsync<TEntity>(ICollection<TEntity>)

```
public Task UpdateRangeAndSaveAsync<TEntity>(ICollection<TEntity> entities) where
TEntity : class
```

Parameters

`entities` ICollection<TEntity>

Returns

Task

Type Parameters

`TEntity`

UpdateRange<TEntity>(ICollection<TEntity>)

```
public void UpdateRange<TEntity>(ICollection<TEntity> entities) where TEntity
: class
```

Parameters

`entities` ICollection<TEntity>

Type Parameters

`TEntity`

Update<TEntity>(TEntity)

```
public void Update<TEntity>(TEntity entity) where TEntity : class
```

Parameters

`entity` TEntity

Type Parameters

`TEntity`

Interface ICoreService<T>

Namespace: Solstice.Applications.Core

Assembly: Solstice.Applications.dll

This interface describes an abstraction for a data manipulation service on a given type of CoreModel. It includes methods for CRUD operations, queries for specific entities, counting entities, starting transactions, and more.

```
public interface ICoreService<T> where T : class
```

Type Parameters

`T`

The type of the class to be processed by the service.

Methods

AddAndSaveAsync(T)

```
Task AddAndSaveAsync(T entity)
```

Parameters

`entity` T

Returns

Task

AddAndSaveAsync<TEntity>(TEntity)

```
Task AddAndSaveAsync<TEntity>(TEntity entity) where TEntity : class
```

Parameters

`entity` TEntity

Returns

Task

Type Parameters

`TEntity`

AddAsync(T)

Add entity to repository

```
Task AddAsync(T entity)
```

Parameters

`entity` T

The entity object

Returns

Task

AddAsync<TEntity>(TEntity)

```
Task AddAsync<TEntity>(TEntity entity) where TEntity : class
```

Parameters

`entity` TEntity

Returns

Task

Type Parameters

TEntity

AddRangeAndSaveAsync(ICollection<T>)

```
Task AddRangeAndSaveAsync(ICollection<T> entities)
```

Parameters

entities ICollection<T>

Returns

Task

AddRangeAndSaveAsync<TEntity>(ICollection<TEntity>)

```
Task AddRangeAndSaveAsync<TEntity>(ICollection<TEntity> entities) where TEntity
: class
```

Parameters

entities ICollection<TEntity>

Returns

Task

Type Parameters

TEntity

AddRangeAsync(ICollection<T>)

Add range of entities to repository

```
Task AddRangeAsync(ICollection<T> entities)
```

Parameters

`entities` ICollection<T>

The entities list

Returns

Task

AddRangeAsync<TEntity>(ICollection<TEntity>)

```
Task AddRangeAsync<TEntity>(ICollection<TEntity> entities) where TEntity : class
```

Parameters

`entities` ICollection<TEntity>

Returns

Task

Type Parameters

`TEntity`

AnyAsyncBy(Expression<Func<T, bool>>)

Checks if any entity in the repository matches the provided expression

```
Task<bool> AnyAsyncBy(Expression<Func<T, bool>> where)
```

Parameters

`where` Expression<Func<T, bool>>

The expression that describes the condition to match

Returns

Task<bool>

True if any entity matches the condition, False otherwise

AnyAsyncBy<TEntity>(Expression<Func<TEntity, bool>>)

```
Task<bool> AnyAsyncBy<TEntity>(Expression<Func<TEntity, bool>> where) where TEntity
: class
```

Parameters

`where` Expression<Func<TEntity, bool>>

Returns

Task<bool>

Type Parameters

`TEntity`

AnyAsyncBy<TEntity>(IQueryable<TEntity>)

```
Task<bool> AnyAsyncBy<TEntity>(IQueryable<TEntity> queryable) where TEntity : class
```

Parameters

`queryable` IQueryable<TEntity>

Returns

Task<bool>

Type Parameters

`TEntity`

BeginTransactionAsync()

Begins a new transaction asynchronously.

```
Task<IDbContextTransaction> BeginTransactionAsync()
```

Returns

Task<IDbContextTransaction>

A task that represents the asynchronous operation. The task result is an IDbContextTransaction object which encapsulates all information about the transaction.

CountAllAsync()

Counts all entities in repository

```
Task<decimal> CountAllAsync()
```

Returns

Task<decimal>

Total count of all entities

CountAllAsyncBy(Expression<Func<T, bool>>)

Counts the total entities that match the provided expression

```
Task<decimal> CountAllAsyncBy(Expression<Func<T, bool>> where)
```

Parameters

`where` Expression<Func<T, bool>>

The expression that describes the condition to match

Returns

Task<decimal>

Total matched entities count

CountAllAsyncBy<TEntity>(Expression<Func<TEntity, bool>>)

```
Task<decimal> CountAllAsyncBy<TEntity>(Expression<Func<TEntity, bool>> where) where
TEntity : class
```

Parameters

`where` Expression<Func<TEntity, bool>>

Returns

Task<decimal>

Type Parameters

`TEntity`

FindAsync(int)

```
Task<T> FindAsync(int id)
```

Parameters

`id` int

Returns

Task<T>

FindAsync<TEntity>(int)

```
Task<TEntity> FindAsync<TEntity>(int id) where TEntity : class
```

Parameters

`id` int

Returns

Task<TEntity>

Type Parameters

`TEntity`

GetAllAsync(ICoreSpecifications<T>?)

Retrieves a collection of all entities from the repository asynchronously.

```
Task<ICollection<T>> GetAllAsync(ICoreSpecifications<T>? coreSpecifications)
```

Parameters

`coreSpecifications` ICoreSpecifications<T>

The specifications that entities must meet to be retrieved from the repository.

Returns

Task<ICollection<T>>

A task that represents the asynchronous operation. The task result contains a collection of entities satisfying The core specifications or an empty collection if no matches.

GetAllAsync(Expression<Func<T, bool>>)

Retrieves a collection of entities from the repository that satisfy the specified condition asynchronously.

```
Task<ICollection<T>> GetAllAsync(Expression<Func<T, bool>> where)
```

Parameters

`where` Expression<Func<T, bool>>

An expression representing a condition to be matched by entities in the repository.

Returns

Task<ICollection<T>>

A task that represents the asynchronous operation. The task result contains a collection of entities satisfying the condition or an empty collection if there are no matches.

GetAllAsync(string, ICoreSpecifications<T>?)

Retrieves a collection of all entities from the repository asyncronously based on a provided SQL-like query and specifications.

```
Task<ICollection<T>> GetAllAsync(string query, ICoreSpecifications<T>?
coreSpecifications)
```

Parameters

`query` string

A SQL-like query that retrieves entities from the repository.

`coreSpecifications` ICoreSpecifications<T>

The specifications that entities must meet to be retrieved from the repository.

Returns

Task<ICollection<T>>

A task that represents the asynchronous operation. The task results contains a collection od entities satisfying the query and The core specifications or an empty collection if no matches.

GetAllAsync(string, ICollection<DbParameter>, ICoreSpecifications<T>?)

Retrieves a collection of all entities from the repository asyncronously based on a provided SQL-like query, parameters and specifications.

```
Task<ICollection<T>> GetAllAsync(string query, ICollection<DbParameter> parameters,
ICoreSpecifications<T>? coreSpecifications)
```

Parameters

`query` string

A SQL-like query that retrieves entities from the repository.

`parameters` ICollection<DbParameter>

A collection of database parameters used in the query

`coreSpecifications` ICoreSpecifications<T>

The specifications that entities must meet to be retrieved from the repository.

Returns

Task<ICollection<T>>

A task that represents the asynchronous operation. The task results contains a collection od entities satisfying the query, parameters and The core specifications or an empty collection if no matches.

GetAllAsync<TEntity>(ICoreSpecifications<TEntity>)

```
Task<ICollection<TEntity>> GetAllAsync<TEntity>(ICoreSpecifications<TEntity>
coreSpecifications) where TEntity : class
```

Parameters

`coreSpecifications` ICoreSpecifications<TEntity>

Returns

Task<ICollection<TEntity>>

Type Parameters

`TEntity`

GetAllAsync<TEntity>(Expression<Func<TEntity, bool>>)

```
Task<ICollection<TEntity>> GetAllAsync<TEntity>(Expression<Func<TEntity, bool>>
where) where TEntity : class
```

Parameters

`where` Expression<Func<TEntity, bool>>

Returns

Task<ICollection<TEntity>>

Type Parameters

`TEntity`

GetAllAsync<TEntity>(string)

```
Task<ICollection<TEntity>> GetAllAsync<TEntity>(string query) where TEntity : class
```

Parameters

`query` string

Returns

Task<ICollection<TEntity>>

Type Parameters

`TEntity`

GetAllAsync<TEntity>(string, ICoreSpecifications<TEntity>)

```
Task<ICollection<TEntity>> GetAllAsync<TEntity>(string query,
ICoreSpecifications<TEntity> coreSpecifications) where TEntity : class
```

Parameters

`query` string

`coreSpecifications` ICoreSpecifications<TEntity>

Returns

Task<ICollection<TEntity>>

Type Parameters

`TEntity`

GetAllAsync<TEntity>(string, ICollection<DbParameter>)

```
Task<ICollection<TEntity>> GetAllAsync<TEntity>(string query,
ICollection<DbParameter> parameters) where TEntity : class
```

Parameters

`query` string

`parameters` ICollection<DbParameter>

Returns

Task<ICollection<TEntity>>

Type Parameters

`TEntity`

GetAllAsync<TEntity>(string, ICollection<DbParameter>, ICoreSpecifications<TEntity>)

```
Task<ICollection<TEntity>> GetAllAsync<TEntity>(string query,
ICollection<DbParameter> parameters, ICoreSpecifications<TEntity>
coreSpecifications) where TEntity : class
```

Parameters

`query` string

`parameters` ICollection<DbParameter>

`coreSpecifications` ICoreSpecifications<TEntity>

Returns

Task<ICollection<TEntity>>

Type Parameters

`TEntity`

GetAllByQueryable<TEntity>(IQueryable<TEntity>)

Retrieves a collection of entities from a specified IQueryable.

```
Task<ICollection<TEntity>> GetAllByQueryable<TEntity>(IQueryable<TEntity> query)
```

Parameters

`query` IQueryable<TEntity>

An IQueryable that retrieves entities from the repository.

Returns

Task<ICollection<TEntity>>

A task that represents the asynchronous operation. The task result contains a collection of entities that satisfy the query or an empty collection if no matches.

Type Parameters

`TEntity`

GetAllQueryable()

Retrieves all entities from the repository asynchronously.

```
IQueryable<T> GetAllQueryable()
```

Returns

IQueryable<T>

A queryable collection of all entities in the repository.

GetAllQueryable(ICoreSpecifications<T>?)

Retrieves entities that match the specified Solstice specifications from the repository asynchronously.

```
IQueryable<T> GetAllQueryable(ICoreSpecifications<T>? coreSpecifications)
```

Parameters

`coreSpecifications` ICoreSpecifications<T>

The specifications that entities must meet to be retrieved from the repository.

Returns

IQueryable<T>

A queryable collection of entities that satisfy The core specifications.

GetAllQueryable(Expression<Func<T, bool>>)

Retrieves entities that match the specified condition from the repository asynchronously.

```
IQueryable<T> GetAllQueryable(Expression<Func<T, bool>> where)
```

Parameters

`where` Expression<Func<T, bool>>

An expression representing a condition to be matched by entities in the repository.

Returns

IQueryable<T>

A queryable collection of entities matching the condition.

GetAllQueryable(string, ICoreSpecifications<T>?)

Retrieves entities from the repository based on a provided SQL-like query and specifications asynchronously.

```
IQueryable<T> GetAllQueryable(string query, ICoreSpecifications<T>?
coreSpecifications)
```

Parameters

`query` string

A SQL-like query that retrieves entities from the repository.

`coreSpecifications` ICoreSpecifications<T>

The specifications that entities must meet to be retrieved from the repository.

Returns

IQueryable<T>

A queryable collection of entities satisfying the query and The core specifications.

GetAllQueryable(string, ICollection<DbParameter>, ICoreSpecifications<T>?)

Retrieves entities from the repository based on a provided SQL-like query, parameters and specifications asynchronously.

```
IQueryable<T> GetAllQueryable(string query, ICollection<DbParameter> parameters,
ICoreSpecifications<T>? coreSpecifications)
```

Parameters

`query` string

A SQL-like query that retrieves entities from the repository.

`parameters` ICollection<DbParameter>

A collection of database parameters used in the query

`coreSpecifications` ICoreSpecifications<T>

The specifications that entities must meet to be retrieved from the repository.

Returns

IQueryable<T>

A queryable collection of entities satisfying the query, parameters and The core specifications.

GetBy(ICoreSpecifications<T>)

Retrieves an entity from the repository that meets the criteria specified by the given Solstice specification.

```
Task<T?> GetBy(ICoreSpecifications<T> coreSpecifications)
```

Parameters

`coreSpecifications` ICoreSpecifications<T>

The specifications that an entity must meet to be retrieved from the repository.

Returns

Task<T>

A task that represents the asynchronous operation. The task result is the first entity that satisfies the specified Solstice specifications. If no entity satisfies the specifications, the task result is null.

GetBy(Expression<Func<T, bool>>)

Retrieves an entity that matches the specified condition from the repository.

```
Task<T?> GetBy(Expression<Func<T, bool>> where)
```

Parameters

`where` Expression<Func<T, bool>>

An expression representing a condition to be matched by entities in the repository.

Returns

Task<T>

A task that represents the asynchronous operation. The task result contains the first matching entity or null if no entity matches the condition.

GetBy<TEntity>(ICoreSpecifications<TEntity>)

```
Task<TEntity?> GetBy<TEntity>(ICoreSpecifications<TEntity> coreSpecifications) where
TEntity : class
```

Parameters

`coreSpecifications` ICoreSpecifications<TEntity>

Returns

Task<TEntity>

Type Parameters

`TEntity`

GetBy<TEntity>(Expression<Func<TEntity, bool>>)

```
Task<TEntity?> GetBy<TEntity>(Expression<Func<TEntity, bool>> where) where TEntity
: class
```

Parameters

`where` Expression<Func<TEntity, bool>>

Returns

Task<TEntity>

Type Parameters

`TEntity`

GetBy<TEntity>(string, ICollection<DbParameter>, ICoreSpecifications<TEntity>)

```
Task<TEntity?> GetBy<TEntity>(string query, ICollection<DbParameter> parameters,
ICoreSpecifications<TEntity> coreSpecifications) where TEntity : class
```

Parameters

`query` string

`parameters` ICollection<DbParameter>

`coreSpecifications` ICoreSpecifications<TEntity>

Returns

Task<TEntity>

Type Parameters

`TEntity`

GetPagedResult(Page)

Retrieves a paginated list of entities from the repository that conforms to the specified page.

```
Task<Paged<T>> GetPagedResult(Page page)
```

Parameters

`page` Page

The page number and size of the entities to be retrieved from the repository.

Returns

Task<Paged<T>>

A task that represents the asynchronous operation. The task result is a collection of entities that meet The core specifications, paginated based on the given page object. If no entities meet the specifications, the task result is an empty collection.

GetPagedResult(Page, ICoreSpecifications<T>)

```
Task<Paged<T>> GetPagedResult(Page page, ICoreSpecifications<T> coreSpecifications)
```

Parameters

`page` Page

`coreSpecifications` ICoreSpecifications<T>

Returns

Task<Paged<T>>

GetPagedResult(Page, string, ICollection<DbParameter>)

```
Task<Paged<T>> GetPagedResult(Page page, string query, ICollection<DbParameter>
parameters)
```

Parameters

`page` Page

`query` string

`parameters` ICollection<DbParameter>

Returns

Task<Paged<T>>

GetPagedResult(Page, string, ICollection<DbParameter>, ICoreSpecifications<T>)

```
Task<Paged<T>> GetPagedResult(Page page, string query, ICollection<DbParameter>
parameters, ICoreSpecifications<T> coreSpecifications)
```

Parameters

`page` Page

`query` string

`parameters` ICollection<DbParameter>

`coreSpecifications` ICoreSpecifications<T>

Returns

Task<Paged<T>>

GetPagedResult<TEntity>(Page)

```
Task<Paged<TEntity>> GetPagedResult<TEntity>(Page page) where TEntity : class
```

Parameters

`page` Page

Returns

Task<Paged<TEntity>>

Type Parameters

TEntity

GetPagedResult<TEntity>(Page, ICoreSpecifications<TEntity>)

```
Task<Paged<TEntity>> GetPagedResult<TEntity>(Page page, ICoreSpecifications<TEntity>
coreSpecifications) where TEntity : class
```

Parameters

page Page

coreSpecifications ICoreSpecifications<TEntity>

Returns

Task<Paged<TEntity>>

Type Parameters

TEntity

GetPagedResult<TEntity>(Page, string, ICollection<DbParameter>)

```
Task<Paged<TEntity>> GetPagedResult<TEntity>(Page page, string query,
ICollection<DbParameter> parameters) where TEntity : class
```

Parameters

page Page

query string

parameters ICollection<DbParameter>

Returns

Task<Paged<TEntity>>

Type Parameters

`TEntity`

GetPagedResult<TEntity>(Page, string, ICollection<DbParameter>, ICoreSpecifications<TEntity>)

```
Task<Paged<TEntity>> GetPagedResult<TEntity>(Page page, string query,
ICollection<DbParameter> parameters, ICoreSpecifications<TEntity>
coreSpecifications) where TEntity : class
```

Parameters

`page` Page

`query` string

`parameters` ICollection<DbParameter>

`coreSpecifications` ICoreSpecifications<TEntity>

Returns

Task<Paged<TEntity>>

Type Parameters

`TEntity`

PageAllAsync(Page, ICoreSpecifications<T>?)

Retrieves a collection of entities from the repository that satisfy the specified condition asyncronously, and do paging on them.

```
Task<ICollection<T>> PageAllAsync(Page page, ICoreSpecifications<T>?
```

```
coreSpecifications)
```

Parameters

`page` Page

The page number and size of the entities to be retrieved from the repository.

`coreSpecifications` ICoreSpecifications<T>

The specifications that entities must meet to be retrieved from the repository.

Returns

Task<ICollection<T>>

A task that represents the asynchronous operation. The task result contains a collection of entities satisfying the condition. or an empty collection if no matches, paged by the given information in Page object.

PageAllQueryable(Page, ICoreSpecifications<T>?)

Retrieves a queryable collection of entities from the repository that satisfy the specified condition asyncronously, and do paging on them.

```
IQueryable<T> PageAllQueryable(Page page, ICoreSpecifications<T>?
coreSpecifications)
```

Parameters

`page` Page

The page number and size of the entities to be retrieved from the repository.

`coreSpecifications` ICoreSpecifications<T>

The specifications that entities must meet to be retrieved from the repository.

Returns

IQueryable<T>

A queryable collection of all entities in the repository that fulfill the condition, paged by the given information in page object.

Remove(T)

Remove entity from repository

```
void Remove(T entity)
```

Parameters

`entity` T

The entity object

RemoveAndSaveAsync(T)

```
Task RemoveAndSaveAsync(T entity)
```

Parameters

`entity` T

Returns

Task

RemoveAndSaveAsync<TEntity>(TEntity)

```
Task RemoveAndSaveAsync<TEntity>(TEntity entity) where TEntity : class
```

Parameters

`entity` TEntity

Returns

Task

Type Parameters

`TEntity`

RemoveRange(ICollection<T>)

Remove range of entities from repository

```
void RemoveRange(ICollection<T> entities)
```

Parameters

`entities` ICollection<T>

The entities list

RemoveRangeAndSaveAsync(ICollection<T>)

```
Task RemoveRangeAndSaveAsync(ICollection<T> entities)
```

Parameters

`entities` ICollection<T>

Returns

Task

RemoveRangeAndSaveAsync<TEntity>(ICollection<TEntity>)

```
Task RemoveRangeAndSaveAsync<TEntity>(ICollection<TEntity> entities) where TEntity
: class
```

Parameters

`entities` ICollection<TEntity>

Returns

Task

Type Parameters

`TEntity`

RemoveRange<TEntity>(ICollection<TEntity>)

```
void RemoveRange<TEntity>(ICollection<TEntity> entities) where TEntity : class
```

Parameters

`entities` ICollection<TEntity>

Type Parameters

`TEntity`

Remove<TEntity>(TEntity)

```
void Remove<TEntity>(TEntity entity) where TEntity : class
```

Parameters

`entity` TEntity

Type Parameters

`TEntity`

SaveAsync()

Save changes in repository

```
Task SaveAsync()
```

Returns

Task

Update(T)

Update entity in repository

```
void Update(T entity)
```

Parameters

`entity` T

The entity object

UpdateAndSaveAsync(T)

```
Task UpdateAndSaveAsync(T entity)
```

Parameters

`entity` T

Returns

Task

UpdateAndSaveAsync<TEntity>(TEntity)

```
Task UpdateAndSaveAsync<TEntity>(TEntity entity) where TEntity : class
```

Parameters

`entity` TEntity

Returns

Task

Type Parameters

`TEntity`

UpdateRange(ICollection<T>)

Update range of entities in repository

```
void UpdateRange(ICollection<T> entities)
```

Parameters

`entities` ICollection<T>

The entities list

UpdateRangeAndSaveAsync(ICollection<T>)

```
Task UpdateRangeAndSaveAsync(ICollection<T> entities)
```

Parameters

`entities` ICollection<T>

Returns

Task

UpdateRangeAndSaveAsync<TEntity>(ICollection<TEntity>)

```
Task UpdateRangeAndSaveAsync<TEntity>(ICollection<TEntity> entities) where TEntity
: class
```

Parameters

`entities` ICollection<TEntity>

Returns

Task

Type Parameters

`TEntity`

UpdateRange<TEntity>(ICollection<TEntity>)

```
void UpdateRange<TEntity>(ICollection<TEntity> entities) where TEntity : class
```

Parameters

`entities` ICollection<TEntity>

Type Parameters

`TEntity`

Update<TEntity>(TEntity)

```
void Update<TEntity>(TEntity entity) where TEntity : class
```

Parameters

`entity` TEntity

Type Parameters

`TEntity`

Namespace Solstice.Applications.Injections

Classes

ServiceInjections

A static helper class for service injections.

Class ServiceInjections

Namespace: Solstice.Applications.Injections

Assembly: Solstice.Applications.dll

A static helper class for service injections.

```
public static class ServiceInjections
```

Inheritance

object ← ServiceInjections

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Methods

AddServices(IServiceCollection)

Extension method for the IServiceCollection interface. Adds services to the collection.

```
public static void AddServices(this IServiceCollection services)
```

Parameters

`services` IServiceCollection

The IServiceCollection to which the services need to be added.

Remarks

This method scans all types in all assemblies of the current application domain. It looks for types that have the ServiceAttribute. If no such types are found, an exception about no services is thrown. If there are such types, they are all added as services with scope lifetime into the passed IServiceCollection.

Exceptions

CoreException

Throws if there are no services with the ServiceAttribute present.

AddServices(IServiceCollection, string)

```
[Obsolete("AddServices with namespace is deprecated, please use AddServices or
ScanServicesIn with Service Attribute instead.")]
public static void AddServices(this IServiceCollection services,
string assemblyName)
```

Parameters

`services` IServiceCollection

`assemblyName` string

ScanServicesIn(IServiceCollection, Assembly)

Extension method for the IServiceCollection interface. Adds services to the collection.

```
public static void ScanServicesIn(this IServiceCollection services,
Assembly assembly)
```

Parameters

`services` IServiceCollection

The IServiceCollection to which the services need to be added.

`assembly` Assembly

The assembly to load

Remarks

This method scans all types in the specified assembly. It looks for types that have the ServiceAttribute. If no such types are found, an exception about no services is thrown. If

there are such types, they are all added as services with scope lifetime into the passed IServiceCollection.

Exceptions

CoreException

Throws if there are no services with the ServiceAttribute present.

ScanServicesIn(IServiceCollection, string)

Extension method for the IServiceCollection interface. Adds services to the collection.

```
public static void ScanServicesIn(this IServiceCollection services,
string assemblyName)
```

Parameters

`services` IServiceCollection

The IServiceCollection to which the services need to be added.

`assemblyName` string

The name of the assembly to load

Remarks

This method scans all types in the specified assembly. It looks for types that have the ServiceAttribute. If no such types are found, an exception about no services is thrown. If there are such types, they are all added as services with scope lifetime into the passed IServiceCollection.

Exceptions

CoreException

Throws if there are no services with the ServiceAttribute present.