Namespace Quickie.Apis.Crud

Classes

CrudController<TViewModel, TRequestHandler, TIdType>

Base class providing CRUD apis.

CrudForCollectionController<TViewModel, TRequestHandler, TIdType>

Base class providing CRUD apis in collection.

Class CrudController<TViewModel, TRequestHandler, TIdType>

Namespace: Quickie.Apis.Crud

Assembly: Quickie.dll

Base class providing CRUD apis.

```
public class CrudController<TViewModel, TRequestHandler, TIdType> : QuickieBaseApi where
TViewModel : CrudDto where TRequestHandler : ICrudRequestHandler<TViewModel, TIdType> where
TIdType : IComparable
```

Type Parameters

`TViewModel`

Dto.

`TRequestHandler`

Request handler (Service layer).

`TIdType`

Type of your primary key.

Inheritance

object ← ControllerBase ← QuickieBaseApi ←
CrudController<TViewModel, TRequestHandler, TIdType>

Inherited Members

ControllerBase.StatusCode(int) , ControllerBase.StatusCode(int, object) ,
ControllerBase.Content(string) , ControllerBase.Content(string, string) ,
ControllerBase.Content(string, string, Encoding) ,
ControllerBase.Content(string, MediaTypeHeaderValue) , ControllerBase.NoContent() ,
ControllerBase.Ok() , ControllerBase.Ok(object) , ControllerBase.Redirect(string) ,
ControllerBase.RedirectPermanent(string) , ControllerBase.RedirectPreserveMethod(string) ,
ControllerBase.RedirectPermanentPreserveMethod(string) , ControllerBase.LocalRedirect(string) ,
ControllerBase.LocalRedirectPermanent(string) , ControllerBase.LocalRedirectPreserveMethod(string) ,
ControllerBase.LocalRedirectPermanentPreserveMethod(string) , ControllerBase.RedirectToAction() ,

ControllerBase.RedirectToAction(string) , ControllerBase.RedirectToAction(string, object) ,
ControllerBase.RedirectToAction(string, string) ,
ControllerBase.RedirectToAction(string, string, object) ,
ControllerBase.RedirectToAction(string, string, string) ,
ControllerBase.RedirectToAction(string, string, object, string) ,
ControllerBase.RedirectToActionPreserveMethod(string, string, object, string) ,
ControllerBase.RedirectToActionPermanent(string) ,
ControllerBase.RedirectToActionPermanent(string, object) ,
ControllerBase.RedirectToActionPermanent(string, string) ,
ControllerBase.RedirectToActionPermanent(string, string, string) ,
ControllerBase.RedirectToActionPermanent(string, string, object) ,
ControllerBase.RedirectToActionPermanent(string, string, object, string) ,
ControllerBase.RedirectToActionPermanentPreserveMethod(string, string, object, string) ,
ControllerBase.RedirectToRoute(string) , ControllerBase.RedirectToRoute(object) ,
ControllerBase.RedirectToRoute(string, object) , ControllerBase.RedirectToRoute(string, string) ,
ControllerBase.RedirectToRoute(string, object, string) ,
ControllerBase.RedirectToRoutePreserveMethod(string, object, string) ,
ControllerBase.RedirectToRoutePermanent(string) ,
ControllerBase.RedirectToRoutePermanent(object) ,
ControllerBase.RedirectToRoutePermanent(string, object) ,
ControllerBase.RedirectToRoutePermanent(string, string) ,
ControllerBase.RedirectToRoutePermanent(string, object, string) ,
ControllerBase.RedirectToRoutePermanentPreserveMethod(string, object, string) ,
ControllerBase.RedirectToPage(string) , ControllerBase.RedirectToPage(string, object) ,
ControllerBase.RedirectToPage(string, string) , ControllerBase.RedirectToPage(string, string, object) ,
ControllerBase.RedirectToPage(string, string, string) ,
ControllerBase.RedirectToPage(string, string, object, string) ,
ControllerBase.RedirectToPagePermanent(string) ,
ControllerBase.RedirectToPagePermanent(string, object) ,
ControllerBase.RedirectToPagePermanent(string, string) ,
ControllerBase.RedirectToPagePermanent(string, string, string) ,
ControllerBase.RedirectToPagePermanent(string, string, object, string) ,
ControllerBase.RedirectToPagePreserveMethod(string, string, object, string) ,
ControllerBase.RedirectToPagePermanentPreserveMethod(string, string, object, string) ,
ControllerBase.File(byte[], string) , ControllerBase.File(byte[], string, bool) ,
ControllerBase.File(byte[], string, string) , ControllerBase.File(byte[], string, string, bool) ,
ControllerBase.File(byte[], string, DateTimeOffset?, EntityTagHeaderValue) ,
ControllerBase.File(byte[], string, DateTimeOffset?, EntityTagHeaderValue, bool) ,
ControllerBase.File(byte[], string, string, DateTimeOffset?, EntityTagHeaderValue) ,
ControllerBase.File(byte[], string, string, DateTimeOffset?, EntityTagHeaderValue, bool) ,

ControllerBase.File(Stream, string) , ControllerBase.File(Stream, string, bool) ,
ControllerBase.File(Stream, string, string) , ControllerBase.File(Stream, string, string, bool) ,
ControllerBase.File(Stream, string, DateTimeOffset?, EntityTagHeaderValue) ,
ControllerBase.File(Stream, string, DateTimeOffset?, EntityTagHeaderValue, bool) ,
ControllerBase.File(Stream, string, string, DateTimeOffset?, EntityTagHeaderValue) ,
ControllerBase.File(Stream, string, string, DateTimeOffset?, EntityTagHeaderValue, bool) ,
ControllerBase.File(string, string) , ControllerBase.File(string, string, bool) ,
ControllerBase.File(string, string, string) , ControllerBase.File(string, string, string, bool) ,
ControllerBase.File(string, string, DateTimeOffset?, EntityTagHeaderValue) ,
ControllerBase.File(string, string, DateTimeOffset?, EntityTagHeaderValue, bool) ,
ControllerBase.File(string, string, string, DateTimeOffset?, EntityTagHeaderValue) ,
ControllerBase.File(string, string, string, DateTimeOffset?, EntityTagHeaderValue, bool) ,
ControllerBase.PhysicalFile(string, string) , ControllerBase.PhysicalFile(string, string, bool) ,
ControllerBase.PhysicalFile(string, string, string) ,
ControllerBase.PhysicalFile(string, string, string, bool) ,
ControllerBase.PhysicalFile(string, string, DateTimeOffset?, EntityTagHeaderValue) ,
ControllerBase.PhysicalFile(string, string, DateTimeOffset?, EntityTagHeaderValue, bool) ,
ControllerBase.PhysicalFile(string, string, string, DateTimeOffset?, EntityTagHeaderValue) ,
ControllerBase.PhysicalFile(string, string, string, DateTimeOffset?, EntityTagHeaderValue, bool) ,
ControllerBase.Unauthorized() , ControllerBase.Unauthorized(object) , ControllerBase.NotFound() ,
ControllerBase.NotFound(object) , ControllerBase.BadRequest() ,
ControllerBase.BadRequest(object) , ControllerBase.BadRequest(ModelStateDictionary) ,
ControllerBase.UnprocessableEntity() , ControllerBase.UnprocessableEntity(object) ,
ControllerBase.UnprocessableEntity(ModelStateDictionary) , ControllerBase.Conflict() ,
ControllerBase.Conflict(object) , ControllerBase.Conflict(ModelStateDictionary) ,
ControllerBase.Problem(string, string, int?, string, string) ,
ControllerBase.Problem(string, string, int?, string, string, IDictionary<string, object>) ,
ControllerBase.ValidationProblem(ValidationProblemDetails) ,
ControllerBase.ValidationProblem(ModelStateDictionary) , ControllerBase.ValidationProblem() ,
ControllerBase.ValidationProblem(string, string, int?, string, string, ModelStateDictionary) ,
ControllerBase.ValidationProblem(string, string, int?, string, string, ModelStateDictionary,
IDictionary<string, object>) ,
ControllerBase.Created() , ControllerBase.Created(string, object) ,
ControllerBase.Created(Uri, object) , ControllerBase.CreatedAtAction(string, object) ,
ControllerBase.CreatedAtAction(string, object, object) ,
ControllerBase.CreatedAtAction(string, string, object, object) ,
ControllerBase.CreatedAtRoute(string, object) , ControllerBase.CreatedAtRoute(object, object) ,
ControllerBase.CreatedAtRoute(string, object, object) , ControllerBase.Accepted() ,
ControllerBase.Accepted(object) , ControllerBase.Accepted(Uri) , ControllerBase.Accepted(string) ,
ControllerBase.Accepted(string, object) , ControllerBase.Accepted(Uri, object) ,

ControllerBase.AcceptedAtAction(string) , ControllerBase.AcceptedAtAction(string, string) , ControllerBase.AcceptedAtAction(string, object) , ControllerBase.AcceptedAtAction(string, string, object) , ControllerBase.AcceptedAtAction(string, object, object) , ControllerBase.AcceptedAtAction(string, string, object, object) , ControllerBase.AcceptedAtRoute(object) , ControllerBase.AcceptedAtRoute(string) , ControllerBase.AcceptedAtRoute(string, object) , ControllerBase.AcceptedAtRoute(object, object) , ControllerBase.AcceptedAtRoute(string, object, object) , ControllerBase.Challenge() , ControllerBase.Challenge(params string[]) , ControllerBase.Challenge(AuthenticationProperties) , ControllerBase.Challenge(AuthenticationProperties, params string[]) , ControllerBase.Forbid() , ControllerBase.Forbid(params string[]) , ControllerBase.Forbid(AuthenticationProperties) , ControllerBase.Forbid(AuthenticationProperties, params string[]) , ControllerBase.SignIn(ClaimsPrincipal) , ControllerBase.SignIn(ClaimsPrincipal, string) , ControllerBase.SignIn(ClaimsPrincipal, AuthenticationProperties) , ControllerBase.SignIn(ClaimsPrincipal, AuthenticationProperties, string) , ControllerBase.SignOut() , ControllerBase.SignOut(AuthenticationProperties) , ControllerBase.SignOut(params string[]) , ControllerBase.SignOut(AuthenticationProperties, params string[]) , ControllerBase.TryUpdateModelAsync<TModel>(TModel) , ControllerBase.TryUpdateModelAsync<TModel>(TModel, string) , ControllerBase.TryUpdateModelAsync<TModel>(TModel, string, IValueProvider) , ControllerBase.TryUpdateModelAsync<TModel>(TModel, string, params Expression<Func<TModel, object>>[]) , ControllerBase.TryUpdateModelAsync<TModel>(TModel, string, Func<ModelMetadata, bool>) , ControllerBase.TryUpdateModelAsync<TModel>(TModel, string, IValueProvider, params Expression<Func<TModel, object>>[]) , ControllerBase.TryUpdateModelAsync<TModel>(TModel, string, IValueProvider, Func<ModelMetadata, bool>) , ControllerBase.TryUpdateModelAsync(object, Type, string) , ControllerBase.TryUpdateModelAsync(object, Type, string, IValueProvider, Func<ModelMetadata, bool>) , ControllerBase.TryValidateModel(object) , ControllerBase.TryValidateModel(object, string) , ControllerBase.HttpContext , ControllerBase.Request , ControllerBase.Response , ControllerBase.RouteData , ControllerBase.ModelState , ControllerBase.ControllerContext , ControllerBase.MetadataProvider , ControllerBase.ModelBinderFactory , ControllerBase.Url , ControllerBase.ObjectValidator , ControllerBase.ProblemDetailsFactory , ControllerBase.User , ControllerBase.Empty , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

CrudController(TRequestHandler)

Base class providing CRUD apis.

```
public CrudController(TRequestHandler requestHandler)
```

Parameters

`requestHandler` TRequestHandler

Request handler (Service layer)

Methods

CreateAsync(string?, TViewModel, CancellationToken)

Create data.

```
[HttpPost]
public Task<ActionResult<ResponseObj<TViewModel>>> CreateAsync(string? idempotencyKey,
TViewModel request, CancellationToken cancellationToken = default)
```

Parameters

`idempotencyKey` string

Idempotency key

`request` TViewModel

Requested data to create

`cancellationToken` CancellationToken

Cancellation token

Returns

Task<ActionResult<ResponseObj<TViewModel>>>

HTTP 201 Created, or 400 BadRequest.

DeleteAsync(TIdType, CancellationToken)

Delete data

```
[HttpDelete("{id}")]
public Task<ActionResult<ResponseObj>> DeleteAsync(TIdType id, CancellationToken
cancellationToken = default)
```

Parameters

`id` TIdType

Identifier of data

`cancellationToken` CancellationToken

Cancellation token

Returns

Task<ActionResult<ResponseObj>>

HTTP 204 NoContent or 400 BadRequest.

GetByIdAsync(TIdType, CancellationToken)

Get data by id.

```
[HttpGet("{id}")]
public Task<ActionResult<ResponseObj<TViewModel>>> GetByIdAsync(TIdType id,
CancellationToken cancellationToken = default)
```

Parameters

`id` TIdType

Identifier

`cancellationToken` CancellationToken

Cancellation token

Returns

Task<ActionResult<ResponseObj<TViewModel>>>

HTTP 200 Ok, 404 NotFound, or 400 BadRequest.

UpdateAsync(TIdType, TViewModel, CancellationToken)

Update data.

```
[HttpPut("{id}")]
public Task<ActionResult<ResponseObj<TViewModel>>> UpdateAsync(TIdType id, TViewModel
request, CancellationToken cancellationToken = default)
```

Parameters

`id` TIdType

Identifier of data

`request` TViewModel

Requested data to update

`cancellationToken` CancellationToken

Cancellation token

Returns

Task<ActionResult<ResponseObj<TViewModel>>>

HTTP 200 Ok, or 400 BadRequest.

Class CrudForCollectionController<TViewModel, TRequestHandler, TIdType>

Namespace: Quickie.Apis.Crud

Assembly: Quickie.dll

Base class providing CRUD apis in collection.

```
public class CrudForCollectionController<TViewModel, TRequestHandler, TIdType> :
QuickieBaseApi where TViewModel : CrudDto where TRequestHandler :
ICrudForCollectionRequestHandler<TViewModel, TIdType> where TIdType : IComparable
```

Type Parameters

`TViewModel`

Dto

`TRequestHandler`

Request handler (Service layer)

`TIdType`

Type of your primary key.

Inheritance

object ← ControllerBase ← QuickieBaseApi ← CrudForCollectionController<TViewModel, TRequestHandler, TIdType>

Inherited Members

ControllerBase.StatusCode(int) , ControllerBase.StatusCode(int, object) , ControllerBase.Content(string) , ControllerBase.Content(string, string) , ControllerBase.Content(string, string, Encoding) , ControllerBase.Content(string, MediaTypeHeaderValue) , ControllerBase.NoContent() , ControllerBase.Ok() , ControllerBase.Ok(object) , ControllerBase.Redirect(string) , ControllerBase.RedirectPermanent(string) , ControllerBase.RedirectPreserveMethod(string) , ControllerBase.RedirectPermanentPreserveMethod(string) , ControllerBase.LocalRedirect(string) , ControllerBase.LocalRedirectPermanent(string) , ControllerBase.LocalRedirectPreserveMethod(string) ,

ControllerBase.LocalRedirectPermanentPreserveMethod(string) , ControllerBase.RedirectToAction() ,
ControllerBase.RedirectToAction(string) , ControllerBase.RedirectToAction(string, object) ,
ControllerBase.RedirectToAction(string, string) ,
ControllerBase.RedirectToAction(string, string, object) ,
ControllerBase.RedirectToAction(string, string, string) ,
ControllerBase.RedirectToAction(string, string, object, string) ,
ControllerBase.RedirectToActionPreserveMethod(string, string, object, string) ,
ControllerBase.RedirectToActionPermanent(string) ,
ControllerBase.RedirectToActionPermanent(string, object) ,
ControllerBase.RedirectToActionPermanent(string, string) ,
ControllerBase.RedirectToActionPermanent(string, string, string) ,
ControllerBase.RedirectToActionPermanent(string, string, object) ,
ControllerBase.RedirectToActionPermanent(string, string, object, string) ,
ControllerBase.RedirectToActionPermanentPreserveMethod(string, string, object, string) ,
ControllerBase.RedirectToRoute(string) , ControllerBase.RedirectToRoute(object) ,
ControllerBase.RedirectToRoute(string, object) , ControllerBase.RedirectToRoute(string, string) ,
ControllerBase.RedirectToRoute(string, object, string) ,
ControllerBase.RedirectToRoutePreserveMethod(string, object, string) ,
ControllerBase.RedirectToRoutePermanent(string) ,
ControllerBase.RedirectToRoutePermanent(object) ,
ControllerBase.RedirectToRoutePermanent(string, object) ,
ControllerBase.RedirectToRoutePermanent(string, string) ,
ControllerBase.RedirectToRoutePermanent(string, object, string) ,
ControllerBase.RedirectToRoutePermanentPreserveMethod(string, object, string) ,
ControllerBase.RedirectToPage(string) , ControllerBase.RedirectToPage(string, object) ,
ControllerBase.RedirectToPage(string, string) , ControllerBase.RedirectToPage(string, string, object) ,
ControllerBase.RedirectToPage(string, string, string) ,
ControllerBase.RedirectToPage(string, string, object, string) ,
ControllerBase.RedirectToPagePermanent(string) ,
ControllerBase.RedirectToPagePermanent(string, object) ,
ControllerBase.RedirectToPagePermanent(string, string) ,
ControllerBase.RedirectToPagePermanent(string, string, string) ,
ControllerBase.RedirectToPagePermanent(string, string, object, string) ,
ControllerBase.RedirectToPagePreserveMethod(string, string, object, string) ,
ControllerBase.RedirectToPagePermanentPreserveMethod(string, string, object, string) ,
ControllerBase.File(byte[], string) , ControllerBase.File(byte[], string, bool) ,
ControllerBase.File(byte[], string, string) , ControllerBase.File(byte[], string, string, bool) ,
ControllerBase.File(byte[], string, DateTimeOffset?, EntityTagHeaderValue) ,
ControllerBase.File(byte[], string, DateTimeOffset?, EntityTagHeaderValue, bool) ,
ControllerBase.File(byte[], string, string, DateTimeOffset?, EntityTagHeaderValue) ,

ControllerBase.File(byte[], string, string, DateTimeOffset?, EntityTagHeaderValue, bool) ,
ControllerBase.File(Stream, string) , ControllerBase.File(Stream, string, bool) ,
ControllerBase.File(Stream, string, string) , ControllerBase.File(Stream, string, string, bool) ,
ControllerBase.File(Stream, string, DateTimeOffset?, EntityTagHeaderValue) ,
ControllerBase.File(Stream, string, DateTimeOffset?, EntityTagHeaderValue, bool) ,
ControllerBase.File(Stream, string, string, DateTimeOffset?, EntityTagHeaderValue) ,
ControllerBase.File(Stream, string, string, DateTimeOffset?, EntityTagHeaderValue, bool) ,
ControllerBase.File(string, string) , ControllerBase.File(string, string, bool) ,
ControllerBase.File(string, string, string) , ControllerBase.File(string, string, string, bool) ,
ControllerBase.File(string, string, DateTimeOffset?, EntityTagHeaderValue) ,
ControllerBase.File(string, string, DateTimeOffset?, EntityTagHeaderValue, bool) ,
ControllerBase.File(string, string, string, DateTimeOffset?, EntityTagHeaderValue) ,
ControllerBase.File(string, string, string, DateTimeOffset?, EntityTagHeaderValue, bool) ,
ControllerBase.PhysicalFile(string, string) , ControllerBase.PhysicalFile(string, string, bool) ,
ControllerBase.PhysicalFile(string, string, string) ,
ControllerBase.PhysicalFile(string, string, string, bool) ,
ControllerBase.PhysicalFile(string, string, DateTimeOffset?, EntityTagHeaderValue) ,
ControllerBase.PhysicalFile(string, string, DateTimeOffset?, EntityTagHeaderValue, bool) ,
ControllerBase.PhysicalFile(string, string, string, DateTimeOffset?, EntityTagHeaderValue) ,
ControllerBase.PhysicalFile(string, string, string, DateTimeOffset?, EntityTagHeaderValue, bool) ,
ControllerBase.Unauthorized() , ControllerBase.Unauthorized(object) , ControllerBase.NotFound() ,
ControllerBase.NotFound(object) , ControllerBase.BadRequest() ,
ControllerBase.BadRequest(object) , ControllerBase.BadRequest(ModelStateDictionary) ,
ControllerBase.UnprocessableEntity() , ControllerBase.UnprocessableEntity(object) ,
ControllerBase.UnprocessableEntity(ModelStateDictionary) , ControllerBase.Conflict() ,
ControllerBase.Conflict(object) , ControllerBase.Conflict(ModelStateDictionary) ,
ControllerBase.Problem(string, string, int?, string, string) ,
ControllerBase.Problem(string, string, int?, string, string, IDictionary<string, object>) ,
ControllerBase.ValidationProblem(ValidationProblemDetails) ,
ControllerBase.ValidationProblem(ModelStateDictionary) , ControllerBase.ValidationProblem() ,
ControllerBase.ValidationProblem(string, string, int?, string, string, ModelStateDictionary) ,
ControllerBase.ValidationProblem(string, string, int?, string, string, ModelStateDictionary,
IDictionary<string, object>) ,
ControllerBase.Created() , ControllerBase.Created(string, object) ,
ControllerBase.Created(Uri, object) , ControllerBase.CreatedAtAction(string, object) ,
ControllerBase.CreatedAtAction(string, object, object) ,
ControllerBase.CreatedAtAction(string, string, object, object) ,
ControllerBase.CreatedAtRoute(string, object) , ControllerBase.CreatedAtRoute(object, object) ,
ControllerBase.CreatedAtRoute(string, object, object) , ControllerBase.Accepted() ,
ControllerBase.Accepted(object) , ControllerBase.Accepted(Uri) , ControllerBase.Accepted(string) ,

ControllerBase.Accepted(string, object) , ControllerBase.Accepted(Uri, object) , ControllerBase.AcceptedAtAction(string) , ControllerBase.AcceptedAtAction(string, string) , ControllerBase.AcceptedAtAction(string, object) , ControllerBase.AcceptedAtAction(string, string, object) , ControllerBase.AcceptedAtAction(string, object, object) , ControllerBase.AcceptedAtAction(string, string, object, object) , ControllerBase.AcceptedAtRoute(object) , ControllerBase.AcceptedAtRoute(string) , ControllerBase.AcceptedAtRoute(string, object) , ControllerBase.AcceptedAtRoute(object, object) , ControllerBase.AcceptedAtRoute(string, object, object) , ControllerBase.Challenge() , ControllerBase.Challenge(params string[]) , ControllerBase.Challenge(AuthenticationProperties) , ControllerBase.Challenge(AuthenticationProperties, params string[]) , ControllerBase.Forbid() , ControllerBase.Forbid(params string[]) , ControllerBase.Forbid(AuthenticationProperties) , ControllerBase.Forbid(AuthenticationProperties, params string[]) , ControllerBase.SignIn(ClaimsPrincipal) , ControllerBase.SignIn(ClaimsPrincipal, string) , ControllerBase.SignIn(ClaimsPrincipal, AuthenticationProperties) , ControllerBase.SignIn(ClaimsPrincipal, AuthenticationProperties, string) , ControllerBase.SignOut() , ControllerBase.SignOut(AuthenticationProperties) , ControllerBase.SignOut(params string[]) , ControllerBase.SignOut(AuthenticationProperties, params string[]) , ControllerBase.TryUpdateModelAsync<TModel>(TModel) , ControllerBase.TryUpdateModelAsync<TModel>(TModel, string) , ControllerBase.TryUpdateModelAsync<TModel>(TModel, string, IValueProvider) , ControllerBase.TryUpdateModelAsync<TModel>(TModel, string, params Expression<Func<TModel, object>>[]) , ControllerBase.TryUpdateModelAsync<TModel>(TModel, string, Func<ModelMetadata, bool>) , ControllerBase.TryUpdateModelAsync<TModel>(TModel, string, IValueProvider, params Expression<Func<TModel, object>>[]) , ControllerBase.TryUpdateModelAsync<TModel>(TModel, string, IValueProvider, Func<ModelMetadata, bool>) , ControllerBase.TryUpdateModelAsync(object, Type, string) , ControllerBase.TryUpdateModelAsync(object, Type, string, IValueProvider, Func<ModelMetadata, bool>) , ControllerBase.TryValidateModel(object) , ControllerBase.TryValidateModel(object, string) , ControllerBase.HttpContext , ControllerBase.Request , ControllerBase.Response , ControllerBase.RouteData , ControllerBase.ModelState , ControllerBase.ControllerContext , ControllerBase.MetadataProvider , ControllerBase.ModelBinderFactory , ControllerBase.Url , ControllerBase.ObjectValidator , ControllerBase.ProblemDetailsFactory , ControllerBase.User , ControllerBase.Empty , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

CrudForCollectionController(TRequestHandler)

Base class providing CRUD apis in collection.

```
public CrudForCollectionController(TRequestHandler requestHandler)
```

Parameters

`requestHandler` TRequestHandler

Methods

CountAsync(CancellationToken)

Total count of data.

```
[HttpGet("items/count")]
public Task<ActionResult<ResponseObj<int>>> CountAsync(CancellationToken cancellationToken
= default)
```

Parameters

`cancellationToken` CancellationToken

Cancellation token

Returns

Task<ActionResult<ResponseObj<int>>>

HTTP 200 Ok or 400 BadRequest.

CreateAsync(string?, ICollection<TViewModel>, CancellationToken)

Create collection of data.

```
[HttpPost("items")]
public Task<ActionResult<ResponseObj<ICollection<TViewModel>>>> CreateAsync(string?
idempotencyKey, ICollection<TViewModel> request, CancellationToken cancellationToken
= default)
```

Parameters

`idempotencyKey` string

Idempotency key.

`request` ICollection<TViewModel>

Requested data to create.

`cancellationToken` CancellationToken

Cancellation token

Returns

Task<ActionResult<ResponseObj<ICollection<TViewModel>>>>

HTTP 201 Created, or 400 BadRequest.

DeleteAsync(ICollection<TIdType>, CancellationToken)

DELETE request for collection of data.

```
[HttpDelete("items")]
public Task<ActionResult<ResponseObj>> DeleteAsync(ICollection<TIdType> ids,
CancellationToken cancellationToken = default)
```

Parameters

`ids` ICollection<TIdType>

Collection of id.

`cancellationToken` CancellationToken

Cancellation token

Returns

Task<ActionResult<ResponseObj>>

HTTP 204 NoContent or 400 BadRequest.

GetAllAsync(RequestForDataObj, CancellationToken)

Get all data. (Paginated)

```
[HttpGet("items")]
public Task<ActionResult<ResponseObj<PaginatedDataObj<TViewModel>>>>
GetAllAsync(RequestForDataObj request, CancellationToken cancellationToken = default)
```

Parameters

`request` RequestForDataObj

Request object.

`cancellationToken` CancellationToken

Cancellation token

Returns

Task<ActionResult<ResponseObj<PaginatedDataObj<TViewModel>>>>

HTTP 200 Ok, 404 NotFound, or 400 BadRequest.

UpdateAsync(ICollection<TViewModel>, CancellationToken)

Update collection of data.

```
[HttpPut("items")]
public Task<ActionResult<ResponseObj<ICollection<TViewModel>>>>
UpdateAsync(ICollection<TViewModel> request, CancellationToken cancellationToken = default)
```

Parameters

`request` ICollection<TViewModel>

Requested data to update

`cancellationToken` CancellationToken

Cancellation token

Returns

Task<ActionResult<ResponseObj<ICollection<TViewModel>>>>

HTTP 200 Ok, or 400 BadRequest.

Namespace Quickie.Apis.Editonly

Classes

EditOnlyController<TViewModel, TRequestHandler, TIdType>

Base class providing Edit-Only APIs.

Class EditOnlyController<TViewModel, TRequestHandler, TIdType>

Namespace: Quickie.Apis.Editonly

Assembly: Quickie.dll

Base class providing Edit-Only APIs.

```
public class EditOnlyController<TViewModel, TRequestHandler, TIdType> : QuickieBaseApi where
TViewModel : EditOnlyDto where TRequestHandler : IEditOnlyRequestHandler<TViewModel,
TIdType> where TIdType : IComparable
```

Type Parameters

`TViewModel`

Dto

`TRequestHandler`

Request handler (Service layer)

`TIdType`

Type of your primary key

Inheritance

object ← ControllerBase ← QuickieBaseApi ←
EditOnlyController<TViewModel, TRequestHandler, TIdType>

Inherited Members

ControllerBase.StatusCode(int) , ControllerBase.StatusCode(int, object) ,
ControllerBase.Content(string) , ControllerBase.Content(string, string) ,
ControllerBase.Content(string, string, Encoding) ,
ControllerBase.Content(string, MediaTypeHeaderValue) , ControllerBase.NoContent() ,
ControllerBase.Ok() , ControllerBase.Ok(object) , ControllerBase.Redirect(string) ,
ControllerBase.RedirectPermanent(string) , ControllerBase.RedirectPreserveMethod(string) ,
ControllerBase.RedirectPermanentPreserveMethod(string) , ControllerBase.LocalRedirect(string) ,
ControllerBase.LocalRedirectPermanent(string) , ControllerBase.LocalRedirectPreserveMethod(string) ,
ControllerBase.LocalRedirectPermanentPreserveMethod(string) , ControllerBase.RedirectToAction() ,

ControllerBase.RedirectToAction(string) , ControllerBase.RedirectToAction(string, object) ,
ControllerBase.RedirectToAction(string, string) ,
ControllerBase.RedirectToAction(string, string, object) ,
ControllerBase.RedirectToAction(string, string, string) ,
ControllerBase.RedirectToAction(string, string, object, string) ,
ControllerBase.RedirectToActionPreserveMethod(string, string, object, string) ,
ControllerBase.RedirectToActionPermanent(string) ,
ControllerBase.RedirectToActionPermanent(string, object) ,
ControllerBase.RedirectToActionPermanent(string, string) ,
ControllerBase.RedirectToActionPermanent(string, string, string) ,
ControllerBase.RedirectToActionPermanent(string, string, object) ,
ControllerBase.RedirectToActionPermanent(string, string, object, string) ,
ControllerBase.RedirectToActionPermanentPreserveMethod(string, string, object, string) ,
ControllerBase.RedirectToRoute(string) , ControllerBase.RedirectToRoute(object) ,
ControllerBase.RedirectToRoute(string, object) , ControllerBase.RedirectToRoute(string, string) ,
ControllerBase.RedirectToRoute(string, object, string) ,
ControllerBase.RedirectToRoutePreserveMethod(string, object, string) ,
ControllerBase.RedirectToRoutePermanent(string) ,
ControllerBase.RedirectToRoutePermanent(object) ,
ControllerBase.RedirectToRoutePermanent(string, object) ,
ControllerBase.RedirectToRoutePermanent(string, string) ,
ControllerBase.RedirectToRoutePermanent(string, object, string) ,
ControllerBase.RedirectToRoutePermanentPreserveMethod(string, object, string) ,
ControllerBase.RedirectToPage(string) , ControllerBase.RedirectToPage(string, object) ,
ControllerBase.RedirectToPage(string, string) , ControllerBase.RedirectToPage(string, string, object) ,
ControllerBase.RedirectToPage(string, string, string) ,
ControllerBase.RedirectToPage(string, string, object, string) ,
ControllerBase.RedirectToPagePermanent(string) ,
ControllerBase.RedirectToPagePermanent(string, object) ,
ControllerBase.RedirectToPagePermanent(string, string) ,
ControllerBase.RedirectToPagePermanent(string, string, string) ,
ControllerBase.RedirectToPagePermanent(string, string, object, string) ,
ControllerBase.RedirectToPagePreserveMethod(string, string, object, string) ,
ControllerBase.RedirectToPagePermanentPreserveMethod(string, string, object, string) ,
ControllerBase.File(byte[], string) , ControllerBase.File(byte[], string, bool) ,
ControllerBase.File(byte[], string, string) , ControllerBase.File(byte[], string, string, bool) ,
ControllerBase.File(byte[], string, DateTimeOffset?, EntityTagHeaderValue) ,
ControllerBase.File(byte[], string, DateTimeOffset?, EntityTagHeaderValue, bool) ,
ControllerBase.File(byte[], string, string, DateTimeOffset?, EntityTagHeaderValue) ,
ControllerBase.File(byte[], string, string, DateTimeOffset?, EntityTagHeaderValue, bool) ,

ControllerBase.File(Stream, string) , ControllerBase.File(Stream, string, bool) ,
ControllerBase.File(Stream, string, string) , ControllerBase.File(Stream, string, string, bool) ,
ControllerBase.File(Stream, string, DateTimeOffset?, EntityTagHeaderValue) ,
ControllerBase.File(Stream, string, DateTimeOffset?, EntityTagHeaderValue, bool) ,
ControllerBase.File(Stream, string, string, DateTimeOffset?, EntityTagHeaderValue) ,
ControllerBase.File(Stream, string, string, DateTimeOffset?, EntityTagHeaderValue, bool) ,
ControllerBase.File(string, string) , ControllerBase.File(string, string, bool) ,
ControllerBase.File(string, string, string) , ControllerBase.File(string, string, string, bool) ,
ControllerBase.File(string, string, DateTimeOffset?, EntityTagHeaderValue) ,
ControllerBase.File(string, string, DateTimeOffset?, EntityTagHeaderValue, bool) ,
ControllerBase.File(string, string, string, DateTimeOffset?, EntityTagHeaderValue) ,
ControllerBase.File(string, string, string, DateTimeOffset?, EntityTagHeaderValue, bool) ,
ControllerBase.PhysicalFile(string, string) , ControllerBase.PhysicalFile(string, string, bool) ,
ControllerBase.PhysicalFile(string, string, string) ,
ControllerBase.PhysicalFile(string, string, string, bool) ,
ControllerBase.PhysicalFile(string, string, DateTimeOffset?, EntityTagHeaderValue) ,
ControllerBase.PhysicalFile(string, string, DateTimeOffset?, EntityTagHeaderValue, bool) ,
ControllerBase.PhysicalFile(string, string, string, DateTimeOffset?, EntityTagHeaderValue) ,
ControllerBase.PhysicalFile(string, string, string, DateTimeOffset?, EntityTagHeaderValue, bool) ,
ControllerBase.Unauthorized() , ControllerBase.Unauthorized(object) , ControllerBase.NotFound() ,
ControllerBase.NotFound(object) , ControllerBase.BadRequest() ,
ControllerBase.BadRequest(object) , ControllerBase.BadRequest(ModelStateDictionary) ,
ControllerBase.UnprocessableEntity() , ControllerBase.UnprocessableEntity(object) ,
ControllerBase.UnprocessableEntity(ModelStateDictionary) , ControllerBase.Conflict() ,
ControllerBase.Conflict(object) , ControllerBase.Conflict(ModelStateDictionary) ,
ControllerBase.Problem(string, string, int?, string, string) ,
ControllerBase.Problem(string, string, int?, string, string, IDictionary<string, object>) ,
ControllerBase.ValidationProblem(ValidationProblemDetails) ,
ControllerBase.ValidationProblem(ModelStateDictionary) , ControllerBase.ValidationProblem() ,
ControllerBase.ValidationProblem(string, string, int?, string, string, ModelStateDictionary) ,
ControllerBase.ValidationProblem(string, string, int?, string, string, ModelStateDictionary,
IDictionary<string, object>) ,
ControllerBase.Created() , ControllerBase.Created(string, object) ,
ControllerBase.Created(Uri, object) , ControllerBase.CreatedAtAction(string, object) ,
ControllerBase.CreatedAtAction(string, object, object) ,
ControllerBase.CreatedAtAction(string, string, object, object) ,
ControllerBase.CreatedAtRoute(string, object) , ControllerBase.CreatedAtRoute(object, object) ,
ControllerBase.CreatedAtRoute(string, object, object) , ControllerBase.Accepted() ,
ControllerBase.Accepted(object) , ControllerBase.Accepted(Uri) , ControllerBase.Accepted(string) ,
ControllerBase.Accepted(string, object) , ControllerBase.Accepted(Uri, object) ,

ControllerBase.AcceptedAtAction(string), ControllerBase.AcceptedAtAction(string, string), ControllerBase.AcceptedAtAction(string, object), ControllerBase.AcceptedAtAction(string, string, object), ControllerBase.AcceptedAtAction(string, object, object), ControllerBase.AcceptedAtAction(string, string, object, object), ControllerBase.AcceptedAtRoute(object), ControllerBase.AcceptedAtRoute(string), ControllerBase.AcceptedAtRoute(string, object), ControllerBase.AcceptedAtRoute(object, object), ControllerBase.AcceptedAtRoute(string, object, object), ControllerBase.Challenge(), ControllerBase.Challenge(params string[]), ControllerBase.Challenge(AuthenticationProperties), ControllerBase.Challenge(AuthenticationProperties, params string[]), ControllerBase.Forbid(), ControllerBase.Forbid(params string[]), ControllerBase.Forbid(AuthenticationProperties), ControllerBase.Forbid(AuthenticationProperties, params string[]), ControllerBase.SignIn(ClaimsPrincipal), ControllerBase.SignIn(ClaimsPrincipal, string), ControllerBase.SignIn(ClaimsPrincipal, AuthenticationProperties), ControllerBase.SignIn(ClaimsPrincipal, AuthenticationProperties, string), ControllerBase.SignOut(), ControllerBase.SignOut(AuthenticationProperties), ControllerBase.SignOut(params string[]), ControllerBase.SignOut(AuthenticationProperties, params string[]), ControllerBase.TryUpdateModelAsync<TModel>(TModel), ControllerBase.TryUpdateModelAsync<TModel>(TModel, string), ControllerBase.TryUpdateModelAsync<TModel>(TModel, string, IValueProvider), ControllerBase.TryUpdateModelAsync<TModel>(TModel, string, params Expression<Func<TModel, object>>[]), ControllerBase.TryUpdateModelAsync<TModel>(TModel, string, Func<ModelMetadata, bool>), ControllerBase.TryUpdateModelAsync<TModel>(TModel, string, IValueProvider, params Expression<Func<TModel, object>>[]), ControllerBase.TryUpdateModelAsync<TModel>(TModel, string, IValueProvider, Func<ModelMetadata, bool>), ControllerBase.TryUpdateModelAsync(object, Type, string), ControllerBase.TryUpdateModelAsync(object, Type, string, IValueProvider, Func<ModelMetadata, bool>), ControllerBase.TryValidateModel(object), ControllerBase.TryValidateModel(object, string), ControllerBase.HttpContext, ControllerBase.Request, ControllerBase.Response, ControllerBase.RouteData, ControllerBase.ModelState, ControllerBase.ControllerContext, ControllerBase.MetadataProvider, ControllerBase.ModelBinderFactory, ControllerBase.Url, ControllerBase.ObjectValidator, ControllerBase.ProblemDetailsFactory, ControllerBase.User, ControllerBase.Empty, object.Equals(object), object.Equals(object, object), object.GetHashCode(), object.GetType(), object.MemberwiseClone(), object.ReferenceEquals(object, object), object.ToString()

Constructors

EditOnlyController(IEditOnlyRequestHandler<TViewModel, TIdType>)

Base class providing Edit-Only APIs.

```
public EditOnlyController(IEditOnlyRequestHandler<TViewModel, TIdType> requestHandler)
```

Parameters

`requestHandler` IEditOnlyRequestHandler<TViewModel, TIdType>

Methods

EditAsync(TIdType, TViewModel, CancellationToken)

Edit a single item.

```
[HttpPut("{id}")]
public Task<ActionResult<ResponseObj<TViewModel>>> EditAsync(TIdType id, TViewModel
requestModel, CancellationToken cancellationToken = default)
```

Parameters

`id` TIdType

Identifier of the item

`requestModel` TViewModel

Data for editing

`cancellationToken` CancellationToken

Cancellation token

Returns

Task<ActionResult<ResponseObj<TViewModel>>>

HTTP 200 Ok, or 400 BadRequest.

EditCollectionAsync(ICollection<TViewModel>, CancellationToken)

Edit a collection of items.

```
[HttpPut("items")]
public Task<ActionResult<ResponseObj<ICollection<TViewModel>>>>
EditCollectionAsync(ICollection<TViewModel> requestModels, CancellationToken
cancellationToken = default)
```

Parameters

`requestModels` ICollection<TViewModel>

Collection of items to edit

`cancellationToken` CancellationToken

Cancellation token

Returns

Task<ActionResult<ResponseObj<ICollection<TViewModel>>>>

HTTP 200 Ok, or 400 BadRequest.

Namespace Quickie.Apis.Readonly

Classes

ReadOnlyCollectionController<TViewModel, TRequestHandler, TRequestForDataModel, TIdType>

Abstract base class for read-only (GET) operations on data collections, utilizing a request handler to manage read operations.

ReadOnlyController<TViewModel, TRequestHandler, TIdType>

Abstract base class for read-only (GETs) operations that utilize a request handler for managing read operations.

Class ReadOnlyCollectionController<TViewModel, TRequestHandler, TRequestForDataModel, TIdType>

Namespace: Quickie.Apis.Readonly

Assembly: Quickie.dll

Abstract base class for read-only (GET) operations on data collections, utilizing a request handler to manage read operations.

```
public class ReadOnlyCollectionController<TViewModel, TRequestHandler, TRequestForDataModel,
TIdType> : QuickieBaseApi where TViewModel : ReadOnlyDto where TRequestHandler :
IReadOnlyCollectionRequestHandler<TViewModel, TIdType> where TRequestForDataModel :
RequestForDataObj where TIdType : IComparable
```

Type Parameters

`TViewModel`

The type of the view model that represents the data transfer object.

`TRequestHandler`

The type of the request handler that processes the read-only operations.

`TRequestForDataModel`

The type to provide request for collection of data.

`TIdType`

Type of identifier (string, int, double, Guid, ...)

Inheritance

object ← ControllerBase ← QuickieBaseApi ← ReadOnlyCollectionController<TViewModel, TRequestHandler, TRequestForDataModel, TIdType>

Inherited Members

ControllerBase.StatusCode(int) , ControllerBase.StatusCode(int, object) ,
ControllerBase.Content(string) , ControllerBase.Content(string, string) ,
ControllerBase.Content(string, string, Encoding) ,
ControllerBase.Content(string, MediaTypeHeaderValue) , ControllerBase.NoContent() ,
ControllerBase.Ok() , ControllerBase.Ok(object) , ControllerBase.Redirect(string) ,
ControllerBase.RedirectPermanent(string) , ControllerBase.RedirectPreserveMethod(string) ,
ControllerBase.RedirectPermanentPreserveMethod(string) , ControllerBase.LocalRedirect(string) ,
ControllerBase.LocalRedirectPermanent(string) , ControllerBase.LocalRedirectPreserveMethod(string) ,
ControllerBase.LocalRedirectPermanentPreserveMethod(string) , ControllerBase.RedirectToAction() ,
ControllerBase.RedirectToAction(string) , ControllerBase.RedirectToAction(string, object) ,
ControllerBase.RedirectToAction(string, string) ,
ControllerBase.RedirectToAction(string, string, object) ,
ControllerBase.RedirectToAction(string, string, string) ,
ControllerBase.RedirectToAction(string, string, object, string) ,
ControllerBase.RedirectToActionPreserveMethod(string, string, object, string) ,
ControllerBase.RedirectToActionPermanent(string) ,
ControllerBase.RedirectToActionPermanent(string, object) ,
ControllerBase.RedirectToActionPermanent(string, string) ,
ControllerBase.RedirectToActionPermanent(string, string, string) ,
ControllerBase.RedirectToActionPermanent(string, string, object) ,
ControllerBase.RedirectToActionPermanent(string, string, object, string) ,
ControllerBase.RedirectToActionPermanentPreserveMethod(string, string, object, string) ,
ControllerBase.RedirectToRoute(string) , ControllerBase.RedirectToRoute(object) ,
ControllerBase.RedirectToRoute(string, object) , ControllerBase.RedirectToRoute(string, string) ,
ControllerBase.RedirectToRoute(string, object, string) ,
ControllerBase.RedirectToRoutePreserveMethod(string, object, string) ,
ControllerBase.RedirectToRoutePermanent(string) ,
ControllerBase.RedirectToRoutePermanent(object) ,
ControllerBase.RedirectToRoutePermanent(string, object) ,
ControllerBase.RedirectToRoutePermanent(string, string) ,
ControllerBase.RedirectToRoutePermanent(string, object, string) ,
ControllerBase.RedirectToRoutePermanentPreserveMethod(string, object, string) ,
ControllerBase.RedirectToPage(string) , ControllerBase.RedirectToPage(string, object) ,
ControllerBase.RedirectToPage(string, string) , ControllerBase.RedirectToPage(string, string, object) ,
ControllerBase.RedirectToPage(string, string, string) ,
ControllerBase.RedirectToPage(string, string, object, string) ,
ControllerBase.RedirectToPagePermanent(string) ,
ControllerBase.RedirectToPagePermanent(string, object) ,
ControllerBase.RedirectToPagePermanent(string, string) ,
ControllerBase.RedirectToPagePermanent(string, string, string) ,

ControllerBase.RedirectToPagePermanent(string, string, object, string) ,
ControllerBase.RedirectToPagePreserveMethod(string, string, object, string) ,
ControllerBase.RedirectToPagePermanentPreserveMethod(string, string, object, string) ,
ControllerBase.File(byte[], string) , ControllerBase.File(byte[], string, bool) ,
ControllerBase.File(byte[], string, string) , ControllerBase.File(byte[], string, string, bool) ,
ControllerBase.File(byte[], string, DateTimeOffset?, EntityTagHeaderValue) ,
ControllerBase.File(byte[], string, DateTimeOffset?, EntityTagHeaderValue, bool) ,
ControllerBase.File(byte[], string, string, DateTimeOffset?, EntityTagHeaderValue) ,
ControllerBase.File(byte[], string, string, DateTimeOffset?, EntityTagHeaderValue, bool) ,
ControllerBase.File(Stream, string) , ControllerBase.File(Stream, string, bool) ,
ControllerBase.File(Stream, string, string) , ControllerBase.File(Stream, string, string, bool) ,
ControllerBase.File(Stream, string, DateTimeOffset?, EntityTagHeaderValue) ,
ControllerBase.File(Stream, string, DateTimeOffset?, EntityTagHeaderValue, bool) ,
ControllerBase.File(Stream, string, string, DateTimeOffset?, EntityTagHeaderValue) ,
ControllerBase.File(Stream, string, string, DateTimeOffset?, EntityTagHeaderValue, bool) ,
ControllerBase.File(string, string) , ControllerBase.File(string, string, bool) ,
ControllerBase.File(string, string, string) , ControllerBase.File(string, string, string, bool) ,
ControllerBase.File(string, string, DateTimeOffset?, EntityTagHeaderValue) ,
ControllerBase.File(string, string, DateTimeOffset?, EntityTagHeaderValue, bool) ,
ControllerBase.File(string, string, string, DateTimeOffset?, EntityTagHeaderValue) ,
ControllerBase.File(string, string, string, DateTimeOffset?, EntityTagHeaderValue, bool) ,
ControllerBase.PhysicalFile(string, string) , ControllerBase.PhysicalFile(string, string, bool) ,
ControllerBase.PhysicalFile(string, string, string) ,
ControllerBase.PhysicalFile(string, string, string, bool) ,
ControllerBase.PhysicalFile(string, string, DateTimeOffset?, EntityTagHeaderValue) ,
ControllerBase.PhysicalFile(string, string, DateTimeOffset?, EntityTagHeaderValue, bool) ,
ControllerBase.PhysicalFile(string, string, string, DateTimeOffset?, EntityTagHeaderValue) ,
ControllerBase.PhysicalFile(string, string, string, DateTimeOffset?, EntityTagHeaderValue, bool) ,
ControllerBase.Unauthorized() , ControllerBase.Unauthorized(object) , ControllerBase.NotFound() ,
ControllerBase.NotFound(object) , ControllerBase.BadRequest() ,
ControllerBase.BadRequest(object) , ControllerBase.BadRequest(ModelStateDictionary) ,
ControllerBase.UnprocessableEntity() , ControllerBase.UnprocessableEntity(object) ,
ControllerBase.UnprocessableEntity(ModelStateDictionary) , ControllerBase.Conflict() ,
ControllerBase.Conflict(object) , ControllerBase.Conflict(ModelStateDictionary) ,
ControllerBase.Problem(string, string, int?, string, string) ,
ControllerBase.Problem(string, string, int?, string, string, IDictionary<string, object>) ,
ControllerBase.ValidationProblem(ValidationProblemDetails) ,
ControllerBase.ValidationProblem(ModelStateDictionary) , ControllerBase.ValidationProblem() ,
ControllerBase.ValidationProblem(string, string, int?, string, string, ModelStateDictionary) ,

ControllerBase.ValidationProblem(string, string, int?, string, string, ModelStateDictionary, IDictionary<string, object>) ,
ControllerBase.Created() , ControllerBase.Created(string, object) ,
ControllerBase.Created(Uri, object) , ControllerBase.CreatedAtAction(string, object) ,
ControllerBase.CreatedAtAction(string, object, object) ,
ControllerBase.CreatedAtAction(string, string, object, object) ,
ControllerBase.CreatedAtRoute(string, object) , ControllerBase.CreatedAtRoute(object, object) ,
ControllerBase.CreatedAtRoute(string, object, object) , ControllerBase.Accepted() ,
ControllerBase.Accepted(object) , ControllerBase.Accepted(Uri) , ControllerBase.Accepted(string) ,
ControllerBase.Accepted(string, object) , ControllerBase.Accepted(Uri, object) ,
ControllerBase.AcceptedAtAction(string) , ControllerBase.AcceptedAtAction(string, string) ,
ControllerBase.AcceptedAtAction(string, object) ,
ControllerBase.AcceptedAtAction(string, string, object) ,
ControllerBase.AcceptedAtAction(string, object, object) ,
ControllerBase.AcceptedAtAction(string, string, object, object) ,
ControllerBase.AcceptedAtRoute(object) , ControllerBase.AcceptedAtRoute(string) ,
ControllerBase.AcceptedAtRoute(string, object) , ControllerBase.AcceptedAtRoute(object, object) ,
ControllerBase.AcceptedAtRoute(string, object, object) , ControllerBase.Challenge() ,
ControllerBase.Challenge(params string[]) , ControllerBase.Challenge(AuthenticationProperties) ,
ControllerBase.Challenge(AuthenticationProperties, params string[]) , ControllerBase.Forbid() ,
ControllerBase.Forbid(params string[]) , ControllerBase.Forbid(AuthenticationProperties) ,
ControllerBase.Forbid(AuthenticationProperties, params string[]) ,
ControllerBase.SignIn(ClaimsPrincipal) , ControllerBase.SignIn(ClaimsPrincipal, string) ,
ControllerBase.SignIn(ClaimsPrincipal, AuthenticationProperties) ,
ControllerBase.SignIn(ClaimsPrincipal, AuthenticationProperties, string) , ControllerBase.SignOut() ,
ControllerBase.SignOut(AuthenticationProperties) , ControllerBase.SignOut(params string[]) ,
ControllerBase.SignOut(AuthenticationProperties, params string[]) ,
ControllerBase.TryUpdateModelAsync<TModel>(TModel) ,
ControllerBase.TryUpdateModelAsync<TModel>(TModel, string) ,
ControllerBase.TryUpdateModelAsync<TModel>(TModel, string, IValueProvider) ,
ControllerBase.TryUpdateModelAsync<TModel>(TModel, string, params Expression<Func<TModel, object>>[]) ,
ControllerBase.TryUpdateModelAsync<TModel>(TModel, string, Func<ModelMetadata, bool>) ,
ControllerBase.TryUpdateModelAsync<TModel>(TModel, string, IValueProvider, params Expression<Func<TModel, object>>[]) ,
ControllerBase.TryUpdateModelAsync<TModel>(TModel, string, IValueProvider, Func<ModelMetadata, bool>) ,
ControllerBase.TryUpdateModelAsync(object, Type, string) ,
ControllerBase.TryUpdateModelAsync(object, Type, string, IValueProvider, Func<ModelMetadata, bool>) ,

ControllerBase.TryValidateModel(object) , ControllerBase.TryValidateModel(object, string) , ControllerBase.HttpContext , ControllerBase.Request , ControllerBase.Response , ControllerBase.RouteData , ControllerBase.ModelState , ControllerBase.ControllerContext , ControllerBase.MetadataProvider , ControllerBase.ModelBinderFactory , ControllerBase.Url , ControllerBase.ObjectValidator , ControllerBase.ProblemDetailsFactory , ControllerBase.User , ControllerBase.Empty , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

ReadOnlyCollectionController(TRequestHandler)

Abstract base class for read-only (GET) operations on data collections, utilizing a request handler to manage read operations.

```
public ReadOnlyCollectionController(TRequestHandler requestHandler)
```

Parameters

`requestHandler` TRequestHandler

Methods

GetAsync(TRequestForDataModel, CancellationToken)

GET request based on some request.

```
[HttpGet("items")]
public Task<ActionResult<PaginatedDataObj<TViewModel>>> GetAsync(TRequestForDataModel
request, CancellationToken cancellationToken = default)
```

Parameters

`request` TRequestForDataModel

Data filtration and pagination request object.

`cancellationToken` CancellationToken

Cancellation token.

Returns

Task<ActionResult<PaginatedDataObj<TViewModel>>>

Paginated data with appropriate HTTP status codes.

Class ReadOnlyController<TViewModel, TRequestHandler, TIdType>

Namespace: Quickie.Apis.Readonly

Assembly: Quickie.dll

Abstract base class for read-only (GETs) operations that utilize a request handler for managing read operations.

```
public class ReadOnlyController<TViewModel, TRequestHandler, TIdType> : QuickieBaseApi where
TViewModel : ReadOnlyDto where TRequestHandler : IReadOnlyRequestHandler<TViewModel,
TIdType> where TIdType : IComparable
```

Type Parameters

`TViewModel`

The type of the view model that represents the data transfer object.

`TRequestHandler`

The type of the request handler that processes the read-only operations.

`TIdType`

Type of identifier (string, int, double, Guid, ...).

Inheritance

object ← ControllerBase ← QuickieBaseApi ←
ReadOnlyController<TViewModel, TRequestHandler, TIdType>

Inherited Members

ControllerBase.StatusCode(int) , ControllerBase.StatusCode(int, object) ,
ControllerBase.Content(string) , ControllerBase.Content(string, string) ,
ControllerBase.Content(string, string, Encoding) ,
ControllerBase.Content(string, MediaTypeHeaderValue) , ControllerBase.NoContent() ,
ControllerBase.Ok() , ControllerBase.Ok(object) , ControllerBase.Redirect(string) ,
ControllerBase.RedirectPermanent(string) , ControllerBase.RedirectPreserveMethod(string) ,
ControllerBase.RedirectPermanentPreserveMethod(string) , ControllerBase.LocalRedirect(string) ,
ControllerBase.LocalRedirectPermanent(string) , ControllerBase.LocalRedirectPreserveMethod(string) ,

ControllerBase.LocalRedirectPermanentPreserveMethod(string) , ControllerBase.RedirectToAction() ,
ControllerBase.RedirectToAction(string) , ControllerBase.RedirectToAction(string, object) ,
ControllerBase.RedirectToAction(string, string) ,
ControllerBase.RedirectToAction(string, string, object) ,
ControllerBase.RedirectToAction(string, string, string) ,
ControllerBase.RedirectToAction(string, string, object, string) ,
ControllerBase.RedirectToActionPreserveMethod(string, string, object, string) ,
ControllerBase.RedirectToActionPermanent(string) ,
ControllerBase.RedirectToActionPermanent(string, object) ,
ControllerBase.RedirectToActionPermanent(string, string) ,
ControllerBase.RedirectToActionPermanent(string, string, string) ,
ControllerBase.RedirectToActionPermanent(string, string, object) ,
ControllerBase.RedirectToActionPermanent(string, string, object, string) ,
ControllerBase.RedirectToActionPermanentPreserveMethod(string, string, object, string) ,
ControllerBase.RedirectToRoute(string) , ControllerBase.RedirectToRoute(object) ,
ControllerBase.RedirectToRoute(string, object) , ControllerBase.RedirectToRoute(string, string) ,
ControllerBase.RedirectToRoute(string, object, string) ,
ControllerBase.RedirectToRoutePreserveMethod(string, object, string) ,
ControllerBase.RedirectToRoutePermanent(string) ,
ControllerBase.RedirectToRoutePermanent(object) ,
ControllerBase.RedirectToRoutePermanent(string, object) ,
ControllerBase.RedirectToRoutePermanent(string, string) ,
ControllerBase.RedirectToRoutePermanent(string, object, string) ,
ControllerBase.RedirectToRoutePermanentPreserveMethod(string, object, string) ,
ControllerBase.RedirectToPage(string) , ControllerBase.RedirectToPage(string, object) ,
ControllerBase.RedirectToPage(string, string) , ControllerBase.RedirectToPage(string, string, object) ,
ControllerBase.RedirectToPage(string, string, string) ,
ControllerBase.RedirectToPage(string, string, object, string) ,
ControllerBase.RedirectToPagePermanent(string) ,
ControllerBase.RedirectToPagePermanent(string, object) ,
ControllerBase.RedirectToPagePermanent(string, string) ,
ControllerBase.RedirectToPagePermanent(string, string, string) ,
ControllerBase.RedirectToPagePermanent(string, string, object, string) ,
ControllerBase.RedirectToPagePreserveMethod(string, string, object, string) ,
ControllerBase.RedirectToPagePermanentPreserveMethod(string, string, object, string) ,
ControllerBase.File(byte[], string) , ControllerBase.File(byte[], string, bool) ,
ControllerBase.File(byte[], string, string) , ControllerBase.File(byte[], string, string, bool) ,
ControllerBase.File(byte[], string, DateTimeOffset?, EntityTagHeaderValue) ,
ControllerBase.File(byte[], string, DateTimeOffset?, EntityTagHeaderValue, bool) ,
ControllerBase.File(byte[], string, string, DateTimeOffset?, EntityTagHeaderValue) ,

ControllerBase.File(byte[], string, string, DateTimeOffset?, EntityTagHeaderValue, bool) ,
ControllerBase.File(Stream, string) , ControllerBase.File(Stream, string, bool) ,
ControllerBase.File(Stream, string, string) , ControllerBase.File(Stream, string, string, bool) ,
ControllerBase.File(Stream, string, DateTimeOffset?, EntityTagHeaderValue) ,
ControllerBase.File(Stream, string, DateTimeOffset?, EntityTagHeaderValue, bool) ,
ControllerBase.File(Stream, string, string, DateTimeOffset?, EntityTagHeaderValue) ,
ControllerBase.File(Stream, string, string, DateTimeOffset?, EntityTagHeaderValue, bool) ,
ControllerBase.File(string, string) , ControllerBase.File(string, string, bool) ,
ControllerBase.File(string, string, string) , ControllerBase.File(string, string, string, bool) ,
ControllerBase.File(string, string, DateTimeOffset?, EntityTagHeaderValue) ,
ControllerBase.File(string, string, DateTimeOffset?, EntityTagHeaderValue, bool) ,
ControllerBase.File(string, string, string, DateTimeOffset?, EntityTagHeaderValue) ,
ControllerBase.File(string, string, string, DateTimeOffset?, EntityTagHeaderValue, bool) ,
ControllerBase.PhysicalFile(string, string) , ControllerBase.PhysicalFile(string, string, bool) ,
ControllerBase.PhysicalFile(string, string, string) ,
ControllerBase.PhysicalFile(string, string, string, bool) ,
ControllerBase.PhysicalFile(string, string, DateTimeOffset?, EntityTagHeaderValue) ,
ControllerBase.PhysicalFile(string, string, DateTimeOffset?, EntityTagHeaderValue, bool) ,
ControllerBase.PhysicalFile(string, string, string, DateTimeOffset?, EntityTagHeaderValue) ,
ControllerBase.PhysicalFile(string, string, string, DateTimeOffset?, EntityTagHeaderValue, bool) ,
ControllerBase.Unauthorized() , ControllerBase.Unauthorized(object) , ControllerBase.NotFound() ,
ControllerBase.NotFound(object) , ControllerBase.BadRequest() ,
ControllerBase.BadRequest(object) , ControllerBase.BadRequest(ModelStateDictionary) ,
ControllerBase.UnprocessableEntity() , ControllerBase.UnprocessableEntity(object) ,
ControllerBase.UnprocessableEntity(ModelStateDictionary) , ControllerBase.Conflict() ,
ControllerBase.Conflict(object) , ControllerBase.Conflict(ModelStateDictionary) ,
ControllerBase.Problem(string, string, int?, string, string) ,
ControllerBase.Problem(string, string, int?, string, string, IDictionary<string, object>) ,
ControllerBase.ValidationProblem(ValidationProblemDetails) ,
ControllerBase.ValidationProblem(ModelStateDictionary) , ControllerBase.ValidationProblem() ,
ControllerBase.ValidationProblem(string, string, int?, string, string, ModelStateDictionary) ,
ControllerBase.ValidationProblem(string, string, int?, string, string, ModelStateDictionary,
IDictionary<string, object>) ,
ControllerBase.Created() , ControllerBase.Created(string, object) ,
ControllerBase.Created(Uri, object) , ControllerBase.CreatedAtAction(string, object) ,
ControllerBase.CreatedAtAction(string, object, object) ,
ControllerBase.CreatedAtAction(string, string, object, object) ,
ControllerBase.CreatedAtRoute(string, object) , ControllerBase.CreatedAtRoute(object, object) ,
ControllerBase.CreatedAtRoute(string, object, object) , ControllerBase.Accepted() ,
ControllerBase.Accepted(object) , ControllerBase.Accepted(Uri) , ControllerBase.Accepted(string) ,

ControllerBase.Accepted(string, object) , ControllerBase.Accepted(Uri, object) , ControllerBase.AcceptedAtAction(string) , ControllerBase.AcceptedAtAction(string, string) , ControllerBase.AcceptedAtAction(string, object) , ControllerBase.AcceptedAtAction(string, string, object) , ControllerBase.AcceptedAtAction(string, object, object) , ControllerBase.AcceptedAtAction(string, string, object, object) , ControllerBase.AcceptedAtRoute(object) , ControllerBase.AcceptedAtRoute(string) , ControllerBase.AcceptedAtRoute(string, object) , ControllerBase.AcceptedAtRoute(object, object) , ControllerBase.AcceptedAtRoute(string, object, object) , ControllerBase.Challenge() , ControllerBase.Challenge(params string[]) , ControllerBase.Challenge(AuthenticationProperties) , ControllerBase.Challenge(AuthenticationProperties, params string[]) , ControllerBase.Forbid() , ControllerBase.Forbid(params string[]) , ControllerBase.Forbid(AuthenticationProperties) , ControllerBase.Forbid(AuthenticationProperties, params string[]) , ControllerBase.SignIn(ClaimsPrincipal) , ControllerBase.SignIn(ClaimsPrincipal, string) , ControllerBase.SignIn(ClaimsPrincipal, AuthenticationProperties) , ControllerBase.SignIn(ClaimsPrincipal, AuthenticationProperties, string) , ControllerBase.SignOut() , ControllerBase.SignOut(AuthenticationProperties) , ControllerBase.SignOut(params string[]) , ControllerBase.SignOut(AuthenticationProperties, params string[]) , ControllerBase.TryUpdateModelAsync<TModel>(TModel) , ControllerBase.TryUpdateModelAsync<TModel>(TModel, string) , ControllerBase.TryUpdateModelAsync<TModel>(TModel, string, IValueProvider) , ControllerBase.TryUpdateModelAsync<TModel>(TModel, string, params Expression<Func<TModel, object>>[]) , ControllerBase.TryUpdateModelAsync<TModel>(TModel, string, Func<ModelMetadata, bool>) , ControllerBase.TryUpdateModelAsync<TModel>(TModel, string, IValueProvider, params Expression<Func<TModel, object>>[]) , ControllerBase.TryUpdateModelAsync<TModel>(TModel, string, IValueProvider, Func<ModelMetadata, bool>) , ControllerBase.TryUpdateModelAsync(object, Type, string) , ControllerBase.TryUpdateModelAsync(object, Type, string, IValueProvider, Func<ModelMetadata, bool>) , ControllerBase.TryValidateModel(object) , ControllerBase.TryValidateModel(object, string) , ControllerBase.HttpContext , ControllerBase.Request , ControllerBase.Response , ControllerBase.RouteData , ControllerBase.ModelState , ControllerBase.ControllerContext , ControllerBase.MetadataProvider , ControllerBase.ModelBinderFactory , ControllerBase.Url , ControllerBase.ObjectValidator , ControllerBase.ProblemDetailsFactory , ControllerBase.User , ControllerBase.Empty , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

ReadOnlyController(TRequestHandler)

Abstract base class for read-only (GETs) operations that utilize a request handler for managing read operations.

```
public ReadOnlyController(TRequestHandler requestHandler)
```

Parameters

`requestHandler` TRequestHandler

Methods

GetByIdAsync(TIdType, CancellationToken)

```
[HttpGet("{id}")]
public Task<ActionResult<ResponseObj<TViewModel>>> GetByIdAsync(TIdType id,
CancellationToken cancellationToken = default)
```

Parameters

`id` TIdType

`cancellationToken` CancellationToken

Returns

Task<ActionResult<ResponseObj<TViewModel>>>

Namespace Quickie.Apis.Writeonly

Classes

WriteOnlyController<TViewModel, TRequestHandler>

Class WriteOnlyController<TViewModel, TRequestHandler>

Namespace: Quickie.Apis.Writeonly

Assembly: Quickie.dll

```
public class WriteOnlyController<TViewModel, TRequestHandler> : QuickieBaseApi where
TViewModel : WriteOnlyDto where TRequestHandler : IWriteOnlyRequestHandler<TViewModel>
```

Type Parameters

`TViewModel`

`TRequestHandler`

Inheritance

object ← ControllerBase ← QuickieBaseApi ← WriteOnlyController<TViewModel, TRequestHandler>

Inherited Members

ControllerBase.StatusCode(int) , ControllerBase.StatusCode(int, object) ,
ControllerBase.Content(string) , ControllerBase.Content(string, string) ,
ControllerBase.Content(string, string, Encoding) ,
ControllerBase.Content(string, MediaTypeHeaderValue) , ControllerBase.NoContent() ,
ControllerBase.Ok() , ControllerBase.Ok(object) , ControllerBase.Redirect(string) ,
ControllerBase.RedirectPermanent(string) , ControllerBase.RedirectPreserveMethod(string) ,
ControllerBase.RedirectPermanentPreserveMethod(string) , ControllerBase.LocalRedirect(string) ,
ControllerBase.LocalRedirectPermanent(string) , ControllerBase.LocalRedirectPreserveMethod(string) ,
ControllerBase.LocalRedirectPermanentPreserveMethod(string) , ControllerBase.RedirectToAction() ,
ControllerBase.RedirectToAction(string) , ControllerBase.RedirectToAction(string, object) ,
ControllerBase.RedirectToAction(string, string) ,
ControllerBase.RedirectToAction(string, string, object) ,
ControllerBase.RedirectToAction(string, string, string) ,
ControllerBase.RedirectToAction(string, string, object, string) ,
ControllerBase.RedirectToActionPreserveMethod(string, string, object, string) ,
ControllerBase.RedirectToActionPermanent(string) ,
ControllerBase.RedirectToActionPermanent(string, object) ,
ControllerBase.RedirectToActionPermanent(string, string) ,
ControllerBase.RedirectToActionPermanent(string, string, string) ,
ControllerBase.RedirectToActionPermanent(string, string, object) ,

ControllerBase.RedirectToActionPermanent(string, string, object, string) ,
ControllerBase.RedirectToActionPermanentPreserveMethod(string, string, object, string) ,
ControllerBase.RedirectToRoute(string) , ControllerBase.RedirectToRoute(object) ,
ControllerBase.RedirectToRoute(string, object) , ControllerBase.RedirectToRoute(string, string) ,
ControllerBase.RedirectToRoute(string, object, string) ,
ControllerBase.RedirectToRoutePreserveMethod(string, object, string) ,
ControllerBase.RedirectToRoutePermanent(string) ,
ControllerBase.RedirectToRoutePermanent(object) ,
ControllerBase.RedirectToRoutePermanent(string, object) ,
ControllerBase.RedirectToRoutePermanent(string, string) ,
ControllerBase.RedirectToRoutePermanent(string, object, string) ,
ControllerBase.RedirectToRoutePermanentPreserveMethod(string, object, string) ,
ControllerBase.RedirectToPage(string) , ControllerBase.RedirectToPage(string, object) ,
ControllerBase.RedirectToPage(string, string) , ControllerBase.RedirectToPage(string, string, object) ,
ControllerBase.RedirectToPage(string, string, string) ,
ControllerBase.RedirectToPage(string, string, object, string) ,
ControllerBase.RedirectToPagePermanent(string) ,
ControllerBase.RedirectToPagePermanent(string, object) ,
ControllerBase.RedirectToPagePermanent(string, string) ,
ControllerBase.RedirectToPagePermanent(string, string, string) ,
ControllerBase.RedirectToPagePermanent(string, string, object, string) ,
ControllerBase.RedirectToPagePreserveMethod(string, string, object, string) ,
ControllerBase.RedirectToPagePermanentPreserveMethod(string, string, object, string) ,
ControllerBase.File(byte[], string) , ControllerBase.File(byte[], string, bool) ,
ControllerBase.File(byte[], string, string) , ControllerBase.File(byte[], string, string, bool) ,
ControllerBase.File(byte[], string, DateTimeOffset?, EntityTagHeaderValue) ,
ControllerBase.File(byte[], string, DateTimeOffset?, EntityTagHeaderValue, bool) ,
ControllerBase.File(byte[], string, string, DateTimeOffset?, EntityTagHeaderValue) ,
ControllerBase.File(byte[], string, string, DateTimeOffset?, EntityTagHeaderValue, bool) ,
ControllerBase.File(Stream, string) , ControllerBase.File(Stream, string, bool) ,
ControllerBase.File(Stream, string, string) , ControllerBase.File(Stream, string, string, bool) ,
ControllerBase.File(Stream, string, DateTimeOffset?, EntityTagHeaderValue) ,
ControllerBase.File(Stream, string, DateTimeOffset?, EntityTagHeaderValue, bool) ,
ControllerBase.File(Stream, string, string, DateTimeOffset?, EntityTagHeaderValue) ,
ControllerBase.File(Stream, string, string, DateTimeOffset?, EntityTagHeaderValue, bool) ,
ControllerBase.File(string, string) , ControllerBase.File(string, string, bool) ,
ControllerBase.File(string, string, string) , ControllerBase.File(string, string, string, bool) ,
ControllerBase.File(string, string, DateTimeOffset?, EntityTagHeaderValue) ,
ControllerBase.File(string, string, DateTimeOffset?, EntityTagHeaderValue, bool) ,
ControllerBase.File(string, string, string, DateTimeOffset?, EntityTagHeaderValue) ,

ControllerBase.File(string, string, string, DateTimeOffset?, EntityTagHeaderValue, bool) ,
ControllerBase.PhysicalFile(string, string) , ControllerBase.PhysicalFile(string, string, bool) ,
ControllerBase.PhysicalFile(string, string, string) ,
ControllerBase.PhysicalFile(string, string, string, bool) ,
ControllerBase.PhysicalFile(string, string, DateTimeOffset?, EntityTagHeaderValue) ,
ControllerBase.PhysicalFile(string, string, DateTimeOffset?, EntityTagHeaderValue, bool) ,
ControllerBase.PhysicalFile(string, string, string, DateTimeOffset?, EntityTagHeaderValue) ,
ControllerBase.PhysicalFile(string, string, string, DateTimeOffset?, EntityTagHeaderValue, bool) ,
ControllerBase.Unauthorized() , ControllerBase.Unauthorized(object) , ControllerBase.NotFound() ,
ControllerBase.NotFound(object) , ControllerBase.BadRequest() ,
ControllerBase.BadRequest(object) , ControllerBase.BadRequest(ModelStateDictionary) ,
ControllerBase.UnprocessableEntity() , ControllerBase.UnprocessableEntity(object) ,
ControllerBase.UnprocessableEntity(ModelStateDictionary) , ControllerBase.Conflict() ,
ControllerBase.Conflict(object) , ControllerBase.Conflict(ModelStateDictionary) ,
ControllerBase.Problem(string, string, int?, string, string) ,
ControllerBase.Problem(string, string, int?, string, string, IDictionary<string, object>) ,
ControllerBase.ValidationProblem(ValidationProblemDetails) ,
ControllerBase.ValidationProblem(ModelStateDictionary) , ControllerBase.ValidationProblem() ,
ControllerBase.ValidationProblem(string, string, int?, string, string, ModelStateDictionary) ,
ControllerBase.ValidationProblem(string, string, int?, string, string, ModelStateDictionary,
IDictionary<string, object>) ,
ControllerBase.Created() , ControllerBase.Created(string, object) ,
ControllerBase.Created(Uri, object) , ControllerBase.CreatedAtAction(string, object) ,
ControllerBase.CreatedAtAction(string, object, object) ,
ControllerBase.CreatedAtAction(string, string, object, object) ,
ControllerBase.CreatedAtRoute(string, object) , ControllerBase.CreatedAtRoute(object, object) ,
ControllerBase.CreatedAtRoute(string, object, object) , ControllerBase.Accepted() ,
ControllerBase.Accepted(object) , ControllerBase.Accepted(Uri) , ControllerBase.Accepted(string) ,
ControllerBase.Accepted(string, object) , ControllerBase.Accepted(Uri, object) ,
ControllerBase.AcceptedAtAction(string) , ControllerBase.AcceptedAtAction(string, string) ,
ControllerBase.AcceptedAtAction(string, object) ,
ControllerBase.AcceptedAtAction(string, string, object) ,
ControllerBase.AcceptedAtAction(string, object, object) ,
ControllerBase.AcceptedAtAction(string, string, object, object) ,
ControllerBase.AcceptedAtRoute(object) , ControllerBase.AcceptedAtRoute(string) ,
ControllerBase.AcceptedAtRoute(string, object) , ControllerBase.AcceptedAtRoute(object, object) ,
ControllerBase.AcceptedAtRoute(string, object, object) , ControllerBase.Challenge() ,
ControllerBase.Challenge(params string[]) , ControllerBase.Challenge(AuthenticationProperties) ,
ControllerBase.Challenge(AuthenticationProperties, params string[]) , ControllerBase.Forbid() ,
ControllerBase.Forbid(params string[]) , ControllerBase.Forbid(AuthenticationProperties) ,

ControllerBase.Forbid(AuthenticationProperties, params string[]) , ControllerBase.SignIn(ClaimsPrincipal) , ControllerBase.SignIn(ClaimsPrincipal, string) , ControllerBase.SignIn(ClaimsPrincipal, AuthenticationProperties) , ControllerBase.SignIn(ClaimsPrincipal, AuthenticationProperties, string) , ControllerBase.SignOut() , ControllerBase.SignOut(AuthenticationProperties) , ControllerBase.SignOut(params string[]) , ControllerBase.SignOut(AuthenticationProperties, params string[]) , ControllerBase.TryUpdateModelAsync<TModel>(TModel) , ControllerBase.TryUpdateModelAsync<TModel>(TModel, string) , ControllerBase.TryUpdateModelAsync<TModel>(TModel, string, IValueProvider) , ControllerBase.TryUpdateModelAsync<TModel>(TModel, string, params Expression<Func<TModel, object>>[]) , ControllerBase.TryUpdateModelAsync<TModel>(TModel, string, Func<ModelMetadata, bool>) , ControllerBase.TryUpdateModelAsync<TModel>(TModel, string, IValueProvider, params Expression<Func<TModel, object>>[]) , ControllerBase.TryUpdateModelAsync<TModel>(TModel, string, IValueProvider, Func<ModelMetadata, bool>) , ControllerBase.TryUpdateModelAsync(object, Type, string) , ControllerBase.TryUpdateModelAsync(object, Type, string, IValueProvider, Func<ModelMetadata, bool>) , ControllerBase.TryValidateModel(object) , ControllerBase.TryValidateModel(object, string) , ControllerBase.HttpContext , ControllerBase.Request , ControllerBase.Response , ControllerBase.RouteData , ControllerBase.ModelState , ControllerBase.ControllerContext , ControllerBase.MetadataProvider , ControllerBase.ModelBinderFactory , ControllerBase.Url , ControllerBase.ObjectValidator , ControllerBase.ProblemDetailsFactory , ControllerBase.User , ControllerBase.Empty , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

WriteOnlyController(IWriteOnlyRequestHandler<TViewModel>)

```
public WriteOnlyController(IWriteOnlyRequestHandler<TViewModel> requestHandler)
```

Parameters

`requestHandler` IWriteOnlyRequestHandler<TViewModel>

Methods

CreateItemAsync(string?, TViewModel, CancellationToken)

Create a single item.

```
[HttpPost]
public Task<ActionResult<ResponseObj<TViewModel>>> CreateItemAsync(string? idempotencyKey,
TViewModel request, CancellationToken cancellationToken = default)
```

Parameters

`idempotencyKey` string

Idempotency key.

`request` TViewModel

Data to create

`cancellationToken` CancellationToken

Cancellation token

Returns

Task<ActionResult<ResponseObj<TViewModel>>>

Response indicating success or failure of the operation.

CreateItemsAsync(string?, ICollection<TViewModel>, CancellationToken)

Create multiple items in bulk.

```
[HttpPost("items")]
public Task<ActionResult<ResponseObj<ICollection<TViewModel>>>> CreateItemsAsync(string?
idempotencyKey, ICollection<TViewModel> requests, CancellationToken cancellationToken
= default)
```

Parameters

`idempotencyKey` string

idempotency key

`requests` ICollection<TViewModel>

Requested data to create

`cancellationToken` CancellationToken

Cancellation token

Returns

Task<ActionResult<ResponseObj<ICollection<TViewModel>>>>

Response indicating success or failure of the operation.

Namespace Quickie.Base

Classes

QuickieBaseApi

Ultimate base of quickie controllers (APIs)

Class QuickieBaseApi

Namespace: Quickie.Base

Assembly: Quickie.dll

Ultimate base of quickie controllers (APIs)

```
[EnableRateLimiting("Quickie-Rl-Policy")]
[ApiController]
[Route("api/[controller]")]
[Produces("application/json", new string[] { })]
public abstract class QuickieBaseApi : ControllerBase
```

Inheritance

object ← ControllerBase ← QuickieBaseApi

Derived

CrudController<TViewModel, TRequestHandler, TIdType>, CrudForCollectionController<TViewModel, TRequestHandler, TIdType>, EditOnlyController<TViewModel, TRequestHandler, TIdType>, ReadOnlyCollectionController<TViewModel, TRequestHandler, TRequestForDataModel, TIdType>, ReadOnlyController<TViewModel, TRequestHandler, TIdType>, WriteOnlyController<TViewModel, TRequestHandler>

Inherited Members

ControllerBase.StatusCode(int) , ControllerBase.StatusCode(int, object) , ControllerBase.Content(string) , ControllerBase.Content(string, string) , ControllerBase.Content(string, string, Encoding) , ControllerBase.Content(string, MediaTypeHeaderValue) , ControllerBase.NoContent() , ControllerBase.Ok() , ControllerBase.Ok(object) , ControllerBase.Redirect(string) , ControllerBase.RedirectPermanent(string) , ControllerBase.RedirectPreserveMethod(string) , ControllerBase.RedirectPermanentPreserveMethod(string) , ControllerBase.LocalRedirect(string) , ControllerBase.LocalRedirectPermanent(string) , ControllerBase.LocalRedirectPreserveMethod(string) , ControllerBase.LocalRedirectPermanentPreserveMethod(string) , ControllerBase.RedirectToAction() , ControllerBase.RedirectToAction(string) , ControllerBase.RedirectToAction(string, object) , ControllerBase.RedirectToAction(string, string) , ControllerBase.RedirectToAction(string, string, object) , ControllerBase.RedirectToAction(string, string, string) , ControllerBase.RedirectToAction(string, string, object, string) , ControllerBase.RedirectToActionPreserveMethod(string, string, object, string) ,

ControllerBase.RedirectToActionPermanent(string) ,
ControllerBase.RedirectToActionPermanent(string, object) ,
ControllerBase.RedirectToActionPermanent(string, string) ,
ControllerBase.RedirectToActionPermanent(string, string, string) ,
ControllerBase.RedirectToActionPermanent(string, string, object) ,
ControllerBase.RedirectToActionPermanent(string, string, object, string) ,
ControllerBase.RedirectToActionPermanentPreserveMethod(string, string, object, string) ,
ControllerBase.RedirectToRoute(string) , ControllerBase.RedirectToRoute(object) ,
ControllerBase.RedirectToRoute(string, object) , ControllerBase.RedirectToRoute(string, string) ,
ControllerBase.RedirectToRoute(string, object, string) ,
ControllerBase.RedirectToRoutePreserveMethod(string, object, string) ,
ControllerBase.RedirectToRoutePermanent(string) ,
ControllerBase.RedirectToRoutePermanent(object) ,
ControllerBase.RedirectToRoutePermanent(string, object) ,
ControllerBase.RedirectToRoutePermanent(string, string) ,
ControllerBase.RedirectToRoutePermanent(string, object, string) ,
ControllerBase.RedirectToRoutePermanentPreserveMethod(string, object, string) ,
ControllerBase.RedirectToPage(string) , ControllerBase.RedirectToPage(string, object) ,
ControllerBase.RedirectToPage(string, string) , ControllerBase.RedirectToPage(string, string, object) ,
ControllerBase.RedirectToPage(string, string, string) ,
ControllerBase.RedirectToPage(string, string, object, string) ,
ControllerBase.RedirectToPagePermanent(string) ,
ControllerBase.RedirectToPagePermanent(string, object) ,
ControllerBase.RedirectToPagePermanent(string, string) ,
ControllerBase.RedirectToPagePermanent(string, string, string) ,
ControllerBase.RedirectToPagePermanent(string, string, object, string) ,
ControllerBase.RedirectToPagePreserveMethod(string, string, object, string) ,
ControllerBase.RedirectToPagePermanentPreserveMethod(string, string, object, string) ,
ControllerBase.File(byte[], string) , ControllerBase.File(byte[], string, bool) ,
ControllerBase.File(byte[], string, string) , ControllerBase.File(byte[], string, string, bool) ,
ControllerBase.File(byte[], string, DateTimeOffset?, EntityTagHeaderValue) ,
ControllerBase.File(byte[], string, DateTimeOffset?, EntityTagHeaderValue, bool) ,
ControllerBase.File(byte[], string, string, DateTimeOffset?, EntityTagHeaderValue) ,
ControllerBase.File(byte[], string, string, DateTimeOffset?, EntityTagHeaderValue, bool) ,
ControllerBase.File(Stream, string) , ControllerBase.File(Stream, string, bool) ,
ControllerBase.File(Stream, string, string) , ControllerBase.File(Stream, string, string, bool) ,
ControllerBase.File(Stream, string, DateTimeOffset?, EntityTagHeaderValue) ,
ControllerBase.File(Stream, string, DateTimeOffset?, EntityTagHeaderValue, bool) ,
ControllerBase.File(Stream, string, string, DateTimeOffset?, EntityTagHeaderValue) ,
ControllerBase.File(Stream, string, string, DateTimeOffset?, EntityTagHeaderValue, bool) ,

ControllerBase.File(string, string) , ControllerBase.File(string, string, bool) ,
ControllerBase.File(string, string, string) , ControllerBase.File(string, string, string, bool) ,
ControllerBase.File(string, string, DateTimeOffset?, EntityTagHeaderValue) ,
ControllerBase.File(string, string, DateTimeOffset?, EntityTagHeaderValue, bool) ,
ControllerBase.File(string, string, string, DateTimeOffset?, EntityTagHeaderValue) ,
ControllerBase.File(string, string, string, DateTimeOffset?, EntityTagHeaderValue, bool) ,
ControllerBase.PhysicalFile(string, string) , ControllerBase.PhysicalFile(string, string, bool) ,
ControllerBase.PhysicalFile(string, string, string) ,
ControllerBase.PhysicalFile(string, string, string, bool) ,
ControllerBase.PhysicalFile(string, string, DateTimeOffset?, EntityTagHeaderValue) ,
ControllerBase.PhysicalFile(string, string, DateTimeOffset?, EntityTagHeaderValue, bool) ,
ControllerBase.PhysicalFile(string, string, string, DateTimeOffset?, EntityTagHeaderValue) ,
ControllerBase.PhysicalFile(string, string, string, DateTimeOffset?, EntityTagHeaderValue, bool) ,
ControllerBase.Unauthorized() , ControllerBase.Unauthorized(object) , ControllerBase.NotFound() ,
ControllerBase.NotFound(object) , ControllerBase.BadRequest() ,
ControllerBase.BadRequest(object) , ControllerBase.BadRequest(ModelStateDictionary) ,
ControllerBase.UnprocessableEntity() , ControllerBase.UnprocessableEntity(object) ,
ControllerBase.UnprocessableEntity(ModelStateDictionary) , ControllerBase.Conflict() ,
ControllerBase.Conflict(object) , ControllerBase.Conflict(ModelStateDictionary) ,
ControllerBase.Problem(string, string, int?, string, string) ,
ControllerBase.Problem(string, string, int?, string, string, IDictionary<string, object>) ,
ControllerBase.ValidationProblem(ValidationProblemDetails) ,
ControllerBase.ValidationProblem(ModelStateDictionary) , ControllerBase.ValidationProblem() ,
ControllerBase.ValidationProblem(string, string, int?, string, string, ModelStateDictionary) ,
ControllerBase.ValidationProblem(string, string, int?, string, string, ModelStateDictionary,
IDictionary<string, object>) ,
ControllerBase.Created() , ControllerBase.Created(string, object) ,
ControllerBase.Created(Uri, object) , ControllerBase.CreatedAtAction(string, object) ,
ControllerBase.CreatedAtAction(string, object, object) ,
ControllerBase.CreatedAtAction(string, string, object, object) ,
ControllerBase.CreatedAtRoute(string, object) , ControllerBase.CreatedAtRoute(object, object) ,
ControllerBase.CreatedAtRoute(string, object, object) , ControllerBase.Accepted() ,
ControllerBase.Accepted(object) , ControllerBase.Accepted(Uri) , ControllerBase.Accepted(string) ,
ControllerBase.Accepted(string, object) , ControllerBase.Accepted(Uri, object) ,
ControllerBase.AcceptedAtAction(string) , ControllerBase.AcceptedAtAction(string, string) ,
ControllerBase.AcceptedAtAction(string, object) ,
ControllerBase.AcceptedAtAction(string, string, object) ,
ControllerBase.AcceptedAtAction(string, object, object) ,
ControllerBase.AcceptedAtAction(string, string, object, object) ,
ControllerBase.AcceptedAtRoute(object) , ControllerBase.AcceptedAtRoute(string) ,

ControllerBase.AcceptedAtRoute(string, object) , ControllerBase.AcceptedAtRoute(object, object) , ControllerBase.AcceptedAtRoute(string, object, object) , ControllerBase.Challenge() , ControllerBase.Challenge(params string[]) , ControllerBase.Challenge(AuthenticationProperties) , ControllerBase.Challenge(AuthenticationProperties, params string[]) , ControllerBase.Forbid() , ControllerBase.Forbid(params string[]) , ControllerBase.Forbid(AuthenticationProperties) , ControllerBase.Forbid(AuthenticationProperties, params string[]) , ControllerBase.SignIn(ClaimsPrincipal) , ControllerBase.SignIn(ClaimsPrincipal, string) , ControllerBase.SignIn(ClaimsPrincipal, AuthenticationProperties) , ControllerBase.SignIn(ClaimsPrincipal, AuthenticationProperties, string) , ControllerBase.SignOut() , ControllerBase.SignOut(AuthenticationProperties) , ControllerBase.SignOut(params string[]) , ControllerBase.SignOut(AuthenticationProperties, params string[]) , ControllerBase.TryUpdateModelAsync<TModel>(TModel) , ControllerBase.TryUpdateModelAsync<TModel>(TModel, string) , ControllerBase.TryUpdateModelAsync<TModel>(TModel, string, IValueProvider) , ControllerBase.TryUpdateModelAsync<TModel>(TModel, string, params Expression<Func<TModel, object>>[]) , ControllerBase.TryUpdateModelAsync<TModel>(TModel, string, Func<ModelMetadata, bool>) , ControllerBase.TryUpdateModelAsync<TModel>(TModel, string, IValueProvider, params Expression<Func<TModel, object>>[]) , ControllerBase.TryUpdateModelAsync<TModel>(TModel, string, IValueProvider, Func<ModelMetadata, bool>) , ControllerBase.TryUpdateModelAsync(object, Type, string) , ControllerBase.TryUpdateModelAsync(object, Type, string, IValueProvider, Func<ModelMetadata, bool>) , ControllerBase.TryValidateModel(object) , ControllerBase.TryValidateModel(object, string) , ControllerBase.HttpContext , ControllerBase.Request , ControllerBase.Response , ControllerBase.RouteData , ControllerBase.ModelState , ControllerBase.ControllerContext , ControllerBase.MetadataProvider , ControllerBase.ModelBinderFactory , ControllerBase.Url , ControllerBase.ObjectValidator , ControllerBase.ProblemDetailsFactory , ControllerBase.User , ControllerBase.Empty , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Namespace Quickie.Configuration

Classes

DependencyResolver

Dependency resolver for data handlers.

QuickieExtension

Quickie extensions

QuickieOptions

Quickie options.

Class DependencyResolver

Namespace: Quickie.Configuration

Assembly: Quickie.dll

Dependency resolver for data handlers.

```
public static class DependencyResolver
```

Inheritance

object ← DependencyResolver

Inherited Members

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

Methods

AddDataHandlersDependency(IServiceCollection)

Add data handlers dependency.

```
public static IServiceCollection AddDataHandlersDependency(this IServiceCollection services)
```

Parameters

`services` IServiceCollection

Service collection

Returns

IServiceCollection

Class QuickieExtension

Namespace: Quickie.Configuration

Assembly: Quickie.dll

Quickie extensions

```
public static class QuickieExtension
```

Inheritance

object ← QuickieExtension

Inherited Members

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

Methods

AddQuickie(IApplicationBuilder)

Configures the application to use Quickie services and middleware based on provided options.

```
public static IApplicationBuilder AddQuickie(this IApplicationBuilder app)
```

Parameters

`app` IApplicationBuilder

Application request pipeline builder.

Returns

IApplicationBuilder

IApplicationBuilder

QuickieConfig(IServiceCollection, Action<QuickieOptions>?)

Registers Quickie services in the DI container, including all custom services.

```
public static IServiceCollection QuickieConfig(this IServiceCollection services,
Action<QuickieOptions>? configureOptions = null)
```

Parameters

`services` IServiceCollection

DI register and resolver.

`configureOptions` Action<QuickieOptions>

Delegate to configure quickie option.

Returns

IServiceCollection

IServiceCollection

Class QuickieOptions

Namespace: Quickie.Configuration

Assembly: Quickie.dll

Quickie options.

```
public class QuickieOptions
```

Inheritance

object ← QuickieOptions

Inherited Members

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

Properties

IdempotencyConfiguration

Idempotency configuration

```
public IdempotentConfiguration? IdempotencyConfiguration { get; set; }
```

Property Value

IdempotentConfiguration

RateLimitingConfiguration

Rate limit configuration

```
public RateLimitConfiguration? RateLimitingConfiguration { get; set; }
```

Property Value

ShowCustomErrorMessage

Instead of exception message, `ResponseObj` will be returning custom message.

```
public bool ShowCustomErrorMessage { get; set; }
```

Property Value

bool

Namespace Quickie.Configuration.Idempotency

Classes

IdempotencyBackgroundService

Background service to remove expired idempotency key.

IdempotentConfiguration

Interfaces

IIdempotencyProvider

Idempotency provider contract

Interface IIdempotencyProvider

Namespace: Quickie.Configuration.Idempotency

Assembly: Quickie.dll

Idempotency provider contract

```
public interface IIdempotencyProvider
```

Methods

ExistsAsync(string)

Check if the `key` is already used.

```
ValueTask<bool> ExistsAsync(string key)
```

Parameters

`key` string

Unique key

Returns

ValueTask<bool>

whether the key is used or not.

MarkAsync(string)

Add and mark `key` as used.

```
ValueTask MarkAsync(string key)
```

Parameters

`key` string

Unique key

Returns

ValueTask

a Task

RemoveExpiredKeys()

Remove idempotency keys.

```
ValueTask RemoveExpiredKeys()
```

Returns

ValueTask

Class IdempotencyBackgroundService

Namespace: Quickie.Configuration.Idempotency

Assembly: Quickie.dll

Background service to remove expired idempotency key.

```
public class IdempotencyBackgroundService : BackgroundService, IHostedService, IDisposable
```

Inheritance

object ← BackgroundService ← IdempotencyBackgroundService

Implements

IHostedService, IDisposable

Inherited Members

BackgroundService.Dispose() , BackgroundService.StartAsync(CancellationToken) , BackgroundService.StopAsync(CancellationToken) , BackgroundService.ExecuteTask , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

IdempotencyBackgroundService(IIdempotencyProvider)

Background service to remove expired idempotency key.

```
public IdempotencyBackgroundService(IIdempotencyProvider idempotencyProvider)
```

Parameters

`idempotencyProvider` IIdempotencyProvider

Methods

ExecuteAsync(CancellationToken)

This method is called when the IHostedService starts. The implementation should return a task that represents the lifetime of the long running operation(s) being performed.

```
protected override Task ExecuteAsync(CancellationToken stoppingToken)
```

Parameters

`stoppingToken` CancellationToken

Triggered when StopAsync(CancellationToken) is called.

Returns

Task

A Task that represents the long running operations.

Remarks

See Worker Services in .NET for implementation guidelines.

Class IdempotentConfiguration

Namespace: Quickie.Configuration.Idempotency

Assembly: Quickie.dll

```
public class IdempotentConfiguration
```

Inheritance

object ← IdempotentConfiguration

Inherited Members

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

Properties

Enable

Enable idempotency. By default, idempotency is `false`

```
public bool Enable { get; set; }
```

Property Value

bool

IdempotencyLifespan

Lifespan of keys. By default, its 1 hour

```
public TimeSpan IdempotencyLifespan { get; set; }
```

Property Value

TimeSpan

Provider

Your custom idempotency provider

```
public IIdempotencyProvider? Provider { get; set; }
```

Property Value

IIdempotencyProvider

RunBackgroundServiceEveryHour

Run background service every x hour? By default, 1 hour

```
[Range(0, 24)]
public int RunBackgroundServiceEveryHour { get; set; }
```

Property Value

int

Namespace Quickie.Configuration.Idempotency.DefaultProvider

Classes

InMemoryIdempotencyProvider

Default InMemory idempotency provider

Class InMemoryIdempotencyProvider

Namespace: Quickie.Configuration.Idempotency.DefaultProvider

Assembly: Quickie.dll

Default InMemory idempotency provider

```
public class InMemoryIdempotencyProvider : IIdempotencyProvider
```

Inheritance

object ← InMemoryIdempotencyProvider

Implements

IIdempotencyProvider

Inherited Members

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

Methods

ExistsAsync(string)

Asynchronous method synchronous operation(check in dictionary).

```
public ValueTask<bool> ExistsAsync(string key)
```

Parameters

`key` string

Key

Returns

ValueTask<bool>

bool

MarkAsync(string)

Asynchronous method synchronous operation (add in dictionary).

```
public ValueTask MarkAsync(string key)
```

Parameters

`key` string

Returns

ValueTask

RemoveExpiredKeys()

Remove expired api idempotency keys

```
public ValueTask RemoveExpiredKeys()
```

Returns

ValueTask

Namespace Quickie.Configuration.Middleware

Classes

IdempotencyMiddleware

Idempotency implementer middleware

Class IdempotencyMiddleware

Namespace: Quickie.Configuration.Middleware

Assembly: Quickie.dll

Idempotency implementer middleware

```
public class IdempotencyMiddleware
```

Inheritance

object ← IdempotencyMiddleware

Inherited Members

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

Constructors

IdempotencyMiddleware(IIdempotencyProvider, RequestDelegate)

Idempotency implementer middleware

```
public IdempotencyMiddleware(IIdempotencyProvider idempotencyProvider, RequestDelegate next)
```

Parameters

`idempotencyProvider` IIdempotencyProvider

Idempotency provider

`next` RequestDelegate

Methods

InvokeAsync(HttpContext)

```
public Task InvokeAsync(HttpContext context)
```

Parameters

`context` HttpContext

Returns

Task

Namespace Quickie.Configuration.Options

Classes

GlobalQuickieConfigData

Global configuration for Quickie library. Must be initialized once during application startup before any Quickie features are used.

RateLimitConfiguration

Rate limit configuration. (ip based)

Class GlobalQuickieConfigData

Namespace: Quickie.Configuration.Options

Assembly: Quickie.dll

Global configuration for Quickie library. Must be initialized once during application startup before any Quickie features are used.

```
public static class GlobalQuickieConfigData
```

Inheritance

object ← GlobalQuickieConfigData

Inherited Members

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

Properties

Options

Configured Quickie options.

```
public static QuickieOptions Options { get; }
```

Property Value

QuickieOptions

Exceptions

InvalidOperationException

Methods

Initialize(QuickieOptions)

Initializes Quickie configuration. This method must be called once during application startup.

```
public static void Initialize(QuickieOptions options)
```

Parameters

`options` QuickieOptions

Configuration options

Exceptions

ArgumentNullException

Class RateLimitConfiguration

Namespace: Quickie.Configuration.Options

Assembly: Quickie.dll

Rate limit configuration. (ip based)

```
public class RateLimitConfiguration
```

Inheritance

object ← RateLimitConfiguration

Inherited Members

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

Properties

DisableRateLimiting

Disable rate limiting. By default, rate limiting is `true`.

```
public bool DisableRateLimiting { get; set; }
```

Property Value

bool

FromSeconds

Duration (seconds) to block. Default is: 6

```
public int? FromSeconds { get; set; }
```

Property Value

int?

PermitLimit

Number of request. Default is: 1 request every 'FromSeconds'

```
public int? PermitLimit { get; set; }
```

Property Value

int?

PolicyName

Name of rate limit policy. Default is "Quickie-Rl-Policy"

```
public string? PolicyName { get; set; }
```

Property Value

string

Namespace Quickie.DataHandlers.Crud.Definition

Interfaces

ICrudDataHandler<TEntity, TIdType>

ICrudForCollectionDataHandler<TEntity, TIdType>

Interface ICrudDataHandler<TEntity, TIdType>

Namespace: Quickie.DataHandlers.Crud.Definition

Assembly: Quickie.dll

```
public interface ICrudDataHandler<TEntity, in TIdType> where TEntity : CrudEntity where
TIdType : IComparable
```

Type Parameters

`TEntity`

`TIdType`

Methods

CreateAsync(TEntity, CancellationToken?)

```
Task<ResponseObj<TEntity>> CreateAsync(TEntity request, CancellationToken?
cancellationToken)
```

Parameters

`request` TEntity

`cancellationToken` CancellationToken?

Returns

Task<ResponseObj<TEntity>>

DeleteAsync(TIdType, CancellationToken?)

```
Task<ResponseObj> DeleteAsync(TIdType id, CancellationToken? cancellationToken = null)
```

Parameters

`id` TIdType

`cancellationToken` CancellationToken?

Returns

Task<ResponseObj>

GetByIdAsync(TIdType, CancellationToken?)

```
Task<ResponseObj<TEntity>> GetByIdAsync(TIdType id, CancellationToken? cancellationToken
= null)
```

Parameters

`id` TIdType

`cancellationToken` CancellationToken?

Returns

Task<ResponseObj<TEntity>>

UpdateAsync(TIdType, TEntity, CancellationToken?)

```
Task<ResponseObj<TEntity>> UpdateAsync(TIdType id, TEntity request, CancellationToken?
cancellationToken = null)
```

Parameters

`id` TIdType

`request` TEntity

`cancellationToken` CancellationToken?

Returns

Task <ResponseObj<TEntity>>

Interface ICrudForCollectionDataHandler<TEntity, TIdType>

Namespace: Quickie.DataHandlers.Crud.Definition

Assembly: Quickie.dll

```
public interface ICrudForCollectionDataHandler<TEntity, TIdType> where TEntity : CrudEntity
where TIdType : IComparable
```

Type Parameters

`TEntity`

`TIdType`

Methods

CountAsync(CancellationToken?)

```
Task<ResponseObj<int>> CountAsync(CancellationToken? cancellationToken = null)
```

Parameters

`cancellationToken` CancellationToken?

Returns

Task<ResponseObj<int>>

CreateRangeAsync(ICollection<TEntity>, CancellationToken?)

```
Task<ResponseObj<ICollection<TEntity>>> CreateRangeAsync(ICollection<TEntity> entities,
CancellationToken? cancellationToken = null)
```

Parameters

`entities` ICollection<TEntity>

`cancellationToken` CancellationToken?

Returns

Task<ResponseObj<ICollection<TEntity>>>

DeleteRangeAsync(ICollection<TIdType>, CancellationToken?)

```
Task<ResponseObj> DeleteRangeAsync(ICollection<TIdType> ids, CancellationToken?
cancellationToken = null)
```

Parameters

`ids` ICollection<TIdType>

`cancellationToken` CancellationToken?

Returns

Task<ResponseObj>

GetAllAsync(RequestForDataObj, CancellationToken?)

```
Task<ResponseObj<PaginatedDataObj<TEntity>>> GetAllAsync(RequestForDataObj request,
CancellationToken? cancellationToken = null)
```

Parameters

`request` RequestForDataObj

`cancellationToken` CancellationToken?

Returns

Task<ResponseObj<PaginatedDataObj<TEntity>>>

UpdateRangeAsync(ICollection<TEntity>, CancellationToken?)

```
Task<ResponseObj<ICollection<TEntity>>> UpdateRangeAsync(ICollection<TEntity> entities,
CancellationToken? cancellationToken = null)
```

Parameters

`entities` ICollection<TEntity>

`cancellationToken` CancellationToken?

Returns

Task<ResponseObj<ICollection<TEntity>>>

Namespace Quickie.DataHandlers.Crud.Implementation

Classes

CrudDataHandler<TEntity, TDbContext, TIdType>

Abstract class serving as CRUD data handler (Repository layer).

CrudForCollectionDataDataHandler<TEntity, TIdType, TDbContext>

Abstract class serving as CRUD for collection request handler.

Class CrudDataHandler<TEntity, TDbContext, TIdType>

Namespace: Quickie.DataHandlers.Crud.Implementation

Assembly: Quickie.dll

Abstract class serving as CRUD data handler (Repository layer).

```
public abstract class CrudDataHandler<TEntity, TDbContext, TIdType> :
ICrudDataHandler<TEntity, TIdType> where TEntity : CrudEntity where TDbContext : DbContext
where TIdType : IComparable
```

Type Parameters

`TEntity`

Entity

`TDbContext`

Db context

`TIdType`

Type of identifier. eg: int, string, ...

Inheritance

object ← CrudDataHandler<TEntity, TDbContext, TIdType>

Implements

ICrudDataHandler<TEntity, TIdType>

Inherited Members

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

Constructors

CrudDataHandler(TDbContext)

Abstract class serving as CRUD data handler (Repository layer).

```
protected CrudDataHandler(TDbContext dbContext)
```

Parameters

`dbContext` TDbContext

Db context

Methods

CreateAsync(TEntity, CancellationToken?)

Data create request.

```
public Task<ResponseObj<TEntity>> CreateAsync(TEntity request, CancellationToken?
cancellationToken = null)
```

Parameters

`request` TEntity

Data requested to create.

`cancellationToken` CancellationToken?

Cancellation token

Returns

Task<ResponseObj<TEntity>>

DeleteAsync(TIdType, CancellationToken?)

Delete request.

```
public Task<ResponseObj> DeleteAsync(TIdType id, CancellationToken? cancellationToken
```

```
= null)
```

Parameters

`id` TIdType

Identifier of the data to remove

`cancellationToken` CancellationToken?

Cancellation token

Returns

Task<ResponseObj>

GetByIdAsync(TIdType, CancellationToken?)

Get data by providing an identifier

```
public Task<ResponseObj<TEntity>> GetByIdAsync(TIdType id, CancellationToken?
cancellationToken = null)
```

Parameters

`id` TIdType

Identifier

`cancellationToken` CancellationToken?

Cancellation token

Returns

Task<ResponseObj<TEntity>>

UpdateAsync(TIdType, TEntity, CancellationToken?)

Update request.

```
public Task<ResponseObj<TEntity>> UpdateAsync(TIdType id, TEntity request,
CancellationToken? cancellationToken = null)
```

Parameters

`id` TIdType

Identifier

`request` TEntity

Requested data identifier.

`cancellationToken` CancellationToken?

Cancellation token

Returns

Task<ResponseObj<TEntity>>

Class CrudForCollectionDataDataHandler<TEntity, TIdType, TDbContext>

Namespace: Quickie.DataHandlers.Crud.Implementation

Assembly: Quickie.dll

Abstract class serving as CRUD for collection request handler.

```
public abstract class CrudForCollectionDataDataHandler<TEntity, TIdType, TDbContext> :
ICrudForCollectionDataHandler<TEntity, TIdType> where TEntity : CrudEntity where TIdType :
IComparable where TDbContext : DbContext
```

Type Parameters

`TEntity`

Entity

`TIdType`

Type of identifier. eg: int, string, ...

`TDbContext`

Db context

Inheritance

object ← CrudForCollectionDataDataHandler<TEntity, TIdType, TDbContext>

Implements

ICrudForCollectionDataHandler<TEntity, TIdType>

Inherited Members

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

Constructors

CrudForCollectionDataDataHandler(TDbContext)

Abstract class serving as CRUD for collection request handler.

```
protected CrudForCollectionDataDataHandler(TDbContext dbContext)
```

Parameters

`dbContext` TDbContext

Db context

Methods

CountAsync(CancellationToken?)

Count.

```
public Task<ResponseObj<int>> CountAsync(CancellationToken? cancellationToken = null)
```

Parameters

`cancellationToken` CancellationToken?

Cancellation token

Returns

Task<ResponseObj<int>>

CreateRangeAsync(ICollection<TEntity>, CancellationToken?)

Data create request.

```
public Task<ResponseObj<ICollection<TEntity>>> CreateRangeAsync(ICollection<TEntity>
entities, CancellationToken? cancellationToken = null)
```

Parameters

`entities` ICollection<TEntity>

Collection of data requested to create.

`cancellationToken` CancellationToken?

Cancellation token

Returns

Task<ResponseObj<ICollection<TEntity>>>

DeleteRangeAsync(ICollection<TIdType>, CancellationToken?)

Delete request.

```
public Task<ResponseObj> DeleteRangeAsync(ICollection<TIdType> ids, CancellationToken?
cancellationToken = null)
```

Parameters

`ids` ICollection<TIdType>

Identifiers of the data to remove.

`cancellationToken` CancellationToken?

Cancellation token

Returns

Task<ResponseObj>

GetAllAsync(RequestForDataObj, CancellationToken?)

Get data.

```
public Task<ResponseObj<PaginatedDataObj<TEntity>>> GetAllAsync(RequestForDataObj request,
CancellationToken? cancellationToken = null)
```

Parameters

`request` RequestForDataObj

Data request filter

`cancellationToken` CancellationToken?

Cancellation token

Returns

Task<ResponseObj<PaginatedDataObj<TEntity>>>

UpdateRangeAsync(ICollection<TEntity>, CancellationToken?)

Update request.

```
public Task<ResponseObj<ICollection<TEntity>>> UpdateRangeAsync(ICollection<TEntity>
entities, CancellationToken? cancellationToken = null)
```

Parameters

`entities` ICollection<TEntity>

Requested collection data

`cancellationToken` CancellationToken?

Cancellation token

Returns

Task<ResponseObj<ICollection<TEntity>>>

Namespace Quickie.DataHandlers.Editonly.Definition

Interfaces

IEditOnlyDataHandler<TEntity, TIdType>

Defines operations to update entities

Interface IEditOnlyDataHandler<TEntity, TIdType>

Namespace: Quickie.DataHandlers.Editonly.Definition

Assembly: Quickie.dll

Defines operations to update entities

```
public interface IEditOnlyDataHandler<TEntity, in TIdType> where TEntity : EditOnlyEntity
where TIdType : IComparable
```

Type Parameters

`TEntity`

Data to update.

`TIdType`

Type of id (int, Guid, string, ...)

Methods

EditAsync(TIdType, TEntity, CancellationToken?)

Update single data.

```
Task<ResponseObj<TEntity>> EditAsync(TIdType id, TEntity requestModel,
CancellationToken? cancellationToken)
```

Parameters

`id` TIdType

Identifier.

`requestModel` TEntity

Data to be updated.

`cancellationToken` CancellationToken?

Cancellation token.

Returns

Task<ResponseObj<TEntity>>

EditCollectionAsync(ICollection<TEntity>, CancellationToken?)

Update collection of data (bulk update).

```
Task<ResponseObj<ICollection<TEntity>>> EditCollectionAsync(ICollection<TEntity>
requestModels, CancellationToken? cancellationToken)
```

Parameters

`requestModels` ICollection<TEntity>

Collection of entity.

`cancellationToken` CancellationToken?

Cancellation token.

Returns

Task<ResponseObj<ICollection<TEntity>>>

Namespace Quickie.DataHandlers.Editonly.Implementation

Classes

EditOnlyDataHandler<TEntity, TIdType>

Abstract class serving as edit only data handler (Repository layer).

Class EditOnlyDataHandler<TEntity, TIdType>

Namespace: Quickie.DataHandlers.Editonly.Implementation

Assembly: Quickie.dll

Abstract class serving as edit only data handler (Repository layer).

```
public abstract class EditOnlyDataHandler<TEntity, TIdType> : IEditOnlyDataHandler<TEntity,
TIdType> where TEntity : EditOnlyEntity where TIdType : IComparable
```

Type Parameters

`TEntity`

Entity

`TIdType`

Type of identifier. eg: int, string, ...

Inheritance

object ← EditOnlyDataHandler<TEntity, TIdType>

Implements

IEditOnlyDataHandler<TEntity, TIdType>

Inherited Members

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

Constructors

EditOnlyDataHandler(DbContext)

Abstract class serving as edit only data handler (Repository layer).

```
protected EditOnlyDataHandler(DbContext dbContext)
```

Parameters

`dbContext` DbContext

Db context

Methods

EditAsync(TIdType, TEntity, CancellationToken?)

Update request.

```
public Task<ResponseObj<TEntity>> EditAsync(TIdType id, TEntity requestModel,
CancellationToken? cancellationToken = null)
```

Parameters

`id` TIdType

Identifier of the entity

`requestModel` TEntity

Requested data to update

`cancellationToken` CancellationToken?

Cancellation token

Returns

Task<ResponseObj<TEntity>>

EditCollectionAsync(ICollection<TEntity>, CancellationToken?)

Update collection of data.

```
public Task<ResponseObj<ICollection<TEntity>>> EditCollectionAsync(ICollection<TEntity>
requestModels, CancellationToken? cancellationToken = null)
```

Parameters

`requestModels` ICollection<TEntity>

Collection of requested data to update

`cancellationToken` CancellationToken?

Cancellation token

Returns

Task<ResponseObj<ICollection<TEntity>>>

Namespace Quickie.DataHandlers.Readonly.Definition

Interfaces

IReadOnlyCollectionDataHandler<TEntity, TIdType>

Defines read-only data access operations for collections of entities. Provides methods to retrieve data with optional filtering and pagination.

IReadOnlyDataHandler<TEntity, TIdType>

Read-only data handler (Repository).

Interface IReadOnlyCollectionDataHandler<TEntity, TIdType>

Namespace: Quickie.DataHandlers.Readonly.Definition

Assembly: Quickie.dll

Defines read-only data access operations for collections of entities. Provides methods to retrieve data with optional filtering and pagination.

```
public interface IReadOnlyCollectionDataHandler<TEntity, in TIdType> where TEntity :
ReadOnlyEntity where TIdType : IComparable
```

Type Parameters

`TEntity`

The type of entity to do read-only operations.

`TIdType`

Type of id (int, Guid, string, ...)

Methods

GetByFilterAsync<TRequestModel>(Expression<Func<TEntity, bool>>?, TRequestModel?, bool?, CancellationToken?)

Get a paginated collection of entities that match the specified filter.

```
Task<PaginatedDataObj<TEntity>> GetByFilterAsync<TRequestModel>(Expression<Func<TEntity,
bool>>? filter, TRequestModel? request, bool? track = false, CancellationToken?
cancellationToken = null) where TRequestModel : RequestForDataObj
```

Parameters

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

A predicate function to define your condition.

`request` TRequestModel

The pagination request object.

`track` bool?

A flag indicating whether to track changes for the retrieved entities. Set to `false` to disable change tracking for read-only queries. Default is `false`.

`cancellationToken` CancellationToken?

Cancellation token.

Returns

Task<PaginatedDataObj<TEntity>>

Paginated data.

Type Parameters

`TRequestModel`

Interface IReadOnlyDataHandler<TEntity, TIdType>

Namespace: Quickie.DataHandlers.Readonly.Definition

Assembly: Quickie.dll

Read-only data handler (Repository).

```
public interface IReadOnlyDataHandler<TEntity, in TIdType> where TEntity : ReadOnlyEntity
```

Type Parameters

`TEntity`

Entity

`TIdType`

Type of id (int, Guid, string, ...)

Methods

GetByIdAsync(TIdType, CancellationToken?)

Get data by id. (ID must be primary key or composite key.)

```
Task<TEntity?> GetByIdAsync(TIdType id, CancellationToken? cancellationToken = null)
```

Parameters

`id` TIdType

Unique identifier.

`cancellationToken` CancellationToken?

Cancellation token.

Returns

Task <TEntity>

Entity object.

Namespace Quickie.DataHandlers.Readonly.Implementation

Classes

ReadOnlyCollectionDataHandler<TEntity, TDbContext, TIdType>

Abstract class serving as read only (collection) data handler (Repository layer).

ReadOnlyDataHandler<TEntity, TDbContext, TIdType>

Abstract base class for handling read-only data requests.

Class ReadOnlyCollectionDataHandler<TEntity, TDbContext, TIdType>

Namespace: Quickie.DataHandlers.Readonly.Implementation

Assembly: Quickie.dll

Abstract class serving as read only (collection) data handler (Repository layer).

```
public abstract class ReadOnlyCollectionDataHandler<TEntity, TDbContext, TIdType> :
IReadOnlyCollectionDataHandler<TEntity, TIdType> where TEntity : ReadOnlyEntity where
TDbContext : DbContext where TIdType : IComparable
```

Type Parameters

`TEntity`

Entity

`TDbContext`

Db Context

`TIdType`

Type of identifier. eg: int, string, ...

Inheritance

object ← ReadOnlyCollectionDataHandler<TEntity, TDbContext, TIdType>

Implements

IReadOnlyCollectionDataHandler<TEntity, TIdType>

Inherited Members

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

Constructors

ReadOnlyCollectionDataHandler(DbContext)

Abstract class serving as read only (collection) data handler (Repository layer).

```
protected ReadOnlyCollectionDataHandler(DbContext context)
```

Parameters

`context` DbContext

Db Context

Methods

GetByFilterAsync<TRequestModel>(Expression<Func<TEntity, bool>>?, TRequestModel?, bool?, CancellationToken?)

Get filtered collection of data.

```
public Task<PaginatedDataObj<TEntity>> GetByFilterAsync<TRequestModel>
(Expression<Func<TEntity, bool>>? filter, TRequestModel? request, bool? track = false,
CancellationToken? cancellationToken = null) where TRequestModel : RequestForDataObj
```

Parameters

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

Data request filter

`request` TRequestModel

Data request filter

`track` bool?

Track entity? Default is false

`cancellationToken` CancellationToken?

Cancellation token

Returns

Task <PaginatedDataObj<TEntity>>

Type Parameters

`TRequestModel`

Type of data request filter

Class ReadOnlyDataHandler<TEntity, TDbContext, TIdType>

Namespace: Quickie.DataHandlers.Readonly.Implementation

Assembly: Quickie.dll

Abstract base class for handling read-only data requests.

```
public abstract class ReadOnlyDataHandler<TEntity, TDbContext, TIdType> :
IReadOnlyDataHandler<TEntity, TIdType> where TEntity : ReadOnlyEntity where TDbContext :
DbContext where TIdType : IComparable
```

Type Parameters

`TEntity`

Entity

`TDbContext`

Db context

`TIdType`

Type of identifier. eg: int, string, ...

Inheritance

object ← ReadOnlyDataHandler<TEntity, TDbContext, TIdType>

Implements

IReadOnlyDataHandler<TEntity, TIdType>

Inherited Members

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

Constructors

ReadOnlyDataHandler(DbContext)

Abstract base class for handling read-only data requests.

```
protected ReadOnlyDataHandler(DbContext context)
```

Parameters

`context` DbContext

Db context

Methods

GetByIdAsync(TIdType, CancellationToken?)

Get data by id.

```
public Task<TEntity?> GetByIdAsync(TIdType id, CancellationToken? cancellationToken = null)
```

Parameters

`id` TIdType

Identifier (unique key identifying your entity)

`cancellationToken` CancellationToken?

Cancellation token

Returns

Task<TEntity>

Namespace Quickie.DataHandlers.Writeonly.Definition

Interfaces

IWriteOnlyDataHandler<TEntity>

Interface IWriteOnlyDataHandler<TEntity>

Namespace: Quickie.DataHandlers.Writeonly.Definition

Assembly: Quickie.dll

```
public interface IWriteOnlyDataHandler<TEntity> where TEntity : WriteOnlyEntity
```

Type Parameters

`TEntity`

Methods

CreateItemAsync(TEntity, CancellationToken?)

```
Task<ResponseObj<TEntity>> CreateItemAsync(TEntity item, CancellationToken?
cancellationToken)
```

Parameters

`item` TEntity

`cancellationToken` CancellationToken?

Returns

Task<ResponseObj<TEntity>>

CreateItemsAsync(ICollection<TEntity>, CancellationToken?)

```
Task<ResponseObj<ICollection<TEntity>>> CreateItemsAsync(ICollection<TEntity> items,
CancellationToken? cancellationToken)
```

Parameters

`items` ICollection<TEntity>

`cancellationToken` CancellationToken?

Returns

Task<ResponseObj<ICollection<TEntity>>>

Namespace Quickie.DataHandlers.Writeonly.Implementation

Classes

WriteOnlyDataHandler<TEntity, TDbContext>

Abstract class serving as write only data handler (Repository layer).

Class WriteOnlyDataHandler<TEntity, TDbContext>

Namespace: Quickie.DataHandlers.Writeonly.Implementation

Assembly: Quickie.dll

Abstract class serving as write only data handler (Repository layer).

```
public abstract class WriteOnlyDataHandler<TEntity, TDbContext> :
IWriteOnlyDataHandler<TEntity> where TEntity : WriteOnlyEntity where TDbContext : DbContext
```

Type Parameters

`TEntity`

Entity

`TDbContext`

Db context

Inheritance

object ← WriteOnlyDataHandler<TEntity, TDbContext>

Implements

IWriteOnlyDataHandler<TEntity>

Inherited Members

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

Constructors

WriteOnlyDataHandler(TDbContext)

Abstract class serving as write only data handler (Repository layer).

```
protected WriteOnlyDataHandler(TDbContext dbContext)
```

Parameters

`dbContext` TDbContext

Db context

Methods

CreateItemAsync(TEntity, CancellationToken?)

Data create request.

```
public Task<ResponseObj<TEntity>> CreateItemAsync(TEntity item, CancellationToken?
cancellationToken = null)
```

Parameters

`item` TEntity

Data requested to create

`cancellationToken` CancellationToken?

Cancellation token

Returns

Task<ResponseObj<TEntity>>

CreateItemsAsync(ICollection<TEntity>, CancellationToken?)

Collection of data create request.

```
public Task<ResponseObj<ICollection<TEntity>>> CreateItemsAsync(ICollection<TEntity> items,
CancellationToken? cancellationToken = null)
```

Parameters

`items` ICollection<TEntity>

Collection of data requested to create

`cancellationToken` CancellationToken?

Cancellation token

Returns

Task<ResponseObj<ICollection<TEntity>>>

Namespace Quickie.DataObj

Classes

DetailedResponseObj

Represents a detailed response object used to return results from an operation.

PaginatedDataObj<TObject>

Represents an object for paginated data and its metadata.

RequestForDataObj

Represents a request for data with pagination.

ResponseObj

Represents a basic response object that contains information about the success of an operation.

ResponseObj<TData>

Represents a response object that contains the result of an operation along with the data.

Class DetailedResponseObj

Namespace: Quickie.DataObj

Assembly: Quickie.dll

Represents a detailed response object used to return results from an operation.

```
public class DetailedResponseObj
```

Inheritance

object ← DetailedResponseObj

Inherited Members

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

Properties

Description

Detailed description that explains the response such as troubleshooting information or guidance.

```
public string Description { get; set; }
```

Property Value

string

ErrorStatus

Status of the error or response, indicating the severity of the result.

```
public ErrorStatusEnum ErrorStatus { get; set; }
```

Property Value

ErrorStatusEnum

Message

Short message that provides high-level information about the response. Can be used like "title" of your detailed error description.

```
public string Message { get; set; }
```

Property Value

string

Show

Value indicating whether to show the response.

```
public bool Show { get; set; }
```

Property Value

bool

Class PaginatedDataObj<TObject>

Namespace: Quickie.DataObj

Assembly: Quickie.dll

Represents an object for paginated data and its metadata.

```
public class PaginatedDataObj<TObject> where TObject : class
```

Type Parameters

`TObject`

Inheritance

object ← PaginatedDataObj<TObject>

Inherited Members

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

Properties

Items

Collection of data

```
public IEnumerable<TObject> Items { get; set; }
```

Property Value

IEnumerable<TObject>

Total

Total no. of data.

```
public int Total { get; set; }
```

Property Value

int

Class RequestForDataObj

Namespace: Quickie.DataObj

Assembly: Quickie.dll

Represents a request for data with pagination.

```
public class RequestForDataObj
```

Inheritance

object ← RequestForDataObj

Inherited Members

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

Properties

PageNumber

The page number to retrieve. Default is `1`.

```
public int PageNumber { get; set; }
```

Property Value

int

PageSize

The number of items to retrieve per page. Default is `10`.

```
public int PageSize { get; set; }
```

Property Value

int

Class ResponseObj

Namespace: Quickie.DataObj

Assembly: Quickie.dll

Represents a basic response object that contains information about the success of an operation.

```
public class ResponseObj
```

Inheritance

object ← ResponseObj

Inherited Members

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

Properties

IsSuccess

Value indicating whether the operation was successful. "true" meaning operation has succeeded and "false" meaning operation has failed.

```
public bool IsSuccess { get; set; }
```

Property Value

bool

Message

Message providing additional information about the operation's result.

```
public string Message { get; set; }
```

Property Value

string

Class ResponseObj<TData>

Namespace: Quickie.DataObj

Assembly: Quickie.dll

Represents a response object that contains the result of an operation along with the data.

```
public class ResponseObj<TData>
```

Type Parameters

`TData`

Data.

Inheritance

object ← ResponseObj<TData>

Inherited Members

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

Properties

Data

Data returned as a result of the operation.

```
public TData Data { get; set; }
```

Property Value

TData

IsSuccess

Value indicating whether the operation was successful. "true" meaning operation has succeeded and "false" meaning operation has failed.

```
public bool IsSuccess { get; set; }
```

Property Value

bool

Message

Message providing additional information about the operation's result.

```
public string Message { get; set; }
```

Property Value

string

Namespace Quickie.Handlers.Crud.Definition

Interfaces

ICrudForCollectionRequestHandler<TViewModel, TIdType>

ICrudRequestHandler<TViewModel, TIdType>

Abstract base class for CRUD operation. Contains method to: Create, Update, Read (GET), and Delete.

Interface ICrudForCollectionRequestHandler<TViewModel, TIdType>

Namespace: Quickie.Handlers.Crud.Definition

Assembly: Quickie.dll

```
public interface ICrudForCollectionRequestHandler<TViewModel, TIdType> where TViewModel :
CrudDto where TIdType : IComparable
```

Type Parameters

`TViewModel`

`TIdType`

Methods

CountAsync(CancellationToken?)

```
Task<ResponseObj<int>> CountAsync(CancellationToken? cancellationToken = null)
```

Parameters

`cancellationToken` CancellationToken?

Returns

Task<ResponseObj<int>>

CreateRangeAsync(ICollection<TViewModel>, CancellationToken?)

```
Task<ResponseObj<ICollection<TViewModel>>> CreateRangeAsync(ICollection<TViewModel>
entities, CancellationToken? cancellationToken = null)
```

Parameters

`entities` ICollection<TViewModel>

`cancellationToken` CancellationToken?

Returns

Task<ResponseObj<ICollection<TViewModel>>>

DeleteRangeAsync(ICollection<TIdType>, CancellationToken?)

```
Task<ResponseObj> DeleteRangeAsync(ICollection<TIdType> ids, CancellationToken?
cancellationToken = null)
```

Parameters

`ids` ICollection<TIdType>

`cancellationToken` CancellationToken?

Returns

Task<ResponseObj>

GetAllAsync(RequestForDataObj, CancellationToken?)

```
Task<ResponseObj<PaginatedDataObj<TViewModel>>> GetAllAsync(RequestForDataObj request,
CancellationToken? cancellationToken = null)
```

Parameters

`request` RequestForDataObj

`cancellationToken` CancellationToken?

Returns

Task<ResponseObj<PaginatedDataObj<TViewModel>>>

UpdateRangeAsync(ICollection<TViewModel>, CancellationToken?)

```
Task<ResponseObj<ICollection<TViewModel>>> UpdateRangeAsync(ICollection<TViewModel>
entities, CancellationToken? cancellationToken = null)
```

Parameters

`entities` ICollection<TViewModel>

`cancellationToken` CancellationToken?

Returns

Task<ResponseObj<ICollection<TViewModel>>>

Interface ICrudRequestHandler<TViewModel, TIdType>

Namespace: Quickie.Handlers.Crud.Definition

Assembly: Quickie.dll

Abstract base class for CRUD operation. Contains method to: Create, Update, Read (GET), and Delete.

```
public interface ICrudRequestHandler<TViewModel, in TIdType> where TViewModel : CrudDto
where TIdType : IComparable
```

Type Parameters

`TViewModel`

`TIdType`

Type of the identifier used in methods (string, int, Guid, ...)

Methods

CreateAsync(TViewModel, CancellationToken?)

Create data.

```
Task<ResponseObj<TViewModel>> CreateAsync(TViewModel request,
CancellationToken? cancellationToken)
```

Parameters

`request` TViewModel

Requested data to be created.

`cancellationToken` CancellationToken?

Cancellation token

Returns

Task<ResponseObj<TViewModel>>

Response object.

DeleteAsync(TIdType, CancellationToken?)

Removes data.

```
Task<ResponseObj> DeleteAsync(TIdType id, CancellationToken? cancellationToken = null)
```

Parameters

`id` TIdType

Identifier of data to be removed

`cancellationToken` CancellationToken?

Cancellation token

Returns

Task<ResponseObj>

Response object

GetByIdAsync(TIdType, CancellationToken?)

Get data by id (unique identifier, a pk).

```
Task<ResponseObj<TViewModel>> GetByIdAsync(TIdType id, CancellationToken? cancellationToken
= null)
```

Parameters

`id` TIdType

Identifier

`cancellationToken` CancellationToken?

Cancellation token

Returns

Task<ResponseObj<TViewModel>>

Response object with data.

UpdateAsync(TIdType, TViewModel, CancellationToken?)

Update requested data.

```
Task<ResponseObj<TViewModel>> UpdateAsync(TIdType id, TViewModel request, CancellationToken?
cancellationToken = null)
```

Parameters

`id` TIdType

Data identifier to update.

`request` TViewModel

Data requested to be updated.

`cancellationToken` CancellationToken?

Cancellation token

Returns

Task<ResponseObj<TViewModel>>

Response object with updated data.

Namespace Quickie.Handlers.Crud.Implementation

Classes

CrudForCollectionRequestHandler<TViewModel, TEntity, TDataHandler, TIdType>

Abstract class serving as CRUD for collection request handler.

CrudRequestHandler<TViewModel, TEntity, TDataHandler, TIdType>

Abstract class serving as CRUD request handler.

Class CrudForCollectionRequestHandler<TViewModel, TEntity, TDataHandler, TIdType>

Namespace: Quickie.Handlers.Crud.Implementation

Assembly: Quickie.dll

Abstract class serving as CRUD for collection request handler.

```
public abstract class CrudForCollectionRequestHandler<TViewModel, TEntity, TDataHandler,
TIdType> : ICrudForCollectionRequestHandler<TViewModel, TIdType> where TViewModel : CrudDto
where TEntity : CrudEntity where TDataHandler : ICrudForCollectionDataHandler<TEntity,
TIdType> where TIdType : IComparable
```

Type Parameters

`TViewModel`

View model or dto

`TEntity`

Entity

`TDataHandler`

Data handler (repository layer)

`TIdType`

Type of identifier. eg: int, string, ...

Inheritance

object ← CrudForCollectionRequestHandler<TViewModel, TEntity, TDataHandler, TIdType>

Implements

ICrudForCollectionRequestHandler<TViewModel, TIdType>

Inherited Members

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

Constructors

CrudForCollectionRequestHandler(TDataHandler)

Abstract class serving as CRUD for collection request handler.

```
protected CrudForCollectionRequestHandler(TDataHandler dataHandler)
```

Parameters

`dataHandler` TDataHandler

Data

Methods

CountAsync(CancellationToken?)

Count.

```
public Task<ResponseObj<int>> CountAsync(CancellationToken? cancellationToken = null)
```

Parameters

`cancellationToken` CancellationToken?

Cancellation token

Returns

Task<ResponseObj<int>>

CreateRangeAsync(ICollection<TViewModel>, CancellationToken?)

Data create request.

```
public Task<ResponseObj<ICollection<TViewModel>>> CreateRangeAsync(ICollection<TViewModel>
entities, CancellationToken? cancellationToken = null)
```

Parameters

`entities` ICollection<TViewModel>

Collection of data requested to create.

`cancellationToken` CancellationToken?

Cancellation token

Returns

Task<ResponseObj<ICollection<TViewModel>>>

DeleteRangeAsync(ICollection<TIdType>, CancellationToken?)

Delete request.

```
public Task<ResponseObj> DeleteRangeAsync(ICollection<TIdType> ids, CancellationToken?
cancellationToken = null)
```

Parameters

`ids` ICollection<TIdType>

Identifiers of the data to remove.

`cancellationToken` CancellationToken?

Cancellation token

Returns

Task<ResponseObj>

GetAllAsync(RequestForDataObj, CancellationToken?)

Get data.

```
public Task<ResponseObj<PaginatedDataObj<TViewModel>>> GetAllAsync(RequestForDataObj
request, CancellationToken? cancellationToken = null)
```

Parameters

`request` RequestForDataObj

Data request filter

`cancellationToken` CancellationToken?

Cancellation token

Returns

Task<ResponseObj<PaginatedDataObj<TViewModel>>>

MapToCollectionOfDto(ICollection<TEntity>)

Maps a collection of `TEntity` to a collection of `TViewModel` dto.

```
protected abstract ICollection<TViewModel> MapToCollectionOfDto(ICollection<TEntity> entity)
```

Parameters

`entity` ICollection<TEntity>

Returns

ICollection<TViewModel>

MapToCollectionOfEntity(ICollection<TViewModel>)

Maps a collection of `TViewModel` to a collection of `TEntity` entity.

```
protected abstract ICollection<TEntity> MapToCollectionOfEntity(ICollection<TViewModel>
```

```
entity)
```

Parameters

`entity` ICollection<TViewModel>

Returns

ICollection<TEntity>

UpdateRangeAsync(ICollection<TViewModel>, CancellationToken?)

Update request.

```
public Task<ResponseObj<ICollection<TViewModel>>> UpdateRangeAsync(ICollection<TViewModel>
entities, CancellationToken? cancellationToken = null)
```

Parameters

`entities` ICollection<TViewModel>

Requested collection data

`cancellationToken` CancellationToken?

Cancellation token

Returns

Task<ResponseObj<ICollection<TViewModel>>>

Class CrudRequestHandler<TViewModel, TEntity, TDataHandler, TIdType>

Namespace: Quickie.Handlers.Crud.Implementation

Assembly: Quickie.dll

Abstract class serving as CRUD request handler.

```
public abstract class CrudRequestHandler<TViewModel, TEntity, TDataHandler, TIdType>
: ICrudRequestHandler<TViewModel, TIdType> where TViewModel : CrudDto where TEntity
: CrudEntity where TDataHandler : ICrudDataHandler<TEntity, TIdType> where TIdType
: IComparable
```

Type Parameters

`TViewModel`

View model or dto

`TEntity`

Entity

`TDataHandler`

Data handler (repository layer)

`TIdType`

Type of identifier. eg: int, string, ...

Inheritance

object ← CrudRequestHandler<TViewModel, TEntity, TDataHandler, TIdType>

Implements

ICrudRequestHandler<TViewModel, TIdType>

Inherited Members

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

Constructors

CrudRequestHandler(ICrudDataHandler<TEntity, TIdType>)

Abstract class serving as CRUD request handler.

```
protected CrudRequestHandler(ICrudDataHandler<TEntity, TIdType> dataHandler)
```

Parameters

`dataHandler` ICrudDataHandler<TEntity, TIdType>

Data handler (repository layer)

Methods

CreateAsync(TViewModel, CancellationToken?)

Data create request.

```
public Task<ResponseObj<TViewModel>> CreateAsync(TViewModel request, CancellationToken?
cancellationToken = null)
```

Parameters

`request` TViewModel

Data requested to create.

`cancellationToken` CancellationToken?

Cancellation token

Returns

Task<ResponseObj<TViewModel>>

DeleteAsync(TIdType, CancellationToken?)

Delete request.

```
public Task<ResponseObj> DeleteAsync(TIdType id, CancellationToken? cancellationToken
= null)
```

Parameters

`id` TIdType

Identifier of the data to remove.

`cancellationToken` CancellationToken?

Cancellation token

Returns

Task<ResponseObj>

Exceptions

ArgumentException

GetByIdAsync(TIdType, CancellationToken?)

Get data by providing an identifier

```
public Task<ResponseObj<TViewModel>> GetByIdAsync(TIdType id, CancellationToken?
cancellationToken = null)
```

Parameters

`id` TIdType

Identifier

`cancellationToken` CancellationToken?

Cancellation token

Returns

Task <ResponseObj<TViewModel>>

Exceptions

ArgumentException

MapToDto(TEntity)

Maps a `TEntity` to a `TViewModel` dto.

```
protected abstract TViewModel MapToDto(TEntity request)
```

Parameters

`request` TEntity

Entity

Returns

TViewModel

Dto

MapToEntity(TViewModel)

Maps a `TViewModel` to a `TEntity` entity.

```
protected abstract TEntity MapToEntity(TViewModel request)
```

Parameters

`request` TViewModel

Dto

Returns

TEntity

Entity

UpdateAsync(TIdType, TViewModel, CancellationToken?)

Update request.

```
public Task<ResponseObj<TViewModel>> UpdateAsync(TIdType id, TViewModel request,
CancellationToken? cancellationToken = null)
```

Parameters

`id` TIdType

Requested data identifier.

`request` TViewModel

Requested data

`cancellationToken` CancellationToken?

Cancellation token

Returns

Task<ResponseObj<TViewModel>>

Namespace Quickie.Handlers.Editonly.Definition

Interfaces

IEditOnlyRequestHandler<TViewModel, TIdType>

Abstract base class for handling update requests, processing operations for the specified view model type.

Interface IEditOnlyRequestHandler<TViewModel, TIdType>

Namespace: Quickie.Handlers.Editonly.Definition

Assembly: Quickie.dll

Abstract base class for handling update requests, processing operations for the specified view model type.

```
public interface IEditOnlyRequestHandler<TViewModel, in TIdType> where TViewModel :
EditOnlyDto where TIdType : IComparable
```

Type Parameters

`TViewModel`

The type of the view model that represents the data transfer object to be updated.

`TIdType`

Type of id (int, string, guid, ...)

Methods

EditAsync(TIdType, TViewModel, CancellationToken?)

Update single entity.

```
Task<ResponseObj<TViewModel>> EditAsync(TIdType id, TViewModel requestModel,
CancellationToken? cancellationToken)
```

Parameters

`id` TIdType

Identifier of the entity.

`requestModel` TViewModel

Data to be updated.

`cancellationToken` CancellationToken?

Cancellation token

Returns

Task<ResponseObj<TViewModel>>

EditCollectionAsync(ICollection<TViewModel>, CancellationToken?)

Update collection of entity. (Bulk)

```
Task<ResponseObj<ICollection<TViewModel>>> EditCollectionAsync(ICollection<TViewModel>
requestModels, CancellationToken? cancellationToken)
```

Parameters

`requestModels` ICollection<TViewModel>

Bulk data to be updated.

`cancellationToken` CancellationToken?

Cancellation token.

Returns

Task<ResponseObj<ICollection<TViewModel>>>

Namespace Quickie.Handlers.Editonly.Implementation

Classes

EditOnlyRequestHandler<TViewModel, TEntity, TDataHandler, TIdType>

Abstract class serving as edit only request handler.

Class EditOnlyRequestHandler<TViewModel, TEntity, TDataHandler, TIdType>

Namespace: Quickie.Handlers.Editonly.Implementation

Assembly: Quickie.dll

Abstract class serving as edit only request handler.

```
public abstract class EditOnlyRequestHandler<TViewModel, TEntity, TDataHandler, TIdType> :
IEditOnlyRequestHandler<TViewModel, TIdType> where TViewModel : EditOnlyDto where TEntity
: EditOnlyEntity where TDataHandler : IEditOnlyDataHandler<TEntity, TIdType> where TIdType
: IComparable
```

Type Parameters

`TViewModel`

View model or dto

`TEntity`

Entity

`TDataHandler`

Data handler (Repository layer)

`TIdType`

Type of identifier. eg: int, string, ...

Inheritance

object ← EditOnlyRequestHandler<TViewModel, TEntity, TDataHandler, TIdType>

Implements

IEditOnlyRequestHandler<TViewModel, TIdType>

Inherited Members

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

Constructors

EditOnlyRequestHandler(IEditOnlyDataHandler<TEntity, TIdType>)

Abstract class serving as edit only request handler.

```
protected EditOnlyRequestHandler(IEditOnlyDataHandler<TEntity, TIdType> dataHandler)
```

Parameters

`dataHandler` IEditOnlyDataHandler<TEntity, TIdType>

Data handler (Repository layer)

Methods

EditAsync(TIdType, TViewModel, CancellationToken?)

Update request.

```
public Task<ResponseObj<TViewModel>> EditAsync(TIdType id, TViewModel requestModel,
CancellationToken? cancellationToken = null)
```

Parameters

`id` TIdType

Identifier of the entity.

`requestModel` TViewModel

Requested data to update

`cancellationToken` CancellationToken?

Cancellation token

Returns

Task<ResponseObj<TViewModel>>

EditCollectionAsync(ICollection<TViewModel>, CancellationToken?)

Update collection of data.

```
public Task<ResponseObj<ICollection<TViewModel>>>
EditCollectionAsync(ICollection<TViewModel> requestModels, CancellationToken?
cancellationToken = null)
```

Parameters

`requestModels` ICollection<TViewModel>

Collection of requested data to update

`cancellationToken` CancellationToken?

Cancellation token

Returns

Task<ResponseObj<ICollection<TViewModel>>>

MapToCollectionOfDto(ICollection<TEntity>)

Maps a collection of `TEntity` entities to a collection of `TViewModel` view models.

```
protected abstract ICollection<TViewModel> MapToCollectionOfDto(ICollection<TEntity>
entities)
```

Parameters

`entities` ICollection<TEntity>

Collection of entity.

Returns

ICollection<TViewModel>

Collection of view model/dto.

MapToCollectionOfEntity(ICollection<TViewModel>)

Maps a collection of `TViewModel` to a collection of `TEntity` entities.

```
protected abstract ICollection<TEntity> MapToCollectionOfEntity(ICollection<TViewModel>
requestModels)
```

Parameters

`requestModels` ICollection<TViewModel>

Collection of view model/dto

Returns

ICollection<TEntity>

Collection of entity.

MapToDto(TEntity)

Maps a `TEntity` entity to a `TViewModel` view model/dto.

```
protected abstract TViewModel MapToDto(TEntity entity)
```

Parameters

`entity` TEntity

Entity.

Returns

TViewModel

View model/dto.

MapToEntity(TViewModel)

Maps a `TViewModel` to a `TEntity` entity.

```
protected abstract TEntity MapToEntity(TViewModel requestModel)
```

Parameters

`requestModel` TViewModel

The view model/dto to map to an entity

Returns

TEntity

Entity.

Namespace Quickie.Handlers.Readonly. Definition

Interfaces

IReadOnlyCollectionRequestHandler<TViewModel, TIdType>

Abstract base class for handling collections of read-only requests, processing operations for the specified view model type.

IReadOnlyRequestHandler<TViewModel, TIdType>

Abstract base class for handling read-only requests, processing operations for the specified view model type.

Interface IReadOnlyCollectionRequestHandler<TViewModel, TIdType>

Namespace: Quickie.Handlers.Readonly.Definition

Assembly: Quickie.dll

Abstract base class for handling collections of read-only requests, processing operations for the specified view model type.

```
public interface IReadOnlyCollectionRequestHandler<TViewModel, in TIdType> where TViewModel
: ReadOnlyDto where TIdType : IComparable
```

Type Parameters

`TViewModel`

The type of the view model that represents the data transfer object for read-only operations.

`TIdType`

Type of id (int, Guid, string, ...)

Methods

GetAsync<TRequestModel>(TRequestModel?, CancellationToken?)

Get collection of data by provided request model.

```
Task<PaginatedDataObj<TViewModel>> GetAsync<TRequestModel>(TRequestModel? request,
CancellationToken? cancellationToken) where TRequestModel : RequestForDataObj
```

Parameters

`request` TRequestModel

Data request model of type RequestForDataDto.

`cancellationToken` CancellationToken?

Cancellation token.

Returns

Task<PaginatedDataObj<TViewModel>>

Collection of data wrapped by response dto.

Type Parameters

`TRequestModel`

Interface IReadOnlyRequestHandler<TViewModel, TIdType>

Namespace: Quickie.Handlers.Readonly.Definition

Assembly: Quickie.dll

Abstract base class for handling read-only requests, processing operations for the specified view model type.

```
public interface IReadOnlyRequestHandler<TViewModel, in TIdType> where TViewModel :
ReadOnlyDto where TIdType : IComparable
```

Type Parameters

`TViewModel`

The type of the view model that represents the data transfer object for read-only operations.

`TIdType`

Type of id (int, Guid, string, ...)

Methods

GetByIdAsync(TIdType, CancellationToken?)

Get exact data (single) by id.

```
Task<ResponseObj<TViewModel>> GetByIdAsync(TIdType id, CancellationToken? cancellationToken)
```

Parameters

`id` TIdType

Identifier of data.

`cancellationToken` CancellationToken?

Cancellation token.

Returns

Task<ResponseObj<TViewModel>>

Single data wrapped by response dto.

Namespace Quickie.Handlers.Readonly.Implementation

Classes

ReadOnlyCollectionRequestHandler<TViewModel, TEntity, TDataHandler, TIdType>

Abstract class serving as read only (collection) request handler.

ReadOnlyRequestHandler<TViewModel, TEntity, TDataHandler, TIdType>

Abstract base class for handling read-only requests, processing operations for the specified view model type.

Class ReadOnlyCollectionRequestHandler<TViewModel, TEntity, TDataHandler, TIdType>

Namespace: Quickie.Handlers.Readonly.Implementation

Assembly: Quickie.dll

Abstract class serving as read only (collection) request handler.

```
public abstract class ReadOnlyCollectionRequestHandler<TViewModel, TEntity, TDataHandler,
TIdType> : IReadOnlyCollectionRequestHandler<TViewModel, TIdType> where TViewModel :
ReadOnlyDto where TEntity : ReadOnlyEntity where TDataHandler :
IReadOnlyCollectionDataHandler<TEntity, TIdType> where TIdType : IComparable
```

Type Parameters

`TViewModel`

View model or dto

`TEntity`

Entity

`TDataHandler`

Data handler (Repository layer)

`TIdType`

Type of identifier. eg: int, string, ...

Inheritance

object ← ReadOnlyCollectionRequestHandler<TViewModel, TEntity, TDataHandler, TIdType>

Implements

IReadOnlyCollectionRequestHandler<TViewModel, TIdType>

Inherited Members

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

Constructors

ReadOnlyCollectionRequestHandler(IReadOnlyCollectionDataHandler<TEntity, TIdType>)

Abstract class serving as read only (collection) request handler.

```
protected ReadOnlyCollectionRequestHandler(IReadOnlyCollectionDataHandler<TEntity,
TIdType> readOnlyCollectionDataHandler)
```

Parameters

`readOnlyCollectionDataHandler` IReadOnlyCollectionDataHandler<TEntity, TIdType>

Data handler (Repository layer)

Methods

GetAsync<TRequestModel>(TRequestModel?, CancellationToken?)

Get data.

```
public Task<PaginatedDataObj<TViewModel>> GetAsync<TRequestModel>(TRequestModel? request,
CancellationToken? cancellationToken = null) where TRequestModel : RequestForDataObj
```

Parameters

`request` TRequestModel

Data request filter

`cancellationToken` CancellationToken?

Cancellation token

Returns

Task<PaginatedDataObj<TViewModel>>

Type Parameters

`TRequestModel`

Type of data request filter

MapToDto(ICollection<TEntity>)

Mapping profile. Map from collection of Entity type to your collection of data transfer object.

```
protected abstract ICollection<TViewModel> MapToDto(ICollection<TEntity> entity)
```

Parameters

`entity` ICollection<TEntity>

Collection of entity object.

Returns

ICollection<TViewModel>

Mapped from collection of Entity, a collection data transfer object.

Class ReadOnlyRequestHandler<TViewModel, TEntity, TDataHandler, TIdType>

Namespace: Quickie.Handlers.Readonly.Implementation

Assembly: Quickie.dll

Abstract base class for handling read-only requests, processing operations for the specified view model type.

```
public abstract class ReadOnlyRequestHandler<TViewModel, TEntity, TDataHandler, TIdType> :
IReadOnlyRequestHandler<TViewModel, TIdType> where TViewModel : ReadOnlyDto where TEntity
: ReadOnlyEntity where TDataHandler : IReadOnlyDataHandler<TEntity, TIdType> where TIdType
: IComparable
```

Type Parameters

`TViewModel`

The type of the view model that represents the data transfer object for read-only operations.

`TEntity`

Entity.

`TDataHandler`

Data handler (Repository)

`TIdType`

Type of identifier. eg: int, string, ...

Inheritance

object ← ReadOnlyRequestHandler<TViewModel, TEntity, TDataHandler, TIdType>

Implements

IReadOnlyRequestHandler<TViewModel, TIdType>

Inherited Members

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

Constructors

ReadOnlyRequestHandler(IReadOnlyDataHandler<TEntity, TIdType>)

Abstract base class for handling read-only requests, processing operations for the specified view model type.

```
protected ReadOnlyRequestHandler(IReadOnlyDataHandler<TEntity, TIdType> dataHandler)
```

Parameters

`dataHandler` IReadOnlyDataHandler<TEntity, TIdType>

Methods

GetByIdAsync(TIdType, CancellationToken?)

Get data by id.

```
public Task<ResponseObj<TViewModel>> GetByIdAsync(TIdType id, CancellationToken?
cancellationToken = null)
```

Parameters

`id` TIdType

Identifier (unique key identifying your entity)

`cancellationToken` CancellationToken?

Cancellation token

Returns

Task<ResponseObj<TViewModel>>

Exceptions

ArgumentException

MapToDto(TEntity)

Mapping profile. Map from Entity type to your data transfer object.

```
protected abstract TViewModel MapToDto(TEntity entity)
```

Parameters

`entity` TEntity

Entity object.

Returns

TViewModel

Mapped from Entity, a data transfer object.

Namespace Quickie.Handlers.Writeonly. Definition

Interfaces

IWriteOnlyRequestHandler<TViewModel>

Interface IWriteOnlyRequestHandler<TViewModel>

Namespace: Quickie.Handlers.Writeonly.Definition

Assembly: Quickie.dll

```
public interface IWriteOnlyRequestHandler<TViewModel> where TViewModel : WriteOnlyDto
```

Type Parameters

`TViewModel`

Methods

CreateItemAsync(TViewModel, CancellationToken?)

```
Task<ResponseObj<TViewModel>> CreateItemAsync(TViewModel item,
CancellationToken? cancellationToken)
```

Parameters

`item` TViewModel

`cancellationToken` CancellationToken?

Returns

Task<ResponseObj<TViewModel>>

CreateItemsAsync(ICollection<TViewModel>, CancellationToken?)

```
Task<ResponseObj<ICollection<TViewModel>>> CreateItemsAsync(ICollection<TViewModel> items,
CancellationToken? cancellationToken)
```

Parameters

`items` ICollection<TViewModel>

`cancellationToken` CancellationToken?

Returns

Task<ResponseObj<ICollection<TViewModel>>>

Namespace Quickie.Handlers.Writeonly.Implementation

Classes

WriteOnlyRequestHandler<TViewModel, TEntity, TDataHandler>

Abstract class serving as write only request handler.

Class WriteOnlyRequestHandler<TViewModel, TEntity, TDataHandler>

Namespace: Quickie.Handlers.Writeonly.Implementation

Assembly: Quickie.dll

Abstract class serving as write only request handler.

```
public abstract class WriteOnlyRequestHandler<TViewModel, TEntity, TDataHandler> :
IWriteOnlyRequestHandler<TViewModel> where TViewModel : WriteOnlyDto where TEntity :
WriteOnlyEntity where TDataHandler : IWriteOnlyDataHandler<TEntity>
```

Type Parameters

`TViewModel`

`TEntity`

`TDataHandler`

Inheritance

object ← WriteOnlyRequestHandler<TViewModel, TEntity, TDataHandler>

Implements

IWriteOnlyRequestHandler<TViewModel>

Inherited Members

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

Constructors

WriteOnlyRequestHandler(IWriteOnlyDataHandler<TEntity>)

Abstract class serving as write only request handler.

```
protected WriteOnlyRequestHandler(IWriteOnlyDataHandler<TEntity> dataHandler)
```

Parameters

`dataHandler` IWriteOnlyDataHandler<TEntity>

Methods

CreateItemAsync(TViewModel, CancellationToken?)

Data create request.

```
public Task<ResponseObj<TViewModel>> CreateItemAsync(TViewModel item,
CancellationToken? cancellationToken)
```

Parameters

`item` TViewModel

Data requested to create.

`cancellationToken` CancellationToken?

Cancellation token

Returns

Task<ResponseObj<TViewModel>>

CreateItemsAsync(ICollection<TViewModel>, CancellationToken?)

Data create request.

```
public Task<ResponseObj<ICollection<TViewModel>>> CreateItemsAsync(ICollection<TViewModel>
items, CancellationToken? cancellationToken)
```

Parameters

`items` ICollection<TViewModel>

Collection of data requested to create.

`cancellationToken` CancellationToken?

Cancellation token

Returns

Task<ResponseObj<ICollection<TViewModel>>>

MapToCollectionOfDto(ICollection<TEntity>)

Maps a collection of `TEntity` entities to a collection of `TViewModel` view models.

```
protected abstract ICollection<TViewModel> MapToCollectionOfDto(ICollection<TEntity>
entities)
```

Parameters

`entities` ICollection<TEntity>

Collection of entity.

Returns

ICollection<TViewModel>

Collection of view model/dto.

MapToCollectionOfEntity(ICollection<TViewModel>)

Maps a collection of `TViewModel` to a collection of `TEntity` entities.

```
protected abstract ICollection<TEntity> MapToCollectionOfEntity(ICollection<TViewModel>
requestModels)
```

Parameters

`requestModels` ICollection<TViewModel>

Collection of view model/dto

Returns

ICollection <TEntity>

Collection of entity.

MapToDto(TEntity)

Maps a `TEntity` entity to a `TViewModel` view model/dto.

```
protected abstract TViewModel MapToDto(TEntity entity)
```

Parameters

`entity` TEntity

Entity.

Returns

TViewModel

View model/dto.

MapToEntity(TViewModel)

Maps a `TViewModel` to a `TEntity` entity.

```
protected abstract TEntity MapToEntity(TViewModel requestModel)
```

Parameters

`requestModel` TViewModel

The view model/dto to map to an entity

Returns

TEntity

Entity.

Namespace Quickie.Types

Enums

ErrorStatusEnum

Status representing the different severity levels of an error or message.

Enum ErrorStatusEnum

Namespace: Quickie.Types

Assembly: Quickie.dll

Status representing the different severity levels of an error or message.

```
public enum ErrorStatusEnum
```

Fields

`Critical = 2`

Represents critical errors that require immediate attention. These typically indicate a failure or something that must be fixed for the application to function properly.

`Info = 0`

Represents informational messages that do not indicate any issues. These are typically non-critical and meant for logging or user information.

`Warning = 1`

Represents warning messages that indicate potential issues. These require attention but do not block the operation or indicate a failure.

Namespace Quickie.Types.Crud

Classes

CrudDto

Dto representing CRUD operation

CrudEntity

Entity representing CRUD operation

Class CrudDto

Namespace: Quickie.Types.Crud

Assembly: Quickie.dll

Dto representing CRUD operation

```
public abstract record CrudDto : IEquatable<CrudDto>
```

Inheritance

object ← CrudDto

Implements

IEquatable<CrudDto>

Inherited Members

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

Constructors

CrudDto()

Dto representing CRUD operation

```
protected CrudDto()
```

Class CrudEntity

Namespace: Quickie.Types.Crud

Assembly: Quickie.dll

Entity representing CRUD operation

```
public abstract class CrudEntity
```

Inheritance

object ← CrudEntity

Inherited Members

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

Namespace Quickie.Types.Editonly

Classes

EditOnlyDto

Dto representing EDIT only operation

EditOnlyEntity

Entity representing EDIT only operation

Class EditOnlyDto

Namespace: Quickie.Types.Editonly

Assembly: Quickie.dll

Dto representing EDIT only operation

```
public abstract record EditOnlyDto : IEquatable<EditOnlyDto>
```

Inheritance

object ← EditOnlyDto

Implements

IEquatable<EditOnlyDto>

Inherited Members

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

Constructors

EditOnlyDto()

Dto representing EDIT only operation

```
protected EditOnlyDto()
```

Class EditOnlyEntity

Namespace: Quickie.Types.Editonly

Assembly: Quickie.dll

Entity representing EDIT only operation

```
public abstract class EditOnlyEntity
```

Inheritance

object ← EditOnlyEntity

Inherited Members

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

Constructors

EditOnlyEntity()

Entity representing EDIT only operation

```
protected EditOnlyEntity()
```

Namespace Quickie.Types.Readonly

Classes

ReadOnlyDto

Dto representing READ only operation

ReadOnlyEntity

Entity representing READ only operation

Class ReadOnlyDto

Namespace: Quickie.Types.Readonly

Assembly: Quickie.dll

Dto representing READ only operation

```
public abstract record ReadOnlyDto : IEquatable<ReadOnlyDto>
```

Inheritance

object ← ReadOnlyDto

Implements

IEquatable<ReadOnlyDto>

Inherited Members

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

Constructors

ReadOnlyDto()

Dto representing READ only operation

```
protected ReadOnlyDto()
```

Class ReadOnlyEntity

Namespace: Quickie.Types.Readonly

Assembly: Quickie.dll

Entity representing READ only operation

```
public abstract class ReadOnlyEntity
```

Inheritance

object ← ReadOnlyEntity

Inherited Members

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

Namespace Quickie.Types.Writable

Classes

WriteOnlyDto

Dto representing WRITE only operation

WriteOnlyEntity

Entity representing WRITE only operation

Class WriteOnlyDto

Namespace: Quickie.Types.Writable

Assembly: Quickie.dll

Dto representing WRITE only operation

```
public record WriteOnlyDto : IEquatable<WriteOnlyDto>
```

Inheritance

object ← WriteOnlyDto

Implements

IEquatable<WriteOnlyDto>

Inherited Members

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

Constructors

WriteOnlyDto()

Dto representing WRITE only operation

```
public WriteOnlyDto()
```

Class WriteOnlyEntity

Namespace: Quickie.Types.Writable

Assembly: Quickie.dll

Entity representing WRITE only operation

```
public abstract class WriteOnlyEntity
```

Inheritance

object ← WriteOnlyEntity

Inherited Members

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